

Single-step Dimension Reduction for High-dimensional Data Analysis  
with Application in Reinforcement Learning  
(高次元データ解析のためのシングルステップ次元削減と  
その強化学習への応用)

by

Tangkaratt Voot  
タンカラット ブット

A Doctor Thesis  
博士論文

Submitted to  
the Graduate School of the University of Tokyo  
on 9 December 2016  
in Partial Fulfillment of the Requirements  
for the Degree of Doctor of Information Science and Technology  
in Computer Science

Thesis Supervisor: Masashi Sugiyama 杉山将  
Professor of Computer Science

## ABSTRACT

Machine learning has become an important field due to the necessity of analyzing complex data. Machine learning is categorized into three paradigms: supervised learning, unsupervised learning, and reinforcement learning. Supervised and unsupervised learning have a long history and they have been extensively studied in the machine learning literature. However, they still have a weakness when they are applied to high-dimensional problems. Reinforcement learning has also been studied for decades, but its study has been rather limited and it has more room for improvement. In this dissertation, we firstly propose a dimension reduction method to mitigate the high dimensionality limitation in supervised and unsupervised learning. Secondly, we propose a dimension reduction method for the conditional density estimation problem. Then, we propose a model-based reinforcement learning method which does not rely on strong assumptions unlike existing methods, and we subsequently improve its performance by utilizing our dimension reduction method. Finally, we propose a contextual reinforcement learning method that effectively learns from high-dimensional contexts by utilizing the idea of dimension reduction.

In the first part of this dissertation, we focus on developing two single-step dimension reduction methods. Dimension reduction is a standard tool in machine learning and many methods were proposed in literature. However, existing methods are sensitive to outliers which are common phenomena in practice. To solve this problem, we propose a dimension reduction method based on the maximization of quadratic mutual information (QMI) which is a robust statistical dependence measure. Solving this maximization problem requires an accurate estimate of the derivative of QMI. A common approach first estimates QMI from data and then computes the derivatives of the estimated QMI. However, this multi-step approach is not appropriate since the derivative of an accurate QMI estimator is not necessarily an accurate estimator of the derivative of QMI. Instead, we propose to directly estimate the derivative of QMI in a single-step manner without estimating QMI itself. Experimental evaluations on high-dimensional regression problems show that our single-step QMI-based dimension reduction method works better and is more robust against outliers than existing methods.

While our QMI-based dimension reduction method is useful for both supervised and unsupervised learning problems, it may not be the optimal method for a supervised learning problem called conditional density estimation. Conditional density estimation aims to estimate a conditional probability density of output given input, and is highly useful for analyzing a relationship between input and output. However, accurately estimating a conditional density from high-dimensional input is challenging. Moreover, a multi-step approach which first performs dimension reduction and then performs conditional density estimation is not always appropriate. To solve this problem, we propose the least-squares conditional entropy (LSCE) method which simultaneously performs non-parametric conditional density estimation and dimension reduction in an integrated manner. Experimental evaluations on high-dimensional conditional density estimation problems show that LSCE estimates conditional densities more accurately than methods based on the multi-step approach.

In the second part of this dissertation, we focus on utilizing dimension reduction in high-dimensional reinforcement learning problems. The goal of reinforcement learning is to learn an optimal policy which controls an agent to receive the maximum cumulative rewards. Among reinforcement learning methods, policy gradient methods are widely applicable. However, accurately estimating policy gradients requires a large amount of data. The model-based approach can cope with this issue by first estimating a transition model from data and then using the model to accurately estimate policy gradients. An advantage of the model-based approach over the model-free approach is that once the transition model is accurately estimated, the agent does not need to collect more data to learn an optimal policy. Moreover, when the budget for collecting data is limited, the model-based approach does not need to determine the sampling schedule, unlike the model-free approach. However, the state-of-the-art method

for transition model estimation is based on the strong assumption that the transition dynamics can be accurately modeled by the Gaussian distribution. To avoid assuming such a strong assumption, we propose a model-based policy gradient method which uses the least-squares conditional density estimation (LSCDE) method to estimate a transition model. LSCDE is an existing non-parametric conditional density estimation method and it can asymptotically estimate any conditional density. We evaluate our proposed method through experiments on benchmark problems and a simulated humanoid robot control problem, and show that our model-based policy gradient method gives better performance than existing methods when the amount of data is limited.

However, LSCDE still requires a relatively large amount of data in high-dimensional reinforcement learning problems. To mitigate this limitation, we propose to improve our model-based policy gradient method by estimating a transition model using our LSCE method. Experimental evaluations on benchmark control problems and a real humanoid robot control problem show that our model-based policy gradient method with LSCE gives the best performance among compared methods.

Although our model-based reinforcement learning method with single-step dimension reduction works well for standard reinforcement learning, it may not be an appropriate method for contextual reinforcement learning. In contextual reinforcement learning, an agent requires to learn different optimal policies for different contexts of a problem. Contextual reinforcement learning is challenging especially when contexts have high dimensionality. To overcome this challenge, we propose a contextual reinforcement learning method where our key idea is to learn a low-rank representation of a model of the cumulative rewards. We show that learning the low-rank representation actually corresponds to a single-step approach to simultaneously performing dimension reduction for model learning. We evaluate our method on a benchmark problem and robot ball hitting problems based on camera images. The experimental results show that our method gives the best performance among compared methods.

In this dissertation, we have proposed five methods that overcome the weaknesses of existing machine learning methods in high-dimensional problems. Based on our empirical evaluations on both benchmark and real-world problems, we conclude that our methods successfully overcome the above mentioned weaknesses of existing machine learning methods.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Machine Learning . . . . .	1
1.2	High-dimensional Data Analysis . . . . .	5
1.2.1	Linear Dimension Reduction . . . . .	6
1.2.2	Non-linear Dimension Reduction and Deep Learning . . . . .	7
1.3	Single-step Approach to Linear Dimension Reduction . . . . .	9
1.4	Contributions . . . . .	9
1.4.1	Development of Single-step Dimension Reduction Methods . . . . .	10
1.4.2	Applications of Single-step Dimension Reduction in Reinforcement Learning . . . . .	11
1.5	Organization . . . . .	12
<b>2</b>	<b>Background of Linear Dimension Reduction</b>	<b>14</b>
2.1	Linear Dimension Reduction . . . . .	15
2.2	Unsupervised Linear Dimension Reduction . . . . .	17
2.2.1	Principal Component Analysis . . . . .	17
2.2.2	Locality Preserving Projection . . . . .	18
2.3	Supervised Linear Dimension Reduction . . . . .	20
2.3.1	Linear Discriminant Analysis . . . . .	20
2.3.2	Sliced Inverse Regression . . . . .	21
2.3.3	Methods based on Gradients of Conditional Density functions . . . . .	22
2.3.4	Minimum Average Variance Estimation based on the Conditional Density Functions . . . . .	24
2.4	Dimension Reduction based on Statistical Dependence . . . . .	25
2.4.1	Dimension Reduction and Statistical Dependence . . . . .	25
2.4.2	Pearson Correlation Coefficient . . . . .	26
2.4.3	Conditional Covariance Operator on Reproducing Kernel Hilbert Spaces . . . . .	26
2.4.4	Mutual Information . . . . .	27
2.4.5	Squared-loss Mutual Information . . . . .	29
2.4.6	Quadratic Mutual Information . . . . .	29
2.5	Non-linearized Linear Dimension Reduction . . . . .	30
2.6	Summary of Dimension Reduction . . . . .	32
<b>3</b>	<b>Dimension Reduction via Single-step Estimation of the Derivative of Quadratic Mutual Information</b>	<b>33</b>
3.1	Introduction . . . . .	33
3.2	Quadratic Mutual Information for Dimension Reduction . . . . .	34
3.2.1	Quadratic Mutual Information . . . . .	35

3.2.2	Estimation method based on Density Estimation . . . . .	36
3.2.3	Least-Squares Quadratic Mutual Information . . . . .	36
3.2.4	Multi-step Supervised Linear Dimension Reduction . . . . .	38
3.3	Derivative of Quadratic Mutual Information . . . . .	39
3.3.1	Single-step Estimation of the Derivative of Quadratic Mutual Information . . . . .	39
3.3.2	Existing Approaches to Estimate the Derivative of the Density Difference . . . . .	40
3.3.3	Direct Estimation of the Derivative of the Density Difference .	41
3.3.4	Basis Function Design . . . . .	43
3.3.5	Model Selection by Cross-Validation . . . . .	43
3.4	Supervised Linear Dimension Reduction via Derivative Estimator . .	44
3.4.1	Gradient Ascent . . . . .	44
3.4.2	Quadratic Mutual Information Approximation via Derivative Estimator . . . . .	45
3.4.3	Fixed-Point Iteration . . . . .	46
3.5	Experiment . . . . .	47
3.5.1	Illustrative Experiment . . . . .	47
3.5.2	Artificial Data . . . . .	50
3.5.3	Benchmark Data . . . . .	54
3.6	Further Extension: Estimation of Higher Order Derivatives of Quadratic Mutual Information . . . . .	60
3.7	Conclusion . . . . .	65
<b>4</b>	<b>Single-step Dimension Reduction for Conditional Density Estimation</b>	<b>66</b>
4.1	Introduction . . . . .	66
4.2	Conditional Density Estimation . . . . .	67
4.2.1	Problem Formulation . . . . .	68
4.2.2	Existing Methods . . . . .	68
4.2.3	Multi-step Dimension Reduction for Conditional Density Es- timation . . . . .	70
4.3	Least-Squares Conditional Entropy . . . . .	70
4.3.1	Squared-loss Conditional Entropy and conditional Indepen- dence . . . . .	71
4.3.2	Estimating Squared-loss Conditional Entropy . . . . .	73
4.3.3	Supervised Linear Dimension Reduction with Squared-loss Conditional Entropy . . . . .	74
4.3.4	Conditional Density Estimation with Squared-loss Condi- tional Entropy . . . . .	76
4.3.5	Model Selection by Cross-Validation . . . . .	76
4.3.6	Basis Function Design . . . . .	77
4.4	Experiment . . . . .	78
4.4.1	Illustration . . . . .	78
4.4.2	Artificial Data . . . . .	79
4.4.3	Benchmark Data . . . . .	80
4.4.4	Humanoid Robot . . . . .	85
4.4.5	Computer Art . . . . .	86
4.5	Conclusion . . . . .	86

<b>5</b>	<b>Background of Reinforcement Learning</b>	<b>88</b>
5.1	Markov Decision Processes . . . . .	88
5.2	Policy Iteration . . . . .	91
5.2.1	State Value Function and State-Action Value Function . . . . .	91
5.2.2	Policy Iteration Framework . . . . .	92
5.2.3	Q-Learning . . . . .	94
5.2.4	Least-Squares Policy Iteration . . . . .	95
5.2.5	Summary of Policy Iteration . . . . .	97
5.3	Direct Policy Search . . . . .	97
5.3.1	Policy Gradient . . . . .	97
5.3.2	Policy Gradient with Parameter-based Exploration . . . . .	100
5.3.3	Expectation-Maximization . . . . .	102
5.3.4	Information-Theoretic Approach . . . . .	104
5.3.5	Summary of Direct Policy Search . . . . .	107
5.4	Model-based Reinforcement Learning . . . . .	108
5.4.1	Learning the Environment Model . . . . .	108
5.4.2	Locally Weighted Linear Regression . . . . .	109
5.4.3	Gaussian Process Regression . . . . .	110
5.4.4	Summary of Model-based Reinforcement Learning . . . . .	111
<b>6</b>	<b>Model-based Policy Gradient with Parameter-based Exploration</b>	<b>113</b>
6.1	Introduction . . . . .	113
6.2	Transition Model Estimation via Least-Squares Conditional Density Estimation . . . . .	115
6.3	Policy Learning Framework . . . . .	116
6.4	Experiment . . . . .	117
6.4.1	Continuous Chainwalk . . . . .	117
6.4.2	Humanoid Robot Control . . . . .	123
6.5	Conclusion . . . . .	128
<b>7</b>	<b>Dimension Reduction for Model-based Policy Gradient with Parameter-based Exploration</b>	<b>130</b>
7.1	Introduction . . . . .	130
7.2	Existing Approach to Dimension Reduction in Reinforcement Learning	131
7.2.1	Feature Selection in Factored Markov Decision Process . . . . .	131
7.2.2	Supervised Dimension Reduction in Reinforcement Learning . . . . .	132
7.3	Model-based PGPE with Single-step Dimension Reduction . . . . .	132
7.3.1	Learning Framework . . . . .	132
7.3.2	Imitation Learning . . . . .	133
7.4	Experiment . . . . .	136
7.4.1	Continuous Chainwalk . . . . .	136
7.4.2	Cartpole Balancing . . . . .	142
7.4.3	Humanoid Robot Control . . . . .	146
7.5	Conclusion . . . . .	149
<b>8</b>	<b>Contextual Policy Search with Single-step Dimension Reduction</b>	<b>153</b>
8.1	Introduction . . . . .	153
8.2	Contextual Policy Search . . . . .	154
8.2.1	Problem Formulation . . . . .	154

8.2.2	Related Work . . . . .	157
8.3	Contextual Model-based Relative Entropy Stochastic Search . . . . .	157
8.3.1	Learning the Search Distribution . . . . .	158
8.3.2	Dual Function Evaluation via the Quadratic Model . . . . .	160
8.3.3	Learning the Quadratic Model . . . . .	162
8.3.4	Dimension Reduction and Low-Rank Representation . . . . .	162
8.3.5	Learning a Low-Rank Matrix with Nuclear Norm Regularization . . . . .	163
8.4	Experiment . . . . .	166
8.4.1	Quadratic Cost Function Optimization . . . . .	166
8.4.2	Ball Hitting with a 2-DoF Robot Arm . . . . .	167
8.4.3	Ball Hitting with a 6-DoF Robot Arm . . . . .	169
8.5	Conclusion . . . . .	170
<b>9</b>	<b>Conclusion and Future Work</b>	<b>171</b>
9.1	Conclusion . . . . .	171
9.2	Future Work . . . . .	172

# Chapter 1

## Introduction

Machine learning has recently become a popular topic due to its ability to analyze complex data. However, existing machine learning methods still have some limitations especially when data has high dimensionality. In this dissertation, we develop machine learning methods that overcome these limitations. In particular, we focus on developing and utilizing dimension reduction to overcome the high-dimensional data limitation.

In this chapter, we firstly describe fundamental concepts of machine learning and issues of learning from high-dimensional data. Then, we give an overview of our contributions and the organization of this dissertation.

### 1.1 Machine Learning

Informally, the goal of machine learning is to construct machines that automatically learn to solve problems from data. Machine learning can be regarded as an intersection between computer science and statistics (Mitchell, 2006). While computer science designs methods that solve problems, machine learning designs methods that automatically learn to solve problems from data. On the other hand, statistics and machine learning are more closely related as they both emphasize on inferring conclusions from data. However, machine learning focuses more on further utilizing these inferred conclusions to solve problems.

More formally, machine learning is concerned with problems of learning some target function from data such that the function satisfies a certain performance criterion. Machine learning can be categorized into *supervised learning*, *unsupervised learning*, and *reinforcement learning* depending on the types of target function, data and performance criteria. We briefly introduce these three learning paradigms below.

- **Supervised learning**

Supervised learning is a learning paradigm that learns from an *input-output data*. This input-output data is represented as a set of paired data points:

$$\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}, \quad (1.1)$$

where  $\mathbf{x}$  is input,  $\mathbf{y}$  is output, and  $(\mathbf{x}_n, \mathbf{y}_n)$  denotes the  $n$ -th data point. The goal of supervised learning is to learn a function representing a relationship from input to output. For example, *regression* learns a function that maps from input to real-valued output, while *classification* learns a function that maps from input to categorical output. Supervised learning is considered the most mature

paradigm of machine learning with many successful applications including *text classification* (Joachims, 1998), *face detection* (Viola and Jones, 2004), *spam filtering* (Zhang et al., 2004; Tretyakov, 2004), and *ranking in information retrieval* (Liu, 2009). Supervised learning is illustrated in Figure 1.1(a).

- **Unsupervised learning**

Unsupervised learning is a learning paradigm that learns from *input-only data* and is opposite to supervised learning. This input-only data is represented as a set of data points:

$$\{\mathbf{x}_1, \dots, \mathbf{x}_N\}. \quad (1.2)$$

The goal of unsupervised learning is to learn interesting structures in data. Since the *true* interesting structures in data is often unknown, the target function in unsupervised learning depends mostly on what structure is considered interesting to users. For example, *clustering* learns a function that assigns a cluster label to each data point, while *unsupervised dimension reduction* learns a function that reduce dimensionality of data. Unsupervised learning are used in applications such as *image segmentation* (Friedman and Russell, 1997) and *fraud analysis* (Hilas and Mastorocostas, 2008). Unsupervised learning is illustrated in Figure 1.1(b).

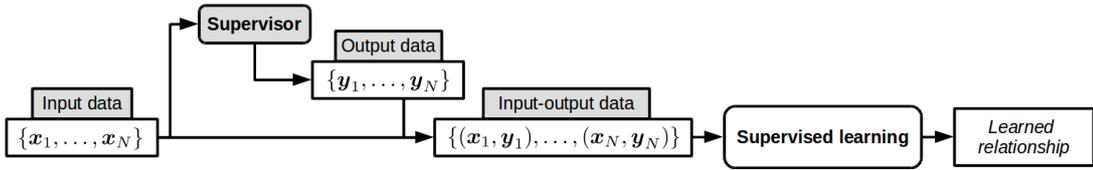
- **Reinforcement learning**

Reinforcement learning considers the sequential decision making problem where at each time step an agent observes an environment’s *state* and chooses an *action* to observe a *reward* and a *next state*. The goal of this problem is to find an optimal *sequence of actions* such that the cumulative reward is maximized. Reinforcement learning solves this problem by learning an optimal *policy function* which maps a state to an action such that the cumulative reward is maximized. This policy is typically learned from data represented as a sequence of a state, an action, a reward, and a next state:

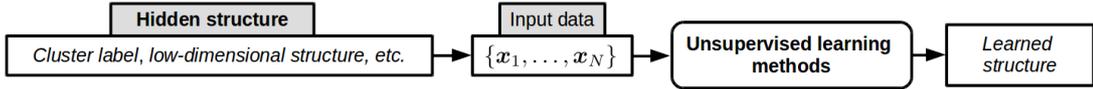
$$(s_1, \mathbf{a}_1, r_1, s_2, \mathbf{a}_2, \dots, \mathbf{a}_T, r_T, s_{T+1}), \quad (1.3)$$

where  $s$  denotes a state,  $\mathbf{a}$  denotes an action,  $r$  denotes a reward, and subscript denotes the time step. Learning from this data is different from learning in supervised learning because this data does not tell us what are optimal actions. Moreover, an action does not only determine the immediate reward, it also determines what will be the next state which subsequently affects the next optimal action. Reinforcement learning has been applied to many problems that involve sequential decision making. These applications include but not limited to *robotics* (Mahadevan and Connell, 1992; Schaal and Atkeson, 1994; Ng. and Jordan, 2000; Kober et al., 2013a), *games* (Tesauro, 1995; Mnih et al., 2015; Silver et al., 2016), *job scheduling* (Zhang and Dietterich, 1995), and *tax collection* (Abe et al., 2010). Reinforcement learning is illustrated in Figure 1.1(c).

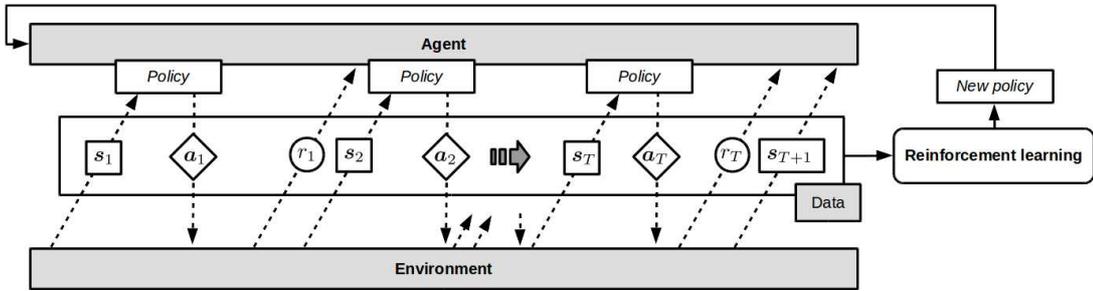
In machine learning, a performance criterion is often evaluated based on unobserved data that is not used for learning. This means that well performed machine learning methods need to *generalize* their learning to unobserved data. Methods that perform well on observed data but perform poorly on unobserved data are said to have



(a) Supervised learning: An unknown supervisor assigns corresponding output  $y_i$  to each input data point  $x_i$ . Then, supervised learning methods use the input-output data to learn a relationship from input to output that is intended by the supervisor.



(b) Unsupervised learning: Input data points are generated with regard to some hidden structures. Then, unsupervised learning methods use the input-only data to learn a hidden structure of interest.



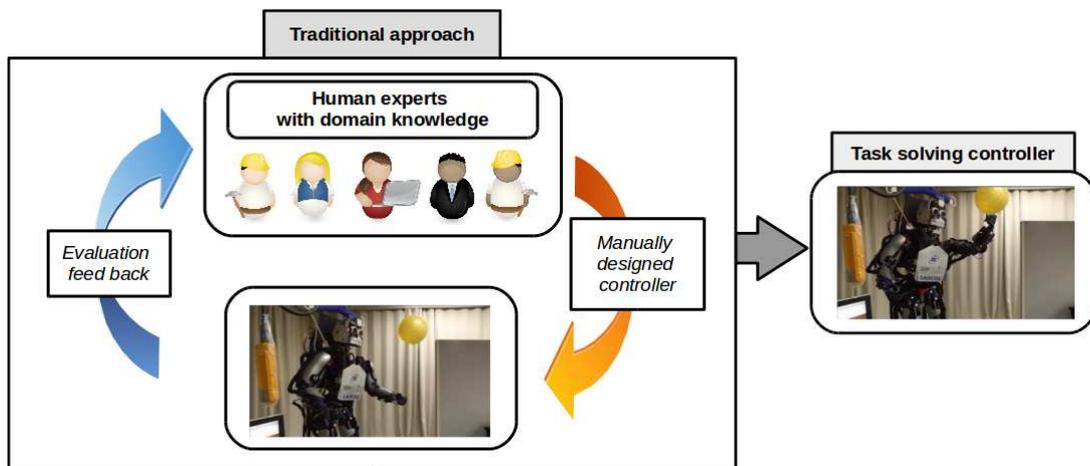
(c) Reinforcement learning: Data is collected through sequential interaction between an agent and an environment. The agent uses its policy to choose actions  $a_t$  depends on observed states  $s_t$ . The environment response with immediate rewards  $r_t$  and next states  $s_{t+1}$ . The sequence of a state, an action, a reward, and a next state is the data used by reinforcement learning to learn a new policy. The process is repeated until an optimal policy is obtained.

Figure 1.1: Illustrations of typical processes of the three paradigms of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

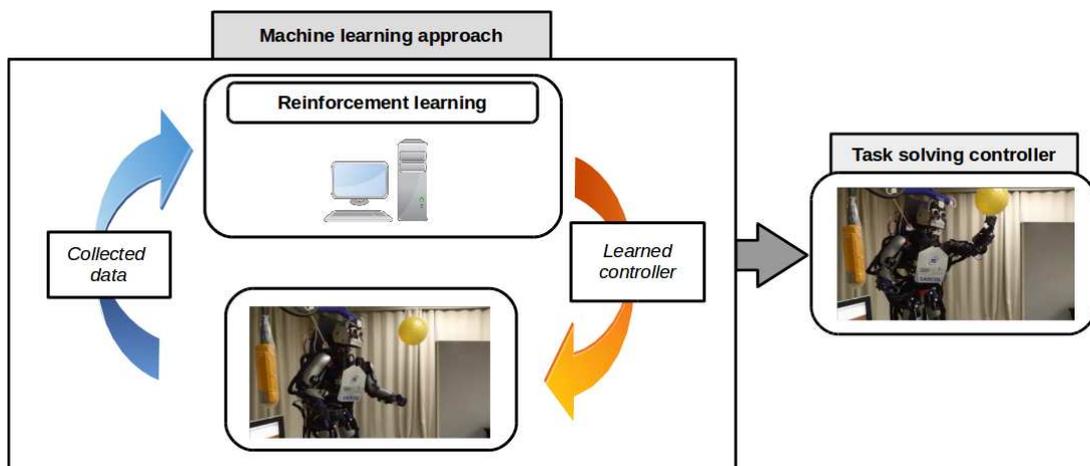
an *overfitting* issue. Avoiding overfitting is important when using machine learning methods.

Machine learning research has a long history<sup>1</sup>. However, it is only until recently that machine learning has gained a noticeable popularity and become an important tool in both academic and industrial. We argue that this is because problems that we want to solve nowadays are very sophisticated and complex to the point where traditional methods cannot solve them well. On the other hand, machine learning offers data-driven methods that allow these problems to be solved relatively well when

<sup>1</sup>The exact origin of machine learning research is debatable since machine learning includes and combines elements from both statistics and computer science. The least squares regression (a supervised learning method) has been used by mathematicians since the 18th century (Stigler, 1981). The concept of a learning machine was later implemented and demonstrated by a checker program (Samuel, 1959), but the learning approach is very different from machine learning nowadays which heavily utilizes statistics and mathematics.



(a) Traditional approach: Human experts design a controller and evaluate it on the robot. Then, the experts adjust the controller based on the evaluation results. This trial and error process is repeated until the experts satisfy with the evaluation results.

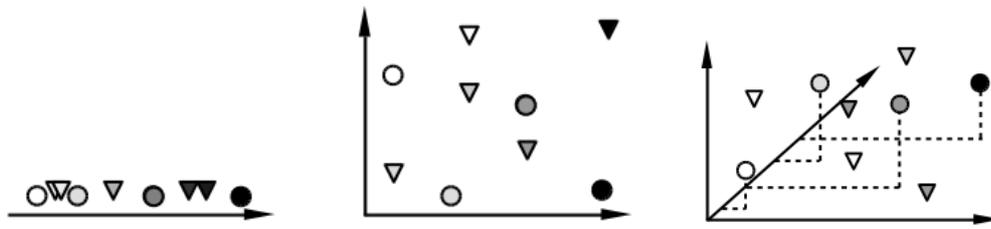


(b) Machine learning approach: A reinforcement learning method collects data using a controller. Then the controller is improved based on the collected data. This learning process is repeated until an optimal controller is learned.

Figure 1.2: Comparison between a traditional approach and a machine learning approach for obtaining a task solving robot controller. The traditional approach is not appropriate without experts with domain knowledge.

compared to traditional non data-driven methods.

As an example, let us consider a problem of controlling a robot to perform a task. A traditional approach to solve this problem is to let human experts manually design a robot controller (see Figure 1.2). However, designing a robot controller requires thorough knowledge about the robot's tasks and dynamics. This knowledge is typically unavailable for complex tasks with high uncertainty, or for robots with many degrees of freedom or with tendon-driven bodies. In contrast, reinforcement learning allows robots to automatically learn to solve the task and it has been showing great successes in doing so (Schaal and Atkeson, 1994; Rombokas et al., 2012; Kormushev et al., 2013; Kober et al., 2013b; Sugimoto et al., 2016).



(a) One-dimensional space    (b) Two-dimensional space    (c) Three-dimensional space

Figure 1.3: Illustration of the curse of dimensionality. The goal is to learn a function that assigns data points with correct shade of color. The circle points represent training data points and the triangle points represent test data points. Figure 1.3(a): In low-dimensional space, training data points are dense and the learned function can generalize well to test data points. Figure 1.3(b) and Figure 1.3(c): In high-dimensional space, training data points are sparsely distributed and there are many empty regions with no training data points. In this case, the learned function poorly generalizes to test data points that lie on these empty regions.

Despite its recent successes, many machine learning methods perform very poorly when data has high dimensionality. This is mainly because analyzing high-dimensional data is a very challenging task, as explained in the next section.

## 1.2 High-dimensional Data Analysis

Like a human learns from experiences, a machine learns from data. Data is a collection of data points where each data point is described by the value of its *features*. Machine learning methods learn target functions by analyzing relationships between data points based on features. Since learning is based on features, it is intuitive to think that we can learn better from data with more features. However, this is only partially true, and in fact many machine learning methods perform very poorly when the number of features is more than necessary. There are many reasons that can explain this behavior. Below, we explain two typical reasons.

Features form a feature space where each data point is represented as a point in this space. In low-dimensional feature space formed by a small number of features, both observed and unobserved data points tend to be densely distributed and located in closed proximities. In this scenario, the learned functions can generalize well to unobserved data since both observed and unobserved data are similar. On the other hand, observed data points in high-dimensional space are sparsely distributed and there are many “empty” regions with no data points. This is problematic since we need to generalize target functions to these empty regions based on observed data points that are located far away. For this reason, many machine learning methods have an overfitting issue when data has high dimensionality. This issue is illustrated in Figure 1.3.

Another explanation for the poor behavior can be given as follows. Many machine learning methods rely on the distance between two data points to learn target functions. If data points are sparsely distributed, the notion of the distance is less informative since all pairs of data points have a large distance. In other words, for

distance-based machine learning methods we may say that sparsely distributed data points do not contain enough information about the underlying problem. For this reason, these machine learning methods often perform poorly even when evaluated on observed data.

While there are many reasons to explain the poor behavior of machine learning methods for high-dimensional data, we commonly describe these reasons together as the *curse of dimensionality*. Originally, this term was used in the domain of dynamic programming to refer to a scenario where the computational complexity of dynamic programming methods increases exponentially as the dimensionality of the state variable increases (Bellman, 1957a, 1961). However, it is used nowadays in many domains including machine learning to refer to scenarios where difficulties of problems sharply increase as the dimensionality of involving variables increases. In machine learning, this “difficulty” is related to both accuracy of the learned function as well as the computational complexity of machine learning methods.

Despite the presence of the curse of dimensionality, it is still of great interest to learn from high-dimensional data since most problems that we wish to solve by machine learning involve high-dimensional data. Among many approaches that were proposed in literature, *linear dimension reduction* and *non-linear dimension reduction and deep learning* are considered the most popular approaches to mitigate the curse of dimensionality.

### 1.2.1 Linear Dimension Reduction

Linear dimension reduction has been considered by many as a standard approach for high-dimensional data analysis (Fodor, 2002; Cunningham and Ghahramani, 2015). Linear dimension reduction aims to find a *linear transformation function* that transforms the original set of features into a smaller set of features such that information in the original set of features is preserved.

Linear dimension reduction is commonly separated into *feature extraction* and *feature selection*. The feature extraction approach finds a new set of features where each new feature is a linear combination of the original features. This linear combination is often represented by a *transformation matrix*. On the other hand, the feature selection approach finds the best subset of the original features and uses this subset as the new set of features. The difference between the two approaches is illustrated in Figure 1.4. In many settings, feature selection can be regarded as an instance of feature extraction, i.e., a transformation matrix in feature selection is a submatrix of a permutation matrix. Therefore in this dissertation, unless specified otherwise dimension reduction will be discussed from the standpoint of the feature extraction approach.

Linear dimension reduction is simple to use and has high interpretability. For instance, many linear dimension reduction methods learn an orthogonal transformation matrix. In such a case, these methods can be understood as methods that learn a subspace of the original feature space (Fodor, 2002; Cunningham and Ghahramani, 2015). Example of subspace learning methods are *principal component analysis* (Jolliffe, 1986) and *sliced inverse regression* (Li, 1991). This high interpretability is also the main reason that makes linear dimension reduction popular.

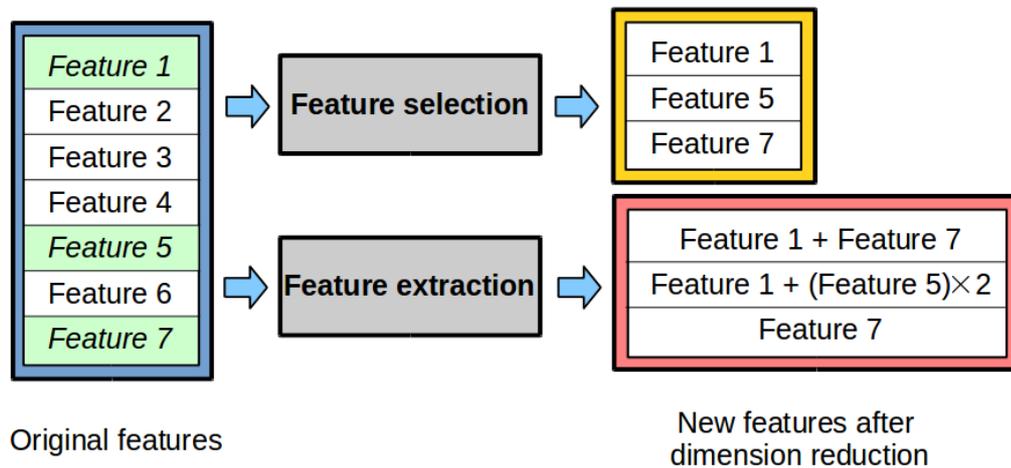


Figure 1.4: A comparison between the feature selection and feature extraction approaches. Assume that features 1, 5, and 7 as well as their linear combinations are informative. Feature selection selects subset of the original features as new features. In contrast, feature extraction computes new features by a linear combination of the original features.

### 1.2.2 Non-linear Dimension Reduction and Deep Learning

Non-linear dimension reduction aims to find a non-linear transformation function that transforms the original set of features into a new set of features. Non-linear dimension reduction methods can be categorized into *non-linearized linear methods* and *manifold learning methods*.

Non-linearized linear methods refers to methods that apply the non-linear transformation to data and then perform linear dimension reduction. Applying non-linear transformation to data is a common technique in machine learning that allows us to use linear learning methods to analyze non-linear relationships in data. An example of non-linearized linear method is *kernel principal component analysis* (Schölkopf et al., 1998) which is regarded as performing principal component analysis (Jolliffe, 1986) on non-linearly transformed data. Non-linearized linear methods will be explained in more details in Chapter 2.

In contrast, manifold learning methods learn a non-linear manifold embedded in the original feature space such that data points lie on this manifold. The key idea of many manifold learning methods is to find a non-linear manifold that preserves distance between data points (Tenenbaum et al., 2000; Roweis and Saul, 2000; Silva and Tenenbaum, 2003; Belkin and Niyogi, 2003; Lee and Verleysen, 2007). These manifold learning methods are highly useful for finding a meaningful visualization of high-dimensional data.

A popular learning approach that is closely related to non-linear dimension reduction is the *deep learning* approach. Deep learning refers to learning a function modelled by a deep neural network (Bengio, 2009) (see Figure 1.5). Deep neural network is a computational model aims to simulate computation in human brains. In a deep neural network, each computation unit called a *neuron* takes some input values from other neurons and outputs some value to other neurons. These neurons are grouped together into *layers*. Computation in the network is done in a layer-by-layer

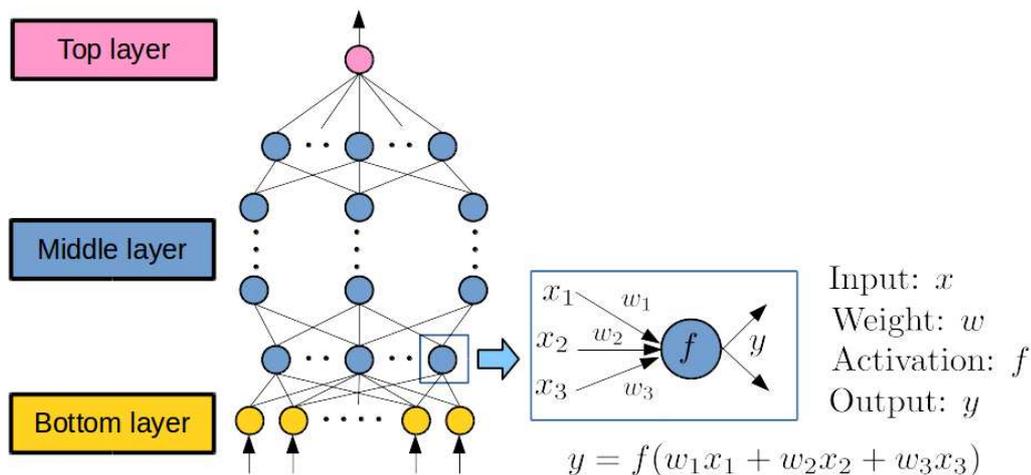


Figure 1.5: An example of a deep neural network. Computation is done from the bottom layer to the top layer. Each neuron computes its output using its weight  $w$  and its activation function  $f$ . In this example, the output  $y$  is computed by applying the function  $f$  to a linear combination of the weight  $x$  and the input  $w$ .

basis from the bottom layer to the top layer where outputs from neurons in a lower layer become inputs of neurons in an upper layer. The computation in each neuron is determined by its *activation function* and *weights* of its input connections. In general, the number of neurons, the number of layers, and activation functions are fixed, while the weights are learned such that the network accurately represents the target function.

The computation in each neuron can be regarded as passing a combination of weighted inputs through an activation function to construct new inputs for an upper layer. By considering inputs as features, deep learning is regarded as progressively and automatically learning more informative features such that features at the top layer are the most informative for computing function output. This progressive feature learning closely resembles non-linear dimension reduction which learns a manifold embedding of feature space (Bengio et al., 2013). This is the key characteristic that makes deep learning popular in applications involving high-dimensional data such as image data (Hinton et al., 2006; Krizhevsky et al., 2012), natural language data (Collobert and Weston, 2008), and speech data (Yu et al., 2012).

Transformation function in non-linear dimension reduction including deep learning is much more complex than a transformation matrix in linear dimension reduction. For this reason, non-linear dimension reduction is often considered more powerful than linear dimension reduction. However, non-linear dimension reduction has two limitations. Firstly, analyzing the behavior of non-linear transformation function is not a trivial task due to their high complexity. For example, a deep neural network for image classification problems may contain up to half a million neurons and more than millions of weights (Krizhevsky et al., 2012). Analyzing this network to explain how it behaves is near impossible without some anticipated behaviors in mind. This “black box” behavior makes deep learning sometimes considered to be untrustworthy in applications such as medical analysis where explanations of results are equally important as accuracies of results (Castelvecchi, 2016). In contrast, as mentioned earlier, most linear dimension reduction methods can be understood much more easily as

learning a subspace of the feature space (Fodor, 2002; Cunningham and Ghahramani, 2015).

Secondly, due to the fact that the amount of data required for learning a model of functions is proportion to the complexity of the model (Vapnik, 1998; Hastie et al., 2001), highly complex non-linear transformation functions such as deep neural networks require an extremely large amount of data to be learned. In domains such as image classification and information retrieval, data can be collected without much effort and we have seen many successful applications of non-linear dimension reduction in these domains. However, in other domains such as robotics, data is a limited resource and collecting data requires effort and careful considerations. For this reason, non-linear dimension reduction may not be suitable for such domains. Nonetheless, with enough data it also works well in these domains (Watter et al., 2015a; Lillicrap et al., 2015). Other domains in which data is a limited resource are domain that involves rare events such as medical diagnostics (Au et al., 2010) and hardware fault prediction (Murray et al., 2005).

### 1.3 Single-step Approach to Linear Dimension Reduction

As discussed in the previous section, linear dimension reduction is a promising approach for mitigating the curse of dimensionality. Furthermore, linear dimension reduction is a popular topic and there are many linear dimension reduction methods in literature.

However, naively using existing linear dimension reduction methods often involves a *multi-step approach*. In a multi-step approach, one learns some intermediate functions and then uses these intermediate functions to learn another function. These intermediate functions are usually learned based on performance criteria that are different from the performance criterion of the latter function. This is not preferable since good performances of these intermediate functions do not always imply a good performance of the latter function. This situation is sometimes described via the *Vapnik's Principle* (Vapnik, 1998):

“When solving a problem of interest, one should not solve a more general problem as an intermediate step.”

— Vladimir N. Vapnik, *Statistical Learning Theory*, 1998.

An alternative to a multi-step approach is to learn the latter function directly. We refer to this approach as a *single-step approach*. A single-step approach is often more preferable since the function is learned directly using an appropriate performance criterion. However, it should be noted that a single-step approach is not always the best approach in practice since some problems can be solved more conveniently by learning intermediate functions. An example of such a case is the *model-based reinforcement learning* approach which learns a model of the environment as an intermediate function. Nonetheless, in this dissertation we claim and show that for linear dimension reduction a single-step approach is more preferable than a multi-step approach.

### 1.4 Contributions

This dissertation contributes to developing and utilizing single-step dimension reduction to solve high-dimensional machine learning problems especially reinforcement

learning problems. Our contributions can be separated into two parts: *development of single-step dimension reduction methods* and *applications of single-step dimension reduction to reinforcement learning*.

Note that even though our contributions are originally linear dimension reduction methods, they can be straightforwardly extended to non-linear dimension reduction methods as well by using the non-linearization approach.

#### 1.4.1 Development of Single-step Dimension Reduction Methods

Although many linear dimension reduction methods were proposed, existing methods still have issues. The first issue occurs when they are applied to data contaminated by outliers, and the second issue occurs when they are applied for the conditional density estimation problem. In this first part, our contributions are two linear dimension reduction methods that separately overcome these issues.

Data from real-world problems may possess some undesired property. One of these properties is the contamination by *outliers*. Outliers are erroneous and unreliable data points that are significantly different from other data points. Outliers are unreliable, do not contain correct information about the problem, and therefore should not be used for learning. However, existing linear dimension reduction methods are sensitive to outliers which makes them performing poorly in the presence of outliers. This issue can be overcome by maximizing the *quadratic mutual information* (QMI) (Principe et al., 2000) which is a statistical dependence measure that is robust against outliers. Maximizing QMI requires an estimate of the derivative of QMI. However, existing approaches estimate these derivatives using a multi-step approach which firstly estimates QMI from data and then computes derivatives of the estimated QMI. This approach is not appropriate since an accurately estimated QMI does not always imply that its derivatives are accurately estimated derivatives of QMI. To solve this problem, we propose a single-step method that directly estimates the derivatives of QMI from data, and then we use this method for QMI-based linear dimension reduction. Experimental evaluations on regression problems show that our single-step method is more robust against outliers and performs better than compared methods.

In our second contribution, we focus on a supervised learning problem called *conditional density estimation*. The goal of conditional density estimation is to learn a probability density of output given input. Conditional density estimation is a challenging problem especially when input has high dimensionality. However, naively performing linear dimension reduction *before* conditional density estimation results in a multi-step procedure where an error from the former dimension reduction step may be magnified by the latter conditional density estimation step. To solve this problem, we propose to use the squared-loss conditional entropy and develop a method called *least-squares conditional entropy* (LSCE). Unlike existing methods, LSCE is a single-step method that simultaneously performs non-parametric conditional density estimation and linear dimension reduction in an integrated manner. Experimental evaluations on high-dimensional data including data from a simulated humanoid robot shows that LSCE performs better than methods based on the multi-step approach.

Note that a part of the second contribution was included in the Master's thesis submitted to the Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology, for graduation in March 2014. The contribution in this dissertation contains substantially improved contents that are published in the *Neural Computation* journal in January 2015.

## 1.4.2 Applications of Single-step Dimension Reduction in Reinforcement Learning

In this second part, we focus our attention on utilizing single-step linear dimension reduction for solving high-dimensional reinforcement learning problems. However, existing reinforcement learning methods still have an issue even for low-dimensional problems. Our third contribution in this dissertation is a model-based reinforcement learning method that overcomes this issue. However, the practicability of this method is still limited to low-dimensional problems. This limitation is overcome in our fourth contributions by using LSCE to perform dimension reduction. As our fifth contribution, we consider the contextual reinforcement learning setting and propose a method that utilizes single-step dimension reduction to effectively learn from high-dimensional contexts.

Reinforcement learning learns an optimal policy by iteratively updating the current policy based on data. In the standard setting, data is collected at each iteration by an agent. However, when the amount of available data is limited, we need to decide before learning how many data points the agent needs to collect at each iteration. Determining this *sampling schedule* is a difficult task since it should be determined based on the final performances, but the final performances can only be obtained after learning. This sampling schedule issue can be overcome by using the *model-based* approach which learns a model of the environment using the available data and then uses the model to generate artificial data for policy update. In this way, the agent no longer needs to determine the sampling schedule and it can generate a huge amount of artificial data for accurate policy update. However, existing model learning methods rely on strong assumptions about the environment and they may perform poorly when these assumptions are violated. Our idea to overcome this weakness is to learn the model by *least-squares conditional density estimation* (LSCDE) (Sugiyama et al., 2010). LSCDE is a non-parametric conditional density estimation method and can asymptotically learn any conditional density function without requiring strong assumptions. We combine LSCDE with a reinforcement learning method and propose the *model-based policy gradient with parameter-based exploration* (M-PGPE) method. Experimental evaluations on benchmark problems and a simulated humanoid robot control problem shows that M-PGPE gives better performance than existing methods when the amount of available data is limited.

Although LSCDE can asymptotically learn any model without any strong assumptions, it still requires a prohibitively large amount of data when the environment involves high-dimensional states and actions. To overcome this limitation, we propose to use our LSCE method to learn a model. As mentioned above, LSCE is a single-step method that simultaneously performs non-parametric conditional density estimation and dimension reduction. Experimental evaluations on benchmark problems and real humanoid robot control problems show that our proposed M-PGPE with LSCE gives the best performance among compared methods.

Our fifth contribution is a method for high-dimensional contextual reinforcement learning problems. Unlike standard reinforcement learning, contextual reinforcement learning assumes that the environment is determined by observable variables called *contexts* and the goal is to find optimal policies that yield the maximum cumulative rewards for different contexts. This is very challenging especially for high-dimensional contexts such as camera images. We firstly develop a contextual reinforcement learning method where the key procedure is to learn a locally quadratic model of the cumu-

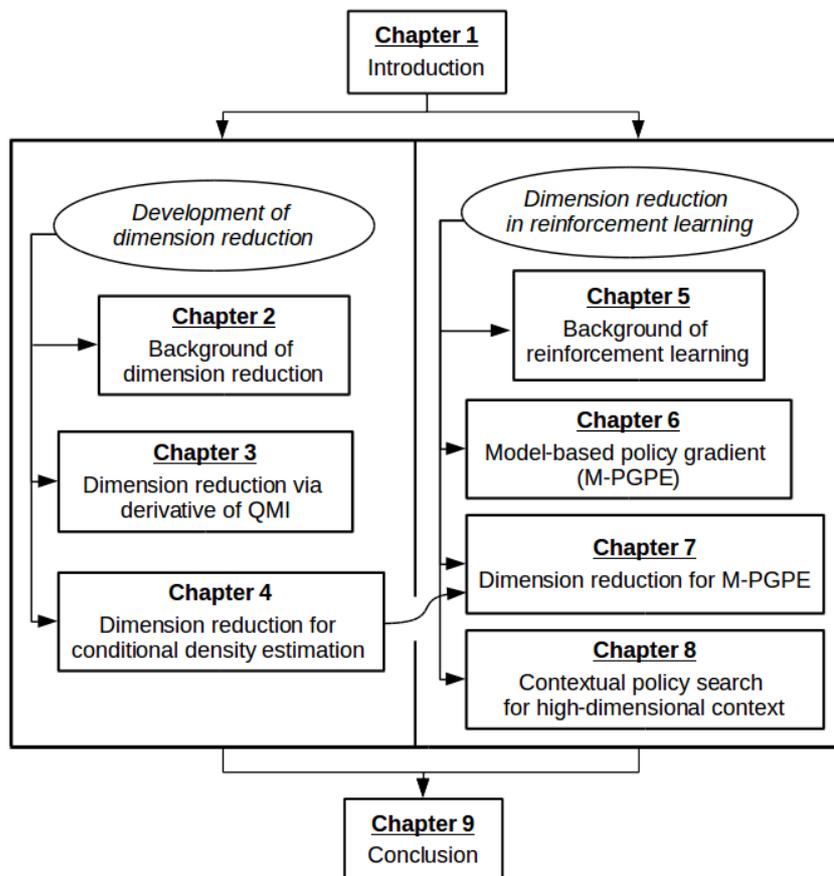


Figure 1.6: The dissertation is organized into nine chapters.

lative rewards. Learning this quadratic model is a high-dimensional regression problem. Instead of naively performing dimension reduction before learning the model, we propose to learn a low-rank representation of the model. We show that learning the low-rank representation can be regarded as a single-step approach that simultaneously performs dimension reduction and regression. We evaluate the proposed method on a benchmark problem and robot ball hitting problems based on camera images. Experimental results show that the proposed method with low-rank representation performs better than methods that explicitly perform dimension reduction before model learning.

## 1.5 Organization

This dissertation consists of nine chapters, as shown in Figure 1.6. In this chapter, we have given the introduction and overview of this dissertation. The remaining chapters are organized as follows.

Chapters 2, 3, and 4 are concerned with the first part; *development of single-step dimension reduction methods*. Chapter 2 covers the background of linear dimension reduction. We cover the supervised linear dimension reduction problem and the unsupervised linear dimension reduction problem in two separated sections. In both sections, we mathematically formulate the problems first and then discuss existing methods and their weaknesses. Then we cover a linear dimension reduction approach based on dependence maximization which is applicable to both supervised and unsu-

pervised scenarios.

Chapter 3 covers our first contribution to the single-step dimension reduction based on derivatives of quadratic mutual information (QMI). We firstly discuss properties of QMI and how to use QMI for dimension reduction. Next, we present our single-step method called *least-squares quadratic mutual information derivative* (LSQMID) which directly learns the derivative of QMI from data. At the same time, we also present the fix-point iteration approach that efficiently uses LSQMID for dimension reduction. We then present our experimental evaluations and close the chapter with conclusion.

Chapter 4 covers our second contribution to the single-step method for conditional density estimation and dimension reduction. We firstly discuss the conditional density estimation problem and an issue of naively using dimension reduction for the problem. We then present the *squared-loss conditional entropy* (SCE) and our approach called *least-squares conditional entropy* (LSCE) that uses SCE for conditional density estimation and dimension reduction. Then, experimental evaluations and conclusion are presented.

Chapters 5, 6, 7, and 8 are concerned with the second part; *applications of single-step dimension reduction in reinforcement learning*. Chapter 5 covers the background of reinforcement learning. We mathematically formulate the reinforcement learning problem and describe methods to solve the problem. We categorize these methods based on two categorizations. They are either categorized into *value iteration* and *direct policy search* depending on how they learn an optimal policy, or categorized into *model-free* and *model-based* depending on whether they learn a model of the environment or not.

Chapter 6 covers our third contribution to the model-based policy gradient with parameter-based exploration (M-PGPE) method. We focus our presentation on the transition model estimation problem which is the key step of our method and review *least-squares conditional density estimation*. Then, we present the learning framework of our M-PGPE method that combines a direct policy search method with LSCDE. Lastly, we demonstrate the performance of M-PGPE through experiments on a benchmark problem and a simulated humanoid robot control problem and conclude the chapter.

Chapter 7 covers our fourth contribution on improving the performance of M-PGPE in high-dimensional problems by using LSCE. We start by discussing existing approaches to performing dimension reduction in reinforcement learning. Then we present an improved learning framework that combines M-PGPE, LSCE, and imitation learning together. Finally, we demonstrate its usefulness through experiments on a benchmark problem and a real humanoid robot control problem and conclude the chapter.

Chapter 8 covers our fifth contribution to contextual reinforcement learning for high-dimensional contexts. We firstly motivate and formulate the contextual reinforcement learning problem. Then, we present our method which consists of the policy learning part and the model learning part. Next, we present our experimental evaluations on high-dimensional contextual reinforcement learning problems including robot ball hitting problems based on camera images, and lastly we conclude the chapter.

Chapter 9 is the final chapter where we conclude the dissertation and discuss future directions along the line of our research.

## Chapter 2

# Background of Linear Dimension Reduction

This chapter gives a background of linear dimension reduction. After we introduce linear dimension reduction and its problem formulation, we review some of existing supervised and unsupervised linear dimension reduction methods. Next, we introduce a linear dimension reduction framework based on statistical dependence that is applicable to both supervised and unsupervised settings. Then we discuss related work on non-linear dimension reduction via non-linear transformation of data. Lastly, we close this chapter by briefly introducing our two contributions on single-step dimension reduction methods.

**Notations:** We use the following mathematical notations. A scalar-valued quantity is denoted by a normal letter, e.g.,  $x$ . A vector-valued quantity is denoted by a bold letter, e.g.,  $\mathbf{x}$ , and its  $i$ -th entry is denoted by a normal letter with superscript, e.g.,  $x^{(i)}$ . All vectors are column vectors unless specified otherwise. A matrix-valued quantity is denoted by a bold capital letter, e.g.,  $\mathbf{X}$ , and its  $(i, j)$ -th entry is denoted by a capital letter with subscript, e.g.,  $X_{(i,j)}$ . The transpose of a matrix  $\mathbf{X}$  is denoted by  $\mathbf{X}^\top$ . The trace of a matrix  $\mathbf{X}$  is denoted by  $\text{tr}[\mathbf{X}]$ . A set of real numbers is denoted by  $\mathbb{R}$ . The *Euclidean norm* or  $\ell_2$  norm of a  $d$ -dimensional vector is denoted by  $\|\cdot\|_2$  and is defined by

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d (x^{(i)})^2}. \quad (2.1)$$

For brevity, we may sometimes omit the subscript and denote the Euclidean norm by  $\|\cdot\|$ . The *Frobenius norm* of a  $d_1$ -by- $d_2$  matrix is denoted by  $\|\cdot\|_{\text{Fro}}$  and is defined by

$$\begin{aligned} \|\mathbf{X}\|_{\text{Fro}} &= \sqrt{\text{tr}[\mathbf{X}^\top \mathbf{X}]} \\ &= \sqrt{\sum_{i=1}^{d_1} \sum_{j=1}^{d_2} (X_{(i,j)})^2}. \end{aligned} \quad (2.2)$$

The *inner product* of two vectors in Euclidean space is denoted by  $\langle \cdot, \cdot \rangle$  and is defined by

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \mathbf{x}_1^\top \mathbf{x}_2. \quad (2.3)$$

We assume that random variables have the following domains:  $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ ,  $\mathbf{y} \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$ , and  $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^{d_z}$ . We also assume that input only data points

are random variables drawn *independently and identically distributed* (i.i.d.) from a probability distribution with density function  $p(\mathbf{x})$ :

$$\{\mathbf{x}_1, \dots, \mathbf{x}_N\} = \{\mathbf{x}_i\}_{i=1}^N \stackrel{\text{i.i.d.}}{\sim} p(\mathbf{x}), \quad (2.4)$$

and input-output data points are random variables drawn i.i.d. from a joint probability distribution with density function  $p(\mathbf{x}, \mathbf{y})$ :

$$\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N \stackrel{\text{i.i.d.}}{\sim} p(\mathbf{x}, \mathbf{y}). \quad (2.5)$$

The *expectation* of  $f(\mathbf{x})$  over a density  $p(\mathbf{x})$  is defined and denoted as

$$\mathbb{E}_{p(\mathbf{x})} [f(\mathbf{x})] = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x}. \quad (2.6)$$

The domain of the integration is assumed to be the whole support of the variable unless specified otherwise. This expectation can be approximated using the i.i.d. data points  $\{\mathbf{x}_i\}_{i=1}^N$  and this approximation is given by

$$\mathbb{E}_{p(\mathbf{x})} [f(\mathbf{x})] \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i). \quad (2.7)$$

The *conditional expectation* of  $f(\mathbf{y})$  over a conditional density  $p(\mathbf{y}|\mathbf{x})$  is defined and denoted as

$$\mathbb{E}_{p(\mathbf{y}|\mathbf{x})} [f(\mathbf{y})] = \int f(\mathbf{y})p(\mathbf{y}|\mathbf{x})d\mathbf{y}. \quad (2.8)$$

When there is no ambiguity, we may omit the subscript and only use the symbol  $\mathbb{E}[\cdot]$ . We also use a matrix  $\mathbf{X} \in \mathbb{R}^{d_x \times N}$  to represent data points  $\{\mathbf{x}_i\}_{i=1}^N$ :

$$\mathbf{X} = [\mathbf{x}_1 \mid \mathbf{x}_2 \mid \dots \mid \mathbf{x}_N] = \begin{bmatrix} x_1^{(1)} & \dots & x_N^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(d_x)} & \dots & x_N^{(d_x)} \end{bmatrix}, \quad (2.9)$$

where the rows correspond to features and columns correspond to data points. We assume that data is centered so that it has zero empirical mean. Under this assumption, the data covariance matrix is estimated from data by

$$\text{cov}(\mathbf{X}) = \frac{1}{N} \mathbf{X} \mathbf{X}^\top. \quad (2.10)$$

## 2.1 Linear Dimension Reduction

Linear dimension reduction transforms the original set of features into a set of low-dimensional features by a *transformation matrix* denoted by  $\mathbf{W}$ . More specifically, given a transformation matrix  $\mathbf{W} \in \mathbb{R}^{d_z \times d_x}$  and a data point  $\mathbf{x}$  with  $d_x$  features,

$$\mathbf{x} = [x^{(1)}, \dots, x^{(d_x)}]^\top, \quad (2.11)$$

a new data point  $\mathbf{z}$  with  $d_z (< d_x)$  features is obtained by

$$\mathbf{z} = \mathbf{W} \mathbf{x}, \quad (2.12)$$

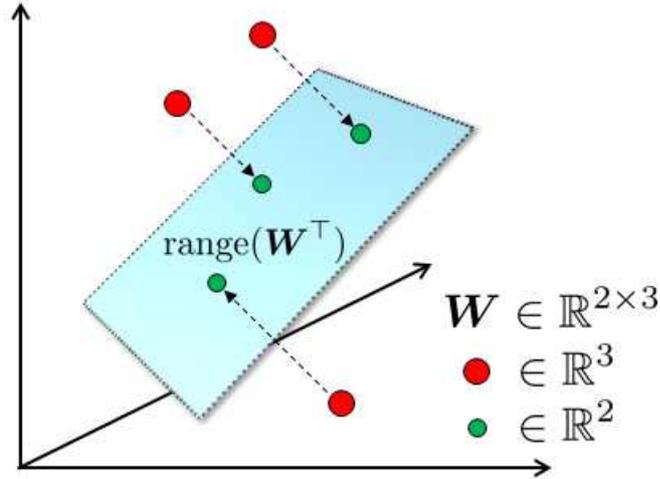


Figure 2.1: Linear dimension reduction by a transformation matrix  $\mathbf{W} \in \mathbb{R}^{2 \times 3}$ . The original data points  $\mathbf{x}$  (red points) in three dimensional feature space is orthogonally projected onto a two-dimensional subspace spanned by rows of  $\mathbf{W}$ . The new data points  $\mathbf{z}$  (green points) now lie in a two-dimensional feature space.

where each new feature is given by

$$\begin{aligned} z^{(i)} &= \vec{w}_i^\top \mathbf{x} \\ &= \sum_{j=1}^{d_x} W_{(i,j)} x^{(j)}, \end{aligned} \quad (2.13)$$

where  $\vec{w}_i$  denotes the  $i$ -th row of  $\mathbf{W}$ . Linear dimension reduction is illustrated in Figure 2.1.

The goal of linear dimension reduction is to learn  $\mathbf{W}$  from data such that the new features preserve *information* contained in the original features. This problem can be mathematically formulated as an optimization problem,

$$\min_{\mathbf{W} \in \mathcal{W}} \mathcal{J}(\mathbf{W}), \quad (2.14)$$

where the objective function  $\mathcal{J}(\mathbf{W})$  is a function that corresponds to the amount of information loss after feature transformation by  $\mathbf{W}$ . That is, we want to find  $\mathbf{W}$  which gives the least amount of information loss. The set  $\mathcal{W}$  determines the domain of  $\mathbf{W}$  and is often defined as a subset of  $d_z$ -by- $d_x$  matrices. This optimization problem is the key problem of linear dimension reduction and we can derive many well-known linear dimension reduction methods by changing  $\mathcal{J}(\mathbf{W})$  and  $\mathcal{W}$  accordingly.

Although any subset of  $d_z$ -by- $d_x$  matrices can be used for  $\mathcal{W}$ , it is common to use a set of  $\mathbf{W}$  satisfying  $\mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}$ . That is, the set  $\mathcal{W}$  is defined as

$$\mathcal{W} = \{\mathbf{W} \in \mathbb{R}^{d_z \times d_x} \mid \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}\}, \quad (2.15)$$

where  $\mathbf{I}_{d_z}$  denotes the  $d_z$ -by- $d_z$  identity matrix. This set is also called the *Stiefel manifold* (Stiefel, 1935; Edelman et al., 1998; Absil et al., 2008) and is denoted by  $\mathcal{S}_{d_z}^{d_x}$ . An optimization problem over this set can be written as

$$\min_{\mathbf{W} \in \mathcal{S}_{d_z}^{d_x}} \mathcal{J}(\mathbf{W}), \quad (2.16)$$

or equivalently as a constrained optimization problem

$$\begin{aligned} \min_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \quad & \mathcal{J}(\mathbf{W}) \\ \text{subject to} \quad & \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}. \end{aligned} \quad (2.17)$$

Most of linear dimension reduction methods that are considered in this dissertation solve an optimization problem in Equation (2.17). In literature, such a constrained optimization problem is popular since the linear transformation is equivalent to an orthogonal projection (see Figure 2.1) which is highly useful for data visualization (Cunningham and Ghahramani, 2015).

The objective function  $\mathcal{J}(\mathbf{W})$  depends on the given data and we can generally categorize linear dimension reduction methods into *unsupervised* and *supervised* methods depending on the type of the data. In the following sections, we give an overview of unsupervised and supervised linear dimension reduction in details.

## 2.2 Unsupervised Linear Dimension Reduction

Unsupervised linear dimension reduction refers to linear dimension reduction that learns the matrix  $\mathbf{W}$  from input-only data  $\{\mathbf{x}_i\}_{i=1}^N$  such that the low-dimensional data  $\mathbf{z} = \mathbf{W}\mathbf{x}$  preserves information contained in the original data  $\mathbf{x}$ . We emphasize that this ‘information’ that we wish to preserve is not mathematically defined and different methods define this information in different ways. In the followings, we briefly review two important unsupervised linear dimension reduction methods in the machine learning literature.

### 2.2.1 Principal Component Analysis

The *principal component analysis* (PCA) method (Jolliffe, 1986) is considered as the most well-known unsupervised linear dimension reduction method. PCA is commonly described as a dimension reduction method that projects  $d_x$ -dimensional data onto  $d_z$  orthogonal directions with maximum variances. These orthogonal directions are obtained by computing eigendecomposition of the data covariance matrix and selects  $d_z$  eigenvectors associated with the  $d_z$  largest eigenvalues. Below, we review PCA in another viewpoint where it aims to minimize the reconstruction error.

Firstly, we consider an objective function defined as

$$\begin{aligned} \mathcal{J}(\mathbf{W}) &= \sum_{i=1}^N \|\mathbf{x}_i - \mathbf{W}^\top \mathbf{W} \mathbf{x}_i\|_2^2 \\ &= \|\mathbf{X} - \mathbf{W}^\top \mathbf{W} \mathbf{X}\|_{\text{Fro}}^2. \end{aligned} \quad (2.18)$$

The quantity  $\mathbf{x}_i - \mathbf{W}^\top \mathbf{W} \mathbf{x}_i$  corresponds to the difference between a data point  $\mathbf{x}_i$  and a reconstructed data point  $\mathbf{W}^\top \mathbf{W} \mathbf{x}_i$  obtained by orthogonal projection of  $\mathbf{W}$ . Therefore, solving an optimization problem

$$\begin{aligned} \min_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \quad & \|\mathbf{X} - \mathbf{W}^\top \mathbf{W} \mathbf{X}\|_{\text{Fro}}^2 \\ \text{subject to} \quad & \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}, \end{aligned} \quad (2.19)$$

gives a solution which minimizes the reconstruction error in the least-squares sense. By using the definition of the Frobenius norm, the above objective function can be rewritten into

$$\begin{aligned}\|\mathbf{X} - \mathbf{W}^\top \mathbf{W} \mathbf{X}\|_{\text{Fro}}^2 &= \text{tr} \left[ (\mathbf{X} - \mathbf{W}^\top \mathbf{W} \mathbf{X})^\top (\mathbf{X} - \mathbf{W}^\top \mathbf{W} \mathbf{X}) \right] \\ &= \text{tr} \left[ \mathbf{X}^\top \mathbf{X} - \mathbf{X}^\top \mathbf{W}^\top \mathbf{W} \mathbf{X} \right] \\ &= \text{tr} \left[ \mathbf{X}^\top \mathbf{X} - \mathbf{W} \mathbf{X} \mathbf{X}^\top \mathbf{W}^\top \right].\end{aligned}\quad (2.20)$$

The term  $\mathbf{X}^\top \mathbf{X}$  can be omitted from the objective function since it is a constant with respect to (w.r.t.) the matrix  $\mathbf{W}$ . Thus, the optimization problem is reduced to

$$\begin{aligned}\max_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \quad & \text{tr} \left[ \mathbf{W} \mathbf{X} \mathbf{X}^\top \mathbf{W}^\top \right] \\ \text{subject to} \quad & \mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_z}.\end{aligned}\quad (2.21)$$

It is known that an optimization problem in this form can be solved by computing an eigendecomposition of the matrix  $\mathbf{X} \mathbf{X}^\top$  and choosing  $d_z$  eigenvectors associated with the  $d_z$  largest eigenvalues (Fukunaga, 1990). Given a  $d_x$ -by- $d_x$  square matrix  $\mathbf{X} \mathbf{X}^\top$ , the eigendecomposition seeks for a set of *eigenvectors*  $\{\mathbf{v}_i\}_{i=1}^{d_x}$  and a set of *eigenvalues*  $\{\lambda_i\}_{i=1}^{d_x}$  such that

$$\mathbf{X} \mathbf{X}^\top \mathbf{v}_i = \lambda_i \mathbf{v}_i. \quad (2.22)$$

It should be noted that the eigenvectors are always orthogonal to each other, and thus the orthogonal constraint is always satisfied by using eigenvectors for a solution  $\mathbf{W}$ . We also emphasize that  $\mathbf{W}$  obtained via eigendecomposition is always the *global solution* that yields the optimal objective value subject to the constraint.

While PCA is simple and computationally efficient, it also has many limitations. Among these limitations, one severe limitation is that PCA assumes that the maximum variance directions are the most informative. This is generally not true when data possesses a cluster structure or multi-modal structure. In other words, PCA tries to preserve the global structure of data and ignore local structure such as a cluster structure.

Many extensions of PCA have been proposed to alleviate its limitations (Bishop, 2006; Mollah et al., 2010; Candès et al., 2011). For example, *local PCA* (Mollah et al., 2010) alleviates the cluster structure issue by locally performing PCA on each cluster. However, these extensions do not completely mitigate the limitations of PCA and they also have their own drawbacks, e.g., local PCA requires an appropriate number of cluster. Next, we review an unsupervised linear dimension reduction method that aims to preserve the local structure of data and is considered as an alternative to PCA.

## 2.2.2 Locality Preserving Projection

In contrast to PCA which preserves the global structure of data, *locality preserving projection* (LPP) (He and Niyogi, 2003) preserves the local structure of data. The goal of LPP is to find a matrix  $\mathbf{W}$  which preserves similarity between data points, and this goal can be achieved by minimizing the following objective function:

$$\mathcal{J}(\mathbf{W}) = \sum_{i,j=1}^N S_{(i,j)} \|\mathbf{W} \mathbf{x}_i - \mathbf{W} \mathbf{x}_j\|_2^2, \quad (2.23)$$

where  $0 \leq S_{(i,j)} \leq 1$  denotes a *similarity* between data points  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . The similarity is computed such that  $S_{(i,j)}$  has a high value if  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are similar and has a low value if they are dissimilar. This similarity allows LPP to focus on the local structure in data as follows. If  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are similar, then low-dimensional data points  $\mathbf{W}\mathbf{x}_i$  and  $\mathbf{W}\mathbf{x}_j$  should be similar as well since  $S_{(i,j)}$  has high value. On the other hand, if  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are dissimilar, then  $\mathbf{W}\mathbf{x}_i$  and  $\mathbf{W}\mathbf{x}_j$  can be either similar or dissimilar since  $S_{(i,j)}$  has a low value. This objective function can be rewritten into a matrix form:

$$\begin{aligned}
\mathcal{J}(\mathbf{W}) &= \sum_{i,j=1}^N S_{(i,j)} \|\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j\|_2^2 \\
&= \sum_{i,j=1}^N S_{(i,j)} (\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j)^\top (\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{x}_j) \\
&= \sum_{i=1}^N \mathbf{x}_i^\top \mathbf{W}^\top \mathbf{W} \mathbf{x}_i \left( \sum_{j=1}^N S_{(i,j)} \right) - \sum_{i,j=1}^N \mathbf{x}_i^\top \mathbf{W}^\top \mathbf{W} \mathbf{x}_j S_{(i,j)} \\
&= \text{tr} [\mathbf{W} \mathbf{X} \mathbf{D} \mathbf{X}^\top \mathbf{W}^\top] - \text{tr} [\mathbf{W} \mathbf{X} \mathbf{S} \mathbf{X}^\top \mathbf{W}^\top] \\
&= \text{tr} [\mathbf{W} \mathbf{X} \mathbf{L} \mathbf{X}^\top \mathbf{W}^\top], \tag{2.24}
\end{aligned}$$

where  $\mathbf{S}$  is an  $N$ -by- $N$  matrix whose  $(i,j)$ -th entry is the similarity  $S_{(i,j)}$ ,  $\mathbf{D}$  is an  $N$ -by- $N$  diagonal matrix with its diagonal entries correspond to  $D_{(i,i)} = \sum_{j=1}^N S_{(i,j)}$ , and  $\mathbf{L} = \mathbf{D} - \mathbf{S}$ . To prevent arbitrary scaling and an uninformative minimizer such as a zero matrix, a normalization term is included into the objective function as

$$\mathcal{J}(\mathbf{W}) = \text{tr} \left[ (\mathbf{W} \mathbf{X} \mathbf{D} \mathbf{X}^\top \mathbf{W}^\top)^{-1} (\mathbf{W} \mathbf{X} \mathbf{L} \mathbf{X}^\top \mathbf{W}^\top) \right], \tag{2.25}$$

where  $(\mathbf{W} \mathbf{X} \mathbf{D} \mathbf{X}^\top \mathbf{W}^\top)^{-1}$  is the normalization term.

Unlike PCA, LPP does not require that  $\mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_z}$ . The optimization problem of LPP is an unconstrained optimization problem over a set of  $d_z$ -by- $d_x$  matrices:

$$\min_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \text{tr} \left[ (\mathbf{W} \mathbf{X} \mathbf{D} \mathbf{X}^\top \mathbf{W}^\top)^{-1} (\mathbf{W} \mathbf{X} \mathbf{L} \mathbf{X}^\top \mathbf{W}^\top) \right]. \tag{2.26}$$

The solution of an optimization problem in this form is obtained by solving a *generalized eigenvalue problem* (Fukunaga, 1990). In a generalized eigenvalue problem, given two  $d_x$ -by- $d_x$  square matrices  $\mathbf{X} \mathbf{L} \mathbf{X}^\top$  and  $\mathbf{X} \mathbf{D} \mathbf{X}^\top$ , we seek for a set of *generalized eigenvectors*  $\{\mathbf{v}_i\}_{i=1}^{d_x}$  and a set of *generalized eigenvalues*  $\{\lambda_i\}_{i=1}^{d_x}$  such that

$$\mathbf{X} \mathbf{L} \mathbf{X}^\top \mathbf{v}_i = \lambda_i \mathbf{X} \mathbf{D} \mathbf{X}^\top \mathbf{v}_i. \tag{2.27}$$

Unlike eigenvectors, these generalized eigenvectors are not necessarily orthonormal to each other. Instead, they are  $\mathbf{C}$ -orthonormal (Bai et al., 2000), i.e.,  $\mathbf{v}_i^\top \mathbf{C} \mathbf{v}_i = 1$  and  $\mathbf{v}_i^\top \mathbf{C} \mathbf{v}_j = 0$  for  $i \neq j$ . Note that  $\mathbf{C} = \mathbf{X} \mathbf{D} \mathbf{X}^\top$  in LPP. The solution  $\mathbf{W}$  consists of  $d_z$  generalized eigenvectors associated with  $d_z$  smallest generalized eigenvalues  $\lambda_i$ . Similarly to eigendecomposition,  $\mathbf{W}$  obtained by solving generalized eigenvalue problem is always the global solution.

The main advantage of LPP is that it preserves pairwise structure of data which is very beneficial since many machine learning methods rely on pairwise relationship between data points. However, the performance and solution of LPP depend heavily

on the similarity measure used for computing  $S$ . While any similarity measure can be used, the two most popular ones are the squared exponential measure defined as

$$S_{(i,j)} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right), \quad (2.28)$$

and the  $k$ -nearest neighbor measure defined as

$$S_{(i,j)} = \begin{cases} 1, & \text{if } \mathbf{x}_j \text{ belongs to the } k \text{ closest neighbor of } \mathbf{x}_i, \\ 0, & \text{otherwise.} \end{cases} \quad (2.29)$$

Unfortunately, there is no systematic approach to determine which measure is better due to the unsupervised nature of the problem. Moreover, these measures contain tuning parameters, namely  $\sigma$  and  $k$ , which need to be appropriately determined. For this reason, dimension reduction via LPP can be subjective and may not be reliable.

As reviewed above, PCA considers information as the global structure of data while LPP considers information as the local structure of data. These two methods are regarded as the classical unsupervised linear dimension reduction methods. Later in Section 2.4, we will show that we can alternatively consider information as an amount of statistical dependence between data.

## 2.3 Supervised Linear Dimension Reduction

Supervised linear dimension reduction methods learn a matrix  $\mathbf{W}$  from input-output data  $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$  such that  $\mathbf{z} = \mathbf{W}\mathbf{x}$  preserves *information in  $\mathbf{x}$  about  $\mathbf{y}$* . Similarly to unsupervised dimension reduction, this ‘information’ is not mathematically defined. In the followings, we firstly review a classical supervised linear dimension reduction method for *classification* setting. Then, we review supervised linear dimension reduction methods in the *sufficient dimension reduction* framework which information is defined based on statistical dependence.

### 2.3.1 Linear Discriminant Analysis

*Fisher discriminant analysis* (FDA) (Fisher, 1936; Fukunaga, 1990) is a classical supervised dimension reduction method for classification. Classification is a supervised learning problem where output  $y \in \{1, \dots, c\}$  is a discrete variable called a *class label* and the goal is to learn a function  $f(\mathbf{x})$  called a *classifier* that assigns  $\mathbf{x}$  a correct class label. The main idea of FDA is to find a matrix  $\mathbf{W}$  such that low-dimensional data points from the same class are closed to each other, and data points from different classes are far from each other. This goal can be achieved by minimizing the following objective function:

$$\mathcal{J}(\mathbf{W}) = \text{tr} \left[ \left( \mathbf{W} \mathbf{S}^{(b)} \mathbf{W}^\top \right)^{-1} \left( \mathbf{W} \mathbf{S}^{(w)} \mathbf{W}^\top \right) \right], \quad (2.30)$$

where  $\mathbf{S}^{(w)} \in \mathbb{R}^{d_x \times d_x}$  denotes the *within-class scatter* matrix and  $\mathbf{S}^{(b)} \in \mathbb{R}^{d_x \times d_x}$  denotes the *between-class scatter* matrix. The within-class scatter is defined by

$$\mathbf{S}^{(w)} = \sum_{m=1}^c \sum_{i: y_i=m} (\mathbf{x}_i - \boldsymbol{\mu}_m)(\mathbf{x}_i - \boldsymbol{\mu}_m)^\top, \quad (2.31)$$

where  $\sum_{i:y_i=m}$  denotes summation of data points  $\mathbf{x}_i$  in class  $m$  and  $\boldsymbol{\mu}_m = \frac{1}{N_m} \sum_{i:y_i=m} \mathbf{x}_i$  denotes the mean of  $N_m$  data points in class  $m$ . The between-class scatter is defined by

$$\mathbf{S}^{(b)} = \sum_{m=1}^c (\boldsymbol{\mu}_m - \boldsymbol{\mu})(\boldsymbol{\mu}_m - \boldsymbol{\mu})^\top, \quad (2.32)$$

where  $\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$  denotes the mean of all data points. Intuitively, this means that FDA finds a matrix  $\mathbf{W}$  which minimizes the distance from data points to their class's center, and maximizes the distance between their classes' center. In other words, FDA aims to preserve *class separability* of data.

FDA does not require that  $\mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}$  and it solves an unconstrained optimization problem

$$\min_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \text{tr} \left[ \left( \mathbf{W}\mathbf{S}^{(b)}\mathbf{W}^\top \right)^{-1} \left( \mathbf{W}\mathbf{S}^{(w)}\mathbf{W}^\top \right) \right]. \quad (2.33)$$

Similarly to LPP, the solution to this optimization problem is obtained by solving a generalized eigenvalue problem of matrices  $\mathbf{S}^{(w)}$  and  $\mathbf{S}^{(b)}$ , and then choosing  $d_z$  generalized eigenvectors associated with  $d_z$  smallest generalized eigenvalues (Fukunaga, 1990).

FDA is a popular dimension reduction method for classification and it has been studied for more than decades. Many researchers have proposed methods that extend and improve LDA in many directions. These methods include but not limited to *local LDA* (Sugiyama, 2007), *semi-supervised LDA* (Sugiyama et al., 2008; Yang et al., 2009), *probabilistic LDA* (Ioffe, 2006; Zhang and Yeung, 2009), and *regularized LDA* (Friedman, 1989; Zhang et al., 2010). However, the main limitation of these LDA-based methods is that they are only applicable to classification where output  $y$  is a categorical variable.

Next, we consider methods in the framework of *sufficient dimension reduction* (Li, 1991; Cook and Ni, 2005) which is applicable to non-categorical output as well.

### 2.3.2 Sliced Inverse Regression

*Sliced inverse regression* (SIR) (Li, 1991) is a pioneer method of *sufficient dimension reduction* (Li, 1991; Cook and Ni, 2005). Sufficient dimension reduction is a supervised linear dimension reduction framework where the goal is to find a matrix  $\mathbf{W} \in \{\mathbf{W} | \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}\}$  which satisfies the equality

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{W}\mathbf{x}). \quad (2.34)$$

This equality is also equivalent to the conditional independence:

$$(\mathbf{y} \perp\!\!\!\perp \mathbf{x}) | \mathbf{W}\mathbf{x}, \quad (2.35)$$

where  $\perp\!\!\!\perp$  denotes statistical independence between two random variables. This conditional independence implies that given  $\mathbf{W}\mathbf{x}$ , input  $\mathbf{x}$  and output  $\mathbf{y}$  are not related to each other. Therefore, we can use  $\mathbf{W}\mathbf{x}$  instead of  $\mathbf{x}$  to describe  $\mathbf{y}$  without losing any information.

In general, output can be any quantity including a real-valued vector. However, the procedure of SIR is only applicable when output is a real-valued scalar. The key principal of SIR lies on the following equality:

$$\mathbb{E}[\mathbf{c}^\top \mathbf{x} | \mathbf{W}\mathbf{x}] = a_0 + \sum_{i=1}^{d_z} a_i \vec{\mathbf{w}}_i^\top \mathbf{x}. \quad (2.36)$$

The importance of this equality is that if the equality holds for any  $\mathbf{c} \in \mathbb{R}^{d_x}$  and some constants  $a_0, a_1, \dots, a_{d_z}$ , then the *inverse regression function*  $\mathbb{E}[\mathbf{x}|y]$  lies on the space spanned by  $\mathbf{W}$  which satisfies Equation (2.34). Based on this fact, SIR estimates  $\mathbf{W}$  as follows. First, the range of  $y$  is sliced into  $H$  slices and  $\mathbb{E}[\mathbf{x}|y]$  is estimated as the mean of  $\mathbf{x}$  for each slice of  $y$ . That is, data  $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$  is firstly split into  $H$  disjoint subsets based on the value of  $y_i$  and the mean of  $\mathbf{x}$  is computed using  $\{\mathbf{x}_i\}_{i=1}^{N_h}$  in each set:

$$\begin{aligned} \{(\mathbf{x}_i, y_i \in \mathcal{D}_1)\}_{i=1}^{N_1} &\rightarrow \mathbb{E}[\mathbf{x}|y \in \mathcal{D}_1] \approx \frac{1}{N_1} \sum_{i=1}^{N_1} \mathbf{x}_i = \bar{\mathbf{x}}_1, \\ \{(\mathbf{x}_i, y_i \in \mathcal{D}_2)\}_{i=1}^{N_2} &\rightarrow \mathbb{E}[\mathbf{x}|y \in \mathcal{D}_2] \approx \frac{1}{N_2} \sum_{i=1}^{N_2} \mathbf{x}_i = \bar{\mathbf{x}}_2, \\ &\vdots \\ \{(\mathbf{x}_i, y_i \in \mathcal{D}_H)\}_{i=1}^{N_H} &\rightarrow \mathbb{E}[\mathbf{x}|y \in \mathcal{D}_H] \approx \frac{1}{N_H} \sum_{i=1}^{N_H} \mathbf{x}_i = \bar{\mathbf{x}}_H, \end{aligned} \quad (2.37)$$

where  $\mathbb{E}[\mathbf{x}|y \in \mathcal{D}_h]$  denotes inverse regression for the  $h$ -th slice. Note that each mean  $\bar{\mathbf{x}}_h$  is a  $d_x$ -dimensional vector. Next, the covariance matrix of the mean is estimated by

$$\Sigma = \frac{1}{N} \sum_{h=1}^H N_h \bar{\mathbf{x}}_h \bar{\mathbf{x}}_h^\top. \quad (2.38)$$

Finally, SIR computes an eigendecomposition of  $\Sigma$  and chooses  $d_z$  eigenvectors associated with the  $d_z$  largest eigenvalues as the matrix  $\mathbf{W}$ .

SIR has been extensively studied and has been extended in many directions including *localized SIR* (Wu et al., 2008) and *regularized SIR* (Zhong et al., 2005; Bernard-Michel et al., 2009). A similar idea to SIR is also adopted in *sliced average variance estimation* (Cook, 2000) and *directional regression* (Li and Wang, 2007) where the inverse regression function is replaced by inverse variance functions.

Methods based on inverse regression are simple and computationally efficient. However, they typically require the input distribution to be elliptically symmetric distribution such as the Gaussian distribution. This requirement is very restrictive and thus the practical usefulness of inverse regression-based methods is quite limited.

### 2.3.3 Methods based on Gradients of Conditional Density functions

Another approach to sufficient dimension reduction is based on gradients of conditional density functions. We firstly describe the fundamental idea of this approach below.

Let  $\mathbf{z} = \mathbf{W}\mathbf{x}$ , then from the equality  $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{z})$  in Equation (2.34) we can verify an important relationship between the matrix  $\mathbf{W}$  and gradients of conditional density  $p(\mathbf{y}|\mathbf{x})$  as follows:

$$\begin{aligned}\frac{\partial p(\mathbf{y}|\mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial p(\mathbf{y}|\mathbf{z})}{\partial \mathbf{x}} \\ &= \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \frac{\partial p(\mathbf{y}|\mathbf{z})}{\partial \mathbf{z}} \\ &= \mathbf{W}^\top \frac{\partial p(\mathbf{y}|\mathbf{z})}{\partial \mathbf{z}},\end{aligned}\tag{2.39}$$

where we used the chain-rule and the fact that  $\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \mathbf{W}^\top$ . By left-multiplying both side with  $\mathbf{W}$ , we can also see that

$$\mathbf{W} \frac{\partial p(\mathbf{y}|\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial p(\mathbf{y}|\mathbf{z})}{\partial \mathbf{z}},\tag{2.40}$$

where on the right hand-sided we have  $\mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}$ . Equation (2.39) suggests that we may learn the matrix  $\mathbf{W}$  by minimizing an error between  $\frac{\partial p(\mathbf{y}|\mathbf{x})}{\partial \mathbf{x}}$  and  $\mathbf{W}^\top \frac{\partial p(\mathbf{y}|\mathbf{z})}{\partial \mathbf{z}}$ . In particular, we minimize the sum of the squared error:

$$\begin{aligned}\mathcal{J}(\mathbf{W}) &= \sum_{i=1}^N \left\| \frac{\partial p(\mathbf{y}|\mathbf{x}_i)}{\partial \mathbf{x}} - \mathbf{W}^\top \frac{\partial p(\mathbf{y}|\mathbf{z}_i)}{\partial \mathbf{z}} \right\|_2^2 \\ &= \sum_{i=1}^N \left\| \frac{\partial p(\mathbf{y}|\mathbf{x}_i)}{\partial \mathbf{x}} - \mathbf{W}^\top \mathbf{W} \frac{\partial p(\mathbf{y}|\mathbf{x}_i)}{\partial \mathbf{x}} \right\|_2^2 \\ &= \left\| \mathbf{M} - \mathbf{W}^\top \mathbf{W} \mathbf{M} \right\|_{\text{Fro}}^2,\end{aligned}\tag{2.41}$$

where  $\mathbf{M}$  is a  $d_x$ -by- $N$  matrix whose  $(i, j)$ -th entry is the partial derivative of  $p(\mathbf{y}|\mathbf{x})$  w.r.t.  $x^{(i)}$  evaluated at data point  $\mathbf{x}_j$ , i.e.,  $M_{(i,j)} = \frac{\partial p(\mathbf{y}|\mathbf{x}_j)}{\partial x^{(i)}}$ . Then, the matrix  $\mathbf{W}$  can be obtained by solving the following optimization problem:

$$\begin{aligned}\min_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \quad & \left\| \mathbf{M} - \mathbf{W}^\top \mathbf{W} \mathbf{M} \right\|_{\text{Fro}}^2 \\ \text{subject to} \quad & \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}.\end{aligned}\tag{2.42}$$

Interestingly, this optimization problem highly resembles the optimization problem of PCA in Equation (2.19). Likewise, the solution to this optimization problem is obtained by computing eigendecomposition of the matrix  $\mathbf{M}\mathbf{M}^\top$  and choosing  $d_z$  eigenvectors associated with the largest  $d_z$  eigenvalues.

As we have shown, sufficient dimension reduction can be performed using the gradients of the conditional density function. However, these gradients are often unknown and need to be estimated from data. The main focus of gradient-based sufficient dimension reduction methods is to accurately estimate the gradients from data. These methods include but not limited to *average derivative estimates* (Samarov, 1993), *outer product of gradient based on the conditional density functions* (Xia, 2007), *gradient-based kernel dimension reduction* (Fukumizu and Leng, 2012), and *least-squares logarithmic conditional density gradients* (Sasaki et al., 2015b). While these methods have their own advantages and disadvantages, they have one common disadvantage when compared with other sufficient dimension reduction approach. That is, the gradient estimation for  $\frac{\partial p(\mathbf{y}|\mathbf{x})}{\partial \mathbf{x}}$  needs to be done in the high-dimensional space of  $\mathbf{x}$  which makes accurate estimation very challenging.

### 2.3.4 Minimum Average Variance Estimation based on the Conditional Density Functions

*Minimum average variance estimation based on the conditional density functions* (dMAVE) (Xia, 2007) is a sufficient dimension reduction method that is not based on inverse functions or gradients of conditional density functions. Instead, dMAVE finds a matrix  $\mathbf{W}$  which yields an accurate non-parametric estimation of the conditional density  $p(y|\mathbf{z})$ .

dMAVE estimates the conditional density  $p(y|\mathbf{z})$  based on the following model:

$$H_b(\tilde{y} - y) = m_b(\mathbf{z}, y) + \varepsilon_b(y|\mathbf{z}), \quad (2.43)$$

where  $H_b$  denotes a symmetric *kernel function* with bandwidth  $b > 0$ ,  $m_b(\mathbf{z}, y)$  denotes a conditional expectation of  $H_b(\tilde{y} - y)$  given  $\mathbf{z}$ , and  $\varepsilon_b(y|\mathbf{z}) = H_b(\tilde{y} - y) - \mathbb{E}[H_b(\tilde{y} - y)|\mathbf{z}]$  with  $\mathbb{E}[\varepsilon_b(y|\mathbf{z})] = 0$ . An important property of this model is that  $m_b(\mathbf{z}, y) \rightarrow p(y|\mathbf{z})$  when  $b \rightarrow 0$  as  $n \rightarrow \infty$ . This means that an estimator  $m_b(\mathbf{z}, y)$  asymptotically converges to  $p(y|\mathbf{z})$ . Then, dMAVE estimates  $m_b(\mathbf{z}, y)$  by a local linear smoother (Fan et al., 1996). More specifically, a local linear smoother of  $m_b(\mathbf{z}_i, y_k)$  is given by

$$\begin{aligned} m_b(\mathbf{z}_i, y_k) &\approx m_b(\mathbf{z}_j, y_k) + \frac{\partial m_b(\mathbf{z}_j, y_k)}{\partial \mathbf{z}} (\mathbf{z}_i - \mathbf{z}_j) \\ &= a_{jk} + \mathbf{b}_{jk}^\top \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j), \end{aligned} \quad (2.44)$$

where  $\mathbf{z}_j$  is an arbitrary point close to  $\mathbf{z}_i$ , and  $a_{jk} \in \mathbb{R}$  and  $\mathbf{b}_{jk} \in \mathbb{R}^{d_z}$  are parameters. This local linear smoother can also be regarded as the first order Taylor approximation. Based on this local linear smoother and the relation to the conditional density above, dMAVE solves the following optimization problem:

$$\begin{aligned} \min_{\mathbf{W}, a_{jk}, \mathbf{b}_{jk}} & \frac{1}{n^3} \sum_{j,k=1}^n \rho(\mathbf{x}_j, y_k) \sum_{i=1}^n [H_b(y_i - y_k) - a_{jk} - \mathbf{b}_{jk}^\top \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j)]^2 K_h(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} & \mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_z}, \end{aligned} \quad (2.45)$$

where  $K_h$  is a symmetric kernel function for  $\mathbf{x}$  with bandwidth  $h > 0$ . The function  $\rho(\mathbf{x}, y)$  is a trimming function which is evaluated as zero when the densities of  $\mathbf{x}$  or  $y$  are lower than some threshold. A solution to this minimization problem is obtained by alternatively solving quadratic programming problems for  $\mathbf{W}$ , and  $(a_{jk}, \mathbf{b}_{jk})$  until convergence.

dMAVE does not require any assumption on the data distribution unlike inverse regression based methods. Moreover, the estimation is done in low-dimensional space of  $\mathbf{z}$  unlike gradient-based methods. However, performance and solution of dMAVE depends on the choice of the kernel bandwidth and trimming threshold, and so far there is no systematic method to choose these tuning parameters. In practice, dMAVE uses a bandwidth selection method based on the *normal-reference* rule of the non-parametric conditional density estimation (Silverman, 1986; Fan et al., 1996), and a fixed trimming threshold. Although this model selection strategy works reasonably well in general, it does not always guarantee good performance.

Unlike linear dimension reduction methods we introduced so far, dMAVE does not compute eigendecomposition or solve generalized eigenvalue problem to obtain the matrix  $\mathbf{W}$ . Instead, it iteratively solves quadratic programming problems until

convergence. However, the optimization problem in Equation (2.45) is non-convex and a solution obtained by this approach can be a local solution. dMAVE alleviates this issue by using a gradient-based sufficient dimension reduction method called dOPG (Xia, 2007) to learn an initial solution. Thus, dMAVE may not perform well if dOPG fails to provide a good initial solution.

Another disadvantage of dMAVE is that the provided learning procedure and theoretical analysis are established only for a real-value scalar output  $y$ . For this reason, the applicability of dMAVE to vector-valued output is currently unclear.

## 2.4 Dimension Reduction based on Statistical Dependence

Linear dimension reduction methods that we reviewed above consider different definition of information that the matrix  $\mathbf{W}$  needs to preserve. For unsupervised methods, PCA defines information as the global structure of data and LPP defines information as the local structure of data. For supervised methods, LDA defines information as the class separability of data and sufficient dimension reduction defines information as an amount of statistical dependence. In this section, we firstly show that statistical dependence can also be used for unsupervised scenarios as well. Then, we overview five approaches that were used to evaluate statistical dependence for dimension reduction.

### 2.4.1 Dimension Reduction and Statistical Dependence

Two random variables  $\mathbf{x}$  and  $\mathbf{y}$  are statistically independent if and only if

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}). \quad (2.46)$$

Assume that we have an additional random variable  $\mathbf{z}$ , then  $\mathbf{x}$  and  $\mathbf{y}$  are statistically independent given  $\mathbf{z}$  if and only if

$$p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z}). \quad (2.47)$$

This conditional independence is also denoted by

$$(\mathbf{x} \perp \mathbf{y})|\mathbf{z}. \quad (2.48)$$

In the sufficient dimension reduction framework, the variables  $\mathbf{x}$ ,  $\mathbf{y}$ , and  $\mathbf{z}$  are input, output, and low-dimensional data, respectively. In this case, the conditional independence implies that given  $\mathbf{z}$ , input  $\mathbf{x}$  and output  $\mathbf{y}$  are not related and we may use  $\mathbf{z}$  instead of  $\mathbf{x}$  to describe  $\mathbf{y}$ . Now, let us assume that the variable  $\mathbf{y}$  is an identical copy of  $\mathbf{x}$  denoted by  $\tilde{\mathbf{x}}$ , i.e.,  $\mathbf{y} = \tilde{\mathbf{x}}$ . In this case, the conditional independence

$$(\mathbf{x} \perp \tilde{\mathbf{x}})|\mathbf{z}, \quad (2.49)$$

implies that given  $\mathbf{z}$ , the input  $\mathbf{x}$  and its copy  $\tilde{\mathbf{x}}$  are statistically independent and not related. This means that  $\mathbf{z}$  contains all necessary information that can describe  $\mathbf{x}$  and we can use  $\mathbf{z}$  instead of  $\mathbf{x}$  without losing any information. This suggests that we may perform unsupervised linear dimension reduction by finding a matrix  $\mathbf{W}$  such that  $\mathbf{z} = \mathbf{W}\mathbf{x}$  satisfies the conditional independence in Equation (2.49).

As we have shown, both supervised and unsupervised linear dimension reduction can be performed based on statistical dependence. The remaining question is how to estimate statistical dependence of random variables from data. Below, we review approaches that estimate statistical dependence from data.

## 2.4.2 Pearson Correlation Coefficient

A classical statistical dependence measure is the *Pearson correlation coefficient* (PCC) (Galton, 1886; Pearson, 1895). PCC between two scalar random variables  $z$  and  $y$  is defined as

$$\begin{aligned} \text{PCC}(Z, Y) &= \frac{\mathbb{E}_{p(z,y)} [(z - \mathbb{E}_{p(z)}[z])(y - \mathbb{E}_{p(y)}[y])]}{\sqrt{\mathbb{E}_{p(z)} [(z - \mathbb{E}_{p(z)}[z])^2]} \sqrt{\mathbb{E}_{p(y)} [(y - \mathbb{E}_{p(y)}[y])^2]}} \\ &= \frac{\text{cov}(z, y)}{\sqrt{\text{var}(z)} \sqrt{\text{var}(y)}}, \end{aligned} \quad (2.50)$$

where  $\text{cov}(z, y) = \mathbb{E}_{p(z,y)} [(z - \mathbb{E}_{p(z)}[z])(y - \mathbb{E}_{p(y)}[y])]$  denotes the covariance between  $z$  and  $y$ , and  $\text{var}(z) = \mathbb{E}_{p(z)} [(z - \mathbb{E}_{p(z)}[z])^2]$  denotes the variance of  $z$ . An estimated PCC can be obtained by estimating the covariance and the variances from data and substitute them into Equation (2.50).

The value of PCC lies between  $-1$  and  $1$  and it can be used to evaluate *linear dependency* between  $z$  and  $y$  where the absolute value of PCC determines the amount of dependency. More specifically, if  $z$  and  $y$  are independent then  $\text{PCC}(Z, Y)$  is zero. However, it should be noted that the reverse is not necessarily true, i.e.,  $\text{PCC}(Z, Y) = 0$  does not imply that  $z$  and  $y$  are independent. The value  $\text{PCC}(Z, Y) > 0$  implies that  $z$  and  $y$  has positive linear dependency where  $z$  increases as  $y$  increases. On the other hand, the value  $\text{PCC}(Z, Y) < 0$  implies that  $z$  and  $y$  has negative dependency where  $z$  decreases as  $y$  increases and vice versa. The main limitation of PCC is that it can only detect linear dependency between random variables. In practice, data often has non-linear dependency and thus PCC is not sufficient.

## 2.4.3 Conditional Covariance Operator on Reproducing Kernel Hilbert Spaces

*Reproducing kernel Hilbert space* (RKHS) (Aronszajn, 1950) is a very important mathematics tool in machine learning. Firstly, we give an overview of RKHS and then describe its relation to statistical independence.

Roughly speaking, an RKHS is a space of real-valued functions equipped with an inner product where there exists a function which has the reproducing property. We use  $\mathcal{H}_{\mathcal{Z}}$  to denote a space of real-valued functions on a set  $\mathcal{Z}$  equipped with an inner product  $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\mathcal{Z}}}$ . That is,  $f \in \mathcal{H}_{\mathcal{Z}}$ ,  $f : \mathcal{Z} \rightarrow \mathbb{R}$ , and  $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\mathcal{Z}}} : \mathcal{H}_{\mathcal{Z}} \times \mathcal{H}_{\mathcal{Z}} \rightarrow \mathbb{R}$ . Then, the space  $\mathcal{H}_{\mathcal{Z}}$  is an RKHS if there exists a *reproducing kernel*  $k_{\mathcal{Z}} : \mathcal{Z} \times \mathcal{Z} \rightarrow \mathbb{R}$  which satisfies the *reproducing property*:

$$\langle f, k_{\mathcal{Z}}(\cdot, z) \rangle_{\mathcal{H}_{\mathcal{Z}}} = f(z), \quad (2.51)$$

where  $z \in \mathcal{Z}$ , and  $k_{\mathcal{Z}}(\cdot, z) \in \mathcal{H}_{\mathcal{Z}}$ . We also use  $\mathcal{H}_{\mathcal{Y}}$  to denote an RKHS on a set  $\mathcal{Y}$  with a kernel  $k_{\mathcal{Y}}$ .

Researchers have shown that statistical dependence can be evaluated through operations between RKHSs on random variables (Bach and Jordan, 2003; Fukumizu et al., 2004; Gretton et al., 2005; Fukumizu et al., 2009). More specifically, given  $\mathcal{H}_{\mathcal{Y}}$  and  $\mathcal{H}_{\mathcal{Z}}$ , the *cross-covariance operator*  $\Sigma_{\mathcal{Y}\mathcal{Z}} : \mathcal{H}_{\mathcal{Z}} \rightarrow \mathcal{H}_{\mathcal{Y}}$  satisfies the following equality for all  $f \in \mathcal{H}_{\mathcal{Z}}$  and  $g \in \mathcal{H}_{\mathcal{Y}}$ :

$$\langle g, \Sigma_{\mathcal{Y}\mathcal{Z}} f \rangle_{\mathcal{H}_{\mathcal{Y}}} = \mathbb{E}_{p(z,y)} [f(z)g(y)] - \mathbb{E}_{p(z)} [f(z)] \mathbb{E}_{p(y)} [g(y)]. \quad (2.52)$$

Next, the *conditional covariance operator*  $\Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{Z}}$  is defined using cross-covariance operators by

$$\Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{Z}} = \Sigma_{\mathcal{Y}\mathcal{Y}} - \Sigma_{\mathcal{Y}\mathcal{Z}}\Sigma_{\mathcal{Z}\mathcal{Z}}^{-1}\Sigma_{\mathcal{Z}\mathcal{Y}}, \quad (2.53)$$

where it is assumed that the inverse operator  $\Sigma_{\mathcal{Z}\mathcal{Z}}^{-1}$  exists. This conditional covariance operator is related to the conditional independence through the following two relations (Fukumizu et al., 2004, 2009):

$$\Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{Z}} \geq \Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{X}}, \quad (2.54)$$

and

$$\Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{Z}} = \Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{X}} \iff (\mathbf{x} \perp \mathbf{y})|z, \quad (2.55)$$

where the inequality refers to the partial order of self-adjoint operators. These relations mean that the conditional independence in Equation (2.48) can be achieved by minimizing  $\Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{Z}}$  in the sense of the partial order of self-adjoint operators.

There are two dimension reduction methods that utilize the conditional covariance operator. The first method is *kernel dimension reduction* (KDR) (Fukumizu et al., 2004) which is a supervised linear dimension reduction method that minimizes an empirical estimator of the trace of  $\Sigma_{\mathcal{Y}\mathcal{Y}|\mathcal{Z}}$  to find the matrix  $\mathbf{W}$ . That is, KDR solves the following optimization problem:

$$\begin{aligned} & \max_{\mathbf{W} \in \mathbb{R}^{d_{\mathcal{Z}} \times d_{\mathcal{X}}}} \text{tr} [\mathbf{K}_{\mathcal{Y}} (\mathbf{K}_{\mathcal{Z}} + \lambda \mathbf{I}_N)^{-1}] \\ & \text{subject to } \mathbf{W}\mathbf{W}^{\top} = \mathbf{I}_{d_{\mathcal{Z}}}, \end{aligned} \quad (2.56)$$

where  $\mathbf{K}_{\mathcal{Z}}$  and  $\mathbf{K}_{\mathcal{Y}}$  denotes centered Gram matrices (Schölkopf et al., 1998; Bach and Jordan, 2002) with kernels  $k_{\mathcal{Z}}$  and  $k_{\mathcal{Y}}$ , respectively. The second method extends KDR to unsupervised dimension reduction and is called *unsupervised kernel dimension reduction* (Wang et al., 2010). In essence, this method replaces output  $\mathbf{y}$  with an identical copy of input  $\mathbf{x}$  and then perform KDR. It also proposes an extension of KDR to non-linear dimension reduction by applying a non-linear transformation to data first and then applying KDR. This non-linearization will be explained in details in Section 2.5.

The main advantage of using the conditional covariance operator is that the operator can detect non-linear dependency. However, using this operator requires us to define the kernels and their parameters. Unfortunately, there seems to be no justifiable model selection method to choose these parameters so far (Fukumizu et al., 2009). Moreover, Equation (2.56) is a non-convex optimization problem and an iterative procedure that were used in KDR and its unsupervised extension may result in a local solution.

#### 2.4.4 Mutual Information

*Mutual information* (MI) is a well-known statistical dependence measure that have been widely used in many applications (Cover and Thomas, 1991). Mutual information between random variables  $z$  and  $\mathbf{y}$  is defined as

$$\text{MI}(\mathbf{Z}, \mathbf{Y}) = \iint p(z, \mathbf{y}) \log \frac{p(z, \mathbf{y})}{p(z)p(\mathbf{y})} dz d\mathbf{y}. \quad (2.57)$$

The important properties of MI are that it is always non-negative and  $\text{MI}(Z, Y)$  is zero if and only if  $z$  and  $\mathbf{y}$  are statistically independent, i.e.,  $p(z, \mathbf{y}) = p(z)p(\mathbf{y})$ .

In statistics, a *divergence* refers to a 'pseudo' measure of distance for two probability distributions and it also plays an important role in machine learning.  $\text{MI}(Z, Y)$  is equivalent to the *Kullback-Leibler* (KL) divergence (Kullback and Leibler, 1951) from  $p(z, \mathbf{y})$  to  $p(z)p(\mathbf{y})$  where the KL divergence from  $p(z)$  to  $q(z)$  is defined as

$$\text{KL}(p||q) = \int p(z) \log \frac{p(z)}{q(z)} dz. \quad (2.58)$$

Note that a divergence is not strictly a distance since it is not required to be symmetric, e.g.,  $\text{KL}(p||q) \neq \text{KL}(q||p)$ . Interestingly, the KL divergence can be regarded as an instance of two divergence classes. The first divergence class is the *f*-divergence (Ali and Silvey, 1966; Csiszár, 1967). The *f*-divergence from  $p(z)$  to  $q(z)$  is defined as

$$D_f(p||q) = \int q(z) f\left(\frac{p(z)}{q(z)}\right) dz, \quad (2.59)$$

where  $f$  is a convex function such that  $f(0) = 1$ . The KL divergence corresponds to the *f*-divergence with  $f(t) = t \log t$ . The second divergence class that the KL divergence belongs to is the *density power* divergence (Basu et al., 1998) and is defined as

$$\text{DP}_\alpha(p||q) = \int \left( p(z)^{1+\alpha} - \left(1 + \frac{1}{\alpha}\right) p(z)q(z)^\alpha + \frac{1}{\alpha} q(z)^{1+\alpha} \right) dz, \quad (2.60)$$

where  $\alpha > 0$  is the parameter of the divergence. The KL divergence corresponds to the density power divergence with  $\alpha \rightarrow 0$  in the limit.

We can verify that MI is related to the conditional independence and can be used for dimension reduction as follows. Given random variables  $\mathbf{x}$ ,  $\mathbf{y}$ , and  $z = \mathbf{W}\mathbf{x}$  we can verify that

$$\begin{aligned} \text{MI}(\mathbf{X}, \mathbf{Y}) - \text{MI}(Z, \mathbf{Y}) &= \iint p(\mathbf{x}, \mathbf{y}) \log \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} d\mathbf{x}d\mathbf{y} \\ &\quad - \iint p(z, \mathbf{y}) \log \frac{p(z, \mathbf{y})}{p(z)p(\mathbf{y})} dzd\mathbf{y} \\ &= \iint p(\mathbf{x}, \mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y}|z)} d\mathbf{x}d\mathbf{y} \\ &\geq 0, \end{aligned} \quad (2.61)$$

where the inequality follows from Jensen's inequality (Jensen, 1906). We can see that the equality holds if and only if  $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|z)$ . This implies that the conditional independence is achieved when  $\text{MI}(\mathbf{X}, \mathbf{Y}) = \text{MI}(Z, \mathbf{Y})$ . Since  $\text{MI}(\mathbf{X}, \mathbf{Y})$  is always larger than  $\text{MI}(Z, \mathbf{Y})$ , dimension reduction can be performed by maximizing  $\text{MI}(Z, \mathbf{Y})$ :

$$\begin{aligned} &\max_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \text{MI}(Z, \mathbf{Y}) \\ &\text{subject to } \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}. \end{aligned} \quad (2.62)$$

MI is a well-known tool and many researchers have used MI in various data analysis problems including dimension reduction (Cover and Thomas, 1991; Peng et al.,

2005; Brown, 2009; Faivishevsky and Goldberger, 2012) and clustering (Agakov and Barber, 2005; Gomes et al., 2010; Romano et al., 2014). The main appeal of MI is that it can detect non-linear dependency. Moreover, an efficient method to estimate  $\text{MI}(\mathbf{Z}, \mathbf{Y})$  from data is also available (Paninski, 2003; Kraskov et al., 2004; Suzuki et al., 2008; Pál et al., 2010; Gao et al., 2015). For these reasons, dimension reduction based on maximizing MI seems to be an appealing approach.

However, MI is not always the optimal choice for measuring statistical dependence because it is not robust against outliers. An intuitive explanation is that  $\text{MI}(\mathbf{Z}, \mathbf{Y})$  contains the log function and the density ratio  $\frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})p(\mathbf{y})}$ . In the presence of outliers, the value of logarithm can be highly sharp near zero, and density ratio can be highly fluctuated and diverge to infinity. Thus, the value of MI tends to be unstable and unreliable in the presence of outliers.

#### 2.4.5 Squared-loss Mutual Information

*Squared-loss mutual information* (SMI) is a statistical dependence measure between random variables. SMI between  $\mathbf{z}$  and  $\mathbf{y}$  is defined as

$$\text{SMI}(\mathbf{Z}, \mathbf{Y}) = \iint p(\mathbf{z})p(\mathbf{y}) \left( \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})p(\mathbf{y})} - 1 \right)^2 d\mathbf{z}d\mathbf{y}. \quad (2.63)$$

SMI is always non-negative and  $\text{SMI}(\mathbf{Z}, \mathbf{Y})$  is zero if and only if  $\mathbf{z}$  and  $\mathbf{y}$  are statistically independent. These properties are identical to those of MI and they suggest us that dimension reduction may be performed based on SMI. In fact, SMI is also an  $f$ -divergence from  $p(\mathbf{z}, \mathbf{y})$  to  $p(\mathbf{z})p(\mathbf{y})$ , where the function  $f(t) = (t - 1)^2$  is applied. The  $f$ -divergence with  $f(t) = (t - 1)^2$  is called the *Pearson divergence* (Pearson, 1900).

The *least-squares dimension reduction* (LSDR) (Suzuki and Sugiyama, 2013) method performs supervised linear dimension reduction by maximizing SMI:

$$\begin{aligned} & \max_{\mathbf{W} \in \mathbb{R}^{d_{\mathbf{z}} \times d_{\mathbf{x}}}} \text{SMI}(\mathbf{Z}, \mathbf{Y}) \\ & \text{subject to } \mathbf{W}\mathbf{W}^{\top} = \mathbf{I}_{d_{\mathbf{z}}}. \end{aligned} \quad (2.64)$$

Since SMI is unknown in practice, LSDR estimates it from data. In particular,  $\text{SMI}(\mathbf{Z}, \mathbf{Y})$  is estimated by *least-squares mutual information* (LSMI) (Suzuki et al., 2009), which directly estimates the density ratio  $\frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})p(\mathbf{y})}$  without performing any density estimation. Although an extension of LSDR to unsupervised linear dimension reduction is not yet explored, we may straightforwardly do so by replacing  $\mathbf{y}$  with an identical copy of  $\mathbf{x}$ .

SMI can detect non-linear dependency similarly to MI. However, SMI does not contain the logarithm function and tends to be more robust against outliers than MI. However, SMI still contains the density ratio which makes an accurate estimation of SMI challenging in the presence of outliers.

#### 2.4.6 Quadratic Mutual Information

*Quadratic mutual information* (QMI) is another statistical dependence measure between random variables (Principe et al., 2000). QMI between  $\mathbf{z}$  and  $\mathbf{y}$  is defined as

$$\text{QMI}(\mathbf{Z}, \mathbf{Y}) = \iint (p(\mathbf{z}, \mathbf{y}) - p(\mathbf{z})p(\mathbf{y}))^2 d\mathbf{z}d\mathbf{y}. \quad (2.65)$$

QMI is always non-negative and  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is zero if and only if  $\mathbf{z}$  and  $\mathbf{y}$  are statistically independent. Again, these properties are identical to those of MI and thus we may use QMI for dimension reduction.

$\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is an  $L_2$  distance from  $p(\mathbf{z}, \mathbf{y})$  to  $p(\mathbf{z})p(\mathbf{y})$ . The  $L_2$  distance can also be regarded as the density power divergence in Equation (2.60) where  $\alpha = 1$ . As discussed in Basu et al. (1998), the parameter  $\alpha$  controls the robustness against outliers of the divergence where a large value of  $\alpha$  indicates high robustness. This means that QMI which is the density power divergence with  $\alpha = 1$  is more robust against outliers than MI which is the density power divergence with  $\alpha \rightarrow 0$ .

While comparison between robustness against outliers of QMI and SMI based on their divergence classes is currently unexplored, we may assume that QMI is more robust than SMI. The reason is that unlike SMI, QMI does not contain the density ratio and thus it suffers less from outliers.

QMI has been used for both supervised linear dimension reduction (Principe et al., 2000; Torkkola, 2003) and unsupervised linear dimension reduction (Sainui and Sugiyama, 2014). These methods perform linear dimension reduction by solving the following optimization problem:

$$\begin{aligned} & \max_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \text{QMI}(\mathbf{Z}, \mathbf{Y}) \\ & \text{subject to } \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}, \end{aligned} \quad (2.66)$$

where  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is estimated by either *kernel density estimation* (KDE) (Silverman, 1986) or *least-squares quadratic mutual information* (LSQMI) (Sainui and Sugiyama, 2013). It was shown that these QMI-based dimension reduction methods work well in the presence of outliers when compared with other dimension reduction methods.

## 2.5 Non-linearized Linear Dimension Reduction

Dimension reduction methods that we introduced so far in this chapter are linear dimension reduction methods which learn a matrix  $\mathbf{W}$  that transforms  $\mathbf{x}$  to  $\mathbf{z}$  by  $\mathbf{z} = \mathbf{W}\mathbf{x}$ . In other words, the transformation  $\mathbf{z} = \mathbf{W}\mathbf{x}$  is linear w.r.t. the variable  $\mathbf{x}$ .

Assume that we non-linearly transform  $\mathbf{x}$  to  $\phi(\mathbf{x})$  where  $\phi : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_\phi}$  is a non-linear function. Then, the transformation  $\mathbf{z} = \mathbf{W}\phi(\mathbf{x})$  is non-linear w.r.t. the variable  $\mathbf{x}$  and can be regarded as non-linear dimension reduction. The function  $\phi$  is often called a *feature map* since it maps data points from the original feature space to a new feature space (Bishop, 2006). Applying a non-linear feature map to data is the idea that is commonly used to extend linear learning methods to non-linear learning methods (Bishop, 2006; Murphy, 2012).

The approach described above can be used to extend linear dimension reduction we have described in this chapter to non-linear dimension reduction (Lee and Verleysen, 2007; Cunningham and Ghahramani, 2015). However, the main concern of such an approach is that we need to choose an appropriate feature map  $\phi(\mathbf{x})$ . In general, a highly complex feature map allows us to better preserve information in data. However, such a feature map often results in a high-dimensional  $\phi(\mathbf{x})$  whose dimensionality can be much higher than the dimensionality of the original data  $\mathbf{x}$ . Since the computation time of dimension reduction methods increases as the dimensionality of data increases, using a highly complex feature map is often computationally inefficient and is not preferable in practice.

However, for linear dimension reduction methods that deal with data points only through the inner product, we may use a highly complex feature map without sacrificing much computational efficiency. More specifically, given a feature map  $\phi$  and two data points  $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ , the *theorem of reproducing kernel Hilbert space* (Aronszajn, 1950) states that there exists a reproducing *positive definite kernel*  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  such that

$$k(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle_{\mathcal{H}}. \quad (2.67)$$

Note that we implicitly assume that  $\mathcal{H}$  is a reproducing kernel Hilbert space on  $\mathcal{X}$  and that  $\phi \in \mathcal{H}$ . Equation (2.67) shows that we can evaluate the kernel between two data points instead of evaluating the inner product between feature maps of two data points. This is advantageous since evaluating kernels is often computationally more efficient than evaluating very high-dimensional feature maps.

As an example, we briefly review the *kernel principal component analysis* (KPCA) (Schölkopf et al., 1997) method as follows. As discussed previously, the essential step of PCA is to compute the eigendecomposition of a matrix  $\mathbf{X}\mathbf{X}^\top$  where  $\mathbf{X} \in \mathbb{R}^{d_x \times N}$  is a data matrix. Now, let us assume that we instead perform PCA with a matrix  $\Phi\Phi^\top$  where  $\Phi \in \mathbb{R}^{d_\phi \times N}$  is a new data matrix. That is, we find  $d_z$  eigenvectors associated with  $d_z$  largest eigenvalues such that

$$\Phi\Phi^\top \mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad (2.68)$$

where  $\mathbf{v}_i$  and  $\lambda_i$  is the eigenvector and the eigenvalue, respectively. These eigenvectors can be used to represent the original data. However, computing eigendecomposition of the matrix  $\Phi\Phi^\top$  is computationally expensive when  $d_\phi$  is large.

KPCA avoids explicitly computing eigendecomposition of a high-dimensional matrix  $\Phi\Phi^\top$  by instead computing eigendecomposition of the *Gram matrix*  $\mathbf{K} \in \mathbb{R}^{N \times N}$  which is a symmetric matrix consists of kernels between data points:

$$K_{(i,j)} = k(\mathbf{x}_i, \mathbf{x}_j). \quad (2.69)$$

The kernel is equivalent to the inner product between feature maps of two data points and thus the Gram matrix can be represented using the matrix  $\Phi$  by

$$\mathbf{K} = \Phi^\top \Phi. \quad (2.70)$$

It was shown that the matrices  $\Phi^\top \Phi$  and  $\Phi\Phi^\top$  have the same eigenvalues (Schölkopf et al., 1997). Indeed, suppose that an eigenvector  $\mathbf{u}_i$  of  $\mathbf{K} = \Phi^\top \Phi$  satisfies

$$\Phi^\top \Phi \mathbf{u}_i = \lambda_i \mathbf{u}_i. \quad (2.71)$$

Then, left-multiplying both side of Equation (2.71) with  $\Phi$  gives

$$\Phi\Phi^\top \Phi \mathbf{u}_i = \lambda_i \Phi \mathbf{u}_i. \quad (2.72)$$

By comparing Equation (2.72) with Equation (2.68), we can see that

$$\mathbf{v}_i = \Phi \mathbf{u}_i. \quad (2.73)$$

Thus, the desired eigenvectors  $\{\mathbf{v}_i\}_{i=1}^{d_z}$  can be obtained from the eigenvectors  $\{\mathbf{u}_i\}_{i=1}^{d_z}$ . Since the matrix  $\mathbf{K}$  has dimensionality equal to the number of data points which is

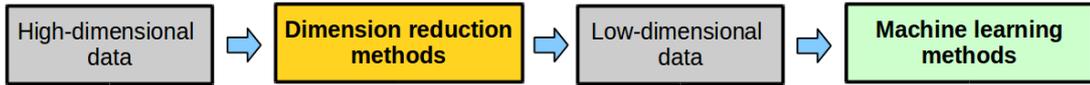


Figure 2.2: Dimension reduction is commonly used as a pre-processing tool for subsequent machine learning methods.

often much smaller than the dimensionality of the feature map, computing eigendecomposition of  $\mathbf{K}$  is computationally more efficient than computing eigendecomposition of  $\Phi\Phi^\top$ . Note that we may need to normalize the eigenvectors  $\{\Phi\mathbf{u}_i\}_{i=1}^{d_z}$  to ensure that they have unit norm.

Kernels allow us to efficiently perform non-linear dimension reduction based on linear methods. However, the performance of such an approach depends on the choice of the kernel or its corresponding feature map. For kernels, the Gaussian kernel is the most commonly used:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right). \quad (2.74)$$

Interestingly, the corresponding feature map of the Gaussian kernel has infinite dimensionality. While the Gaussian kernel was shown to work well, it contains the tuning parameter  $\sigma$  which is called the kernel bandwidth. In practice, this kernel bandwidth is often fixed or chosen by model selection procedure such as cross-validation (Cawley and Talbot, 2007; Murphy, 2012).

## 2.6 Summary of Dimension Reduction

In this chapter we formulated the linear dimension reduction problem and introduced existing linear dimension reduction methods for both unsupervised and supervised scenarios. Then, we showed that both unsupervised and supervised linear dimension reduction can be performed by maximizing a statistical dependence measure. Among statistical dependence measures that we introduced, *quadratic mutual information* (QMI) is highly attractive due to its robustness against outliers. However, as we will show in the next chapter, an existing approach which utilizes QMI for dimension reduction involves a multi-step approach and is not appropriate. In the next chapter, we present our first contribution which overcomes the multi-step issue of QMI-based dimension reduction.

Although dimension reduction is useful by itself for finding important features of data, the most common usage of dimension reduction is to use it as a pre-processing tool before applying machine learning methods. That is, high-dimensional data is processed by a dimension reduction method to obtain a low-dimensional data and then the low-dimensional data is used by a subsequent machine learning method to solve the target problem (see Figure 2.2). However, this naive two-step approach is not always appropriate when the target problem is the *conditional density estimation* problem. This issue will be overcome in Chapter 4 where we will present our second contribution in this dissertation.

## Chapter 3

# Dimension Reduction via Single-step Estimation of the Derivative of Quadratic Mutual Information

In this chapter, we describe our first contribution on dimension reduction based on the single-step estimation of the derivative of quadratic mutual information.

### 3.1 Introduction

As introduced in Chapter 1, machine learning methods often perform poorly when data has high dimensionality due to the curse of dimensionality (Bishop, 2006). An attractive approach to mitigate this issue is to perform linear dimension reduction on data. The goal of linear dimension reduction is to find a low-dimensional subspace of the feature space such that the projected data preserves maximal information in the original data. Then, a subsequent learning method can use the low-dimensional projection of data with a minimal loss of information.

Among many approaches to perform linear dimension reduction, the dependence maximization approach is highly attractive since it is applicable to both unsupervised and supervised scenarios. For unsupervised linear dimension reduction, this approach finds a subspace which maximizes a statistical dependence measure between projected input and an identical copy of input. For supervised linear dimension reduction, this approach finds a subspace which maximizes a statistical dependence measure between projected input and output. Although our proposed method in this chapter is developed from the viewpoint of supervised linear dimension reduction, i.e., a statistical dependence between projected input and output is maximized, the proposed method can be straightforwardly applied to the unsupervised linear dimension reduction as well.

For dimension reduction methods based on statistical dependence maximization, the choice of statistical dependence measures heavily affects their performance. Moreover, these statistical dependence measures are often unknown and need to be estimated from data. The most well-known and popular statistical dependence measure is the *mutual information* (MI) (Cover and Thomas, 1991). MI is well-studied and many methods were proposed to estimate MI from data (Paninski, 2003; Kraskov et al., 2004; Suzuki et al., 2008; Pál et al., 2010; Gao et al., 2015). A notable method is the *maximum likelihood MI* (MLMI) (Suzuki et al., 2008), which does not require any assumption on the data distribution and can perform model selection via cross-validation. For these reasons, MLMI seems to be an appealing tool for dimension reduction.

duction based on statistical dependence maximization. However, MI is defined based on the *Kullback-Leibler* divergence (Kullback and Leibler, 1951), which is known to be sensitive to outliers (Basu et al., 1998). Hence, MI is not an appropriate tool when it is applied on data containing outliers.

*Quadratic MI* (QMI) is a variant of MI (Principe et al., 2000). Unlike MI, QMI is defined based on the  $L_2$  distance. A notable advantage of the  $L_2$  distance over the KL divergence is that the  $L_2$  distance is more robust against outliers (Basu et al., 1998). Moreover, a computationally efficient method to estimate QMI from data, called *least-squares QMI* (LSQMI) (Sainui and Sugiyama, 2013), was proposed recently. LSQMI does not require any assumption on the data distribution and it can perform model selection via cross-validation. For these reasons, developing a dimension reduction method based on LSQMI is more promising.

An existing approach to use LSQMI for supervised linear dimension reduction is to firstly estimate QMI between projected input and output by LSQMI, then iteratively search for a subspace which maximizes the estimated QMI by a nonlinear optimization method such as gradient ascent. However, the essential quantity of this subspace search is the derivative of QMI w.r.t. the subspace, not QMI itself. Thus, LSQMI may not be an appropriate tool for developing a supervised linear dimension reduction method since the derivative of an accurate QMI estimator is not necessarily an accurate estimator of the derivative of QMI.

To cope with the above problem, in this chapter, we propose a novel method to *directly* estimate the derivative of QMI without estimating QMI itself. The proposed method has the following advantageous properties: it does not require any assumption on the data distribution, the estimator can be computed analytically, and the tuning parameters can be objectively chosen by cross-validation. We show through experiments that the proposed single-step estimator of the derivative of QMI is more accurate than the derivative of the LSQMI estimator. Then, we develop a fixed-point iteration which efficiently uses the proposed estimator to perform supervised linear dimension reduction. Finally, we demonstrate the usefulness of the proposed dimension reduction method through experiments and show that the proposed method is more robust against outliers than existing methods.

The organization of this chapter is as follows. We firstly reintroduce QMI and review existing QMI estimation methods in Section 3.2. Then, we describe the details of our proposed single-step derivative of QMI estimator in Section 3.3. Next, in Section 3.4 we develop a supervised dimension reduction method based on the proposed derivative estimator. The experimental results are given in Section 3.5. A further extension of the proposed derivative estimator is presented in Section 3.6. Finally, the conclusion of this chapter is given in Section 3.7.

## 3.2 Quadratic Mutual Information for Dimension Reduction

In this section, we briefly reintroduce quadratic mutual information and review existing methods to estimate it from data. Then, we discuss the issue of an existing multi-step approach to utilize QMI for dimension reduction.

### 3.2.1 Quadratic Mutual Information

*Quadratic mutual information* (QMI) is a measure of statistical dependence between random variables (Principe et al., 2000), and is defined for two random variables  $\mathbf{z}$  and  $\mathbf{y}$  as

$$\text{QMI}(\mathbf{Z}, \mathbf{Y}) = \frac{1}{2} \iint (p(\mathbf{z}, \mathbf{y}) - p(\mathbf{z})p(\mathbf{y}))^2 d\mathbf{z}d\mathbf{y}. \quad (3.1)$$

$\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is always non-negative and equals to zero if and only if  $\mathbf{z}$  and  $\mathbf{y}$  are statistically independent, i.e.,  $p(\mathbf{z}, \mathbf{y}) = p(\mathbf{z})p(\mathbf{y})$ . Such a property of QMI is similar to that of the ordinary *mutual information* (MI), which is defined as

$$\text{MI}(Z, Y) = \iint p(\mathbf{z}, \mathbf{y}) \log \left( \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})p(\mathbf{y})} \right) d\mathbf{z}d\mathbf{y}. \quad (3.2)$$

The essential difference between QMI and MI is the discrepancy measure.  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is the  $L_2$  distance between  $p(\mathbf{z}, \mathbf{y})$  and  $p(\mathbf{z})p(\mathbf{y})$ , while  $\text{MI}(Z, Y)$  is the *Kullback-Leibler* (KL) divergence (Kullback and Leibler, 1951).

MI has been studied and applied to many data analysis tasks (Cover and Thomas, 1991). Moreover, an efficient method to estimate MI from data is also available (Paninski, 2003; Kraskov et al., 2004; Suzuki et al., 2008; Pál et al., 2010; Gao et al., 2015). However, MI is not always the optimal choice for measuring statistical dependence because it is not robust against outliers. An intuitive explanation is that MI contains the log function and the density ratio: the value of logarithm can be highly sharp near zero, and density ratio can be highly fluctuated and diverge to infinity. Thus, the value of MI tends to be unstable and unreliable in the presence of outliers. In contrast, QMI does not contain the log function and the density ratio, and thus QMI should be more robust against outliers than MI.

Another explanation of the robustness of QMI and MI can be understood based on their discrepancy measures. Both  $L_2$  distance (QMI) and KL divergence (MI) can be regarded as members of a more general divergence class called the *density power divergence* (Basu et al., 1998):

$$\text{DP}_\alpha(p||q) = \int \left( p(\mathbf{x})^{1+\alpha} - \left( 1 + \frac{1}{\alpha} \right) p(\mathbf{x})q(\mathbf{x})^\alpha + \frac{1}{\alpha} q(\mathbf{x})^{1+\alpha} \right) d\mathbf{x}, \quad (3.3)$$

where  $\alpha > 0$ . Based on this divergence class, the  $L_2$  distance and the KL divergence can be obtained by setting  $\alpha = 1$  and  $\alpha \rightarrow 0$ , respectively. As discussed in Basu et al. (1998), the parameter  $\alpha$  controls the robustness against outliers of the divergence, where a large value of  $\alpha$  indicates high robustness. This means that the  $L_2$  distance ( $\alpha = 1$ ) is more robust against outliers than the KL divergence ( $\alpha \rightarrow 0$ ).

In dimension reduction, robustness against outliers is an important requirement because outliers often make dimension reduction methods work poorly. Thus, developing a supervised linear dimension reduction method based on QMI is attractive since QMI is robust against outliers. This QMI-based supervised linear dimension reduction method is performed by finding a matrix  $\mathbf{W}$  which maximizes  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$ :

$$\begin{aligned} & \max_{\mathbf{W} \in \mathbb{R}^{d_{\mathbf{z}} \times d_{\mathbf{x}}}} \text{QMI}(\mathbf{Z}, \mathbf{Y}) \\ & \text{subject to } \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_{\mathbf{z}}}. \end{aligned} \quad (3.4)$$

d The motivation is that, if  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is maximized then  $\mathbf{z}$  and  $\mathbf{y}$  are maximally dependent on each other, and thus we may disregard  $\mathbf{x}$  with a minimal loss of information about  $\mathbf{y}$ .

Since  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is typically unknown, it needs to be estimated from data. Below, we review existing QMI estimation methods and then discuss a weakness of performing supervised linear dimension reduction based on these QMI estimators.

### 3.2.2 Estimation method based on Density Estimation

Expanding Equation (3.1) allows us to express  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  as

$$\text{QMI}(\mathbf{Z}, \mathbf{Y}) = \frac{1}{2} \iint (p(\mathbf{z}, \mathbf{y})^2 - 2p(\mathbf{z}, \mathbf{y})p(\mathbf{z})p(\mathbf{y}) + p(\mathbf{z})^2p(\mathbf{y})^2) d\mathbf{z}d\mathbf{y}. \quad (3.5)$$

A naive approach to estimate  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is to separately estimate the unknown densities  $p(\mathbf{z}, \mathbf{y})$ ,  $p(\mathbf{z})$ , and  $p(\mathbf{y})$  by density estimation methods such as *kernel density estimation* (KDE) (Silverman, 1986), and then plug the estimates into Equation (3.5).

Following this approach, the KDE-based QMI estimator has been studied and applied to many problems such as *feature extraction for classification* (Torkkola, 2003; Principe et al., 2000), *blind source separation* (Principe et al., 2000), and *image registration* (Atif et al., 2003). Although this density estimation based approach was shown to work well, accurately estimating densities for high-dimensional data is known to be one of the most challenging tasks (Vapnik, 1998). Moreover, the densities contained in Equation (3.5) are estimated independently without regarding the accuracy of the QMI estimator. Thus, even if each density is accurately estimated, the QMI estimator obtained from these density estimates does not necessarily give an accurate QMI. An approach to mitigate this problem is to consider density estimators whose combination minimizes the estimation error of QMI. Although this approach shows better performance than the independent density estimation approach, it still performs poorly in high-dimensional problems (Sugiyama et al., 2013).

### 3.2.3 Least-Squares Quadratic Mutual Information

To avoid the separate density estimation, an alternative method called *least-squares QMI* (LSQMI) (Sainui and Sugiyama, 2013) was proposed. Below, we briefly review the LSQMI method.

First, notice that  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  can be expressed in terms of the density difference as

$$\text{QMI}(\mathbf{Z}, \mathbf{Y}) = \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y}, \quad (3.6)$$

where

$$f(\mathbf{z}, \mathbf{y}) = p(\mathbf{z}, \mathbf{y}) - p(\mathbf{z})p(\mathbf{y}). \quad (3.7)$$

The key idea of LSQMI is to directly estimate the density difference  $f(\mathbf{z}, \mathbf{y})$  without going through any density estimation by the procedure of the *least-squares density difference* (Sugiyama et al., 2013). Letting  $d(\mathbf{z}, \mathbf{y})$  be a model of the density difference, LSQMI learns  $d(\mathbf{z}, \mathbf{y})$  so that it is fitted to the true density difference under the squared loss:

$$\frac{1}{2} \iint (d(\mathbf{z}, \mathbf{y}) - f(\mathbf{z}, \mathbf{y}))^2 d\mathbf{z}d\mathbf{y}. \quad (3.8)$$

By expanding the integrand, we obtain

$$\frac{1}{2} \iint d(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} - \iint d(\mathbf{z}, \mathbf{y})f(\mathbf{z}, \mathbf{y})d\mathbf{z}d\mathbf{y} + \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y}. \quad (3.9)$$

Since the last term is a constant w.r.t. the model  $d(\mathbf{z}, \mathbf{y})$ , we omit it and obtain the following criterion:

$$\frac{1}{2} \iint d(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} - \iint d(\mathbf{z}, \mathbf{y})f(\mathbf{z}, \mathbf{y})d\mathbf{z}d\mathbf{y}. \quad (3.10)$$

Then, the density difference estimator  $\hat{d}(\mathbf{z}, \mathbf{y})$  is obtained as the solution of the following minimization problem:

$$\hat{d} = \underset{d}{\operatorname{argmin}} \left[ \frac{1}{2} \iint d(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} - \iint d(\mathbf{z}, \mathbf{y})f(\mathbf{z}, \mathbf{y})d\mathbf{z}d\mathbf{y} \right]. \quad (3.11)$$

The solution of the minimization problem in Equation (3.11) depends on the choice of the model  $d(\mathbf{z}, \mathbf{y})$ . LSQMI employs the following linear-in-parameter model

$$d(\mathbf{z}, \mathbf{y}) = \boldsymbol{\alpha}^\top \boldsymbol{\psi}(\mathbf{z}, \mathbf{y}), \quad (3.12)$$

where  $\boldsymbol{\alpha}$  is a parameter vector and  $\boldsymbol{\psi}(\mathbf{z}, \mathbf{y})$  is a basis function vector. For this model, finding the solution of Equation (3.11) is equivalent to solving

$$\min_{\boldsymbol{\alpha}} \left[ \frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{D} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{q} \right], \quad (3.13)$$

where

$$\mathbf{D} = \iint \boldsymbol{\psi}(\mathbf{z}, \mathbf{y}) \boldsymbol{\psi}(\mathbf{z}, \mathbf{y})^\top d\mathbf{z}d\mathbf{y}, \quad (3.14)$$

$$\begin{aligned} \mathbf{q} &= \iint \boldsymbol{\psi}(\mathbf{z}, \mathbf{y}) f(\mathbf{z}, \mathbf{y}) d\mathbf{z}d\mathbf{y} \\ &= \iint \boldsymbol{\psi}(\mathbf{z}, \mathbf{y}) p(\mathbf{z}, \mathbf{y}) d\mathbf{z}d\mathbf{y} - \iint \boldsymbol{\psi}(\mathbf{z}, \mathbf{y}) p(\mathbf{z}) p(\mathbf{y}) d\mathbf{z}d\mathbf{y}. \end{aligned} \quad (3.15)$$

By approximating the expectation over the densities  $p(\mathbf{z}, \mathbf{y})$ ,  $p(\mathbf{z})$ , and  $p(\mathbf{y})$  with sample averages, we obtain the following empirical minimization problem

$$\min_{\boldsymbol{\alpha}} \left[ \frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{D} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \hat{\mathbf{q}} \right], \quad (3.16)$$

where  $\hat{\mathbf{q}}$  is the sample approximation of Equation (3.15):

$$\hat{\mathbf{q}} = \frac{1}{N} \sum_{i=1}^N \boldsymbol{\psi}(\mathbf{z}_i, \mathbf{y}_i) - \frac{1}{N^2} \sum_{i,j=1}^N \boldsymbol{\psi}(\mathbf{z}_i, \mathbf{y}_j). \quad (3.17)$$

By including the  $\ell_2$  regularization term, we obtain

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \left[ \frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{D} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \hat{\mathbf{q}} + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \boldsymbol{\alpha} \right], \quad (3.18)$$

where  $\lambda \geq 0$  is the regularization parameter. Then, the solution is obtained analytically as

$$\hat{\alpha} = (\mathbf{D} + \lambda \mathbf{I})^{-1} \hat{\mathbf{q}}. \quad (3.19)$$

Therefore, the density difference estimator is obtained as

$$\hat{d}(\mathbf{z}, \mathbf{y}) = \hat{\alpha}^\top \psi(\mathbf{z}, \mathbf{y}). \quad (3.20)$$

Finally, QMI estimator is obtained by substituting the density difference estimator into Equation (3.6). A direct substitution yields two possible QMI estimators:

$$\widehat{\text{QMI}}(\mathbf{Z}, \mathbf{Y}) = \frac{1}{2} \hat{\alpha}^\top \hat{\mathbf{q}}, \quad (3.21)$$

$$\widehat{\text{QMI}}(\mathbf{Z}, \mathbf{Y}) = \frac{1}{2} \hat{\alpha}^\top \mathbf{D} \hat{\alpha}. \quad (3.22)$$

However, it was shown in Sugiyama et al. (2013) that a linear combination of the two estimators defined as

$$\widehat{\text{QMI}}(\mathbf{Z}, \mathbf{Y}) = \hat{\alpha}^\top \hat{\mathbf{q}} - \frac{1}{2} \hat{\alpha}^\top \mathbf{D} \hat{\alpha}, \quad (3.23)$$

provides smaller bias and is a more appropriate QMI estimator.

As shown above, LSQMI avoids multi-step density estimation by directly estimating the density difference contained in QMI. It was shown that such direct estimation procedure tends to be more accurate than multi-step estimation (Sugiyama et al., 2013). Moreover, LSQMI is able to objectively choose the tuning parameter contained in the basis function  $\psi(\mathbf{z}, \mathbf{y})$  and the regularization parameter  $\lambda$  based on cross-validation. This property allows LSQMI to solve challenging tasks such as *clustering* (Sainui and Sugiyama, 2013) and *unsupervised linear dimension reduction* (Sainui and Sugiyama, 2014) in an objective way.

### 3.2.4 Multi-step Supervised Linear Dimension Reduction

Given an efficient QMI estimation method such as LSQMI, supervised linear dimension reduction can be performed by solving the following optimization problem:

$$\begin{aligned} & \max_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \widehat{\text{QMI}}(\mathbf{W}\mathbf{X}, \mathbf{Y}) \\ & \text{subject to } \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}, \end{aligned} \quad (3.24)$$

where  $\widehat{\text{QMI}}(\mathbf{W}\mathbf{X}, \mathbf{Y})$  is defined as an QMI estimator learned from data  $\{(\mathbf{W}\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ . A straightforward approach to find a local maximizer in Equation (3.24) is to iteratively update a matrix  $\mathbf{W}$  by gradient ascent:

$$\mathbf{W} \leftarrow \mathbf{W} + t \frac{\partial \widehat{\text{QMI}}(\mathbf{W}\mathbf{X}, \mathbf{Y})}{\partial \mathbf{W}}, \quad (3.25)$$

where  $t > 0$  denotes the step size. This update rule means that the essential point of the QMI-based supervised linear dimension reduction method is not the accuracy of the QMI estimator, but the accuracy of the estimator of the derivative of the QMI. Thus, the existing multi-step approach which first estimates QMI and then compute the derivatives of the QMI estimator is not necessarily appropriate since an accurate estimator of QMI does not necessarily mean that its derivative is an accurate estimator of the derivative of QMI. Next, we describe our proposed single-step estimation method which overcomes this issue.

### 3.3 Derivative of Quadratic Mutual Information

To cope with the weakness of the QMI estimation methods when performing supervised linear dimension reduction, we propose to *directly* estimate the derivative of QMI in a single-step manner without estimating QMI itself.

#### 3.3.1 Single-step Estimation of the Derivative of Quadratic Mutual Information

Let  $\mathbf{z} = \mathbf{W}\mathbf{x}$ , then from Equation (3.6), the derivative of the  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  w.r.t. the  $(\ell, \ell')$ -th element of  $\mathbf{W}$  can be expressed by<sup>1</sup>

$$\begin{aligned}
\frac{\partial \text{QMI}(\mathbf{W})}{\partial W_{\ell, \ell'}} &= \frac{\partial}{\partial W_{\ell, \ell'}} \left( \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y})^2 d\mathbf{z} d\mathbf{y} \right) \\
&= \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial W_{\ell, \ell'}} d\mathbf{z} d\mathbf{y} \\
&= \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial f(\mathbf{z}, \mathbf{y})^\top}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial W_{\ell, \ell'}} d\mathbf{z} d\mathbf{y} \\
&= \iint p(\mathbf{z}, \mathbf{y}) \frac{\partial f(\mathbf{z}, \mathbf{y})^\top}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial W_{\ell, \ell'}} d\mathbf{z} d\mathbf{y} \\
&\quad - \iint p(\mathbf{z}) p(\mathbf{y}) \frac{\partial f(\mathbf{z}, \mathbf{y})^\top}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial W_{\ell, \ell'}} d\mathbf{z} d\mathbf{y}, \tag{3.26}
\end{aligned}$$

where in the second line we assume that the order of the derivative and the integration is interchangeable. By approximating the expectations over the densities  $p(\mathbf{z}, \mathbf{y})$ ,  $p(\mathbf{z})$ , and  $p(\mathbf{y})$  with sample averages, we obtain an approximation of the derivative of QMI as

$$\frac{\widehat{\partial \text{QMI}(\mathbf{W})}}{\partial W_{\ell, \ell'}} = \frac{1}{N} \sum_{i=1}^N \frac{\partial f(\mathbf{z}_i, \mathbf{y}_i)^\top}{\partial \mathbf{z}} \frac{\partial \mathbf{z}_i}{\partial W_{\ell, \ell'}} - \frac{1}{N^2} \sum_{i,j=1}^N \frac{\partial f(\mathbf{z}_i, \mathbf{y}_j)^\top}{\partial \mathbf{z}} \frac{\partial \mathbf{z}_i}{\partial W_{\ell, \ell'}}. \tag{3.27}$$

Note that since  $\mathbf{z}^{(\ell)} = \sum_{\ell'=1}^{d_{\mathbf{x}}} W_{\ell, \ell'} x^{(\ell')}$ , we have that  $\frac{\partial \mathbf{z}}{\partial W_{\ell, \ell'}}$  is the  $d_{\mathbf{z}}$ -dimensional vector with zero everywhere except at the  $\ell$ -th dimension which has value  $x^{(\ell')}$ . Hence, Equation (3.27) can be simplified as

$$\frac{\widehat{\partial \text{QMI}(\mathbf{W})}}{\partial W_{\ell, \ell'}} = \frac{1}{N} \sum_{i=1}^N \frac{\partial f(\mathbf{z}_i, \mathbf{y}_i)}{\partial z^{(\ell)}} x_i^{(\ell')} - \frac{1}{N^2} \sum_{i,j=1}^N \frac{\partial f(\mathbf{z}_i, \mathbf{y}_j)}{\partial z^{(\ell)}} x_i^{(\ell')}. \tag{3.28}$$

This means that the derivative of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  w.r.t.  $\mathbf{W}$  can be obtained once we know the derivatives of the density difference w.r.t.  $z^{(\ell)}$  for all  $\ell \in \{1, \dots, d_{\mathbf{z}}\}$ . However, these derivatives are often unknown and need to be estimated from data. Below, we first discuss existing approaches and their drawbacks. Then we propose our approach which can overcome the drawbacks.

---

<sup>1</sup>Throughout this section, we use  $\text{QMI}(\mathbf{W})$  instead of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  when we consider its derivative for notational convenience. However, they still represent the QMI between random variables  $\mathbf{z} = \mathbf{W}\mathbf{x}$  and  $\mathbf{y}$ .

### 3.3.2 Existing Approaches to Estimate the Derivative of the Density Difference

Our current goal is to obtain the derivative of the density difference w.r.t.  $z^{(\ell)}$  which can be rewritten as

$$\frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} = \frac{\partial p(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} - \frac{\partial p(\mathbf{z})}{\partial z^{(\ell)}} p(\mathbf{y}). \quad (3.29)$$

All terms in Equation (3.29) are unknown in practice and need to be estimated from data. There are three existing approaches to estimate them.

#### (A) Density estimation

Separately estimate the densities  $p(\mathbf{z}, \mathbf{y})$ ,  $p(\mathbf{z})$ , and  $p(\mathbf{y})$  by, e.g., *kernel density estimation*. Then estimate the right-hand side of Equation (3.29) as

$$\frac{\partial \widehat{p}(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} - \frac{\partial \widehat{p}(\mathbf{z})}{\partial z^{(\ell)}} \widehat{p}(\mathbf{y}), \quad (3.30)$$

where  $\widehat{p}(\mathbf{z}, \mathbf{y})$ ,  $\widehat{p}(\mathbf{z})$ , and  $\widehat{p}(\mathbf{y})$  denote the estimated densities.

#### (B) Density derivative estimation

Estimate the density  $p(\mathbf{y})$  by e.g., kernel density estimation. Next, separately estimate the densities derivative  $\frac{\partial p(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}}$  and  $\frac{\partial p(\mathbf{z})}{\partial z^{(\ell)}}$  by e.g., the method of *mean integrated square error for derivatives* (Sasaki et al., 2015a), which can estimate the density derivative without estimating the density itself. Then estimate the right-hand side of Equation (3.29) as

$$\widehat{\frac{\partial p(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}}} - \widehat{\frac{\partial p(\mathbf{z})}{\partial z^{(\ell)}}} \widehat{p}(\mathbf{y}), \quad (3.31)$$

where  $\widehat{p}(\mathbf{y})$  denotes the estimated density, and  $\widehat{\frac{\partial p(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}}}$  and  $\widehat{\frac{\partial p(\mathbf{z})}{\partial z^{(\ell)}}}$  denote the (directly) estimated density derivatives.

#### (C) Density difference estimation

Estimate the density difference  $f(\mathbf{z}, \mathbf{y})$  by e.g., *least-squares density difference* (Sugiyama et al., 2013), which can estimate the density difference without estimating the densities themselves. Then estimate the left-hand side of Equation (3.29) as

$$\widehat{\frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}}}, \quad (3.32)$$

where  $\widehat{f}(\mathbf{z}, \mathbf{y})$  denotes the (directly) estimated density difference.

The problem of approaches (A) and (B) is that they involve multiple estimation steps where some quantities are estimated first and then they are plugged into Equation (3.29). Such multiple-step methods are not appropriate since each estimated quantity is obtained without regarding the others and the succeeding plug-in step using these estimates can magnify the estimation error contained in each estimated quantity.

On the other hand, approach (C) seems more promising than the previous two approaches since there is only one estimated quantity  $f(\mathbf{z}, \mathbf{y})$ . However, it is still not the optimal approach due to the fact that an accurate estimator of the density difference does not necessarily mean that its derivative is an accurate estimator of the derivative of the density difference.

To avoid the above drawbacks, we propose a new approach which directly estimates the derivative of the density difference.

### 3.3.3 Direct Estimation of the Derivative of the Density Difference

We propose to estimate the derivative of the density difference w.r.t.  $z^{(\ell)}$  using a model  $g_\ell(\mathbf{z}, \mathbf{y})$ :

$$\frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} \approx g_\ell(\mathbf{z}, \mathbf{y}). \quad (3.33)$$

The model  $g_\ell(\mathbf{z}, \mathbf{y})$  is learned so that it is fitted to its corresponding derivative under the squared error:

$$\frac{1}{2} \iint \left( g_\ell(\mathbf{z}, \mathbf{y}) - \frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} \right)^2 d\mathbf{z}d\mathbf{y}. \quad (3.34)$$

By expanding the square, we obtain

$$\frac{1}{2} \iint g_\ell(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} - \iint g_\ell(\mathbf{z}, \mathbf{y}) \frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z}d\mathbf{y} + \frac{1}{2} \iint \left( \frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} \right)^2 d\mathbf{z}d\mathbf{y}. \quad (3.35)$$

Since the last term is a constant w.r.t. the model  $g_\ell(\mathbf{z}, \mathbf{y})$ , we omit it and obtain the following criterion:

$$\frac{1}{2} \iint g_\ell(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} - \iint g_\ell(\mathbf{z}, \mathbf{y}) \frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z}d\mathbf{y}. \quad (3.36)$$

The second term is intractable due to the unknown derivative of the density difference. To make this term tractable, we use *integration by parts* (Kasube, 1983) to obtain the following:

$$\begin{aligned} & \iint [g_\ell(\mathbf{z}, \mathbf{y}) f(\mathbf{z}, \mathbf{y})]_{z^{(\ell)}=-\infty}^{z^{(\ell)}=\infty} d\mathbf{z}_{\setminus z^{(\ell)}} d\mathbf{y} \\ &= \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial g_\ell(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z}d\mathbf{y} + \iint g_\ell(\mathbf{z}, \mathbf{y}) \frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z}d\mathbf{y}, \end{aligned} \quad (3.37)$$

where  $\int \cdot d\mathbf{z}_{\setminus z^{(\ell)}}$  denotes an integration over  $\mathbf{z}$  except for the  $\ell$ -th element. Here, we require

$$[g_\ell(\mathbf{z}, \mathbf{y}) f(\mathbf{z}, \mathbf{y})]_{z^{(\ell)}=-\infty}^{z^{(\ell)}=\infty} = 0, \quad (3.38)$$

which is a mild assumption since the tails of the density difference  $p(\mathbf{z}, \mathbf{y}) - p(\mathbf{z})p(\mathbf{y})$  often vanish when  $z^{(\ell)}$  approaches infinity. Applying the assumption to the left-hand side of Equation (3.37) allows us to express Equation (3.36) as

$$\frac{1}{2} \iint g_\ell(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} + \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial g_\ell(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z}d\mathbf{y}. \quad (3.39)$$

Then, the estimator  $\hat{g}_\ell(\mathbf{z}, \mathbf{y})$  is obtained as a solution of the following minimization problem:

$$\hat{g}_\ell = \underset{g_\ell}{\operatorname{argmin}} \left[ \frac{1}{2} \iint g_\ell(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} + \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial g_\ell(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z}d\mathbf{y} \right]. \quad (3.40)$$

The solution of Equation (3.40) depends on the choice of the model. Let us employ the following linear-in-parameter model as  $g_\ell(\mathbf{z}, \mathbf{y})$ :

$$g_\ell(\mathbf{z}, \mathbf{y}) = \boldsymbol{\theta}_\ell^\top \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y}), \quad (3.41)$$

where  $\boldsymbol{\theta}_\ell$  is a parameter vector and  $\boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y})$  is a basis function vector whose practical choice will be discussed later in detail. For this model, finding the solution of Equation (3.40) is equivalent to solving

$$\min_{\boldsymbol{\theta}_\ell} \left[ \frac{1}{2} \boldsymbol{\theta}_\ell^\top \mathbf{H}_\ell \boldsymbol{\theta}_\ell + \boldsymbol{\theta}_\ell^\top \mathbf{h}_\ell \right], \quad (3.42)$$

where we define

$$\mathbf{H}_\ell = \iint \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y}) \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y})^\top d\mathbf{z} d\mathbf{y}, \quad (3.43)$$

$$\begin{aligned} \mathbf{h}_\ell &= \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z} d\mathbf{y} \\ &= \iint p(\mathbf{z}, \mathbf{y}) \frac{\partial \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z} d\mathbf{y} - \iint p(\mathbf{z}) p(\mathbf{y}) \frac{\partial \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} d\mathbf{z} d\mathbf{y}. \end{aligned} \quad (3.44)$$

By approximating the expectation over the densities  $p(\mathbf{z}, \mathbf{y})$ ,  $p(\mathbf{z})$ , and  $p(\mathbf{y})$  with sample averages, we obtain the following empirical minimization problem:

$$\min_{\boldsymbol{\theta}_\ell} \left[ \frac{1}{2} \boldsymbol{\theta}_\ell^\top \mathbf{H}_\ell \boldsymbol{\theta}_\ell + \boldsymbol{\theta}_\ell^\top \widehat{\mathbf{h}}_\ell \right], \quad (3.45)$$

where  $\widehat{\mathbf{h}}_\ell$  is the sample approximation of Equation (3.44):

$$\widehat{\mathbf{h}}_\ell = \frac{1}{N} \sum_{i=1}^N \frac{\partial \boldsymbol{\varphi}_\ell(\mathbf{z}_i, \mathbf{y}_i)}{\partial z^{(\ell)}} - \frac{1}{N^2} \sum_{i,j=1}^N \frac{\partial \boldsymbol{\varphi}_\ell(\mathbf{z}_i, \mathbf{y}_j)}{\partial z^{(\ell)}}. \quad (3.46)$$

By including the  $\ell_2$  regularization term to control the model complexity, we obtain

$$\widehat{\boldsymbol{\theta}}_\ell = \operatorname{argmin}_{\boldsymbol{\theta}_\ell} \left[ \frac{1}{2} \boldsymbol{\theta}_\ell^\top \mathbf{H}_\ell \boldsymbol{\theta}_\ell + \boldsymbol{\theta}_\ell^\top \widehat{\mathbf{h}}_\ell + \frac{\lambda_\ell}{2} \boldsymbol{\theta}_\ell^\top \boldsymbol{\theta}_\ell \right], \quad (3.47)$$

where  $\lambda_\ell \geq 0$  denotes the regularization parameter. This minimization problem is convex w.r.t. the parameter  $\boldsymbol{\theta}_\ell$ , and the solution can be obtained analytically as

$$\widehat{\boldsymbol{\theta}}_\ell = -(\mathbf{H}_\ell + \lambda_\ell \mathbf{I})^{-1} \widehat{\mathbf{h}}_\ell, \quad (3.48)$$

where  $\mathbf{I}$  denotes the identity matrix. Finally, the estimator of the derivative of the density difference is obtained by substituting the solution into the model Equation (3.41) as

$$\widehat{g}_\ell(\mathbf{z}, \mathbf{y}) = \widehat{\boldsymbol{\theta}}_\ell^\top \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y}). \quad (3.49)$$

Using this solution, an estimator of the derivative of QMI can be directly obtained by substituting Equation (3.49) into Equation (3.28) as

$$\frac{\partial \widehat{\text{QMI}}(\mathbf{W})}{\partial W_{\ell, \ell'}} = \frac{1}{N} \sum_{i=1}^N \widehat{\boldsymbol{\theta}}_\ell^\top \boldsymbol{\varphi}_\ell(\mathbf{z}_i, \mathbf{y}_i) x_i^{(\ell')} - \frac{1}{N^2} \sum_{i,j=1}^N \widehat{\boldsymbol{\theta}}_\ell^\top \boldsymbol{\varphi}_\ell(\mathbf{z}_i, \mathbf{y}_j) x_i^{(\ell')}. \quad (3.50)$$

We call this method the *least-squares QMI derivative* (LSQMID).

### 3.3.4 Basis Function Design

As basis function  $\varphi_\ell(\mathbf{z}, \mathbf{y})$ , we propose to use

$$\varphi_\ell(\mathbf{z}, \mathbf{y}) = \left[ \varphi_\ell^{(1)}(\mathbf{z}, \mathbf{y}), \dots, \varphi_\ell^{(b)}(\mathbf{z}, \mathbf{y}) \right]^\top, \quad (3.51)$$

where  $b \leq N$ . First, let us define the  $k$ -th Gaussian function as

$$\phi_\ell^{(k)}(\mathbf{z}, \mathbf{y}) = \exp \left( -\frac{\|\mathbf{z} - \mathbf{u}_k\|^2 + \|\mathbf{y} - \mathbf{v}_k\|^2}{2\sigma_\ell^2} \right), \quad (3.52)$$

where  $\mathbf{u}_k$  and  $\mathbf{v}_k$  denote Gaussian centers chosen randomly from the data  $\{(\mathbf{z}_i, \mathbf{y}_i)\}_{i=1}^N$ , and  $\sigma_\ell$  denotes the Gaussian width. We may use different Gaussian widths for  $\mathbf{z}$  and  $\mathbf{y}$ , but this approach significantly increases the computation time for model selection which will be discussed in Section 3.3.5. In our implementation, we standardize each dimension of  $\mathbf{x}$  and  $\mathbf{y}$  to have unit variance and zero mean, and then use the common Gaussian width for both  $\mathbf{z}$  and  $\mathbf{y}$ . We also set  $b = \min(N, 200)$  in the experiments.

Based on the above Gaussian function, we propose to use the following function as the  $k$ -th basis for the  $\ell$ -th model of the derivative of the density difference:

$$\begin{aligned} \varphi_\ell^{(k)}(\mathbf{z}, \mathbf{y}) &= \frac{\partial \phi_\ell^{(k)}(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} \\ &= -\frac{1}{\sigma_\ell^2} (z^{(\ell)} - u_k^{(\ell)}) \phi_\ell^{(k)}(\mathbf{z}, \mathbf{y}). \end{aligned} \quad (3.53)$$

This function is the derivative of the  $k$ -th Gaussian basis function w.r.t.  $z^{(\ell)}$ . A benefit of this basis function design is that the integral appeared in  $\mathbf{H}_\ell$  can be computed analytically. Through a simple calculation, we obtain the  $(k, k')$ -th element of  $\mathbf{H}_\ell$  as follows:

$$H_\ell^{(k, k')} = \frac{1}{\sigma_\ell^4} (\sqrt{\pi} \sigma_\ell)^{d_{\mathbf{z}} + d_{\mathbf{y}}} \exp \left( -\frac{\|\mathbf{u}_k - \mathbf{u}_{k'}\|^2 - \|\mathbf{v}_k - \mathbf{v}_{k'}\|^2}{4\sigma_\ell^2} \right) \quad (3.54)$$

$$\times \left( u_k^{(\ell)} u_{k'}^{(\ell)} - \frac{(u_k^{(\ell)} + u_{k'}^{(\ell)})^2}{2} + \left( \frac{u_k^{(\ell)} + u_{k'}^{(\ell)}}{2} \right)^2 + \frac{\sigma_\ell^2}{2} \right). \quad (3.55)$$

As will be discussed in Section 3.4, this basis function choice has further benefits when we develop a linear supervised dimension reduction method.

### 3.3.5 Model Selection by Cross-Validation

The practical performance of the LSQMID method depends on the choice of the Gaussian width  $\sigma_\ell$  and the regularization parameter  $\lambda_\ell$  included in the estimator  $\hat{g}_\ell(\mathbf{z}, \mathbf{y})$ . These tuning parameters can be objectively chosen by the  $K$ -fold cross-validation (CV) procedure which is described below.

1. Divide the training data  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$  into  $K$  disjoint subsets  $\{\mathcal{D}_j\}_{j=1}^K$  with approximately the same size. In the experiments, we choose  $K = 5$ .
2. For each candidate  $M = (\tilde{\sigma}_\ell, \tilde{\lambda}_\ell)$  and each subset  $\mathcal{D}_j$ , compute a solution  $\hat{\boldsymbol{\theta}}_{\ell, M, \setminus j}$  by Equation (3.48) with the candidate  $M$  and data from  $\mathcal{D} \setminus \mathcal{D}_j$  (i.e., all data points except data points in  $\mathcal{D}_j$ ).

3. Compute the CV score of each candidate pair  $M$  by

$$\text{CV}_\ell(M) = \frac{1}{K} \sum_{j=1}^K \left[ \frac{1}{2} \widehat{\boldsymbol{\theta}}_{\ell, M, \setminus j}^\top \mathbf{H}_{\ell, M} \widehat{\boldsymbol{\theta}}_{\ell, M, \setminus j} + \widehat{\boldsymbol{\theta}}_{\ell, M, \setminus j}^\top \widehat{\mathbf{h}}_{\ell, M, j} \right], \quad (3.56)$$

where  $\widehat{\mathbf{h}}_{\ell, M, j}$  denotes  $\widehat{\mathbf{h}}_\ell$  computed from the candidate  $M$  and data in  $\mathcal{D}_j$ .

4. Choose the tuning parameter pair such that it minimizes the CV score as

$$(\widehat{\sigma}_\ell, \widehat{\lambda}_\ell) = \underset{M}{\operatorname{argmin}} \text{CV}_\ell(M). \quad (3.57)$$

### 3.4 Supervised Linear Dimension Reduction via Derivative Estimator

In this section, we propose a linear supervised dimension reduction method based on the proposed LSQMID estimator.

#### 3.4.1 Gradient Ascent

Recall that our goal in supervised dimension reduction is to find the matrix  $\mathbf{W}^*$ :

$$\begin{aligned} & \max_{\mathbf{W} \in \mathbb{R}^{d_{\mathbf{z}} \times d_{\mathbf{x}}}} \text{QMI}(\mathbf{Z}, \mathbf{Y}) \\ & \text{subject to} \quad \mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_{\mathbf{z}}}. \end{aligned} \quad (3.58)$$

A straightforward approach to find a solution of Equation (3.58) using the proposed method is to perform gradient ascent as

$$\mathbf{W} \leftarrow \mathbf{W} + t \frac{\partial \widehat{\text{QMI}}(\mathbf{W})}{\partial \mathbf{W}}, \quad (3.59)$$

where  $t > 0$  denotes the step size. It is known that choosing a good step size is a difficult task in practice (Nocedal and Wright, 2006). *Line search* is an algorithm to choose a good step size by finding a step size which satisfies certain conditions such as the *Armijo rule* (Armijo, 1966). However, these conditions often require access to the objective value  $\text{QMI}(\mathbf{W})$  which is unavailable in our current setup since the QMI derivative is directly estimated without estimating QMI. Thus, if we want to perform line search, QMI needs to be estimated separately. However, this is problematic since the estimation of the derivative of the QMI and the estimation of the QMI are performed independently without regard to the other, and thus they may not be consistent. For example, the gradient  $\frac{\partial \widehat{\text{QMI}}(\mathbf{W})}{\partial \mathbf{W}}$ , which is supposed to be an ascent direction, may be regarded as a descent direction on the surface of the estimated QMI. For such a case, the step size chosen by any line search algorithm is unreliable and the resulting  $\mathbf{W}$  may not be a good solution.

Below, we consider two approaches which can cope with this problem.

### 3.4.2 Quadratic Mutual Information Approximation via Derivative Estimator

To avoid separate QMI estimation, we consider an approximated QMI which is obtained as a by-product of the proposed method. Recall that the proposed method models the derivative of the density difference as

$$\begin{aligned}
\frac{\partial f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} &\approx g_\ell(\mathbf{z}, \mathbf{y}) \\
&= \boldsymbol{\theta}_\ell^\top \boldsymbol{\varphi}_\ell(\mathbf{z}, \mathbf{y}) \\
&= \boldsymbol{\theta}_\ell^\top \frac{\partial \boldsymbol{\phi}_\ell(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell)}} \\
&= \frac{\partial (\boldsymbol{\theta}_\ell^\top \boldsymbol{\phi}_\ell(\mathbf{z}, \mathbf{y}))}{\partial z^{(\ell)}}.
\end{aligned} \tag{3.60}$$

This means that the density difference can be approximated by

$$\tilde{f}_\ell(\mathbf{z}, \mathbf{y}) = \hat{\boldsymbol{\theta}}_\ell^\top \boldsymbol{\phi}_\ell(\mathbf{z}, \mathbf{y}) + c_\ell, \tag{3.61}$$

where  $c_\ell$  is an unknown quantity which is a constant w.r.t.  $z^{(\ell)}$ .

In a special case where  $d_z = 1$ , we can use Equation (3.61) to obtain a proper approximator of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  in a similar fashion to the LSQMI method. To verify this, let us substitute Equation (3.61) into one of the  $f(z, \mathbf{y})$  in Equation (3.6) to obtain

$$\begin{aligned}
\widetilde{\text{QMI}}(\mathbf{Z}, \mathbf{Y}) &= \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y}) \tilde{f}(\mathbf{z}, \mathbf{y}) d\mathbf{z} d\mathbf{y} \\
&= \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y}) (\hat{\boldsymbol{\theta}}^\top \boldsymbol{\phi}(\mathbf{z}, \mathbf{y}) + c) d\mathbf{z} d\mathbf{y} \\
&= \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y}) \hat{\boldsymbol{\theta}}^\top \boldsymbol{\phi}(\mathbf{z}, \mathbf{y}) d\mathbf{z} d\mathbf{y} + \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y}) c d\mathbf{z} d\mathbf{y} \\
&= \frac{1}{2} \iint f(\mathbf{z}, \mathbf{y}) \hat{\boldsymbol{\theta}}^\top \boldsymbol{\phi}(\mathbf{z}, \mathbf{y}) d\mathbf{z} d\mathbf{y},
\end{aligned} \tag{3.62}$$

where the last line follows from

$$\iint f(\mathbf{z}, \mathbf{y}) c d\mathbf{z} d\mathbf{y} = \iint p(\mathbf{z}, \mathbf{y}) c d\mathbf{z} d\mathbf{y} - \iint p(\mathbf{z}) p(\mathbf{y}) c d\mathbf{z} d\mathbf{y} \tag{3.63}$$

$$= 0. \tag{3.64}$$

By approximating the expectation with sample averages, we obtain a QMI approximator as

$$\widetilde{\text{QMI}}(\mathbf{Z}, \mathbf{Y}) = \frac{1}{2N} \sum_{i=1}^N \hat{\boldsymbol{\theta}}^\top \boldsymbol{\phi}(z_i, \mathbf{y}_i) - \frac{1}{2N^2} \sum_{i,j=1}^N \hat{\boldsymbol{\theta}}^\top \boldsymbol{\phi}(z_i, \mathbf{y}_j). \tag{3.65}$$

The main advantage of using  $\widetilde{\text{QMI}}(\mathbf{Z}, \mathbf{Y})$  is that it is obtained from the derivative estimation, and thus should be consistent with the estimated derivative. This allows us to perform line search for the gradient ascent in a consistent manner. However, such an approximation is unavailable when  $d_z > 1$ . Next, we consider an alternative optimization strategy which does not require an access to the QMI value.

### 3.4.3 Fixed-Point Iteration

To avoid the problem of choosing the step size which requires an access to the QMI value, we propose to use a fixed-point iteration for finding a solution of Equation (3.58). Note that from the first order optimality condition, a solution  $\mathbf{W}^*$  is a stationary point which satisfies

$$\frac{\partial \text{QMI}(\mathbf{W}^*)}{\partial \mathbf{W}} = \mathbf{0}_{d_{\mathbf{z}}, d_{\mathbf{x}}}, \quad (3.66)$$

where  $\mathbf{0}_{d_{\mathbf{z}}, d_{\mathbf{x}}}$  denotes  $d_{\mathbf{z}}$ -by- $d_{\mathbf{x}}$  zero matrix. By using the proposed basis function in Equation (3.53), Equation (3.50) can be expressed as

$$\frac{\partial \widehat{\text{QMI}}(\mathbf{W})}{\partial W_{\ell, \ell'}} = \frac{1}{\sigma_{\ell}^2} \left( F_1^{(\ell, \ell')} - F_2^{(\ell, \ell')} - W_{\ell, \ell'} F_3^{(\ell, \ell')} \right), \quad (3.67)$$

where we define

$$\begin{aligned} F_1^{(\ell, \ell')} &= \widehat{\boldsymbol{\theta}}_{\ell}^{\top} \left( \mathbf{u}^{(\ell)} \odot \left( \frac{1}{N} \sum_{i=1}^N \phi_{\ell}(\mathbf{z}_i, \mathbf{y}_i) x_i^{(\ell')} - \frac{1}{N^2} \sum_{i,j=1}^N \phi_{\ell}(\mathbf{z}_i, \mathbf{y}_j) x_i^{(\ell')} \right) \right), \\ F_2^{(\ell, \ell')} &= \sum_{m \neq \ell'}^{d_{\mathbf{x}}} W_{\ell, m} \widehat{\boldsymbol{\theta}}_{\ell}^{\top} \left( \frac{1}{N} \sum_{i=1}^N \phi_{\ell}(\mathbf{z}_i, \mathbf{y}_i) x_i^{(m)} x_i^{(\ell')} - \frac{1}{N^2} \sum_{i,j=1}^N \phi_{\ell}(\mathbf{z}_i, \mathbf{y}_j) x_i^{(m)} x_i^{(\ell')} \right), \\ F_3^{(\ell, \ell')} &= \widehat{\boldsymbol{\theta}}_{\ell}^{\top} \left( \frac{1}{N} \sum_{i=1}^N \phi_{\ell}(\mathbf{z}_i, \mathbf{y}_i) (x_i^{(\ell')})^2 - \frac{1}{N^2} \sum_{i,j=1}^N \phi_{\ell}(\mathbf{z}_i, \mathbf{y}_j) (x_i^{(\ell')})^2 \right), \end{aligned} \quad (3.68)$$

with  $\mathbf{u}^{(\ell)}$  be the column vector of length  $b$  consisting of the  $\ell$ -th dimension over all  $\mathbf{u}_k$  and the symbol  $\odot$  represents the element-wise vector product. Then, an approximated solution may be obtained by finding  $W_{\ell, \ell'}$  for all  $(\ell, \ell')$  such that the left-hand side of Equation (3.67) is zero. This optimization strategy results in a fixed-point iteration for each dimension of  $\mathbf{W}$ :

$$W_{\ell, \ell'} \leftarrow \frac{F_1^{(\ell, \ell')} - F_2^{(\ell, \ell')}}{F_3^{(\ell, \ell')}}. \quad (3.69)$$

Finally, we orthonormalize the solution after each iteration as

$$\mathbf{W} \leftarrow (\mathbf{W} \mathbf{W}^{\top})^{-\frac{1}{2}} \mathbf{W}. \quad (3.70)$$

In practice, we perform this orthonormalization only every several iterations for computational efficiency.

There is a relation between the fixed point iteration and gradient method. By substituting Equation (3.67) into Equation (3.69), we obtain

$$W_{\ell, \ell'} \leftarrow W_{\ell, \ell'} + \frac{\sigma_{\ell}^2}{F_3^{(\ell, \ell')}} \frac{\partial \widehat{\text{QMI}}(\mathbf{W})}{\partial W_{\ell, \ell'}}. \quad (3.71)$$

This means that the fixed point update step is a gradient method with an adaptive stepsize  $1/F_3^{(\ell, \ell')}$ . Thus, if  $F_3^{(\ell, \ell')}$  is always positive, then the fixed point iteration will converge to a local maxima. However, unfortunately there is no guarantee that

$F_3^{(\ell, \ell')}$  is always positive in our formulation. Indeed, in our numerical experiments,  $F_3^{(\ell, \ell')}$  sometimes took a negative value. A heuristic remedy would be to update the matrix  $\mathbf{W}$  only when  $F_3^{(\ell, \ell')}$  is positive. However, this approach did not work well in our preliminary experiments, so we decided not to modify anything. We will further investigate this issue in the future work.

The optimization problem in Equation (3.58) is non-convex and may have saddle points and local solutions. To avoid obtaining a saddle point or a poor local solution, we perform the optimization starting from several initial guesses and choose the solution which gives the maximum estimated QMI as the final solution.

## 3.5 Experiment

In this section, we demonstrate the usefulness of the proposed method through experiments.

### 3.5.1 Illustrative Experiment

Firstly, we perform the following experiment to illustrate the behavior of the proposed method in terms of the QMI derivative estimation. Let  $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$  denote the Gaussian distribution with mean  $\boldsymbol{\mu}$  and covariance  $\boldsymbol{\Sigma}$ . Then, for  $\epsilon \sim \mathcal{N}(0, 0.15^2)$ , we generate a dataset  $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$  and a matrix  $\mathbf{W}$  as follows:

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\mathbf{0}_2, \mathbf{I}_2), \\ y &= (x^{(1)})^2 + \epsilon, \\ \mathbf{W} &= [\cos \theta \quad \sin \theta], \end{aligned}$$

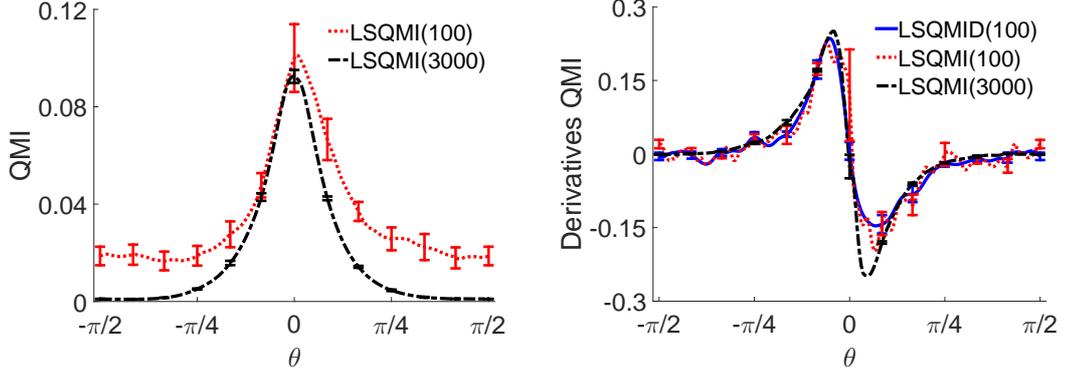
where  $\mathbf{0}_2$  denotes the zero vector of length 2. Thus we have  $z = x^{(1)} \cos \theta + x^{(2)} \sin \theta$ . The goal is to estimate

$$\frac{\partial \text{QMI}(\mathbf{Z}, \mathbf{Y})}{\partial \theta} = \frac{\partial \text{QMI}(\mathbf{Z}, \mathbf{Y})}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial \theta} \quad (3.72)$$

at different value of  $\theta$ . Note that  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is maximized at  $\theta = 0$ , i.e.,  $\mathbf{W} = [1 \quad 0]$ .

Figure 3.1(a) shows the averaged value over 20 trials of the estimated  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  by LSQMI. The vertical axis indicates the value of the estimated QMI and the horizontal axis indicates the value of  $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ . We use  $N = 3000$  and  $N = 100$  for estimating QMI and denote the results by LSQMI(3000) and LSQMI(100), respectively. We perform cross validation at  $\theta = 0$  and use the chosen tuning parameters for all values of  $\theta$ . The result shows that LSQMI accurately estimates  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  when the data size is large. However, when the data size is small, the estimated  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  has high fluctuation.

Figure 3.1(b) shows the averaged value over 20 trials of the derivative of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  w.r.t.  $\theta$  computed by LSQMI(3000), LSQMI(100), and the proposed method with  $N = 100$  which is denoted by LSQMID(100). For the proposed method, we perform cross validation at  $\theta = 0$  and use the chosen tuning parameters for all values of  $\theta$ . The result shows that LSQMID(100) gives a smoother estimate than LSQMI(100) which has high fluctuation. To further explain the cause of the fluctuation of LSQMI(100), we plot experiment results of 4 trials in Figure 3.2, where



(a) The averaged estimated QMI. (b) The averaged estimated derivative of QMI.

Figure 3.1: The mean and standard error of the estimated  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  and the estimated derivative of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  w.r.t.  $\theta$  over 20 trials.

the left column corresponds to the value of the estimated  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  while the right column corresponds to the value of the estimated derivative of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  w.r.t.  $\theta$ . These results show that for LSQMI(100), a small fluctuation in the estimated QMI can cause a large fluctuation in the estimated derivative of QMI. On the other hand, LSQMID directly estimates the derivative of QMI and thus does not suffer from this problem.

Next, we investigate the behavior of the proposed method when the target dimensionality  $d_z$  increases. For  $\epsilon \sim \mathcal{N}(0, 0.15^2)$ , we generate dataset  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  as follows:

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\mathbf{0}_{20}, \mathbf{I}_{20}), \\ y &= \sum_{i=1}^{d_z} (x^{(i)})^2 + \epsilon. \end{aligned}$$

The goal is to estimate  $\frac{\partial \text{QMI}(\mathbf{W}^{\mathbf{X}, \mathbf{Y}})}{\partial \mathbf{W}}$  at different values of  $\mathbf{W}$ . The estimated derivative  $\frac{\partial \widehat{\text{QMI}}(\mathbf{W})}{\partial \mathbf{W}}$  is evaluated by the mean squared error defined as

$$\text{MSE} = \frac{1}{K} \sum_{k=1}^K \left\| \frac{\partial \widehat{\text{QMI}}(\mathbf{W}_k)}{\partial \mathbf{W}} - \frac{\partial \text{QMI}^*(\mathbf{W}_k)}{\partial \mathbf{W}} \right\|_{\text{Frobenius}}^2, \quad (3.73)$$

where  $\frac{\partial \text{QMI}^*(\mathbf{W})}{\partial \mathbf{W}}$  denotes the derivative of QMI estimated by LSQMI with sample size  $n = 3000$ . The matrices  $\{\mathbf{W}_k\}_{k=1}^K$  with  $K = 500$  are generated randomly such that  $\mathbf{W}_k \mathbf{W}_k^\top = \mathbf{I}_{d_z}$ .

Figure 3.3 shows the mean and standard error over 10 trials of the mean squared error on sample sizes  $n \in \{200, 300, 400, 500\}$  and target dimensionalities  $d_z \in \{3, 5, 10\}$ . The results show that for  $d_z = 3$  and  $d_z = 5$  LSQMID gives much more accurate estimated derivatives than that of LSQMI, especially when the sample sizes are small.

For  $d_z = 10$ , LSQMID performs better only when the sample size is small. When the sample size increases, the improvement of LSQMI is better than that of LSQMID and LSQMI eventually outperforms LSQMID. The main reason behind this phenomena is that derivative estimation is very challenging when the target dimensionality

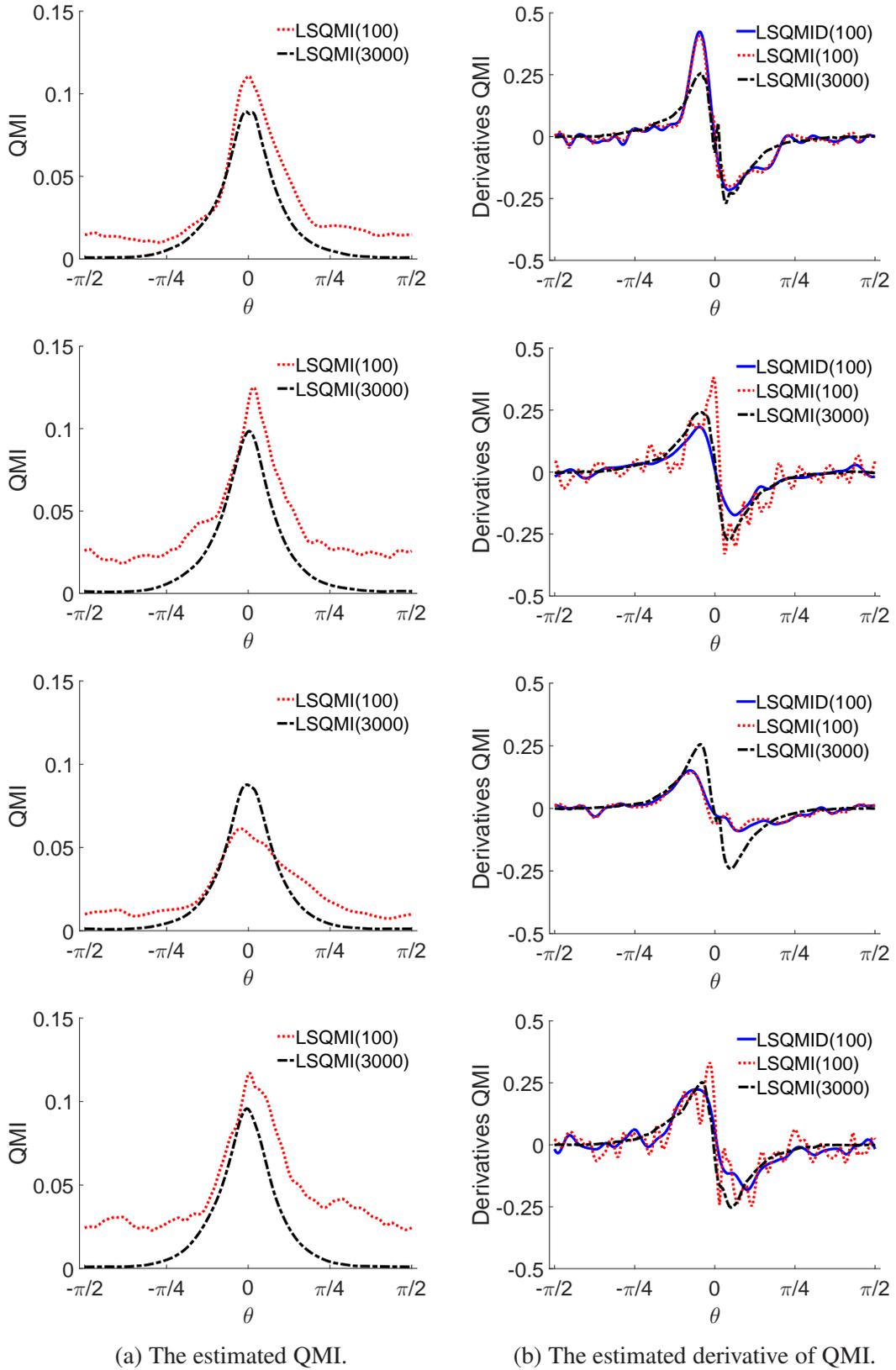


Figure 3.2: Examples of the estimated QMI and the estimated derivative of QMI. The left column shows the estimated  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$ , and the right column shows the estimated derivative of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  w.r.t.  $\theta$ . Each row indicates each trial.

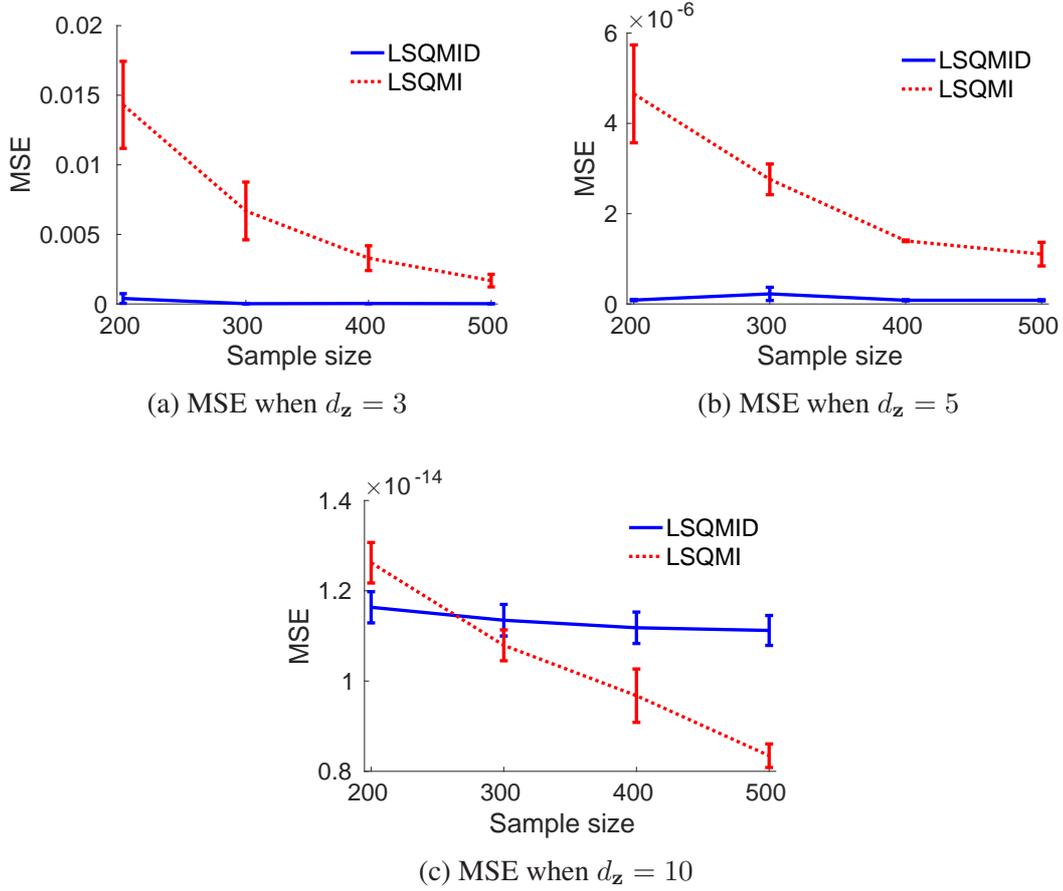


Figure 3.3: The mean and standard error of the mean squared error (MSE) of the estimated QMI derivatives over 10 trials on different sample sizes and different dimensionalities.

$d_z$  is large, and LSQMID would require a much larger number of samples in order to accurately estimate these derivatives.

### 3.5.2 Artificial Data

Next, we evaluate the usefulness of the proposed method in linear supervised dimension reduction using artificial datasets.

#### Setup

Firstly, let  $U(a, b)$  denote the uniform distribution over an interval  $[a, b]$ ,  $\Gamma(a, b)$  denote the gamma distribution with shape parameter  $a$  and scale parameter  $b$ , and  $\text{Laplace}(a, b)$  denote the Laplace distribution with mean  $a$  and scale parameter  $b$ . Then we consider datasets with the output dimensionality  $d_y = 1$ , and the optimal matrix  $\mathbf{W}_{\text{opt}} \in \mathbb{R}^{d_z \times d_x}$  (including their rotations) as follows:

**Dataset A:** For  $\epsilon \sim \Gamma(0.25, 0.25)$ , we use

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\mathbf{0}_{d_x}, \mathbf{I}_{d_x}), \\ y &= \exp\left(-\frac{(x^{(1)} + x^{(2)})^2}{0.5}\right) + \epsilon, \\ \mathbf{W}_{\text{opt}} &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & \cdots & 0 \end{bmatrix}. \end{aligned}$$

**Dataset B:** For  $\epsilon \sim \Gamma(0.25, 0.5)$  and  $i \in \{1, \dots, d_x\}$ , we use

$$\begin{aligned} x^{(i)} &\sim \text{U}(-1, 1), \\ y &= \frac{1}{\sqrt{2}}x^{(1)}x^{(2)} - \epsilon, \\ \mathbf{W}_{\text{opt}} &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix}. \end{aligned}$$

**Dataset C:** For  $\epsilon \sim \mathcal{N}(0, 0.25)$  and  $i \in \{1, \dots, d_x\}$ , we use

$$\begin{aligned} x^{(i)} &\sim \Gamma(0.25, 0.5), \\ y &= \sqrt{x^{(1)}} + \sqrt{2x^{(2)}} + \epsilon, \\ \mathbf{W}_{\text{opt}} &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix}. \end{aligned}$$

**Dataset D:** For  $\epsilon \sim \mathcal{N}(0, 0.25)$  and  $i \in \{1, \dots, d_x\}$ , we use

$$\begin{aligned} x^{(i)} &\sim \text{Laplace}(0, 0.5), \\ y &= \text{sinc}\left(\frac{x^{(1)}\pi}{2}\right) + x^{(2)}\epsilon, \\ \mathbf{W}_{\text{opt}} &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix}. \end{aligned}$$

For the datasets **A**, **B**, and **C**,  $\epsilon$  is an additive noise, while for the datasets **D**,  $\epsilon$  is a multiplicative noise. Figure 3.4 shows the plot of these datasets (after standardization). Note the presence of outliers in the datasets.

To estimate  $\mathbf{W}$  from  $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ , we execute the following methods:

**LSQMID:** The proposed method. Linear supervised dimension reduction is performed by maximizing  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  where the derivative of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is estimated by the proposed method. The solution  $\widehat{\mathbf{W}}$  is obtained by fixed-point iteration<sup>2</sup>.

**LSQMI:** Linear supervised dimension reduction is performed by maximizing  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  where  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  is estimated by LSQMI and the derivative of  $\text{QMI}(\mathbf{Z}, \mathbf{Y})$  w.r.t.  $\mathbf{W}$  is computed from the QMI estimator. The solution  $\widehat{\mathbf{W}}$  is obtained by gradient ascent with linesearch over the Grassmann manifold<sup>3</sup>.

<sup>2</sup>Our code is publicly available: [www.ms.k.u-tokyo.ac.jp/software.html\#LSQMID](http://www.ms.k.u-tokyo.ac.jp/software.html\#LSQMID)

<sup>3</sup>We use the manifold optimization toolbox (Boumal et al., 2014) to perform the optimization.

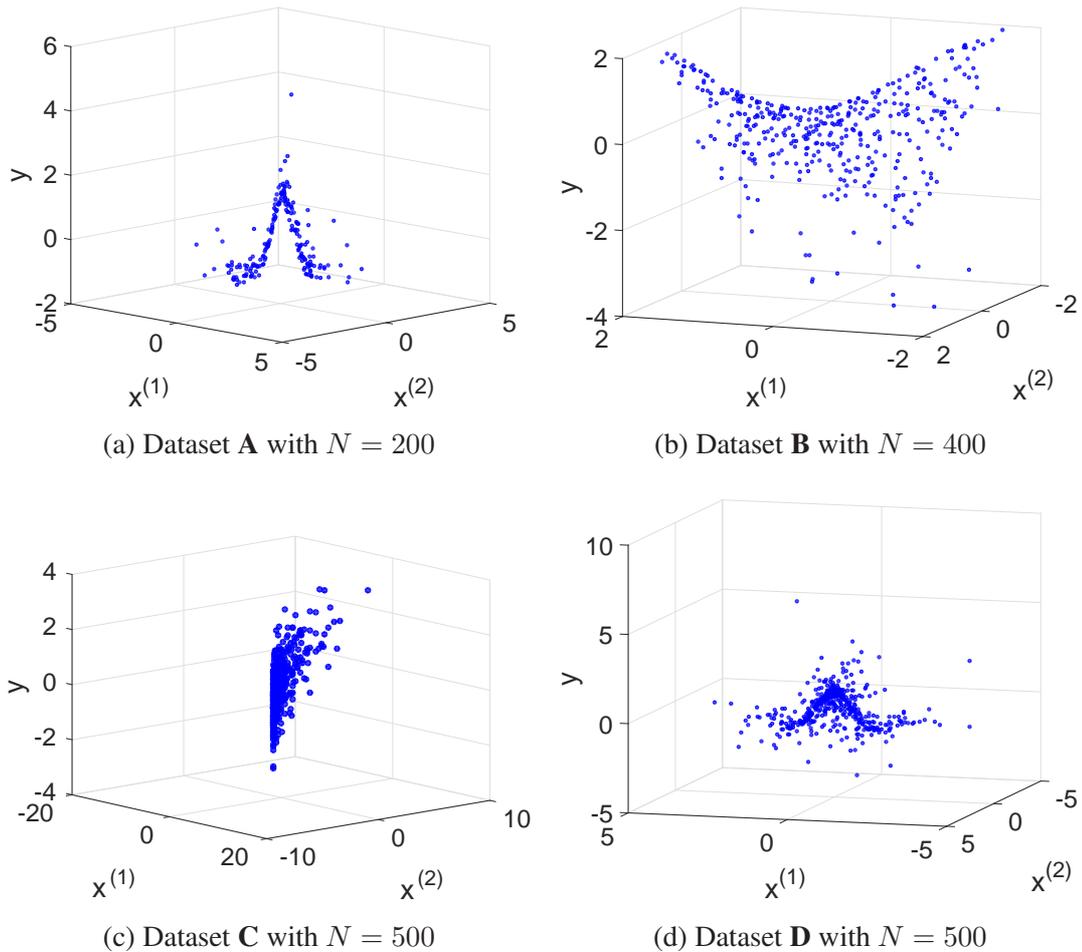


Figure 3.4: Artificial datasets.

**LSDR** (Suzuki and Sugiyama, 2013): Linear supervised dimension reduction is performed by maximizing  $\text{SMI}(Z, Y)$ . The solution  $\widehat{\mathbf{W}}$  is obtained by gradient ascent with linesearch over the Grassmann manifold<sup>4</sup>.

**LSCE**: Linear supervised dimension reduction is performed by minimizing a squared loss variant of conditional entropy. The solution  $\widehat{\mathbf{W}}$  is obtained by gradient descent with linesearch over the Grassmann manifold. This method is our second contribution in this dissertation and its details will be given in Chapter 4.

**dMAVE** (Xia, 2007): Linear supervised dimension reduction is performed by minimizing an error of the local linear smoother of the conditional density  $p(y|z)$ . The solution  $\widehat{\mathbf{W}}$  is obtained by solving quadratic programming problems<sup>5</sup>.

**KDR** (Fukumizu et al., 2004): Linear supervised dimension reduction is performed by minimizing the trace of the conditional covariance operator  $\Sigma_{\mathbf{Y}\mathbf{Y}|\mathbf{Z}}$ . The solution  $\widehat{\mathbf{W}}$  is obtained by gradient descent with linesearch over the Stiefel manifold<sup>6</sup>.

<sup>4</sup>We use the code: [www.ms.k.u-tokyo.ac.jp/software.html\#LSDR](http://www.ms.k.u-tokyo.ac.jp/software.html\#LSDR)

<sup>5</sup>We use the code: [www.stat.nus.edu.sg/~staxyc/](http://www.stat.nus.edu.sg/~staxyc/)

<sup>6</sup>We use the code: [www.ism.ac.jp/~fukumizu/software.html](http://www.ism.ac.jp/~fukumizu/software.html)

We set the number of basis functions to  $b = \min(200, N)$ . For LSQMID, LSQMI, LSDR, and LSCE, we randomly generate 10 orthonormal matrices and use them as the initial solutions. For dMAVE, we use a solution obtained by dOPG (Xia, 2007) as the initial solution. For KDR, we consider two approaches for obtaining the initial solutions. KDR (gKDR) uses a solution obtained by gKDR (Fukumizu and Leng, 2012) as the initial solution. KDR (Random) uses 10 randomly generated orthonormal matrices as the initial solutions and chooses a solution with the minimum objective value as the final solution. We also compare these methods with a randomly generated orthonormal matrix. The obtained solution  $\widehat{\mathbf{W}}$  is evaluated by the dimension reduction error defined as

$$\text{Error}_{\text{DR}} = \|\mathbf{W}_{\text{opt}}^{\top} \mathbf{W}_{\text{opt}} - \widehat{\mathbf{W}}^{\top} \widehat{\mathbf{W}}\|_{\text{Frobenius}}, \quad (3.74)$$

where  $\|\cdot\|_{\text{Frobenius}}$  denotes the Frobenius norm of a matrix. This dimension reduction error is invariant to rotation within the subspace, i.e., an error of  $\widehat{\mathbf{W}}$  is the same as that of  $\mathbf{A}\widehat{\mathbf{W}}$  where  $\mathbf{A}$  is a  $d_z$ -by- $d_z$  orthogonal matrix.

### Results on Different Data Sizes

We firstly evaluate the methods on different data sizes. Table 3.1 shows the mean and standard error over 50 trials of the dimension reduction error with different data sizes where the input dimensionality is fixed to  $d_x = 5$ . The randomly generated matrices are uninformative and give large error. LSQMID works very well for datasets **A**, **C**, and **D**, but it works quite poorly for dataset **B** when compared with other methods. However, LSQMID gives the most informative results for dataset **C** where outliers are in the input domain. These results demonstrate the weakness of existing methods in terms of robustness against outliers.

On the other hand, LSQMI tends to be unstable and works poorly, except for dataset **D** when the data size is large. Note that LSQMI is comparable to the best method (in term of the mean error) in dataset **A** due to its unstable behavior. The cause of this instability could be the high fluctuation of the derivative of QMI by LSQMI, as shown previously in the illustrative experiment.

The two variants of KDR work quite well on datasets **A**, **B**, and **D**. However, KDR (gKDR) is quite unstable for dataset **A** when the data size is small which can be seen by its relatively large standard errors. In contrast, KDR (Random) gives much more stable results. This implies that gKDR might provide a poor initial solution to KDR in some trials, which makes KDR fail to find a good solution. On the other hand, dMAVE works quite poorly overall which might be because its model selection strategy is not suitable for these datasets.

Table 3.2 shows the mean and standard error over 50 trials of the computation time on different data sizes<sup>7</sup>. All methods take longer time as the number of data increases. The results also show that LSQMID is computationally more demanding than other methods except LSQMI and KDR (Random). dMAVE and KDR (gKDR) is computationally very efficient because they do not perform cross-validation for parameter tuning and they do not restart optimization with many initial solutions<sup>8</sup>. On the other

<sup>7</sup>The computation time is measured using MATLAB<sup>®</sup> on 2.10 GHz 8 cores processor with 128 GB memory.

<sup>8</sup>gKDR performs cross-validation based on the regression error to choose its tuning parameter (Fukumizu and Leng, 2012). However, gKDR is not an iterative method and the computation time of KDR (gKDR) is mostly dominated by KDR.

hand, LSQMID, LSQMI, LSDR, and LSCE perform cross-validation and restart optimization with 10 initial solutions. Despite these similarities, LSQMID is computationally more demanding than LSDR and LSCE for two reasons. Firstly, LSQMID performs orthonormalization while LSDR and LSCE utilize manifold optimization. It is known that manifold optimization tends to be computationally more efficient than orthonormalization (Absil et al., 2008). Secondly, LSQMID estimates the derivative of QMI w.r.t.  $\mathbf{W}$  by  $d_z$  estimators for  $d_z$  derivatives of the density difference, while LSDR and LSCE estimate a single quantity.

LSQMI is computationally the most inefficient even though it also utilizes manifold optimization and estimates a single quantity. The reason could be that backtracking linesearch parameters that we used for the toolbox (Boumal et al., 2014) are not suitable for LSQMI which results in many backtracking steps per iteration. We believe that the computation time of LSQMI can be improved with a more proper backtracking linesearch parameter tuning.

KDR (Random) also utilizes manifold optimization and estimates a single quantity. However, it takes longer computation time than LSQMID for datasets **B**, **C** and **D** when  $N = 500$ . The reason is that, KDR inverts a  $N$ -by- $N$  matrix while LSQMID inverts  $d_z$  number of  $b$ -by- $b$  matrices (see Equation (3.48)). Thus, when  $N$  is much larger than  $b$  and  $d_z$  is small, inverting a single  $N$ -by- $N$  matrix can take much longer time than inverting  $d_z$  number of  $b$ -by- $b$  matrices.

### Results on Different Input Dimensionalities

Next, we evaluate the methods on different input dimensionalities. Table 3.3 shows the mean and standard error over 50 trials of the dimension reduction error with different input dimensionalities where the data size is fixed. We use  $N = 200$  for datasets **A**, and use  $N = 400$  for datasets **B**, **C** and **D**. The randomly generated matrices are uninformative and give large error. All methods perform best on low input dimensionalities. For all datasets, LSQMID works well overall except when  $d_x = 15$ . For dataset **C**, only LSQMID and LSQMI gives informative result.

Table 3.4 shows the mean and standard error over 50 trials of the computation time on different input dimensionalities. All methods take longer time as the dimensionality increases. However, dMAVE has the largest relative increment among all methods, i.e., it takes approximately three times longer when  $d_x$  increases from 10 to 15.

### 3.5.3 Benchmark Data

Finally, we evaluate the usefulness of the proposed method on benchmark datasets. In the following experiments, we consider linear supervised dimension reduction for classification and regression tasks.

#### Classification

We firstly evaluate the proposed method on a classification task. We consider the ‘Wine’ dataset from the UCI repository (Bache and Lichman, 2013). The input variables  $\mathbf{x}$  have dimensionality  $d_x = 13$  and the output variable  $y$  determines one of the three classes. We standardize the input so that it has zero mean and unit variance. The

Table 3.1: Mean and standard error of the dimension reduction error over 50 trials for artificial datasets on different data sizes with input dimensionality  $d_x = 5$ . The best method in terms of the mean error and comparable methods according to the paired  $t$ -test at the significance level 5% are specified by bold face.

Dataset	$N$	LSQMID	LSQMI	LSDR	LSCE	dMAVE	KDR (gKDR)	KDR (Random)	Random
<b>A</b>	50	0.464(0.080)	0.990(0.066)	<b>0.149(0.013)</b>	0.652(0.083)	0.233(0.033)	0.418(0.071)	<b>0.190(0.044)</b>	1.304(0.016)
	100	<b>0.111(0.024)</b>	0.473(0.077)	<b>0.070(0.005)</b>	0.160(0.044)	0.127(0.008)	<b>0.124(0.035)</b>	<b>0.075(0.006)</b>	1.304(0.016)
	150	<b>0.059(0.005)</b>	0.165(0.044)	0.058(0.004)	<b>0.054(0.005)</b>	0.095(0.005)	<b>0.056(0.004)</b>	<b>0.056(0.004)</b>	1.304(0.016)
	200	<b>0.045(0.004)</b>	<b>0.072(0.027)</b>	<b>0.046(0.004)</b>	<b>0.052(0.009)</b>	0.080(0.005)	<b>0.047(0.004)</b>	<b>0.047(0.004)</b>	1.304(0.016)
	250	<b>0.040(0.003)</b>	<b>0.070(0.027)</b>	<b>0.041(0.003)</b>	<b>0.041(0.004)</b>	0.070(0.004)	0.045(0.004)	0.045(0.004)	1.304(0.016)
<b>B</b>	100	0.362(0.037)	1.290(0.057)	0.370(0.032)	<b>0.226(0.022)</b>	<b>0.248(0.016)</b>	0.421(0.042)	0.433(0.042)	1.465(0.028)
	200	0.221(0.022)	0.700(0.081)	0.196(0.007)	<b>0.116(0.008)</b>	0.155(0.009)	0.168(0.010)	0.168(0.010)	1.465(0.028)
	300	0.103(0.008)	0.359(0.066)	0.138(0.005)	<b>0.075(0.003)</b>	0.109(0.004)	0.122(0.006)	0.122(0.006)	1.465(0.028)
	400	<b>0.081(0.005)</b>	0.111(0.009)	0.128(0.004)	<b>0.080(0.006)</b>	0.104(0.005)	<b>0.089(0.005)</b>	<b>0.089(0.005)</b>	1.465(0.028)
	500	0.081(0.004)	0.130(0.021)	0.114(0.004)	<b>0.069(0.006)</b>	<b>0.075(0.003)</b>	<b>0.068(0.004)</b>	<b>0.068(0.004)</b>	1.465(0.028)
<b>C</b>	100	<b>1.108(0.069)</b>	1.316(0.057)	1.371(0.024)	<b>1.240(0.039)</b>	<b>1.164(0.036)</b>	1.437(0.023)	1.395(0.023)	1.465(0.028)
	200	<b>0.819(0.092)</b>	1.086(0.089)	1.336(0.026)	1.205(0.043)	<b>1.015(0.054)</b>	1.325(0.020)	1.358(0.019)	1.465(0.028)
	300	<b>0.333(0.061)</b>	0.618(0.081)	1.346(0.029)	1.120(0.048)	0.981(0.047)	1.271(0.024)	1.279(0.026)	1.465(0.028)
	400	<b>0.224(0.054)</b>	0.404(0.080)	1.327(0.028)	1.133(0.044)	0.863(0.056)	1.198(0.033)	1.250(0.023)	1.465(0.028)
	500	<b>0.267(0.069)</b>	0.461(0.087)	1.347(0.027)	1.084(0.050)	0.756(0.054)	1.215(0.032)	1.217(0.020)	1.465(0.028)
<b>D</b>	100	<b>0.602(0.070)</b>	1.033(0.070)	0.706(0.055)	0.630(0.059)	0.877(0.056)	0.610(0.046)	<b>0.466(0.036)</b>	1.465(0.028)
	200	<b>0.401(0.049)</b>	0.569(0.064)	<b>0.408(0.037)</b>	<b>0.338(0.028)</b>	0.630(0.057)	<b>0.371(0.026)</b>	<b>0.338(0.028)</b>	1.465(0.028)
	300	<b>0.274(0.040)</b>	<b>0.334(0.043)</b>	<b>0.276(0.021)</b>	<b>0.293(0.030)</b>	0.453(0.045)	<b>0.266(0.021)</b>	<b>0.263(0.020)</b>	1.465(0.028)
	400	<b>0.216(0.035)</b>	<b>0.176(0.016)</b>	0.223(0.013)	<b>0.214(0.018)</b>	0.324(0.043)	0.252(0.013)	0.238(0.013)	1.465(0.028)
	500	<b>0.137(0.013)</b>	<b>0.151(0.013)</b>	0.191(0.012)	0.218(0.018)	0.258(0.028)	0.205(0.012)	0.195(0.012)	1.465(0.028)

Table 3.2: Mean and standard error of the computation time in seconds over 50 trials for artificial datasets on different data sizes with input dimensionality  $d_x = 5$ .

Dataset	$N$	LSQMID	LSQMI	LSDR	LSCE	dMAVE	KDR (gKDR)	KDR (Random)
<b>A</b>	50	9.429(0.057)	34.110(0.591)	6.174(0.120)	7.305(0.099)	0.094(0.002)	0.395(0.016)	4.231(0.079)
	100	27.814(0.274)	92.052(1.738)	14.150(0.227)	21.928(0.323)	0.360(0.013)	1.292(0.050)	16.573(0.490)
	150	59.452(1.064)	124.585(2.866)	21.368(0.354)	36.561(0.498)	0.583(0.012)	2.064(0.038)	32.998(0.190)
	200	93.544(1.900)	181.314(3.732)	30.290(0.473)	53.591(0.733)	0.895(0.024)	3.895(0.123)	56.161(0.644)
	250	89.871(1.923)	174.112(3.526)	31.211(0.615)	56.277(0.749)	1.197(0.022)	4.915(0.082)	72.194(0.220)
<b>B</b>	100	49.036(0.429)	154.616(3.053)	12.783(0.246)	18.447(0.200)	0.363(0.012)	1.210(0.026)	13.001(0.150)
	200	145.692(2.185)	300.697(6.914)	24.349(0.432)	46.142(0.514)	0.852(0.018)	3.624(0.094)	38.819(0.287)
	300	168.623(3.047)	251.485(5.868)	26.052(0.469)	47.725(0.531)	1.735(0.032)	7.823(0.127)	86.127(0.798)
	400	183.009(2.868)	231.134(6.435)	28.021(0.430)	46.014(0.503)	2.681(0.055)	13.670(0.209)	144.049(0.332)
	500	203.555(3.401)	223.437(5.363)	30.299(0.523)	48.906(0.538)	4.130(0.094)	24.843(0.321)	241.030(0.592)
<b>C</b>	100	49.381(0.287)	155.790(2.051)	13.448(0.198)	16.040(0.167)	0.320(0.002)	1.060(0.008)	11.186(0.061)
	200	132.830(0.152)	358.054(7.186)	29.940(0.474)	43.118(0.600)	0.812(0.011)	3.667(0.026)	38.001(0.253)
	300	148.501(0.210)	331.875(6.626)	32.432(0.627)	38.933(0.448)	1.591(0.013)	6.766(0.045)	74.014(0.454)
	400	169.348(0.375)	343.421(7.810)	36.026(0.672)	40.473(0.483)	2.450(0.016)	13.352(0.070)	140.416(0.343)
	500	186.787(0.357)	352.525(8.282)	39.465(0.813)	45.053(0.552)	3.806(0.032)	22.837(0.126)	240.166(0.707)
<b>D</b>	100	48.305(0.372)	153.212(3.692)	15.120(0.342)	18.610(0.231)	0.392(0.015)	1.283(0.048)	18.654(0.457)
	200	161.487(2.647)	322.414(6.317)	32.599(0.605)	47.629(0.679)	0.920(0.020)	3.856(0.098)	52.280(0.430)
	300	181.158(2.660)	271.453(6.801)	36.607(0.663)	49.746(0.685)	1.792(0.035)	7.559(0.099)	102.684(0.256)
	400	202.340(3.286)	259.462(5.090)	40.922(0.732)	50.476(0.671)	2.880(0.078)	13.556(0.229)	157.811(0.571)
	500	222.527(3.043)	261.036(5.512)	46.453(0.882)	54.916(0.804)	4.456(0.120)	23.930(0.302)	276.041(2.123)

Table 3.3: Mean and standard error of the dimension reduction error over 50 trials for artificial datasets on different input dimensionalities with fixed data sizes;  $N = 200$  for Datasets **A**, and  $N = 400$  for Datasets **B**, **C** and **D**. The best method in terms of the mean error and comparable methods according to the paired  $t$ -test at the significance level 5% are specified by bold face.

Dataset	$d_x$	LSQMID	LSQMI	LSDR	LSCE	dMAVE	KDR (gKDR)	KDR (Random)	Random
<b>A</b>	3	<b>0.045(0.004)</b>	<b>0.046(0.005)</b>	<b>0.047(0.004)</b>	<b>0.045(0.004)</b>	0.076(0.005)	<b>0.047(0.004)</b>	<b>0.047(0.004)</b>	1.182(0.034)
	5	<b>0.045(0.004)</b>	<b>0.072(0.027)</b>	<b>0.046(0.004)</b>	<b>0.052(0.009)</b>	0.080(0.005)	<b>0.047(0.004)</b>	<b>0.047(0.004)</b>	1.304(0.016)
	8	<b>0.060(0.005)</b>	0.321(0.068)	<b>0.054(0.004)</b>	0.540(0.082)	0.100(0.004)	<b>0.057(0.004)</b>	<b>0.057(0.004)</b>	1.341(0.011)
	10	<b>0.055(0.004)</b>	0.597(0.086)	<b>0.056(0.004)</b>	0.883(0.083)	0.116(0.004)	0.061(0.003)	0.061(0.003)	1.355(0.009)
	15	0.151(0.037)	0.885(0.088)	<b>0.061(0.004)</b>	1.253(0.050)	0.136(0.005)	0.456(0.084)	0.069(0.004)	1.376(0.007)
<b>B</b>	3	<b>0.039(0.003)</b>	<b>0.049(0.006)</b>	0.062(0.004)	<b>0.038(0.003)</b>	0.053(0.003)	0.058(0.004)	0.058(0.004)	1.062(0.043)
	5	<b>0.081(0.005)</b>	0.111(0.009)	0.128(0.004)	<b>0.080(0.006)</b>	0.104(0.005)	<b>0.089(0.005)</b>	<b>0.089(0.005)</b>	1.465(0.028)
	8	<b>0.143(0.007)</b>	0.784(0.100)	0.163(0.006)	<b>0.120(0.012)</b>	<b>0.139(0.004)</b>	<b>0.137(0.004)</b>	<b>0.137(0.004)</b>	1.702(0.021)
	10	<b>0.201(0.025)</b>	1.065(0.103)	<b>0.180(0.004)</b>	<b>0.155(0.013)</b>	<b>0.168(0.005)</b>	<b>0.179(0.007)</b>	<b>0.179(0.007)</b>	1.771(0.017)
	15	0.368(0.046)	1.682(0.053)	0.227(0.006)	<b>0.172(0.008)</b>	0.207(0.004)	0.227(0.006)	0.226(0.006)	1.853(0.012)
<b>C</b>	3	<b>0.212(0.054)</b>	<b>0.219(0.065)</b>	1.187(0.055)	0.695(0.071)	0.577(0.062)	0.946(0.043)	0.963(0.037)	1.062(0.043)
	5	<b>0.224(0.054)</b>	0.404(0.080)	1.327(0.028)	1.133(0.044)	0.863(0.056)	1.198(0.033)	1.250(0.023)	1.465(0.028)
	8	<b>0.589(0.087)</b>	<b>0.746(0.094)</b>	1.391(0.013)	1.204(0.037)	1.086(0.047)	1.259(0.029)	1.279(0.020)	1.702(0.021)
	10	<b>0.765(0.088)</b>	<b>0.829(0.089)</b>	1.404(0.009)	1.328(0.022)	1.227(0.031)	1.295(0.023)	1.313(0.021)	1.771(0.017)
	15	1.308(0.068)	<b>1.095(0.073)</b>	1.426(0.010)	1.399(0.012)	1.360(0.019)	1.362(0.014)	1.315(0.022)	1.853(0.012)
<b>D</b>	3	<b>0.067(0.010)</b>	<b>0.069(0.010)</b>	0.131(0.012)	0.095(0.011)	0.317(0.049)	0.129(0.013)	0.124(0.012)	1.062(0.043)
	5	<b>0.216(0.035)</b>	<b>0.176(0.016)</b>	0.223(0.013)	<b>0.214(0.018)</b>	0.324(0.043)	0.252(0.013)	0.238(0.013)	1.465(0.028)
	8	<b>0.343(0.036)</b>	0.727(0.071)	<b>0.314(0.023)</b>	<b>0.348(0.033)</b>	<b>0.380(0.030)</b>	0.356(0.020)	<b>0.345(0.018)</b>	1.702(0.021)
	10	<b>0.473(0.049)</b>	0.809(0.063)	<b>0.387(0.020)</b>	0.484(0.034)	0.605(0.061)	0.484(0.036)	<b>0.420(0.022)</b>	1.771(0.017)
	15	0.689(0.049)	1.400(0.063)	<b>0.616(0.040)</b>	0.757(0.044)	0.936(0.056)	0.632(0.035)	<b>0.558(0.018)</b>	1.853(0.012)

Table 3.4: Mean and standard error of the computation time in seconds over 50 trials for artificial datasets on different input dimensionalities with fixed data sizes;  $N = 200$  for Datasets **A**, and  $N = 400$  for Datasets **B**, **C** and **D**.

Dataset	$d_x$	LSQMID	LSQMI	LSDR	LSCE	dMAVE	KDR (gKDR)	KDR (Random)
<b>A</b>	3	87.730(1.582)	116.106(2.398)	21.621(0.517)	44.211(0.551)	0.412(0.009)	3.539(0.090)	41.643(0.627)
	5	93.544(1.900)	181.314(3.732)	30.290(0.473)	53.591(0.733)	0.895(0.024)	3.895(0.123)	56.161(0.644)
	8	95.935(1.649)	308.600(5.879)	36.921(0.690)	47.501(0.711)	2.147(0.034)	3.793(0.093)	54.826(0.605)
	10	88.833(1.398)	363.697(6.154)	40.374(0.602)	50.793(0.641)	3.335(0.043)	3.695(0.085)	63.552(0.706)
	15	109.637(1.437)	476.093(3.908)	48.330(0.756)	59.981(1.006)	12.079(0.261)	4.350(0.111)	70.260(0.741)
<b>B</b>	3	169.567(2.992)	106.728(1.305)	29.736(0.499)	48.965(0.704)	1.254(0.025)	13.023(0.237)	137.888(1.222)
	5	183.009(2.868)	231.134(6.435)	28.021(0.430)	46.014(0.503)	2.681(0.055)	13.670(0.209)	144.049(0.332)
	8	205.578(3.333)	438.931(7.086)	36.879(0.659)	52.329(0.698)	6.704(0.159)	16.084(0.308)	154.936(0.421)
	10	220.762(3.240)	499.952(6.730)	43.026(0.821)	57.738(0.727)	10.746(0.249)	17.482(0.319)	161.003(0.446)
	15	263.757(3.493)	577.488(3.999)	61.165(1.145)	72.184(0.993)	31.961(0.747)	21.231(0.360)	188.233(0.408)
<b>C</b>	3	154.131(0.343)	92.523(1.908)	25.316(0.478)	43.993(0.635)	1.230(0.011)	11.587(0.073)	124.230(0.498)
	5	169.348(0.375)	343.421(7.810)	36.026(0.672)	40.473(0.483)	2.450(0.016)	13.352(0.070)	140.416(0.343)
	8	191.035(0.345)	500.047(4.296)	50.586(0.708)	47.681(0.606)	6.112(0.044)	15.460(0.081)	158.438(0.479)
	10	208.191(0.615)	529.924(2.915)	56.769(0.689)	48.858(0.621)	9.749(0.068)	15.328(0.094)	162.992(0.593)
	15	250.683(0.865)	570.572(1.730)	74.443(0.774)	55.368(0.682)	28.005(0.072)	18.953(0.089)	192.165(0.457)
<b>D</b>	3	186.339(2.808)	108.504(2.145)	37.751(0.687)	55.153(0.787)	1.333(0.032)	12.895(0.174)	149.617(0.489)
	5	202.340(3.286)	259.462(5.090)	40.922(0.732)	50.476(0.671)	2.880(0.078)	13.556(0.229)	157.811(0.571)
	8	226.080(2.683)	471.307(6.142)	55.825(1.108)	59.997(0.878)	6.873(0.140)	16.407(0.178)	176.396(0.405)
	10	242.822(2.923)	540.447(5.618)	64.376(0.967)	65.743(1.044)	11.472(0.269)	17.919(0.212)	191.839(0.575)
	15	275.041(3.748)	594.448(3.833)	88.215(1.650)	81.971(1.304)	32.900(0.722)	20.652(0.322)	219.071(0.640)

Table 3.5: Mean and standard error of the misclassification rate over 20 trials for the ‘Wine’ dataset with different target dimensionalities. The best method in terms of the mean error and comparable methods according to the paired *t-test* at the significance level 5% are specified by bold face.

$d_z$	LSQMID	LSQMI	LSDR	dMAVE	KDR (Random)	PCA
1	23.33(10.82)	31.54(4.71)	<b>8.59(3.09)</b>	15.77(5.08)	<b>8.53(2.70)</b>	15.38(3.25)
2	<b>3.27(1.21)</b>	24.81(5.94)	4.94(2.25)	<b>3.33(1.46)</b>	<b>3.14(1.64)</b>	<b>3.65(2.05)</b>
3	<b>2.95(2.00)</b>	19.87(7.09)	6.03(3.14)	<b>3.33(2.01)</b>	<b>3.53(1.66)</b>	<b>3.53(1.60)</b>
4	<b>3.53(3.65)</b>	21.03(10.75)	6.09(3.58)	<b>3.59(2.55)</b>	<b>3.40(2.01)</b>	<b>3.59(2.02)</b>

dataset contains 178 samples. We randomly choose  $n_{tr} = 100$  samples for training purposes, and use the rest  $n_{te} = 78$  for testing purposes. We execute linear supervised dimension reduction methods and principal component analysis (PCA) (Jolliffe, 1986) with target dimensionality  $d_z = \{1, 2, 3, 4\}$  to obtain solutions  $\widehat{\mathbf{W}}$ . Then, we train a *support vector machine classifier* (SVM)<sup>9</sup> (Cortes and Vapnik, 1995). The performance of a classifier  $f(\widehat{\mathbf{W}}\mathbf{x})$  is evaluated by the misclassification rate for test samples:

$$\frac{100}{n_{te}} \sum_{i=1}^{n_{te}} I(f(\widehat{\mathbf{W}}\mathbf{x}_i) \neq y_i), \quad (3.75)$$

where  $I(a)$  denotes the indicator function which equals to 1 when the expression  $a$  is true and equals to 0 otherwise.

The misclassification rate in Table 3.5 shows that LSQMID performs very well for this dataset when  $d_z \in \{2, 3, 4\}$  and it gives the lowest misclassification rate when  $d_z = 3$ . In contrast, LSQMI performs very poorly and is also highly unstable as can be seen by a relatively large standard error. We expect that this is because the sample size is quite small which makes the performance of LSQMI relatively poor, as demonstrated in our previous experiments.

Figure 3.5 shows data points projected by  $\widehat{\mathbf{W}}$  with  $d_z = 2$ . We can see that all methods except LSQMI give good projections and we can easily distinguish data points between classes in the new data spaces. In contrast, for LSQMI many data points from one class (denoted by purple color) cannot be distinguished from the other two classes in the new data spaces.

## Regression

Next, we evaluate the proposed method on regression task using datasets from the UCI repository. To make the tasks more challenging, we append the original input  $\mathbf{x}$  with noise features of dimensionality 5. More specifically, for the original input  $\mathbf{x}$  with dimensionality  $d_x$ , we consider the augmented input  $\tilde{\mathbf{x}}$  with dimensionality  $d_{\tilde{\mathbf{x}}} = d_x + 5$  as

$$\tilde{\mathbf{x}} = [\mathbf{x}^\top, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5]^\top,$$

where  $\gamma_i \sim \Gamma(1, 2)$  for  $i \in \{1, \dots, 5\}$ . Then we use the paired data  $\{(\tilde{\mathbf{x}}_i, y_i)\}_{i=1}^N$  to perform experiments. We randomly choose  $N_{tr}$  data points for training purposes, and use the rest  $N_{te} = N - N_{tr}$  for testing purposes. We execute the supervised dimension

<sup>9</sup>We use the *LibSVM* implementation by Chang and Lin (2011).

reduction methods with target dimensionality  $d_z \in \{1, 2, 3, 4\}$  to obtain solutions  $\widehat{\mathbf{W}}$ . Then we use a *kernel ridge regressor* and a *k-nearest neighbor regressor* to evaluate the performance. The performance of a regressor  $f(\widehat{\mathbf{W}}\tilde{\mathbf{x}})$  is measured by the *root mean squared error* (RMSE) for test data points:

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{te}}} \sum_{i=1}^{N_{\text{te}}} \left( y_i - f(\widehat{\mathbf{W}}\tilde{\mathbf{x}}_i) \right)^2}. \quad (3.76)$$

We use a kernel ridge regressor with the Gaussian kernel where the Gaussian width and the regularization parameter are chosen by 5-fold cross-validation. Table 3.6 shows the RMSE averaged over 30 trials. LSQMID performs well overall for all datasets. LSQMI also performs well for the ‘Fertility’ and ‘Bike’ datasets where it outperforms LSQMID in terms of the mean error. However, LSQMI does not work for the other datasets. LSCE and dMAVE perform well only on some datasets, and LSDR, KDR (gKDR), and KDR (Random) perform poorly on these datasets.

Next, we use a *k-nearest neighbor regressor* where  $k \in \{1, \dots, 10\}$  is chosen by 5-fold cross validation. Table 3.7 shows the RMSE averaged over 30 trials. It shows that the *k-nearest neighbor regressor* gives smaller RMSEs than the kernel ridge regressor, except for the ‘Fertility’ dataset. This is perhaps because *k-nearest neighbor* tends to work well when the data has low dimensionality. The results between linear supervised dimension reduction methods are quite similar to those of the kernel ridge regressor, with the exception that LSDR and dMAVE also perform well on the ‘Bike’ dataset.

These results show that LSQMID works well as a linear supervised dimension reduction method for both kernel ridge regressor and *k-nearest neighbor regressor*.

### 3.6 Further Extension: Estimation of Higher Order Derivatives of Quadratic Mutual Information

We have shown that the (first order) derivative of QMI can be estimated once we know the (first order) derivative of the density difference, and we proposed a least-squares estimator to directly estimate the (first order) derivative of the density difference from data. Below, we further show that a higher order derivative of QMI can also be estimated in a similar manner.

From an approximation of the derivative of QMI in Eq.(3.28), the *k*-th order derivative of QMI w.r.t.  $W_{\ell_1, \ell'_1}, \dots, W_{\ell_k, \ell'_k}$  can be obtained from data as

$$\begin{aligned} \frac{\partial^k \widehat{\text{QMI}}(\mathbf{W})}{\partial W_{\ell_1, \ell'_1} \dots \partial W_{\ell_k, \ell'_k}} &= \frac{1}{N} \sum_{i=1}^N \frac{\partial^k f(\mathbf{z}_i, \mathbf{y}_i)}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} \left( \prod_{m=1}^{k'} x_i^{(\ell'_m)} \right) \\ &\quad - \frac{1}{N^2} \sum_{i,j=1}^N \frac{\partial^k f(\mathbf{z}_i, \mathbf{y}_j)}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} \left( \prod_{m=1}^{k'} x_i^{(\ell'_m)} \right). \end{aligned} \quad (3.77)$$

This means that the *k*-th order derivative of QMI w.r.t.  $W_{\ell_1, \ell'_1}, \dots, W_{\ell_k, \ell'_k}$  can be obtained once we know the *k*-th order derivative of the density difference w.r.t.  $z^{(\ell_1)}, \dots, z^{(\ell_k)}$ . A least-squares estimator for this derivative can be obtained

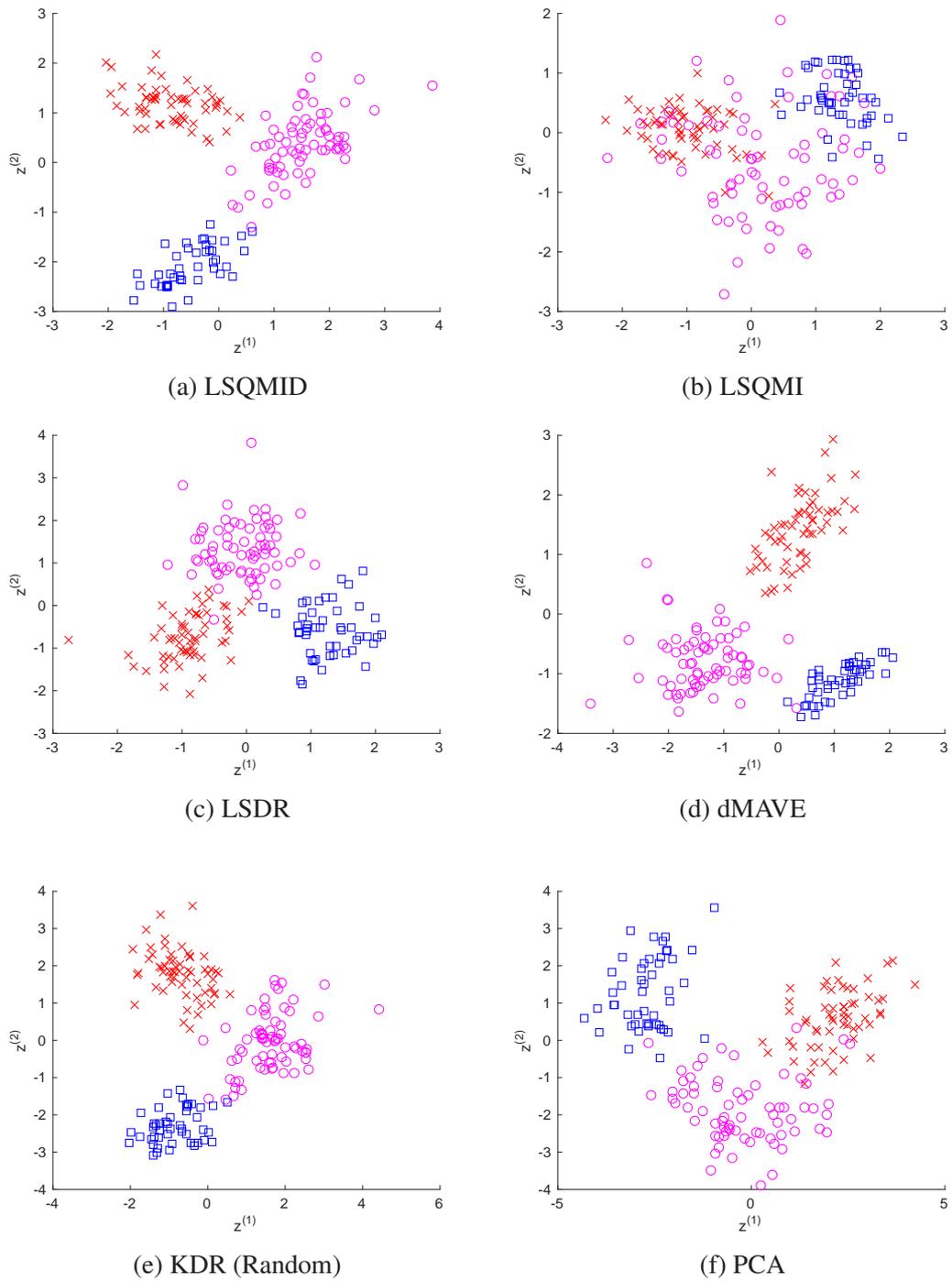


Figure 3.5: Data points of the ‘wine’ dataset after projection by each linear dimension reduction method. Data points from the same class are indicated by the same color.

Table 3.6: Mean and standard error of the root mean squared error over 30 trials for benchmark datasets using kernel ridge regressor. The best method in terms of the mean error and comparable methods according to the paired  $t$ -test at the significance level 5% are specified by bold face.

Dataset	$N_{\text{tr}}$	$d_{\tilde{\mathbf{x}}}$	$d_{\mathbf{z}}$	LSQMID	LSQMI	LSDR	LSCE	dMAVE	KDR (gKDR)	KDR (Random)
Fertility	50	14	1	1.215(0.049)	<b>1.092(0.043)</b>	1.315(0.043)	1.185(0.050)	1.321(0.063)	<b>1.116(0.050)</b>	1.174(0.047)
			2	<b>1.051(0.045)</b>	<b>1.029(0.043)</b>	1.199(0.031)	1.080(0.047)	1.340(0.052)	1.104(0.044)	1.247(0.049)
			3	<b>1.052(0.044)</b>	<b>1.038(0.047)</b>	1.104(0.044)	1.091(0.041)	1.288(0.048)	1.121(0.043)	1.231(0.037)
			4	<b>1.046(0.042)</b>	<b>1.026(0.042)</b>	1.092(0.039)	1.083(0.044)	1.271(0.033)	1.146(0.044)	1.202(0.035)
Yacht	100	11	1	<b>0.120(0.005)</b>	0.546(0.042)	0.180(0.012)	0.718(0.051)	0.213(0.017)	<b>0.124(0.007)</b>	<b>0.124(0.007)</b>
			2	<b>0.154(0.011)</b>	0.675(0.047)	0.344(0.023)	0.275(0.013)	0.224(0.014)	0.278(0.033)	0.248(0.012)
			3	<b>0.314(0.024)</b>	0.690(0.037)	0.425(0.018)	0.319(0.017)	<b>0.265(0.013)</b>	0.353(0.028)	0.318(0.015)
			4	0.413(0.021)	0.732(0.043)	0.494(0.015)	<b>0.355(0.013)</b>	<b>0.352(0.017)</b>	0.399(0.012)	0.400(0.015)
Concrete	200	13	1	0.621(0.013)	<b>0.606(0.014)</b>	0.606(0.008)	0.604(0.009)	<b>0.582(0.006)</b>	0.791(0.030)	0.637(0.012)
			2	0.568(0.010)	0.591(0.009)	0.568(0.010)	0.567(0.011)	<b>0.529(0.009)</b>	0.614(0.025)	<b>0.541(0.014)</b>
			3	<b>0.557(0.009)</b>	0.579(0.011)	0.576(0.012)	0.571(0.010)	<b>0.539(0.007)</b>	0.579(0.016)	<b>0.558(0.012)</b>
			4	<b>0.545(0.012)</b>	0.667(0.025)	0.568(0.010)	0.577(0.010)	<b>0.540(0.008)</b>	0.571(0.014)	0.583(0.014)
Breast-cancer	200	15	1	0.447(0.011)	0.523(0.018)	0.442(0.010)	0.453(0.016)	<b>0.375(0.007)</b>	0.447(0.012)	0.465(0.014)
			2	<b>0.435(0.010)</b>	0.473(0.012)	<b>0.437(0.009)</b>	<b>0.437(0.011)</b>	<b>0.420(0.012)</b>	0.454(0.014)	<b>0.440(0.011)</b>
			3	<b>0.376(0.004)</b>	0.462(0.010)	0.431(0.007)	0.438(0.009)	0.426(0.008)	0.430(0.007)	0.430(0.009)
			4	<b>0.377(0.005)</b>	0.419(0.008)	0.436(0.007)	0.425(0.012)	0.426(0.011)	0.433(0.007)	0.435(0.009)
Bike	300	19	1	0.043(0.011)	0.070(0.019)	<b>0.016(0.001)</b>	<b>0.015(0.004)</b>	0.139(0.051)	0.513(0.059)	0.194(0.005)
			2	<b>0.036(0.005)</b>	<b>0.035(0.003)</b>	0.049(0.002)	<b>0.031(0.005)</b>	0.081(0.007)	0.291(0.050)	0.086(0.006)
			3	<b>0.037(0.005)</b>	<b>0.032(0.003)</b>	0.065(0.002)	0.043(0.005)	0.086(0.008)	0.243(0.037)	0.090(0.006)
			4	<b>0.060(0.006)</b>	<b>0.051(0.007)</b>	0.077(0.002)	<b>0.045(0.005)</b>	0.071(0.005)	0.213(0.029)	0.074(0.006)

Table 3.7: Mean and standard error of the root mean squared error over 30 trials for benchmark datasets using  $k$ -nearest neighbour regressor. The best method in terms of the mean error and comparable methods according to the paired  $t$ -test at the significance level 5% are specified by bold face.

Dataset	$N_{tr}$	$d_{\tilde{x}}$	$d_z$	LSQMID	LSQMI	LSDR	LSCE	dMAVE	KDR (gKDR)	KDR (Random)
Fertility	50	14	1	1.875(0.154)	<b>1.467(0.103)</b>	2.330(0.146)	<b>1.451(0.124)</b>	2.162(0.149)	<b>1.367(0.117)</b>	<b>1.440(0.121)</b>
			2	1.581(0.107)	<b>1.387(0.100)</b>	1.998(0.107)	<b>1.344(0.102)</b>	2.206(0.120)	<b>1.407(0.130)</b>	1.718(0.140)
			3	<b>1.517(0.119)</b>	<b>1.383(0.103)</b>	1.794(0.117)	1.661(0.149)	1.953(0.140)	<b>1.439(0.102)</b>	1.677(0.126)
			4	1.546(0.100)	<b>1.236(0.091)</b>	1.842(0.139)	1.696(0.126)	1.759(0.105)	1.575(0.124)	1.655(0.115)
Yacht	100	11	1	<b>0.020(0.002)</b>	0.368(0.049)	0.031(0.003)	0.629(0.077)	0.040(0.005)	<b>0.019(0.002)</b>	<b>0.018(0.002)</b>
			2	<b>0.026(0.003)</b>	0.510(0.078)	0.194(0.022)	0.147(0.011)	0.101(0.011)	0.201(0.029)	0.191(0.017)
			3	<b>0.171(0.021)</b>	0.577(0.048)	0.311(0.022)	0.257(0.020)	<b>0.186(0.014)</b>	0.319(0.026)	0.337(0.024)
			4	<b>0.369(0.031)</b>	0.674(0.066)	0.422(0.025)	<b>0.344(0.026)</b>	<b>0.324(0.023)</b>	0.437(0.025)	0.459(0.034)
Concrete	200	13	1	0.411(0.019)	0.391(0.017)	0.379(0.009)	0.382(0.010)	<b>0.356(0.007)</b>	0.669(0.048)	0.428(0.016)
			2	0.343(0.010)	0.369(0.009)	0.345(0.009)	0.349(0.013)	<b>0.307(0.010)</b>	0.404(0.033)	<b>0.316(0.019)</b>
			3	<b>0.356(0.012)</b>	0.373(0.013)	0.375(0.012)	0.381(0.012)	<b>0.347(0.010)</b>	0.401(0.018)	0.388(0.014)
			4	<b>0.369(0.012)</b>	0.525(0.034)	0.398(0.013)	0.397(0.014)	<b>0.382(0.012)</b>	0.440(0.014)	0.448(0.015)
Breast-cancer	200	15	1	0.203(0.009)	0.279(0.019)	0.209(0.010)	0.233(0.019)	<b>0.139(0.006)</b>	0.224(0.013)	0.234(0.015)
			2	<b>0.199(0.010)</b>	0.236(0.012)	<b>0.198(0.011)</b>	<b>0.221(0.017)</b>	<b>0.190(0.011)</b>	<b>0.215(0.013)</b>	<b>0.208(0.011)</b>
			3	<b>0.145(0.005)</b>	0.218(0.011)	0.180(0.008)	0.202(0.012)	0.197(0.010)	0.195(0.010)	0.197(0.011)
			4	<b>0.140(0.004)</b>	0.179(0.008)	0.187(0.008)	0.194(0.014)	0.193(0.011)	0.189(0.011)	0.187(0.010)
Bike	300	19	1	<b>0.007(0.004)</b>	0.016(0.006)	<b>0.001(0.000)</b>	<b>0.001(0.000)</b>	<b>0.104(0.052)</b>	0.390(0.075)	0.042(0.002)
			2	<b>0.006(0.001)</b>	<b>0.005(0.001)</b>	<b>0.006(0.001)</b>	<b>0.007(0.002)</b>	<b>0.006(0.001)</b>	0.167(0.051)	0.018(0.001)
			3	<b>0.008(0.002)</b>	<b>0.007(0.002)</b>	<b>0.009(0.001)</b>	0.011(0.001)	<b>0.009(0.001)</b>	0.123(0.035)	0.037(0.001)
			4	<b>0.018(0.003)</b>	<b>0.019(0.003)</b>	<b>0.019(0.002)</b>	<b>0.019(0.002)</b>	<b>0.014(0.001)</b>	0.107(0.019)	0.055(0.002)

as follows. Let  $g_{\ell,k}(\mathbf{z}, \mathbf{y})$  be a model of the  $k$ -th order derivatives of the density difference, i.e.,

$$g_{\ell,k}(\mathbf{z}, \mathbf{y}) \approx \frac{\partial^k f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}}. \quad (3.78)$$

A least-squares estimator minimizes the following squared loss:

$$\frac{1}{2} \iint \left( g_{\ell,k}(\mathbf{z}, \mathbf{y}) - \frac{\partial^k f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} \right)^2 d\mathbf{z}d\mathbf{y}. \quad (3.79)$$

By expanding the square and ignoring the constant term, we obtain

$$\frac{1}{2} \iint g_{\ell,k}(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} - \iint g_{\ell,k}(\mathbf{z}, \mathbf{y}) \frac{\partial^k f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} d\mathbf{z}d\mathbf{y}. \quad (3.80)$$

Under the mild assumption, applying the integration by parts for  $k$  times yields

$$\iint g_{\ell,k}(\mathbf{z}, \mathbf{y}) \frac{\partial^k f(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} d\mathbf{z}d\mathbf{y} = (-1)^k \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial^k g_{\ell,k}(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} d\mathbf{z}d\mathbf{y}. \quad (3.81)$$

Then, the estimator  $g_{\ell,k}(\mathbf{z}, \mathbf{y})$  is obtained as a solution of the following minimization problem:

$$\min_{g_{\ell,k}} \left[ \frac{1}{2} \iint g_{\ell,k}(\mathbf{z}, \mathbf{y})^2 d\mathbf{z}d\mathbf{y} - (-1)^k \iint f(\mathbf{z}, \mathbf{y}) \frac{\partial^k g_{\ell,k}(\mathbf{z}, \mathbf{y})}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} d\mathbf{z}d\mathbf{y} \right]. \quad (3.82)$$

For a linear-in-parameter model  $g_{\ell,k}(\mathbf{z}, \mathbf{y}) = \boldsymbol{\theta}_{\ell,k}^\top \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}, \mathbf{y})$ , the parameter  $\widehat{\boldsymbol{\theta}}_{\ell,k}$  which minimizes a regularized empirical minimization problem can be obtained analytically as

$$\widehat{\boldsymbol{\theta}}_{\ell,k} = (-1)^k (\mathbf{H}_{\ell,k} + \lambda_{\ell,k} \mathbf{I})^{-1} \widehat{\mathbf{h}}_{\ell,k}, \quad (3.83)$$

where  $\lambda_{\ell,k} \geq 0$  denotes the regularization parameter and

$$\mathbf{H}_{\ell,k} = \iint \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}, \mathbf{y}) \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}, \mathbf{y})^\top d\mathbf{z}d\mathbf{y}, \quad (3.84)$$

$$\widehat{\mathbf{h}}_{\ell,k} = \frac{1}{N} \sum_{i=1}^N \frac{\partial^k \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}_i, \mathbf{y}_i)}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}} - \frac{1}{N^2} \sum_{i,j=1}^N \frac{\partial^k \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}_i, \mathbf{y}_j)}{\partial z^{(\ell_1)} \dots \partial z^{(\ell_k)}}. \quad (3.85)$$

Finally, an estimator of the  $k$ -th order derivative of the density difference is given by

$$\widehat{g}_{\ell,k}(\mathbf{z}, \mathbf{y}) = \widehat{\boldsymbol{\theta}}_{\ell,k}^\top \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}, \mathbf{y}). \quad (3.86)$$

Substituting this estimator to the  $k$ -th order of the derivative of QMI in Eq.(3.77) yields

$$\begin{aligned} \frac{\widehat{\partial^k \text{QMI}(\mathbf{W})}}{\partial W_{\ell_1, \ell'_1} \dots \partial W_{\ell_k, \ell'_k}} &= \frac{1}{N} \sum_{i=1}^N \widehat{\boldsymbol{\theta}}_{\ell,k}^\top \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}_i, \mathbf{y}_i) \left( \prod_{m=1}^{k'} x_i^{(\ell'_m)} \right) \\ &\quad - \frac{1}{N^2} \sum_{i,j=1}^N \widehat{\boldsymbol{\theta}}_{\ell,k}^\top \boldsymbol{\varphi}_{\ell,k}(\mathbf{z}_i, \mathbf{y}_j) \left( \prod_{m=1}^{k'} x_i^{(\ell'_m)} \right). \end{aligned} \quad (3.87)$$

We may also use the  $k$ -th order derivative of the Gaussian function as the basis function. In such a case, the Gaussian width and the regularization parameter can be objectively chosen by the  $K$ -fold cross-validation procedure in Section 3.3.5 where the score of each candidate pair  $M$  is

$$\text{CV}(M) = \frac{1}{K} \sum_{j=1}^K \left[ \frac{1}{2} \widehat{\boldsymbol{\theta}}_{\ell,k,M,\setminus j}^\top \mathbf{H}_{\ell,k,M} \widehat{\boldsymbol{\theta}}_{\ell,k,M,\setminus j} + \widehat{\boldsymbol{\theta}}_{\ell,k,M,\setminus j}^\top \widehat{\mathbf{h}}_{\ell,k,M,j} \right]. \quad (3.88)$$

It should be noted that we implicitly assume that the density difference is at least  $k$  times differentiable for estimating the  $k$ -th order derivatives of the density difference. Moreover, we also implicitly assume that the derivatives are smooth and can be accurately estimated by a linear combination of smooth basis functions such as the derivatives of the Gaussian function.

### 3.7 Conclusion

In this chapter, we proposed a novel linear supervised dimension reduction method based on maximization of quadratic mutual information (QMI). Our key idea was to *directly* estimate the derivative of QMI in a single-step manner without estimating QMI itself. We firstly developed a method to directly estimate the derivative of QMI, and then developed fixed-point iteration which efficiently uses the derivative estimator to find a maximizer of QMI. In addition to the robustness against outliers thanks to the property of QMI, the proposed method is widely applicable because it does not require any assumption on the data distribution and tuning parameters can be objectively chosen via cross-validation. The experiment results on artificial and benchmark datasets showed that the proposed method is promising.

The proposed method seems to be computationally expensive. The main reason for this inefficiency is that we restart the optimization with several initial guesses in order to avoid obtaining a poor local solution. Our future work includes a computationally more efficient approach to obtain a better solution; exploring *geodesic convexity* (Udriste, 1994) would also be an interesting direction.

The experimental results on the artificial datasets showed that the performance of LSQMI, which aims at maximizing an estimated QMI, decreases significantly when the input dimensionality increases. Our single-step method significantly improves the performance of the QMI-based supervised dimension reduction approach by directly estimating the derivative of QMI. On the other hand, the performance of LSDR, which aims at maximizing an estimated SMI, does not affect much when the input dimensionality increases. Hence, by the same analogy, it is intuitive to say that a single-step supervised dimension reduction method based on direct estimation of the derivative of SMI would work even better than LSDR. Developing a single-step method which directly estimates the derivative of SMI will be our future work.

In this chapter, we proposed the single-step QMI-based linear dimension reduction method and describe its procedure for supervised linear dimension reduction setting. However, it is straightforward to apply our proposed method to unsupervised linear dimension reduction setting by replacing the output  $\mathbf{y}$  with an identical copy of input  $\mathbf{x}$ . Our future work includes investigating the performance of the proposed method in unsupervised linear dimension reduction setting. Nonetheless, we expect that the performance of our single-step method would be better than existing methods based on multi-step approaches

## Chapter 4

# Single-step Dimension Reduction for Conditional Density Estimation

This chapter presents our second contribution on a single-step dimension reduction method for conditional density estimation. We firstly introduce the conditional density estimation problem and briefly review existing methods. Then, we present our single-step dimension reduction method. Lastly, we present experimental evaluations and conclude the chapter.

### 4.1 Introduction

Analyzing input-output relationship from input-output data is one of most important problem in supervised learning. The most common approach is *regression*, which estimates the conditional mean of output  $\mathbf{y}$  given input  $\mathbf{x}$ , i.e., regression estimates  $\mathbb{E}_{p(\mathbf{y}|\mathbf{x})}[\mathbf{y}]$ . However, just analyzing the conditional mean is not informative enough when the conditional probability density function  $p(\mathbf{y}|\mathbf{x})$  possesses multimodality, asymmetry, or heteroscedasticity (i.e., input-dependent variance). In such cases, it would be more appropriate to estimate the conditional density  $p(\mathbf{y}|\mathbf{x})$  itself. (see Figure 4.3).

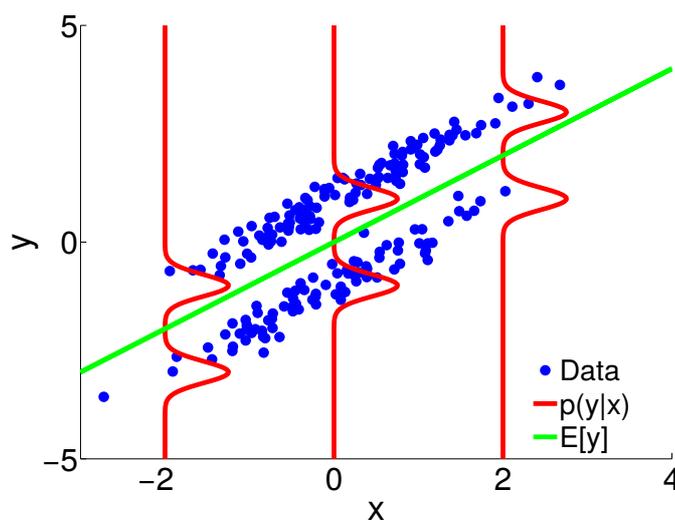


Figure 4.1: Illustration of bi-modal data. The conditional mean (green line) poorly capture information in the data.

The most naive approach to solve the *conditional density estimation* problem would be  $\epsilon$ -neighbor kernel density estimation ( $\epsilon$ -KDE), which performs standard KDE along  $\mathbf{y}$  only with nearby data points in the input domain. However,  $\epsilon$ -KDE do not work well when input has high dimensionality because the number of nearby data points is too few. To avoid this issue, KDE may be applied twice to estimate  $p(\mathbf{x}, \mathbf{y})$  and  $p(\mathbf{x})$  separately and the estimated densities may be plugged into the decomposed form  $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{x}, \mathbf{y})/p(\mathbf{x})$  to estimate the conditional density. However, this approach is not reliable since taking the ratio of two estimated densities significantly magnifies the estimation error. To overcome this issue, an approach to directly estimating the density ratio  $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{x}, \mathbf{y})/p(\mathbf{x})$  without separate estimation of densities  $p(\mathbf{x}, \mathbf{y})$  and  $p(\mathbf{x})$  has been explored (Sugiyama et al., 2010). This method, called *least-squares conditional density estimation* (LSCDE), does not require a strong assumption about the conditional density and it was proved to possess the optimal non-parametric learning rate in the mini-max sense. Moreover, its solution can also be efficiently and analytically computed. Nevertheless, estimating conditional densities when input has high dimensionality is still challenging due to the curse of dimensionality.

The curse of dimensionality can be mitigated by pre-processing data with supervised linear dimension reduction before performing conditional density estimation. However, such a multi-step approach is not optimal because dimension reduction in the former step is performed without regard to conditional density estimation in the latter step, and estimation error incurred in the dimension reduction step can be magnified in the conditional density estimation step.

In this chapter, we overcome the issue of the multi-step approach. More specifically, we propose a single-step dimension reduction method which performs both supervised linear dimension reduction and conditional density estimation under a unified objective. Our key idea is to formulate supervised linear dimension reduction in terms of the *squared-loss conditional entropy* which includes the conditional density in its definition. Optimizing this squared-loss conditional entropy means that supervised linear dimension reduction is performed simultaneously with conditional density estimation. Therefore, when supervised linear dimension reduction is completed, the final conditional density estimator has already been obtained without requiring an additional conditional density estimation step.

The organization of this chapter are as follows. In Section 4.2, we firstly formulate the conditional density estimation problem and review existing methods. Next, in Section 4.3 we present the squared-loss conditional entropy and our method which is called *least-squares conditional entropy* (LSCE). Then, in Section 4.4 we demonstrate its usefulness through experiments on benchmark data, humanoid robot data, and computer art data. Finally, we conclude this chapter in Section 4.5.

## 4.2 Conditional Density Estimation

This section formulates the conditional density estimation problem and review existing methods.

### 4.2.1 Problem Formulation

The goal of conditional density estimation is to estimate the conditional probability density function  $p(\mathbf{y}|\mathbf{x})$  from input-output data  $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$  which are drawn i.i.d. from a joint probability distribution with density function  $p(\mathbf{x}, \mathbf{y})$ . In this chapter, we assume that a parametric form of the conditional density function  $p(\mathbf{y}|\mathbf{x})$  is unknown. Therefore, we do not consider parametric conditional density estimation methods such as *Gaussian process* (Rasmussen and Williams, 2006) in this chapter.

We also assume that the input  $\mathbf{x}$  can be decomposed into an informative component  $\mathbf{z}$  and an uninformative component  $\mathbf{z}_\perp$ . We further assume that there exists an orthogonal matrix  $[\mathbf{W}^\top, \mathbf{W}_\perp^\top]$  such that  $\mathbf{z} = \mathbf{W}\mathbf{x}$  and  $\mathbf{z}_\perp = \mathbf{W}_\perp\mathbf{x}$ . We emphasize that this informative component satisfies the conditional independent:

$$(\mathbf{y} \perp \mathbf{x}) | \mathbf{z}. \quad (4.1)$$

### 4.2.2 Existing Methods

In this section, we review existing (non-parametric) conditional density estimation methods.

#### $\epsilon$ -Neighbor Kernel Density Estimation

The  $\epsilon$ -neighbor kernel density estimation ( $\epsilon$ -KDE) is a simple method for estimating a conditional density function from data. Given a test input point  $\tilde{\mathbf{x}}$ ,  $\epsilon$ -KDE performs KDE using a subset of training output data points whose associated input data points are located near  $\tilde{\mathbf{x}}$ . More specifically, an estimated conditional density of a given test input point  $\tilde{\mathbf{x}}$  is computed in  $\epsilon$ -KDE as

$$\hat{p}(\mathbf{y}|\tilde{\mathbf{x}}) = \frac{1}{|\mathcal{I}_{\tilde{\mathbf{x}},\epsilon}|} \sum_{i \in \mathcal{I}_{\tilde{\mathbf{x}},\epsilon}} K(\mathbf{y} - \mathbf{y}_i), \quad (4.2)$$

where  $\mathcal{I}_{\tilde{\mathbf{x}},\epsilon}$  is a set of data indexes such that  $\|\tilde{\mathbf{x}} - \mathbf{x}_i\| \leq \epsilon$ . The function  $K(\cdot)$  is a *kernel* function which is a positive real-valued function that is integrated to one:

$$\int K(\mathbf{u}) d\mathbf{u} = 1. \quad (4.3)$$

Note that the kernel function in the context of kernel density estimation is defined quite differently from the kernel in the context of reproducing kernel Hilbert space (Aronszajn, 1950) which we introduced in Chapter 2. Among many kernel functions, the most commonly used kernel function is the Gaussian kernel which is defined as

$$K(\mathbf{y} - \tilde{\mathbf{y}}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\mathbf{y} - \tilde{\mathbf{y}})^\top(\mathbf{y} - \tilde{\mathbf{y}})}{2\sigma^2}\right). \quad (4.4)$$

The distance threshold  $\epsilon$  and the tuning parameters contained in the kernel, e.g., the Gaussian width  $\sigma$ , can be chosen by cross-validation.

The main appeal of  $\epsilon$ -KDE is that it is very simple and easy to use. However,  $\epsilon$ -KDE tends to perform very poorly when input has high dimensionality because the number of nearby data points within the  $\epsilon$  distance is usually too few.

## Ratio of Kernel Density Estimators

The conditional density function can also be estimated based on a ratio of two density functions. Firstly, notice that the conditional density function  $p(\mathbf{y}|\mathbf{x})$  can be expressed as a ratio of two (unconditional) density functions, i.e.,

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})}. \quad (4.5)$$

Based on this fact, a conditional density estimator can be obtained by

$$\hat{p}(\mathbf{y}|\mathbf{x}) = \frac{\hat{p}(\mathbf{x}, \mathbf{y})}{\hat{p}(\mathbf{x})}, \quad (4.6)$$

where  $\hat{p}(\mathbf{x}, \mathbf{y})$  and  $\hat{p}(\mathbf{x})$  are an estimated joint density function and an estimated marginal density function, respectively. In the ratio of kernel density estimators method, the estimated joint and marginal density functions are separately obtained by KDE and they are given by

$$\hat{p}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x} - \mathbf{x}_i) H(\mathbf{y} - \mathbf{y}_i) \quad (4.7)$$

$$\hat{p}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N K(\mathbf{x} - \mathbf{x}_i), \quad (4.8)$$

where  $K(\cdot)$  and  $H(\cdot)$  are kernel functions which may be the Gaussian kernel defined in Equation (4.4). Similarly to the  $\epsilon$ -KDE method, the tuning parameters in the kernel functions  $K(\cdot)$  and  $H(\cdot)$  can be chosen by cross-validation.

The important step of this approach is to compute the ratio between the estimated densities  $\hat{p}(\mathbf{x}, \mathbf{y})$  and  $\hat{p}(\mathbf{x})$ . In general, computing such a ratio is not preferable since estimation errors of the two estimated densities can be significantly magnified. For this reason, this approach is usually unreliable except when the two estimated densities are very accurate.

## Least-Squares Conditional Density Estimation

To avoid computing the ratio of two estimated density functions, the *least-squares conditional density estimation* (LSCDE) method (Sugiyama et al., 2010) was proposed. The key idea of LSCDE is to directly estimate the density ratio  $p(\mathbf{x}, \mathbf{y})/p(\mathbf{x})$  from data. More specifically, LSCDE learn a conditional density estimator  $\hat{p}(\mathbf{y}|\mathbf{x})$  such that it minimizes the following squared error:

$$\frac{1}{2} \iint (\hat{p}(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y}. \quad (4.9)$$

The solution of LSCDE depends on the model of the conditional density estimator  $\hat{p}(\mathbf{y}|\mathbf{x})$ . A practical choice is the linear-in-parameter-model:

$$\hat{p}(\mathbf{y}|\mathbf{x}) = \boldsymbol{\alpha}^\top \boldsymbol{\varphi}(\mathbf{x}, \mathbf{y}), \quad (4.10)$$

where  $\boldsymbol{\alpha} \in \mathbb{R}^b$  is a vector parameter to be learned and  $\boldsymbol{\varphi}(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^b$  is a basis function vector. For this model, the parameter vector which minimizes the squared error can

be computed in a closed form. More details of LSCDE will be explained in Chapter 6 where we utilize LSCDE in our contribution on a model-based reinforcement learning method.

The main advantage of LSCDE is that the conditional density estimator  $\hat{p}(\mathbf{y}|\mathbf{x})$  can be obtained in a closed form when a linear-in-parameter model is used as a model of the estimator. Moreover, it was proved that LSCDE achieve the optimal non-parametric asymptotic convergence rate (Sugiyama et al., 2010).

### 4.2.3 Multi-step Dimension Reduction for Conditional Density Estimation

Although LSCDE possesses many desirable properties, it still suffers from the curse of dimensionality and may perform poorly when input has high dimensionality. A common approach to mitigate this issue is to pre-process the input  $\mathbf{x}$  by a supervised linear dimension reduction method before performing LSCDE. For example, we may use any supervised linear dimension reduction methods that we introduced in Chapter 2 or our proposed LSQMID method in Chapter 3 to firstly transform input  $\mathbf{x}$  into new input  $\mathbf{z} = \mathbf{W}\mathbf{x}$ , and then apply LSCDE on the new input-output data  $\{(\mathbf{z}_i, \mathbf{y}_i)\}_{i=1}^N$ .

Although this multi-step approach is a common practice, it is not an appropriate approach for the following reason. Typically, the former dimension reduction step is performed independently without regarding the accuracy of the latter conditional density estimation step. In an ideal case, supervised linear dimension reduction could find an optimal transformation matrix  $\mathbf{W}$  such that the new input  $\mathbf{z} = \mathbf{W}\mathbf{x}$  satisfies the conditional independence:

$$(\mathbf{y} \perp \mathbf{x})|\mathbf{z}, \quad (4.11)$$

or equivalently

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{z}). \quad (4.12)$$

In this ideal case, the latter conditional density estimation step can use the new input  $\mathbf{z}$  without any loss of information. In practice, however, the transformation matrix  $\mathbf{W}$  often contains estimation error and is quite different from the optimal transformation matrix that satisfies the conditional independence. In this case, the learned conditional density estimator  $\hat{p}(\mathbf{y}|\mathbf{z})$  would be inaccurate and quite different from the conditional density  $p(\mathbf{y}|\mathbf{x})$ .

A simple idea to avoid the issue of the multi-step approach is to perform linear dimension reduction while also considering an accuracy of the conditional density estimator. In the next section, we show that this idea can be mathematically formalized by using the definition of squared-loss conditional entropy.

## 4.3 Least-Squares Conditional Entropy

In this section, we present our single-step method that overcomes the issue of multi-step approach. Our key idea is to minimize the *squared-loss conditional entropy* in order to obtain both transformation matrix  $\mathbf{W}$  and the conditional density estimator  $\hat{p}(\mathbf{y}|\mathbf{W}\mathbf{x})$ . Below, we firstly introduce the squared-loss conditional entropy and explain its relation to the conditional independence. Then, we show that it can be utilized as a unified objective for both supervised linear dimension reduction and conditional density estimation.

### 4.3.1 Squared-loss Conditional Entropy and conditional Independence

Let us consider a squared-loss variant of conditional entropy named *squared-loss conditional entropy* (SCE):

$$\text{SCE}(\mathbf{Y}|\mathbf{Z}) = -\frac{1}{2} \iint \left( p(\mathbf{y}|\mathbf{z}) - 1 \right)^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y}. \quad (4.13)$$

By expanding the squared term in Equation (4.13), we obtain

$$\begin{aligned} \text{SCE}(\mathbf{Y}|\mathbf{Z}) &= -\frac{1}{2} \iint \left( p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) + p(\mathbf{y}|\mathbf{z}) p(\mathbf{z}) - \frac{1}{2} p(\mathbf{z}) \right) d\mathbf{z} d\mathbf{y} \\ &= -\frac{1}{2} \iint p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y} + 1 - \frac{1}{2} \int d\mathbf{y} \\ &= \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) + 1 - \frac{1}{2} \int d\mathbf{y}, \end{aligned} \quad (4.14)$$

where  $\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z})$  is defined as

$$\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) = -\frac{1}{2} \iint p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y}. \quad (4.15)$$

Then we have the following theorem which forms the basis of our proposed method:

**Theorem 1.**

$$\begin{aligned} \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) - \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{X}) &= \frac{1}{2} \iint \left( \frac{p(\mathbf{z}_\perp, \mathbf{y}|\mathbf{z})}{p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{y}|\mathbf{z})} - 1 \right)^2 p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y} \\ &\geq 0. \end{aligned} \quad (4.16)$$

The above theorem can be proved as follows. Firstly, we can see that  $\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z})$  and  $\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{X})$  have the following relation:

$$\begin{aligned} \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) - \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{X}) &= \frac{1}{2} \iint p(\mathbf{y}|\mathbf{x})^2 p(\mathbf{x}) d\mathbf{y} d\mathbf{x} - \frac{1}{2} \iint p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y} \\ &= \frac{1}{2} \iint p(\mathbf{y}|\mathbf{x})^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y} + \frac{1}{2} \iint p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y} \\ &\quad - \iint p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y}. \end{aligned} \quad (4.17)$$

Let  $p(\mathbf{x}) = p(\mathbf{z}, \mathbf{z}_\perp)$ ,  $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{z}, \mathbf{z}_\perp, \mathbf{y})$ , and  $d\mathbf{x} = d\mathbf{z} d\mathbf{z}_\perp$ . Then the final term can be expressed as

$$\begin{aligned} \iint p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y} &= \iint \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})} \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})} p(\mathbf{z}) d\mathbf{z} d\mathbf{y} \\ &= \iint \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})} p(\mathbf{z}, \mathbf{y}) d\mathbf{z} d\mathbf{y} \\ &= \iint \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})} p(\mathbf{z}_\perp|\mathbf{z}, \mathbf{y}) p(\mathbf{z}, \mathbf{y}) d\mathbf{z} d\mathbf{z}_\perp d\mathbf{y} \\ &= \iint \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})} p(\mathbf{z}, \mathbf{z}_\perp, \mathbf{y}) d\mathbf{z} d\mathbf{z}_\perp d\mathbf{y} \\ &= \iint \frac{p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z})} p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= \iint p(\mathbf{y}|\mathbf{z}) p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} d\mathbf{y}. \end{aligned} \quad (4.18)$$

Therefore,

$$\begin{aligned}
\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) - \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{X}) &= \frac{1}{2} \iint p(\mathbf{y}|\mathbf{x})^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y} + \frac{1}{2} \iint p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{z}) d\mathbf{z} d\mathbf{y} \\
&\quad - \iint p(\mathbf{y}|\mathbf{z}) p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} d\mathbf{y} \\
&= \frac{1}{2} \iint (p(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{z}))^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y}. \tag{4.19}
\end{aligned}$$

We can also express  $p(\mathbf{y}|\mathbf{x})$  in term of  $p(\mathbf{y}|\mathbf{z})$  as

$$\begin{aligned}
p(\mathbf{y}|\mathbf{x}) &= \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})} \\
&= \frac{p(\mathbf{x}, \mathbf{y}) p(\mathbf{z}, \mathbf{y})}{p(\mathbf{x}) p(\mathbf{z}, \mathbf{y})} \\
&= \frac{p(\mathbf{x}, \mathbf{y}) p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{z}) p(\mathbf{y}|\mathbf{z}) p(\mathbf{z})} \\
&= \frac{p(\mathbf{z}, \mathbf{z}_\perp, \mathbf{y}) p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{z}) p(\mathbf{y}|\mathbf{z}) p(\mathbf{z})} \\
&= \frac{p(\mathbf{z}_\perp, \mathbf{y}|\mathbf{z}) p(\mathbf{z}, \mathbf{y})}{p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{y}|\mathbf{z}) p(\mathbf{z})} \\
&= \frac{p(\mathbf{z}_\perp, \mathbf{y}|\mathbf{z})}{p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{y}|\mathbf{z})} p(\mathbf{y}|\mathbf{z}). \tag{4.20}
\end{aligned}$$

Finally, we obtain

$$\begin{aligned}
\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) - \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{X}) &= \frac{1}{2} \iint (p(\mathbf{y}|\mathbf{x}) - p(\mathbf{y}|\mathbf{z}))^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y} \\
&= \frac{1}{2} \iint \left( \frac{p(\mathbf{z}_\perp, \mathbf{y}|\mathbf{z})}{p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{y}|\mathbf{z})} p(\mathbf{y}|\mathbf{z}) - p(\mathbf{y}|\mathbf{z}) \right)^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y} \\
&= \frac{1}{2} \iint \left( \frac{p(\mathbf{z}_\perp, \mathbf{y}|\mathbf{z})}{p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{y}|\mathbf{z})} - 1 \right)^2 p(\mathbf{y}|\mathbf{z})^2 p(\mathbf{x}) d\mathbf{x} d\mathbf{y} \\
&\geq 0, \tag{4.21}
\end{aligned}$$

which concludes the proof.

The importance of the above theorem is that it shows that  $\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) \geq \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{X})$ , and that the equality holds if and only if

$$p(\mathbf{z}_\perp, \mathbf{y}|\mathbf{z}) = p(\mathbf{z}_\perp|\mathbf{z}) p(\mathbf{y}|\mathbf{z}). \tag{4.22}$$

This is equivalent to the conditional independence in Equation (4.11), and therefore supervised linear dimensionality reduction can be performed by minimizing  $\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z})$  with respect to  $\mathbf{W}$ :

$$\begin{aligned}
&\min_{\mathbf{W} \in \mathbb{R}^{d_{\mathbf{z}} \times d_{\mathbf{x}}}} \widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) \\
&\text{subject to } \mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_{\mathbf{z}}}. \tag{4.23}
\end{aligned}$$

Since  $p(\mathbf{y}|\mathbf{z}) = p(\mathbf{z}, \mathbf{y})/p(\mathbf{z})$ ,  $\text{SCE}(\mathbf{Y}|\mathbf{Z})$  is equivalent to the negative *Pearson divergence* (Pearson, 1900) from  $p(\mathbf{z}, \mathbf{y})$  to  $p(\mathbf{z})$ , which is a member of the *f-divergence* class (Ali and Silvey, 1966; Csiszár, 1967) with the squared-loss function. On the other hand, ordinary conditional entropy (CE), defined by

$$\text{CE}(\mathbf{Y}|\mathbf{Z}) = - \iint p(\mathbf{z}, \mathbf{y}) \log p(\mathbf{y}|\mathbf{z}) d\mathbf{z} d\mathbf{y}, \quad (4.24)$$

is the negative *Kullback-Leibler* (KL) divergence (Kullback and Leibler, 1951) from  $p(\mathbf{z}, \mathbf{y})$  to  $p(\mathbf{z})$ . This analogy is similar to the case of the *squared-loss mutual information* (SMI) and the *mutual information* (MI) which we introduced in Chapter 2 where SMI is an instance of the Pearson divergence and MI is an instance of the KL divergence. As we have discussed earlier in Chapter 2, the Pearson divergence is more appealing than the KL divergence since it is more robust against outliers than the KL divergence.

Next, we propose our approach to estimate the unknown SCE. Then, we show that our SCE estimation is related to performing conditional density estimation.

### 4.3.2 Estimating Squared-loss Conditional Entropy

Since  $\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z})$  in Equation (4.23) is unknown in practice, we estimate it using input-output data  $\{(\mathbf{z}_i, \mathbf{y}_i) \mid \mathbf{z}_i = \mathbf{W} \mathbf{x}_i\}_{i=1}^N$ .

The trivial inequality  $(a - b)^2/2 \geq 0$  yields  $a^2/2 \geq ab - b^2/2$ , and thus we have

$$\frac{a^2}{2} = \max_b \left[ ab - \frac{b^2}{2} \right]. \quad (4.25)$$

If we set  $a = p(\mathbf{y}|\mathbf{z})$ , we have

$$\frac{p(\mathbf{y}|\mathbf{z})^2}{2} \geq \max_b \left[ p(\mathbf{y}|\mathbf{z})b(\mathbf{z}, \mathbf{y}) - \frac{b(\mathbf{z}, \mathbf{y})^2}{2} \right]. \quad (4.26)$$

If we multiply both sides of the above inequality with  $-p(\mathbf{z})$ , and integrated over  $\mathbf{z}$  and  $\mathbf{y}$ , we have

$$\widetilde{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) \leq \min_b \iint \left[ \frac{b(\mathbf{z}, \mathbf{y})^2 p(\mathbf{z})}{2} - b(\mathbf{z}, \mathbf{y}) p(\mathbf{z}, \mathbf{y}) \right] d\mathbf{z} d\mathbf{y}, \quad (4.27)$$

where minimization with respect to  $b$  is now performed as a function of  $\mathbf{z}$  and  $\mathbf{y}$ .

Now, let us consider a linear-in-parameter model for the function  $b$ :

$$b(\mathbf{z}, \mathbf{y}) = \boldsymbol{\alpha}^\top \boldsymbol{\varphi}(\mathbf{z}, \mathbf{y}), \quad (4.28)$$

where  $\boldsymbol{\alpha} \in \mathbb{R}^b$  is a parameter vector to be learned and  $\boldsymbol{\varphi}(\mathbf{z}, \mathbf{y}) \in \mathbb{R}^b$  is a basis function vector. If the expectations over densities  $p(\mathbf{z})$  and  $p(\mathbf{z}, \mathbf{y})$  are approximated by samples averages and the  $\ell_2$ -regularizer  $\lambda \boldsymbol{\alpha}^\top \boldsymbol{\alpha} / 2$  ( $\lambda \geq 0$ ) is included, the above minimization problem yields

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\text{argmin}} \left[ \frac{1}{2} \boldsymbol{\alpha}^\top \hat{\mathbf{G}} \boldsymbol{\alpha} - \hat{\mathbf{h}}^\top \boldsymbol{\alpha} + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \boldsymbol{\alpha} \right], \quad (4.29)$$

where

$$\begin{aligned}\widehat{\mathbf{G}} &= \frac{1}{N} \sum_{i=1}^N \bar{\Phi}(z_i), \\ \widehat{\mathbf{h}} &= \frac{1}{N} \sum_{i=1}^N \varphi(z_i, \mathbf{y}_i), \\ \bar{\Phi}(z) &= \int \varphi(z, \mathbf{y}) \varphi(z, \mathbf{y})^\top d\mathbf{y}.\end{aligned}\tag{4.30}$$

The solution  $\widehat{\boldsymbol{\alpha}}$  is analytically given by

$$\widehat{\boldsymbol{\alpha}} = \left( \widehat{\mathbf{G}} + \lambda \mathbf{I} \right)^{-1} \widehat{\mathbf{h}},\tag{4.31}$$

which yields  $\widehat{b}(z, \mathbf{y}) = \widehat{\boldsymbol{\alpha}}^\top \varphi(z, \mathbf{y})$ . Then, from Equation (4.27), an estimator of  $\widehat{\text{SCE}}(\mathbf{Y}|\mathbf{Z})$  is obtained analytically as

$$\widehat{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) = \frac{1}{2} \widehat{\boldsymbol{\alpha}}^\top \widehat{\mathbf{G}} \widehat{\boldsymbol{\alpha}} - \widehat{\mathbf{h}}^\top \widehat{\boldsymbol{\alpha}}.\tag{4.32}$$

We call this method *least-squares conditional entropy* (LSCE).

### 4.3.3 Supervised Linear Dimension Reduction with Squared-loss Conditional Entropy

Our current goal is to find a transformation matrix  $\mathbf{W}$  which maximizes our SCE estimator:

$$\begin{aligned}\min_{\mathbf{W} \in \mathbb{R}^{d_z \times d_x}} \quad & \widehat{\text{SCE}}(\mathbf{Y}|\mathbf{Z}) \\ \text{subject to} \quad & \mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_z}.\end{aligned}\tag{4.33}$$

This optimization problem may be solved by performing gradient descent to obtain intermediate solutions and then projecting the intermediate solution so that they satisfies the constraint  $\mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_z}$ . However, performing such a projection can be computationally expensive. An alternative approach is to perform optimization on matrix manifold. Firstly, as we discussed in Chapter 2, the optimization problem in Equation (4.33) is the same as the following optimization problem:

$$\min_{\mathbf{W} \in \mathcal{S}_{d_z}^{d_x}} \widehat{\text{SCE}}(\mathbf{Y}|\mathbf{Z}),\tag{4.34}$$

where  $\mathcal{S}_{d_z}^{d_x}$  is the Stiefel manifold (Stiefel, 1935; Edelman et al., 1998; Absil et al., 2008) defined as

$$\mathcal{S}_{d_z}^{d_x} = \{ \mathbf{W} \in \mathbb{R}^{d_z \times d_x} \mid \mathbf{W} \mathbf{W}^\top = \mathbf{I}_{d_z} \}.\tag{4.35}$$

This optimization problem can be solved by performing gradient descent on the manifold.

It can also be noticed that rotations within the subspace are not important in linear dimension reduction. More specifically, rotations within the subspace do not change

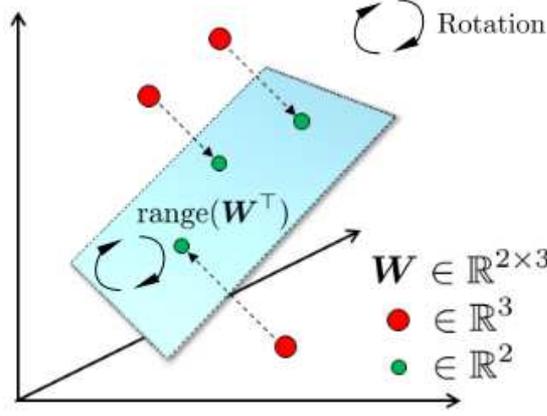


Figure 4.2: Illustration of rotation within subspace. Linear dimension reduction finds a transformation matrix  $\mathbf{W}$  and linearly project data onto the subspace spanned by the rows of  $\mathbf{W}$ . As can be seen, rotation within the subspace does not alter the positions of the projected data points (the green points).

orthogonal projection of the data points (see Figure 4.2). Therefore, instead of considering the optimization problem over the Stiefel manifold  $S_{d_z}^{d_x}$ , we may instead consider an optimization problem over the Grassmann manifold  $G_{d_z}^{d_x}$  defined as

$$\mathcal{G}_{d_z}^{d_x} = \{\mathbf{W} \in \mathbb{R}^{d_x \times d_z} \mid \mathbf{W}\mathbf{W}^\top = \mathbf{I}_{d_z}\} / \sim, \quad (4.36)$$

where  $\sim$  represents the equivalence relation:  $\mathbf{W}$  and  $\widetilde{\mathbf{W}}$  are written as  $\mathbf{W} \sim \widetilde{\mathbf{W}}$  if their rows span the same subspace. Performing the optimization over the Grassmann manifold is more computationally efficient since updates which rotates the subspace within itself are not considered.

Based on the Grassmann manifold, we aim to solve the following optimization problem:

$$\min_{\mathbf{W} \in \mathcal{G}_{d_z}^{d_x}} \widehat{\text{SCE}}(\mathbf{Y} \mid \mathbf{Z}). \quad (4.37)$$

We propose to solve this optimization by performing gradient descent on the Grassmann manifold. Firstly, the  $(l, l')$ -th entry of the gradient of  $\widehat{\text{SCE}}(\mathbf{Y} \mid \mathbf{Z})$  w.r.t.  $\mathbf{W}$  is given by

$$\frac{\partial \widehat{\text{SCE}}}{\partial W_{l,l'}} = \widehat{\boldsymbol{\alpha}}^\top \frac{\partial \widehat{\mathbf{G}}}{\partial W_{l,l'}} \left( \frac{3}{2} \widehat{\boldsymbol{\alpha}} - \widehat{\boldsymbol{\beta}} \right) + \frac{\partial \widehat{\mathbf{h}}^\top}{\partial W_{l,l'}} (\widehat{\boldsymbol{\beta}} - 2\widehat{\boldsymbol{\alpha}}), \quad (4.38)$$

where  $\widehat{\boldsymbol{\beta}} = (\widehat{\mathbf{G}} + \lambda \mathbf{I})^{-1} \widehat{\mathbf{G}} \widehat{\boldsymbol{\alpha}}$ . In the Euclidean space, the above gradient gives the steepest direction. However, on a manifold, the *natural gradient* (Amari, 1998) gives the steepest direction. The natural gradient  $\nabla \widehat{\text{SCE}}(\mathbf{W})$  at  $\mathbf{W}$  is the projection of the ordinary gradient  $\frac{\partial \widehat{\text{SCE}}}{\partial W_{l,l'}}$  to the tangent space of  $\mathcal{G}_{d_z}^{d_x}$  at  $\mathbf{W}$ . If the tangent space is equipped with the canonical metric  $\langle \mathbf{W}, \mathbf{W}' \rangle = \frac{1}{2} \text{tr}(\mathbf{W}^\top \mathbf{W}')$ , the natural gradient is given as follows (Edelman et al., 1998):

$$\nabla \widehat{\text{SCE}} = \frac{\partial \widehat{\text{SCE}}}{\partial \mathbf{W}} - \frac{\partial \widehat{\text{SCE}}}{\partial \mathbf{W}} \mathbf{W}^\top \mathbf{W} = \frac{\partial \widehat{\text{SCE}}}{\partial \mathbf{W}} \mathbf{W}_\perp^\top \mathbf{W}_\perp, \quad (4.39)$$

where  $\mathbf{W}_\perp$  is a  $(d_x - d_z) \times d_x$  matrix such that  $[\mathbf{W}^\top, \mathbf{W}_\perp^\top]$  is an orthogonal matrix. Then, the *geodesic* from  $\mathbf{W}$  to the direction of the natural gradient  $\nabla \widehat{\text{SCE}}$  over  $\mathcal{G}_{d_z}^{d_x}$  can be expressed using  $t \in \mathbb{R}$  as

$$\mathbf{W}_t = \begin{bmatrix} \mathbf{I}_{d_z} & \mathbf{O}_{d_z, (d_x - d_z)} \end{bmatrix} \times \exp \left( -t \begin{bmatrix} \mathbf{O}_{d_z, d_z} & \frac{\partial \widehat{\text{SCE}}}{\partial \mathbf{W}} \mathbf{W}_\perp^\top \\ -\mathbf{W}_\perp \frac{\partial \widehat{\text{SCE}}}{\partial \mathbf{W}}^\top & \mathbf{O}_{d_x - d_z, d_x - d_z} \end{bmatrix} \right) \begin{bmatrix} \mathbf{W} \\ \mathbf{W}_\perp \end{bmatrix}, \quad (4.40)$$

where “exp” for a matrix denotes the matrix exponential and  $\mathbf{O}_{d,d'}$  denotes the  $d \times d'$  zero matrix. Note that the derivative  $\partial_t \mathbf{W}_t$  at  $t = 0$  coincides with the natural gradient  $\nabla \widehat{\text{SCE}}$  (Edelman et al., 1998). Thus, line search along the geodesic in the natural gradient direction is equivalent to finding the minimizer from  $\{\mathbf{W}_t \mid t \geq 0\}$ .

Once  $\mathbf{W}$  is updated, SCE is re-estimated with the new  $\mathbf{W}$  and gradient descent is performed again. This entire procedure is repeated until  $\mathbf{W}$  converges. It should be noted that the optimization problem in Equation (4.37) is non-convex. To avoid obtaining a poor local solution, the gradient descent procedure is executed 20 times with randomly chosen initial solutions and the one achieving the smallest value of  $\widehat{\text{SCE}}$  is chosen.

#### 4.3.4 Conditional Density Estimation with Squared-loss Conditional Entropy

Next, we show that an estimated SCE can be used to obtain a conditional density estimator. Firstly, it can be seen that the maximum of Equation (4.25) is attained at  $b = a$  and  $a = p(\mathbf{y}|\mathbf{z})$ . Thus, the optimal  $b(\mathbf{z}, \mathbf{y})$  is actually the conditional density  $p(\mathbf{y}|\mathbf{z})$  itself. Therefore,  $\widehat{\alpha}^\top \varphi(\mathbf{z}, \mathbf{y})$  obtained by LSCE is a conditional density estimator. In order to make sure that the estimator  $\widehat{\alpha}^\top \varphi(\mathbf{z}, \mathbf{y})$  is a proper conditional density, we may post-process the solution and normalized the estimator as

$$\widehat{p}(\mathbf{y}|\mathbf{z} = \tilde{\mathbf{z}}) = \frac{\tilde{\alpha}^\top \varphi(\tilde{\mathbf{z}}, \mathbf{y})}{\int \tilde{\alpha}^\top \varphi(\tilde{\mathbf{z}}, \mathbf{y}') d\mathbf{y}'}, \quad (4.41)$$

where  $\tilde{\alpha} = \max(\widehat{\alpha}, 0)$  and the maximum operator is applied in an element-wise manner. Interestingly, the conditional density estimator obtained by LSCE is equivalent to the conditional density estimator obtained by LSCDE when the linear-in-parameter model and data  $\{(\mathbf{z}_i, \mathbf{y}_i)\}_{i=1}^N$  are used for LSCDE. Thus, this implies that the conditional density estimator obtained by LSCE may also achieve the same optimal convergence rate of the LSCDE solution (Sugiyama et al., 2010).

#### 4.3.5 Model Selection by Cross-Validation

The accuracy of the SCE estimator in Equation (4.32) depends on the choice of models, i.e., the basis function  $\varphi(\mathbf{z}, \mathbf{y})$  and the regularization parameter  $\lambda$ . The parameters of the basis function and the regularization parameter can be objectively selected by cross-validation as follows:

1. The training dataset  $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$  is divided into  $K$  disjoint subsets  $\{\mathcal{S}_j\}_{j=1}^K$  with (approximately) the same size.
2. **For** each model  $M$  in the candidate set,

(a) For  $j = 1, \dots, K$ ,

- i. For model  $M$ , the LSCE solution  $\widehat{b}^{(M,j)}$  is computed from  $\mathcal{S} \setminus \mathcal{S}_j$  (i.e., all samples except  $\mathcal{S}_j$ ).
- ii. Evaluate the upper bound of  $\widehat{\text{SCE}}$  obtained by  $\widehat{b}^{(M,j)}$  using the hold-out data  $\mathcal{S}_j$ :

$$\text{CV}_j(M) = \frac{1}{2|\mathcal{S}_j|} \sum_{\mathbf{z} \in \mathcal{S}_j} \int \widehat{b}^{(M,j)}(\mathbf{z}, \mathbf{y})^2 d\mathbf{y} - \frac{1}{|\mathcal{S}_j|} \sum_{(\mathbf{z}, \mathbf{y}) \in \mathcal{S}_j} \widehat{b}^{(M,j)}(\mathbf{z}, \mathbf{y}), \quad (4.42)$$

where  $|\mathcal{S}_j|$  denotes the cardinality of  $\mathcal{S}_j$ .

(b) The average score is computed as

$$\text{CV}(M) = \frac{1}{K} \sum_{j=1}^K \text{CV}_j(M). \quad (4.43)$$

3. The model that minimizes the average score is chosen:

$$\widehat{M} = \underset{M}{\text{argmin}} \text{CV}(M). \quad (4.44)$$

4. For the chosen model  $\widehat{M}$ , the LSCE solution  $\widehat{b}$  is computed from all samples  $\mathcal{S}$  and the approximator  $\widehat{\text{SCE}}(\mathbf{Y}|\mathbf{Z})$  is computed.

In the experiments, we use  $K = 5$ .

### 4.3.6 Basis Function Design

For the basis function, we use the following Gaussian function for the  $k$ -th dimension of  $\varphi(\mathbf{z}, \mathbf{y})$ :

$$\varphi_k(\mathbf{z}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{z} - \mathbf{u}_k\|^2 + \|\mathbf{y} - \mathbf{v}_k\|^2}{2\sigma^2}\right), \quad (4.45)$$

where  $(\mathbf{u}_k, \mathbf{v}_k)$  denotes the  $k$ -th Gaussian center located at  $(\mathbf{z}_k, \mathbf{y}_k)$ . When the sample size  $N$  is too large, we may use only a subset of samples as Gaussian centers. The Gaussian bandwidth  $\sigma$  is chosen by cross-validation as we explained previously. We may use different bandwidths for  $\mathbf{z}$  and  $\mathbf{y}$ , but this will increase the computation time for model selection. In our implementation, we normalize each element of  $\mathbf{z}$  and  $\mathbf{y}$  to have the unit variance in advance and then use the common bandwidth for  $\mathbf{z}$  and  $\mathbf{y}$ .

A notable advantage of using the Gaussian function is that the integral over  $\mathbf{y}$  appeared in  $\bar{\Phi}(\mathbf{z})$  (see Equation (4.30)) can be computed analytically as

$$\bar{\Phi}_{k,k'}(\mathbf{z}) = (\sqrt{\pi}\sigma)^{d_y} \exp\left(-\frac{2\|\mathbf{z} - \mathbf{u}_k\|^2 + 2\|\mathbf{z} - \mathbf{u}_{k'}\|^2 + \|\mathbf{v}_k - \mathbf{v}_{k'}\|^2}{4\sigma^2}\right). \quad (4.46)$$

Similarly, the normalization term in Equation (4.41) can also be computed analytically as

$$\int \tilde{\alpha}^\top \varphi(\mathbf{z}, \mathbf{y}) d\mathbf{y} = (\sqrt{2\pi}\sigma)^{d_y} \sum_k \tilde{\alpha}_k \exp\left(-\frac{\|\mathbf{z} - \mathbf{u}_k\|^2}{2\sigma^2}\right). \quad (4.47)$$

## 4.4 Experiment

In this section, we experimentally investigate the practical usefulness of the proposed method. We consider the following supervised linear dimension reduction schemes:

**None:** No dimension reduction is performed.

**dMAVE:** The density-minimum average variance estimation method where supervised linear dimension reduction is performed through local linear estimation for the conditional density<sup>1</sup> (Xia, 2007).

**BDR:** The Bayesian dimension reduction method where the conditional density is modeled by a Gaussian mixture model and supervised linear dimension reduction is performed by sampling from the prior distribution of low-dimensional input<sup>2</sup> (Reich et al., 2011).

**LSDR:** Supervised linear dimension reduction is performed by maximizing an SMI estimator using natural gradients over the Grassmann manifold (Suzuki and Sugiyama, 2013).

**LSCE (proposed):** Supervised linear dimension reduction is performed by minimizing the proposed LSCE using natural gradients over the Grassmann manifold.

**True (reference)** The “true” subspace is used (only for artificial data).

After supervised linear dimension reduction, we execute the following conditional density estimation methods:

**$\epsilon$ -KDE:**  $\epsilon$ -neighbor kernel density estimation, where  $\epsilon$  is chosen by least-squares cross-validation.

**LSCDE:** Least-squares conditional density estimation (Sugiyama et al., 2010).

Note that the proposed method, which is the combination of LSCE and LSCDE, does not explicitly require the post-LSCDE step because LSCDE is already executed inside LSCE, as we have shown in Section 4.3.4. Since the dMAVE and BDR methods are applicable only to univariate output, they are not included in experiments with multivariate output data.

### 4.4.1 Illustration

First, we illustrate the behavior of the plain LSCDE (None/LSCDE) and the proposed method (LSCE/LSCDE). The datasets illustrated in Figure 4.3 have  $d_x = 5$ ,  $d_y = 1$ , and  $d_z = 1$ . The first dimension of input  $\mathbf{x}$  and output  $y$  of the samples are plotted in the graphs, and other 4 dimensions of  $\mathbf{x}$  are just standard normal noise. The results show that the plain LSCDE does not perform well due to the irrelevant noise dimensions of  $\mathbf{x}$ , while the proposed method gives much better estimates.

---

<sup>1</sup>We use the program code provided by the author.

<sup>2</sup>We use the program code available at “<http://www4.stat.ncsu.edu/~reich/code/BayesSDR.R>”.

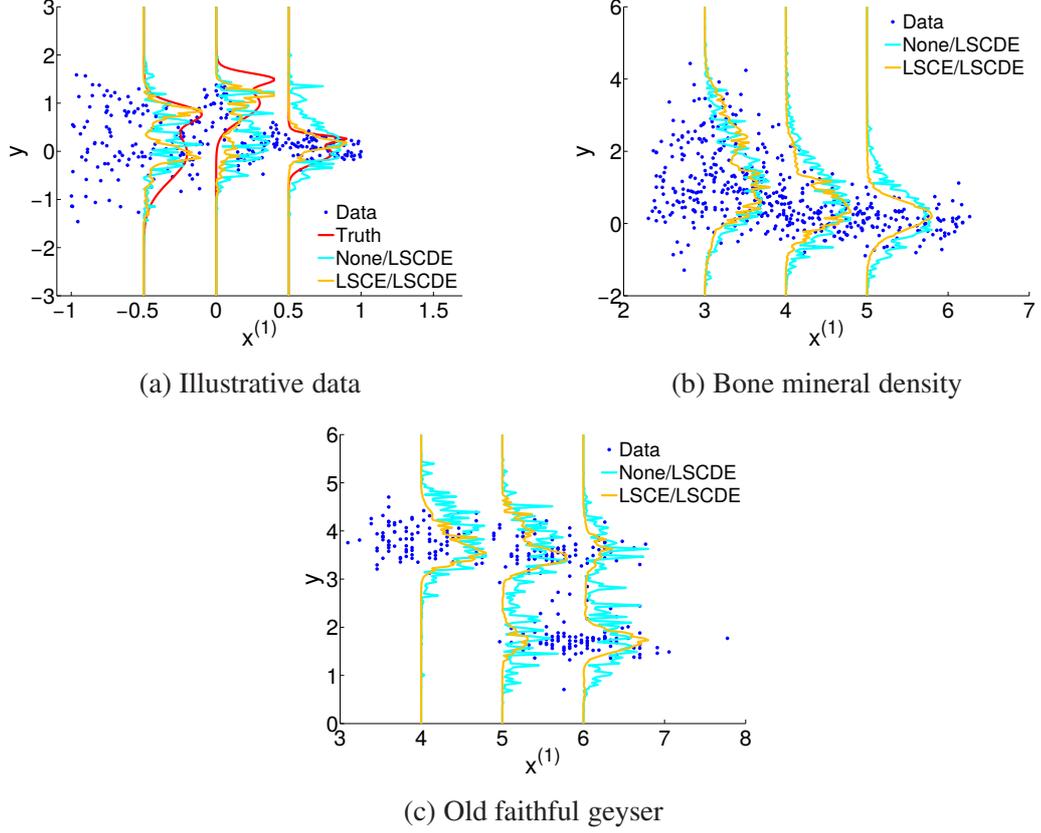


Figure 4.3: Examples of conditional density estimation by plain LSCDE (None/LSCDE) and the proposed method (LSCE/LSCDE).

#### 4.4.2 Artificial Data

Next, we compare the proposed method with the existing dimension reduction methods on conditional density estimation by LSCDE in artificial data.

For  $d_x = 5$ ,  $d_y = 1$ ,  $\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}_5)$ , and  $\epsilon \sim \mathcal{N}(\epsilon|0, 0.25^2)$ , where  $\mathcal{N}(\cdot|\boldsymbol{\mu}, \boldsymbol{\Sigma})$  denotes the normal distribution with mean  $\boldsymbol{\mu}$  and covariance matrix  $\boldsymbol{\Sigma}$ , we consider the following artificial datasets:

- (a)  $d_z = 2$  and  $y = (x^{(1)})^2 + (x^{(2)})^2 + \epsilon$ .
- (b)  $d_z = 1$  and  $y = x^{(2)} + (x^{(2)})^2 + (x^{(2)})^3 + \epsilon$ .
- (c)  $d_z = 1$  and  $y = \begin{cases} (x^{(1)})^2 + \epsilon & \text{with 0.85 probability,} \\ 2\epsilon - 4 & \text{with 0.15 probability.} \end{cases}$

The first row of Figure 4.4 shows the dimensionality reduction error between true  $\mathbf{W}^*$  and its estimate  $\widehat{\mathbf{W}}$  for different sample size  $N$ , measured by

$$\text{Error}_{\text{DR}} = \|\widehat{\mathbf{W}}^\top \widehat{\mathbf{W}} - \mathbf{W}^{*\top} \mathbf{W}^*\|_{\text{Frobenius}}, \quad (4.48)$$

where  $\|\cdot\|_{\text{Frobenius}}$  denotes the Frobenius norm. All methods perform similarly for the dataset (a), and the dMAVE and BDR methods outperform LSCE and LSDR when  $N = 50$ .

In the dataset (b), LSDR does not work well compare to other methods especially when  $N \geq 250$ . To explain this behavior, we plot the histograms of  $\{y\}_{i=1}^{400}$  in the left

column of Figure 4.5. They show that the profile of the histogram (which is a sample approximation of  $p(y)$ ) in the dataset (b) is much sharper than that in the dataset (a). This sharp  $p(y)$  suggests that the density ratio  $\frac{p(z,y)}{p(z)p(y)}$  would be highly non-smooth and is hard to approximate. Thus, LSDR, which estimates the density ratio  $\frac{p(z,y)}{p(z)p(y)}$ , would perform poorly in the dataset (b). On the other hand, other methods are based on the conditional density  $p(y|z)$  where  $p(y)$  is not included. Therefore, the conditional density  $p(y|z)$  would be smoother than the density ratio  $\frac{p(z,y)}{p(z)p(y)}$  and is easier to be estimated.

For the dataset (c), we consider the situation where the output  $y$  contain outliers which are not related to  $x$ . The data profile of dataset (c) in the right column of Figure 4.5 illustrates such a situation. The result on dataset (c) shows that the proposed LSCE method is robust against outliers and gives the best subspace estimation accuracy, while the BDR method performs unreliably with large standard errors.

The right column of Figure 4.4 plots the conditional density estimation error between true  $p(y|x)$  and its estimate  $\hat{p}(y|x)$ , evaluated by the squared-loss:

$$\text{Error}_{\text{CDE}} = \frac{1}{2N'} \sum_{i=1}^{N'} \int \hat{p}(y|\tilde{x}_i)^2 dy - \frac{1}{N'} \sum_{i=1}^{N'} \hat{p}(\tilde{y}_i|\tilde{x}_i), \quad (4.49)$$

where  $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{N'}$  is a set of test samples that have not been used for training. We set  $N' = 1000$ . For the dataset (a) and (c), all methods with dimension reduction perform equally well, which are much better than no dimension reduction (None/LSCDE) and are comparable to the method with the true subspace (True/LSCDE). For the dataset (b), all method except LSDR/LSCDE perform well overall and comparable to the method with the true subspace.

### 4.4.3 Benchmark Data

Next, we use the UCI benchmark datasets (Bache and Lichman, 2013). We randomly select  $N$  samples from each dataset for training, and the rest are used to measure the conditional density estimation error in the test phase. Since the dimensionality of the subspace  $d_z$  is unknown, we chose it by cross-validation. More specifically, 5-fold cross-validation is performed for each combination of the dimension reduction and conditional density estimation methods to choose subspace dimensionality  $d_z$  such that the conditional density estimation error is minimized. Note that tuning parameters  $\lambda$  and  $\sigma$  are also chosen based on cross-validation for each method. Since the conditional density estimation error is equivalent to SCE, choosing the subspace dimensionalities by the conditional density estimation error in LSCE is equivalent to choosing subspace dimensionality which gives the minimum SCE value.

The results of univariate output benchmark datasets averaged over 10 runs are summarized in Table 4.1, showing that LSCDE tends to outperform  $\epsilon$ -KDE and the proposed LSCE/LSCDE method works well overall. Both LSDR/LSCDE and dMAVE/LSCDE methods also perform well in all datasets, while BDR/LSCDE does not work well in the datasets containing outliers such as ‘‘Red Wine’’, ‘‘White Wine’’, and ‘‘Forest Fires’’. Table 4.2 describes the subspace dimensionalities chosen by cross-validation averaged over 10 runs. It shows that all dimensionality reduction methods reduce the input dimension significantly, especially for ‘‘Yacht’’, ‘‘Red Wine’’, and ‘‘White Wine’’ where the best method always chooses  $d_z = 1$  in all runs.

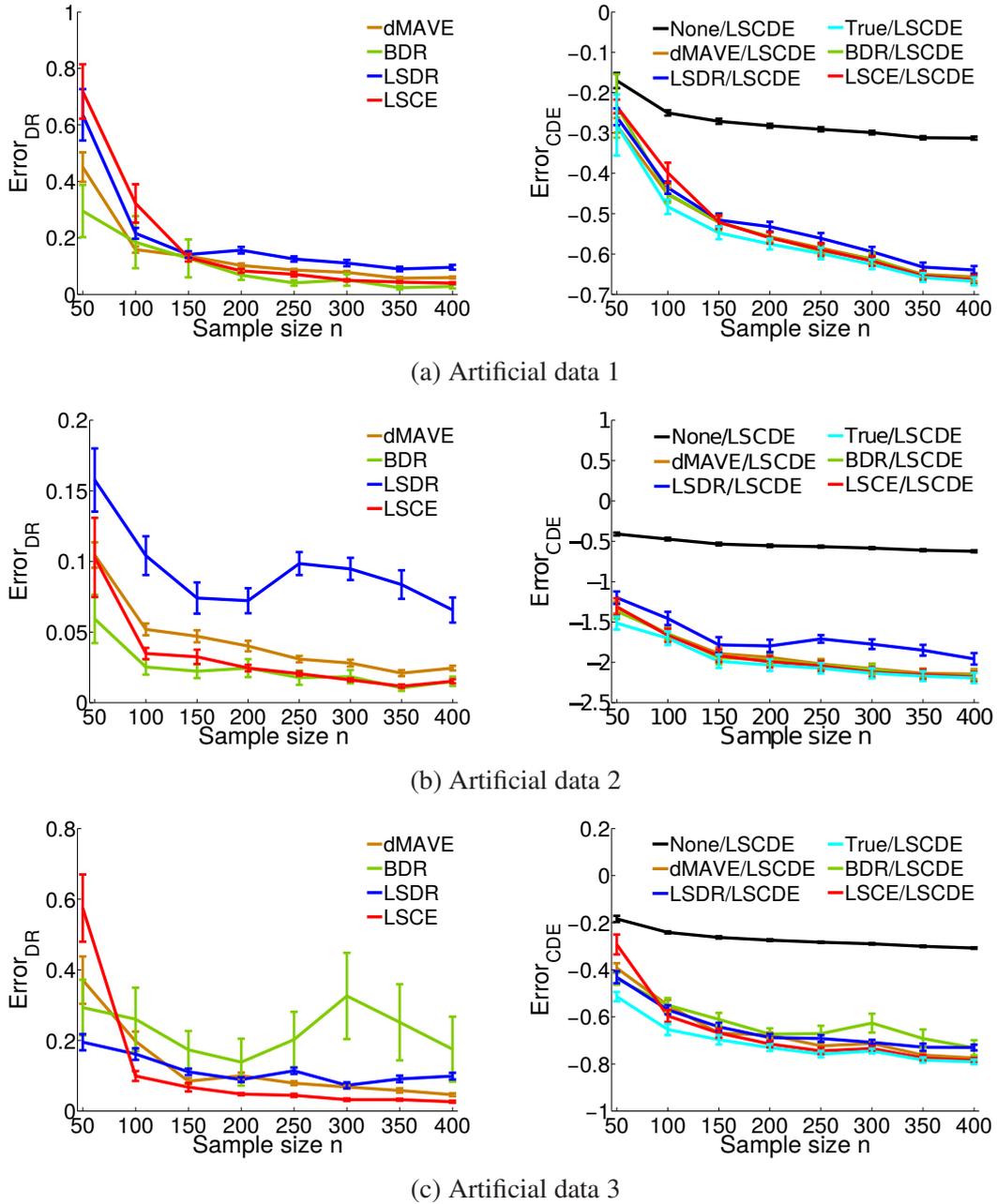
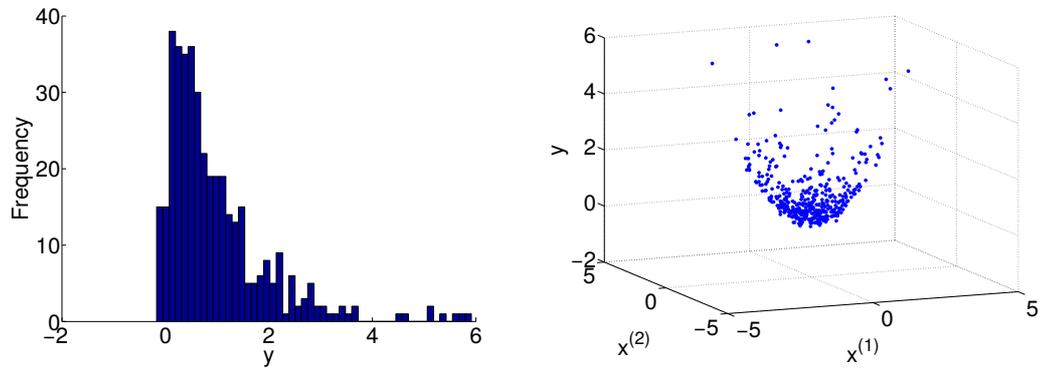
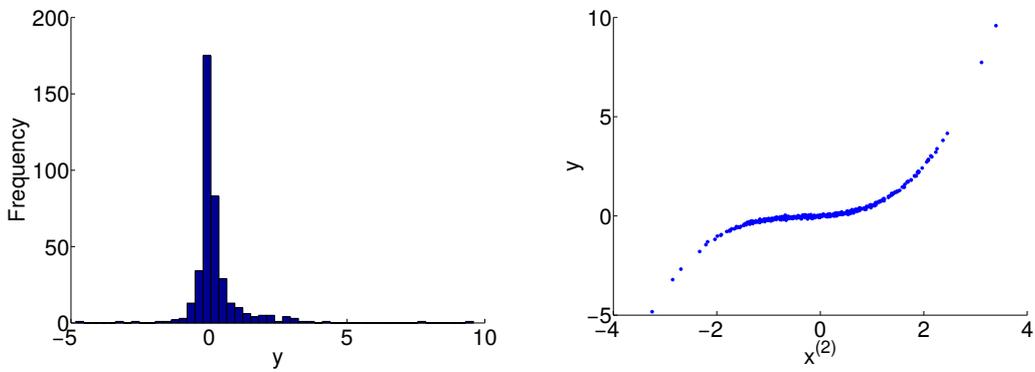


Figure 4.4: Left column: The mean and standard error of the dimensionality reduction error over 20 runs. Right column: The mean and standard error of the conditional density estimation error over 20 runs.

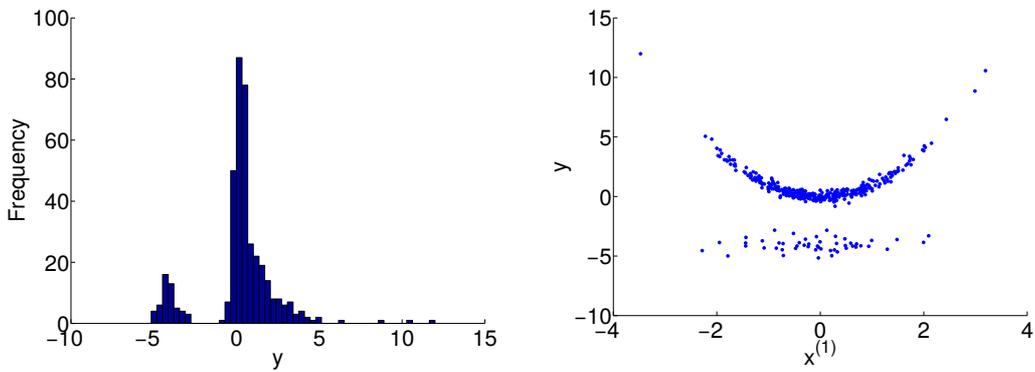
The result of multivariate output “Stock” and “Energy” benchmark datasets are summarized in Table 4.3, showing that the proposed LSCE/LSCDE method also works well for multivariate output datasets and significantly outperforms methods without dimensionality reduction. Table 4.4 describes the subspace dimensionalities selected by cross-validation, showing that LSDR/LSCDE tends to more aggressively reduce the dimensionality than LSCE/LSCDE.



(a) Artificial data 1



(b) Artificial data 2



(c) Artificial data 3

Figure 4.5: Left column: Example histograms of  $\{\mathbf{y}_i\}_{i=1}^{400}$  on the artificial datasets. Right column: Example data plot of relevant features of  $\mathbf{x}$  against  $y$  when  $N = 400$  on the artificial datasets. The left distribution in the histogram of dataset (c) is regarded as outliers.

Table 4.1: Mean and standard error of the conditional density estimation error over 10 runs for univariate output datasets. The best method in term of the mean error and comparable methods according to the two-sample paired  $t$ -test at the significance level 5% are specified by bold face.

Dataset	LSCE		LSDR		dMAVE		BDR		No reduction	
	LSCDE	$\epsilon$ -KDE	LSCDE	$\epsilon$ -KDE						
Servo	<b>-2.95(.17)</b>	<b>-3.03(.14)</b>	-2.69(.18)	<b>-2.95(.11)</b>	<b>-3.13(.13)</b>	<b>-3.17(.10)</b>	<b>-2.96(.10)</b>	<b>-2.95(.12)</b>	-2.62(.09)	-2.72(.06)
Yacht	<b>-6.46(.02)</b>	<b>-6.30(.14)</b>	-5.63(.26)	-5.47(.29)	-6.25(.06)	-5.97(.12)	<b>-6.45(.04)</b>	-6.05(.18)	-1.72(.04)	-2.95(.02)
Auto MPG	-1.80(.04)	-1.75(.05)	-1.85(.04)	-1.77(.05)	<b>-1.98(.04)</b>	<b>-1.97(.04)</b>	<b>-1.91(.04)</b>	-1.84(.05)	-1.75(.04)	-1.46(.04)
Concrete	<b>-1.37(.03)</b>	-1.18(.06)	<b>-1.30(.03)</b>	-1.18(.04)	<b>-1.42(.06)</b>	-1.15(.05)	<b>-1.37(.04)</b>	-1.10(.04)	-1.11(.02)	-0.80(.03)
Physicochem	<b>-1.19(.01)</b>	-0.99(.02)	<b>-1.20(.01)</b>	-0.97(.02)	-1.17(.01)	-0.93(.02)	-1.13(.02)	-0.96(.02)	<b>-1.19(.01)</b>	-0.91(.01)
Red Wine	<b>-2.85(.02)</b>	-1.95(.17)	<b>-2.82(.03)</b>	-1.93(.17)	<b>-2.82(.02)</b>	-1.93(.20)	-2.66(.03)	-2.18(.14)	-2.03(.02)	-1.13(.04)
White Wine	<b>-2.31(.01)</b>	<b>-2.47(.15)</b>	<b>-2.35(.02)</b>	<b>-2.60(.12)</b>	-2.17(.01)	<b>-2.65(.20)</b>	-1.97(.02)	-1.91(.02)	-2.06(.01)	-1.89(.01)
Forest Fires	<b>-7.18(.02)</b>	-6.91(.03)	-6.93(.04)	-6.96(.02)	-7.10(.03)	-6.93(.04)	-7.08(.03)	-6.97(.01)	-3.40(.07)	-6.96(.02)
Housing	<b>-1.72(.09)</b>	-1.58(.08)	<b>-1.91(.05)</b>	-1.62(.08)	<b>-1.76(.11)</b>	-1.50(.13)	<b>-1.86(.09)</b>	-1.74(.03)	-1.41(.05)	-1.13(.01)

Table 4.2: Mean and standard error of the chosen subspace dimensionality over 10 runs for univariate output datasets.

Data set	$(d_x, d_y)$	$N$	LSCE		LSDR		dMAVE		BDR	
			LSCDE	$\epsilon$ -KDE						
Servo	(4, 1)	50	1.6(0.27)	2.4(0.40)	2.2(0.33)	1.6(0.31)	1.5(0.22)	1.5(0.31)	1.2(0.13)	2.0(0.37)
Yacht	(6, 1)	80	1.0(0)	1.0(0)	1.0(0)	1.0(0)	1.2(0.13)	1.0(0)	1.0(0)	1.0(0)
Auto MPG	(7, 1)	100	3.2(0.66)	1.3(0.15)	2.1(0.67)	1.1(0.10)	1.5(0.22)	1.0(0)	1.4(0.16)	1.2(0.13)
Concrete	(8, 1)	300	1.0(0)	1.0(0)	1.2(0.13)	1.0(0)	1.7(0.15)	1.0(0)	2.3(0.21)	1.0(0)
Physicochem	(9, 1)	500	6.5(0.58)	1.9(0.28)	6.6(0.58)	2.6(0.86)	7.5(0.48)	5.0(1.33)	2.6(0.16)	1.7(0.26)
Red Wine	(11, 1)	300	1.0(0)	1.3(0.15)	1.2(0.20)	1.0(0)	1.0(0)	1.1(0.10)	1.5(0.22)	1.0(0)
White Wine	(11, 1)	400	1.2(0.13)	1.0(0)	1.4(0.31)	1.0(0)	1.8(0.70)	1.0(0)	3.1(0.23)	2.7(0.30)
Forest Fires	(12, 1)	100	1.2(0.20)	4.4(0.87)	1.4(0.22)	5.6(1.25)	1.5(0.27)	5.2(1.31)	1.2(0.20)	2.8(0.33)
Housing	(13, 1)	100	3.9(0.74)	1.9(0.80)	2.0(0.39)	1.3(0.15)	3.0(0.77)	1.2(0.13)	1.6(0.22)	1.0(0)

Table 4.3: Mean and standard error of the conditional density estimation error over 10 runs for multivariate output datasets. The best method in term of the mean error and comparable methods according to the two-sample paired  $t$ -test at the significance level 5% are specified by bold face.

Dataset	LSCE		LSDR		No reduction		Scale
	LSCDE	$\epsilon$ -KDE	LSCDE	$\epsilon$ -KDE	LSCDE	$\epsilon$ -KDE	
Stock	-8.37(0.53)	<b>-9.75(0.37)</b>	<b>-9.42(0.50)</b>	<b>-10.27(0.33)</b>	-7.35(0.13)	-9.25(0.14)	$\times 1$
Energy	<b>-7.13(0.04)</b>	-4.18(0.22)	-6.04(0.47)	-3.41(0.49)	-2.12(0.06)	-1.95(0.14)	$\times 10$
2 Joints	<b>-10.49(0.86)</b>	-7.50(0.54)	<b>-8.00(0.84)</b>	-7.44(0.60)	-3.95(0.13)	-3.65(0.14)	$\times 1$
4 Joints	<b>-2.81(0.21)</b>	-1.73(0.14)	-2.06(0.25)	-1.38(0.16)	-0.83(0.03)	-0.75(0.01)	$\times 10$
9 Joints	<b>-8.37(0.83)</b>	-2.44(0.17)	<b>-9.74(0.63)</b>	-2.37(0.51)	-1.60(0.36)	-0.89(0.02)	$\times 100$
Sumi-e 1	<b>-9.96(1.60)</b>	-1.49(0.78)	<b>-6.00(1.28)</b>	1.24(1.99)	-5.98(0.80)	-0.17(0.44)	$\times 10$
Sumi-e 2	<b>-16.83(1.70)</b>	-2.22(0.97)	-9.54(1.31)	-3.12(0.75)	-7.69(0.62)	-0.66(0.13)	$\times 10$
Sumi-e 3	<b>-24.92(1.92)</b>	-6.61(1.25)	-18.0(2.61)	-4.47(0.68)	-8.98(0.66)	-1.45(0.43)	$\times 10$

Table 4.4: Mean and standard error of the chosen subspace dimensionality over 10 runs for multivariate output datasets.

Data set	$(d_x, d_y)$	$N$	LSCE		LSDR	
			LSCDE	$\epsilon$ -KDE	LSCDE	$\epsilon$ -KDE
Stock	(7, 2)	100	3.2(0.83)	2.1(0.59)	2.1(0.60)	2.7(0.67)
Energy	(8, 2)	200	5.9(0.10)	3.9(0.80)	2.1(0.10)	2.0(0.30)
2 Joints	(6, 4)	100	2.9(0.31)	2.7(0.21)	2.5(0.31)	2.0(0)
4 Joints	(12, 8)	200	5.2(0.68)	6.2(0.63)	5.4(0.67)	4.6(0.43)
9 Joints	(27, 18)	500	13.8(1.28)	15.3(0.94)	11.4(0.75)	13.2(1.02)
Sumi-e 1	(9, 6)	200	5.3(0.72)	2.9(0.85)	4.5(0.45)	3.2(0.76)
Sumi-e 2	(9, 6)	250	4.2(0.55)	4.4(0.85)	4.6(0.87)	2.5(0.78)
Sumi-e 3	(9, 6)	300	3.6(0.50)	2.7(0.76)	2.6(0.40)	1.6(0.27)

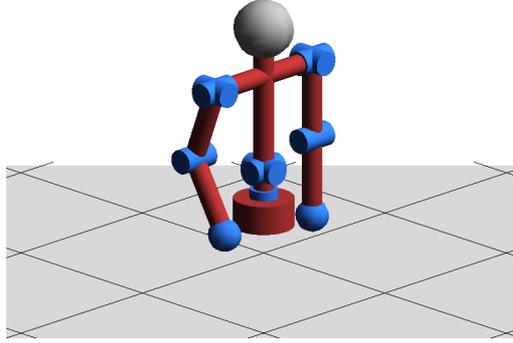


Figure 4.6: Simulator of the upper-body part of the humanoid robot *CB-i*.

#### 4.4.4 Humanoid Robot

We evaluate the performance of the proposed method on humanoid robot transition estimation. We use a simulator of the upper-body part of the humanoid robot *CB-i* (Cheng et al., 2007) (see Figure 4.6). The robot has 9 controllable joints: shoulder pitch, shoulder roll, elbow pitch of the right arm, shoulder pitch, shoulder roll, elbow pitch of the left arm, waist yaw, torso roll, and torso pitch joints.

Posture of the robot is described by 18-dimensional real-valued state vector  $\mathbf{s}$ , which corresponds to the angle and angular velocity of each joint in radians and radians per seconds, respectively. We can control the robot by sending the action command  $\mathbf{a}$  to the system. The action command  $\mathbf{a}$  is a 9-dimensional real-valued vector, which corresponds to the target angle of each joint. When the robot is currently at state  $\mathbf{s}$  and receives action  $\mathbf{a}$ , the physical control system of the simulator calculates the amount of torques to be applied to each joint. These torques are calculated by the *proportional-derivative* (PD) controller as

$$\tau_i = K_{p_i}(a_i - s_i) - K_{d_i}\dot{s}_i, \quad (4.50)$$

where  $s_i$ ,  $\dot{s}_i$ , and  $a_i$  denote the current angle, the current angular velocity, and the received target angle of the  $i$ -th joint, respectively.  $K_{p_i}$  and  $K_{d_i}$  denote the position and velocity gains for the  $i$ -th joint, respectively. We set  $K_{p_i} = 2000$  and  $K_{d_i} = 100$  for all joints except that  $K_{p_i} = 200$  and  $K_{d_i} = 10$  for the elbow pitch joints. After the torques are applied to the joints, the physical control system update the state of the robot to  $\mathbf{s}'$ .

In the experiment, we randomly choose the action vector  $\mathbf{a}$  and simulate a noisy control system by adding a bimodal Gaussian noise vector. More specifically, the action  $a_i$  of the  $i$ -th joint is first drawn from uniform distribution on  $[s_i - 0.087, s_i + 0.087]$ . The drawn action is then contaminated by Gaussian noise with mean 0 and standard deviation 0.034 with probability 0.6 and Gaussian noise with mean -0.087 and standard deviation 0.034 with probability 0.4. By repeatedly control the robot  $N$  times, we obtain the transition samples  $\{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j)\}_{j=1}^N$ . Our goal is to learn the system dynamic as a transition probability  $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$  from these samples. Thus, as the conditional density estimation problem, the state-action pair  $(\mathbf{s}^\top, \mathbf{a}^\top)^\top$  is regarded as input variable  $\mathbf{x}$ , while the next state  $\mathbf{s}'$  is regarded as output variable  $\mathbf{y}$ . Later in

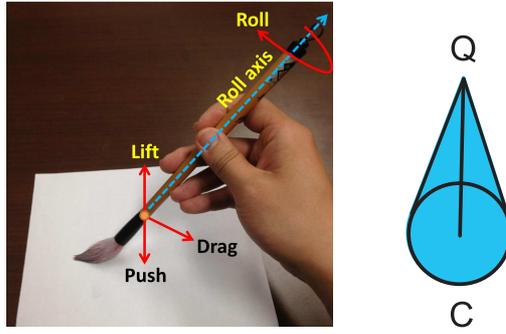


Figure 4.7: Three actions of the brush, which is modeled as the footprint on a paper canvas.

Chapter 6, we will show that an estimated of the transition probability is highly useful in model-based reinforcement learning (Sutton and Barto, 1998).

We consider three scenarios: Using only 2 joints (right shoulder pitch and right elbow pitch), only 4 joints (in addition, right shoulder roll and waist yaw), and all 9 joints. Thus,  $d_x = 6$  and  $d_y = 4$  for the 2-joint case,  $d_x = 12$  and  $d_y = 8$  for the 4-joint case, and  $d_x = 27$  and  $d_y = 18$  for the 9-joint case. We generate 500, 1000, and 1500 transition samples for the 2-joint, 4-joint, and 9-joint cases. We then randomly choose  $N = 100, 200,$  and  $500$  samples for training, and use the rest for evaluating the test error. The results are summarized also in Table 4.3, showing that the proposed method performs well for the all three cases. Table 4.4 describes the dimensionalities selected by cross-validation, showing that the humanoid robot’s transition is highly redundant.

#### 4.4.5 Computer Art

Finally, we consider the transition estimation problem in *sumi-e* style brush drawings for non-photorealistic rendering (Xie et al., 2012). Our aim is to learn the brush dynamics as state transition probability  $p(s'|s, \mathbf{a})$  from the real artists’ stroke-drawing samples.

From a video of real brush strokes, we extract footprints and identify corresponding 3-dimensional actions (see Figure 4.7). The state vector consists of six measurements: the angle of the velocity vector and the heading direction of the footprint relative to the medial axis of the drawing shape, the ratio of the offset distance from the center of the footprint to the nearest point on the medial axis over the radius of the footprint, the relative curvatures of the nearest current point and the next point on the medial axis, and the binary signal of the reverse driving or not. Thus, the state transition probability  $p(s'|s, \mathbf{a})$  has 9-dimensional input and 6-dimensional output. We collect 722 transition samples in total. We randomly choose  $N = 200, 250,$  and  $300$  for training and use the rest for testing.

The estimation results summarized at the bottom of Table 4.3 and Table 4.4. These tables show that there exists a low-dimensional dimension reduction subspace and the proposed method can successfully find it.

### 4.5 Conclusion

In this chapter, we proposed a single-step dimension reduction method for conditional density estimation. Our key idea is to perform supervised linear dimension reduction

by minimizing the square-loss conditional entropy (SCE) which contains conditional density in its formulation. By minimizing SCE, dimension reduction and conditional density estimation are carried out simultaneously in an integrated manner.

The effectiveness of LSCE in conditional density estimation can be understood from the upper-bound minimization in LSCE (see Equation (4.27)) which is equivalent to the minimization of the squared error (see Equation (4.9)). This implies that LSCE learns the conditional density  $\hat{p}(\mathbf{y}|\mathbf{z})$  and the matrix  $\mathbf{W}$  such that they *jointly* minimize the squared error.

We have shown that SCE and the squared-loss mutual information (SMI) are similar but different in that the output density is included in the denominator of the density ratio in SMI. This means that estimation of SMI is hard when the output density is fluctuated, while the proposed method using SCE does not suffer from this problem. The proposed method is also robust against outliers since minimization of the Pearson divergence automatically weights down effects of outlier points. Moreover, the proposed method is applicable to multivariate output data, which is not straightforward to handle in other dimensionality reduction methods based on conditional probability density. However, it should be noted that SCE is less robust against outliers when compared with quadratic mutual information, as we experimentally showed in Chapter 3.

In this chapter, the effectiveness of the proposed method was demonstrated through extensive experiments including humanoid robot and computer art data. Later in Chapter 7, we will further utilize the proposed method for model-based reinforcement learning.

# Chapter 5

## Background of Reinforcement Learning

This chapter gives a background of reinforcement learning. We firstly introduce the Markov decision process which is a standard formulation of reinforcement learning. Then, we review approaches and methods that solve reinforcement learning problems.

### 5.1 Markov Decision Processes

As briefly introduced in Chapter 1, reinforcement learning is concerned with the sequential decision making problem that involves an agent and an unknown environment. This problem is commonly formulated as a discrete-time *Markov decision process* (MDP) (Bellman, 1957b; Puterman, 1994). An MDP is described by a quintuple

$$\{\mathcal{S}, \mathcal{A}, p(\mathbf{s}'|\mathbf{s}, \mathbf{a}), p_1(\mathbf{s}), r(\mathbf{s}, \mathbf{a}, \mathbf{s}')\}. \quad (5.1)$$

The set  $\mathcal{S} \subseteq \mathbb{R}^{d_s}$  denotes the set of *states*  $\mathbf{s}$  and is called the state space. The set  $\mathcal{A} \subseteq \mathbb{R}^{d_a}$  denotes the set of *actions*  $\mathbf{a}$  and is called the action space. The *transition probability density* function  $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$  determines the probability density of the agent observing a new state  $\mathbf{s}'$  after it chooses an action  $\mathbf{a}$  in a current state  $\mathbf{s}$ . The *initial state probability density* function  $p_1(\mathbf{s})$  determines the probability density that the agent initially observes state  $\mathbf{s}$ . The *reward function*  $r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  determines a reward the agent observes after it chooses an action  $\mathbf{a}$  in a state  $\mathbf{s}$  and observes a next state  $\mathbf{s}'$ . The environment is comprised of the transition probability density, the initial state density, and the reward function.

The *policy*  $\pi$  is an important component of the agent. It is a mapping from state to action and it controls how the agent chooses an action in a given state. The policy can be a deterministic policy which gives the same action for the same state, i.e.,  $\mathbf{a} = \pi(\mathbf{s})$ , or it can be a stochastic policy which gives a probability density over actions in a given state, i.e.,  $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$ . Although the deterministic policy behaves differently from the stochastic policy, it can be represented by the stochastic policy using the *Dirac delta* function:

$$\pi(\mathbf{a}|\mathbf{s}) = \delta(\mathbf{a} - \pi(\mathbf{s})) \quad (5.2)$$

where the Dirac delta function  $\delta$  is defined as

$$\delta(x) = \begin{cases} \infty, & \text{if } x = 0, \\ 0 & \text{if } x \neq 0 \end{cases} \quad (5.3)$$

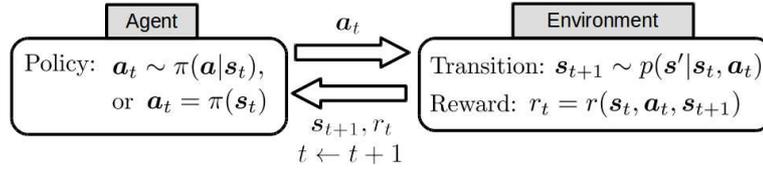


Figure 5.1: An illustration of an MDP governing its process. At the first time step  $t = 1$  the agent observes the initial state  $\mathbf{s}_1 \sim p_1(\mathbf{s})$  (not shown in this illustration). Then at each time step  $t$ , the agent in the state  $\mathbf{s}_t$  chooses an action  $\mathbf{a}_t$  using its policy  $\pi$ . This action changes the state according to the transition probability density  $p(\mathbf{s}'|\mathbf{s}_t, \mathbf{a}_t)$ . Then, the agent observes the next state  $\mathbf{s}_{t+1}$  and an immediate reward  $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ . Finally, the time step increases by 1 and the process is repeated until the final time step  $T$  is reached.

Note that  $\delta(\mathbf{a}, \mathbf{b}) = \infty$  is used for real-valued action space to ensure that the identity  $\int \pi(\mathbf{a}|\mathbf{s})d\mathbf{a} = 1$  holds<sup>1</sup>. This means that any action drawn from the stochastic policy  $\pi(\mathbf{a}|\mathbf{s}) = \delta(\mathbf{a}, \pi(\mathbf{s}))$  is always equivalent to the action obtained by the deterministic policy  $\pi(\mathbf{s})$ .

An MDP governs its process in a sequential manner as follows. Initially at time step  $t = 1$ , an agent observes an initial state  $\mathbf{s}_1 \sim p_1(\mathbf{s})$ . At time step  $t$ , the agent uses a policy  $\pi$  to choose an action  $\mathbf{a}_t$  depending on the current state  $\mathbf{s}_t$ . The action  $\mathbf{a}_t$  causes the environment to change the state from  $\mathbf{s}_t$  to  $\mathbf{s}_{t+1}$  according to the transition probability density, i.e.,  $\mathbf{s}_{t+1} \sim p(\mathbf{s}'|\mathbf{s}_t, \mathbf{a}_t)$ . Then, the agent observes the new state  $\mathbf{s}_{t+1}$  and an immediate reward  $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ . Finally, the above decision making step is repeated with time step increased as  $t \leftarrow t+1$ . The process terminates when the final time step  $T$  is reached. This process is illustrated in Figure 5.1. The important property of MDP is that the next state  $\mathbf{s}_{t+1}$  only depends on the current state  $\mathbf{s}_t$  and the chosen action  $\mathbf{a}_t$ , and does not depend on past states and actions. This property is called the *first order Markov chain* property.

The final time step  $T$  can be either infinite or finite. An MDP with  $T = \infty$  is called an *infinite horizon MDP* and an MDP with finite  $T$  is called a *finite horizon MDP* or an *episodic MDP*. While both infinite horizon MDP and finite horizon MDP have been studied by many researchers (Bellman, 1957b; Puterman, 1994; Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998), finite horizon MDP is considered more popular for real-world applications. This is mainly because in real-world applications, an execute time of an agent is often fixed or limited. For this reason, in this dissertation we only focus on an episodic MDP. A *trajectory* of length  $T$  is defined as a sequence of a state, an action, a reward, and a next state:

$$\boldsymbol{\tau} = (\mathbf{s}_1, \mathbf{a}_1, r_1, \mathbf{s}_2, \mathbf{a}_2, \dots, \mathbf{a}_T, r_T, \mathbf{s}_{T+1}). \quad (5.4)$$

The probability of obtaining a trajectory  $\boldsymbol{\tau}$  depends on the policy, the transition probability density, and the initial state probability density. The *trajectory probability density* function is given by

$$p(\boldsymbol{\tau}; \pi) = p(\mathbf{s}_1) \prod_{t=1}^T p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t), \quad (5.5)$$

<sup>1</sup>For discrete action space, we can define  $\delta(x) = 1$  if  $x = 0$  to have the same interpretation.

where we use  $p(\boldsymbol{\tau}; \pi)$  to emphasize the dependency on  $\pi$ . Given a trajectory  $\boldsymbol{\tau}$ , we can define a discounted cumulative reward or a *return* by

$$R(\boldsymbol{\tau}) = \sum_{t=1}^T \gamma^{t-1} r_t, \quad (5.6)$$

where the discount factor  $0 \leq \gamma \leq 1$  determines the importance of future rewards relative to early rewards. More specifically, if  $\gamma = 1$  then there is no discount and rewards at all time steps are equally important. On the other hand, the return only depends on the first immediate reward  $r_1$  when  $\gamma = 0$  where we define  $0^0 = 1$ . Then, given a policy  $\pi$ , its *expected return* is defined as

$$\begin{aligned} \mathcal{R}(\pi) &= \mathbb{E}_{p(\boldsymbol{\tau}; \pi)} [R(\boldsymbol{\tau})] \\ &= \int R(\boldsymbol{\tau}) p(\boldsymbol{\tau}; \pi) d\boldsymbol{\tau}. \end{aligned} \quad (5.7)$$

The goal of solving an MDP is to find an *optimal policy*  $\pi^*$  which maximizes the expected return.

Methods to solve MDPs have been studied for decades since MDPs were formally introduced. A traditional approach to solve this problem is based on *dynamic programming* (Bellman, 1957a). However, the dynamic programming approach has two major limitations which make it unsuitable for real-world problems. Firstly, dynamic programming is a computation method that requires the complete knowledge about the environment. In real-world problems, the complete knowledge about the environment is usually unavailable, especially for the transition probability density function. The second limitation of dynamic programming is the *curse of dimensionality*. As we introduced earlier in Chapter 1, the curse of dimensionality in the context of dynamic programming refers to a scenario where the computational complexity of dynamic programming increases exponentially as the dimensionality of states increases. Real-world problems often involve high-dimensional states and thus dynamic programming is impractical.

*Reinforcement learning* is a modern approach to find an optimal policy in MDPs. Reinforcement learning methods learn an optimal policy using data that the agent collects by interacting with the environment. Learning from data is the main characteristic of reinforcement learning that distinguishes it from traditional approaches which need a complete knowledge about the environment and do not rely on data.

There are two criteria that categorize reinforcement learning methods. The first criterion categorizes reinforcement learning methods into *policy iteration* methods and *direct policy search* methods depending on how an optimal policy is learned. The second criterion categorizes reinforcement learning methods into *model-free* methods and *model-based* methods depending on whether they learn a model of the environment or not.

In the following sections, we firstly give an overview of the policy iteration and direct policy search approaches from the model-free reinforcement learning viewpoint. Then, we explain the idea of model-based reinforcement learning and review methods for learning a model of the environment.

## 5.2 Policy Iteration

Policy iteration refers to reinforcement learning methods that firstly learn a *value function* from data and then derive an optimal policy based on the learned value function. There are two types of value functions; *state value function* and *state-action value function*. In this section, we firstly explain these value functions and show that they can be used to obtain an optimal policy. Then, we review two policy iteration methods that use different approaches to learn the value functions.

### 5.2.1 State Value Function and State-Action Value Function

The state value function  $V^\pi(\mathbf{s})$  determines the goodness of a policy  $\pi$  in state  $\mathbf{s}$ . It is mathematically defined as the expected return obtained after following  $\pi$  starting from  $\mathbf{s}$ :

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[ \sum_{t=1}^T \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) | \mathbf{s}_1 = \mathbf{s} \right]. \quad (5.8)$$

The expectation  $\mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})}$  denotes the conditional expectation of actions  $\mathbf{a}_t$  and states  $\mathbf{s}_{t+1}$  where  $\mathbf{a}_t$  are drawn from  $\pi(\mathbf{a}|\mathbf{s}_t)$ ,  $\mathbf{s}_{t+1}$  are drawn from  $p(\mathbf{s}'|\mathbf{s}_t, \mathbf{a}_t)$  and  $\mathbf{s}_1 = \mathbf{s}$ . The state value function  $V^\pi(\mathbf{s})$  admits a recursive relation called the *Bellman equation*:

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})} [r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V^\pi(\mathbf{s}')]. \quad (5.9)$$

That is, the value of a state  $\mathbf{s}$  can be computed from the expected reward and the (discounted) expected value of the next states. Since an optimal policy maximizes the expected return, it also maximizes the state value function of every state as well. More specifically, the state value function of an optimal policy  $\pi^*$  is the *optimal state value function* and is defined as

$$V^*(\mathbf{s}) = \max_{\pi} V^\pi(\mathbf{s}). \quad (5.10)$$

The recursive relation of the optimal state value function is called the *Bellman optimality equation*:

$$V^*(\mathbf{s}) = \max_{\pi} \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})} [r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V^*(\mathbf{s}')]. \quad (5.11)$$

We can see that once the optimal state value function  $V^*$  is learned, an optimal policy can be obtained as the maximizer of the right-hand side of the Bellman optimality equation.

An alternative to the state value function is the *state-action value function*  $Q^\pi(\mathbf{s}, \mathbf{a})$  which determines the expected return obtained after choosing an action  $\mathbf{a}$  in state  $\mathbf{s}$  and then following  $\pi$  afterwards:

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[ \sum_{t=1}^T \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) | \mathbf{s}_1 = \mathbf{s}, \mathbf{a}_1 = \mathbf{a} \right]. \quad (5.12)$$

The state-action value function also admits the Bellman equation:

$$\begin{aligned} Q^\pi(\mathbf{s}, \mathbf{a}) &= \mathbb{E}_{p(\mathbf{s}'|\mathbf{s},\mathbf{a})} [r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')} [Q^\pi(\mathbf{s}', \mathbf{a}')] ] \\ &= r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})} [Q^\pi(\mathbf{s}', \mathbf{a}')], \end{aligned} \quad (5.13)$$

where  $r(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [r(\mathbf{s}, \mathbf{a}, \mathbf{s}')] ]$  denotes an expected immediate reward. The state-action value function of an optimal policy is defined as

$$Q^*(\mathbf{s}, \mathbf{a}) = \max_{\pi} Q^{\pi}(\mathbf{s}, \mathbf{a}), \quad (5.14)$$

and its Bellman optimality equation is

$$Q^*(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} \left[ \max_{\mathbf{a}'} Q^*(\mathbf{s}', \mathbf{a}') \right], \quad (5.15)$$

Given the optimal state-action value function  $Q^*$ , a deterministic optimal policy in state  $\mathbf{s}$  which gives the maximum expected return can be obtained as

$$\pi^*(\mathbf{s}) = \operatorname{argmax}_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}). \quad (5.16)$$

In general, the agent requires some amount of stochastic behavior in order to explore the state space and find better policies. To obtain a stochastic policy from a deterministic policy  $\pi(\mathbf{s})$ , a simple approach is to employ the Gaussian distribution with  $\pi(\mathbf{s})$  as the mean, i.e.,

$$\pi(\mathbf{a}|\mathbf{s}) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{a} - \pi(\mathbf{s}))^{\top} \Sigma^{-1}(\mathbf{a} - \pi(\mathbf{s}))\right), \quad (5.17)$$

where the matrix  $\Sigma$  is a covariance matrix and  $|\Sigma|$  denotes the determinant of  $\Sigma$ . A more sophisticated approach to obtain a stochastic policy from the optimal state-action value function is to assign high probabilities to actions with high state-action values. For instance, the *Gibbs softmax distribution* gives the following stochastic policy:

$$\pi(\mathbf{a}|\mathbf{s}) = \frac{\exp(Q^*(\mathbf{s}, \mathbf{a})/\tau)}{\int \exp(Q^*(\mathbf{s}, \mathbf{a})/\tau) d\mathbf{a}}, \quad (5.18)$$

where the parameter  $\tau > 0$  controls the amount of stochasticity.

Both of the state value function and the state-action value function can be used to obtain an optimal policy. However, using the state-action value function is more convenient since Equation (5.16) does not involve an expectation. In the remainder, we focus on policy iteration based on the state-action value function and simply refer to the state-action value function as the value function.

## 5.2.2 Policy Iteration Framework

We have shown that the optimal (state-action) value function is useful for obtaining an optimal policy. A natural question that follows is how can we learn the optimal value function from data. This question is answered in the policy iteration framework by alternately performing the *policy evaluation* step and the *policy improvement* step. Given an arbitrary initial policy  $\pi$ , the policy evaluation step learns the value function  $Q^{\pi}(\mathbf{s}, \mathbf{a})$  based on data. Then after the policy evaluation step, the policy improvement step greedily computes a new policy  $\pi'$  based on the learned value function  $Q^{\pi}(\mathbf{s}, \mathbf{a})$ , i.e.,  $\pi'(\mathbf{s}) = \operatorname{argmax}_{\mathbf{a}} Q^{\pi}(\mathbf{s}, \mathbf{a})$ . These two steps are alternately performed until the value function and the policy converge. Figure 5.2 illustrates the policy iteration framework.

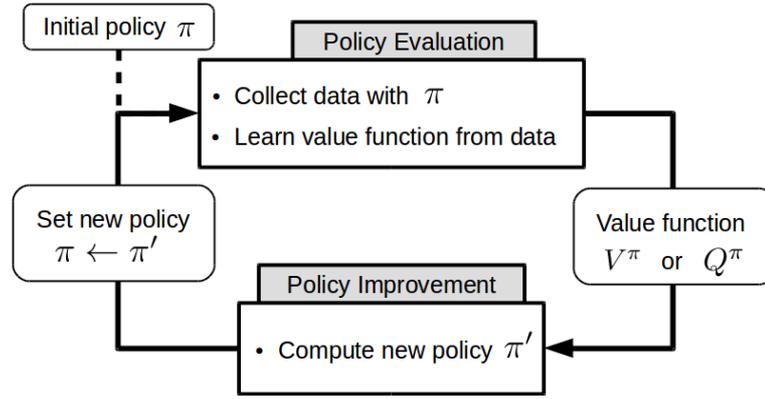


Figure 5.2: An illustration of the policy iteration framework. Starting from an arbitrary initial policy  $\pi$ , the policy evaluation step learns the value function of  $\pi$  from data. Then the policy improvement step computes the new policy from the learned value function. The procedure is repeated until convergence.

For policy iteration in MDPs with discrete action space, the *policy improvement theorem* (Bertsekas and Tsitsiklis, 1996) guarantees that the new policy  $\pi'$  from the policy improvement step indeed improves the current policy  $\pi$ . This improvement is expressed by an inequality

$$Q^\pi(\mathbf{s}, \pi'(\mathbf{s})) \geq V^\pi(\mathbf{s}), \forall \mathbf{s} \in \mathcal{S}. \quad (5.19)$$

The theorem also holds true for a stochastic policy  $\pi(\mathbf{a}|\mathbf{s})$  where  $Q^\pi(\mathbf{s}, \pi'(\mathbf{s}))$  is defined as the expected state-action value over the policy (Sutton and Barto, 1998), i.e.,

$$Q^\pi(\mathbf{s}, \pi'(\mathbf{s})) = \sum_{\mathbf{a} \in \mathcal{A}} \pi'(\mathbf{a}|\mathbf{s}) Q^\pi(\mathbf{s}, \mathbf{a}). \quad (5.20)$$

This inequality means that in state  $\mathbf{s}$ , choosing an action using  $\pi'$  always gives expected return no less than that of choosing an action using  $\pi$ . Thus, the policy in state  $\mathbf{s}$  should change from  $\pi$  to  $\pi'$ . Moreover, the inequality also implies that

$$V^{\pi'}(\mathbf{s}) \geq V^\pi(\mathbf{s}), \forall \mathbf{s} \in \mathcal{S}. \quad (5.21)$$

Then, by using the definition of the optimal policy in Equation (5.10), we can see that the equality in Equation (5.21) only holds when  $\pi$  is already the optimal policy  $\pi^*$ . Thus, policy iteration is guaranteed to converge to the optimal policy. However, it should be emphasized that the theorem guarantees the convergence to the optimal policy only when the action space is discrete and the true value function is used. In the case of continuous action space, the policy improvement step may not improve value functions (Bertsekas and Tsitsiklis, 1996). In the case of learned value functions, policy iteration may not converge to the optimal policy, but it will converge to a near optimal policy when approximation errors of the value functions are bounded (Lagoudakis and Parr, 2003).

An important concern in policy iteration is the policy evaluation step. That is, how to learn the value function  $Q^\pi(\mathbf{s}, \mathbf{a})$  from data collected by a policy  $\pi$ . Next, we review two policy iteration methods that use two different approach to learn the value function.

### 5.2.3 Q-Learning

*Q-learning* (Watkins and Dayan, 1992) is one of the most well-known and popular policy iteration methods. Here, we review the simplest form of Q-learning called a one-step Q-learning which updates the value function for every one-step state transitions. A one-step state transition is simply a pair of a state, an action, a reward, and a next state that the agent observes, i.e.,

$$(\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \mathbf{s}_{t+1}). \quad (5.22)$$

The main idea of Q-learning lies in the Bellman equation in Equation (5.13). Based on the Bellman equation, Q-learning updates the value function by

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q^\pi(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left( r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}) - Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \right), \quad (5.23)$$

where  $0 < \alpha < 1$  denotes the step size. The difference term inside the parenthesis,

$$r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}) - Q^\pi(\mathbf{s}_t, \mathbf{a}_t), \quad (5.24)$$

is called the *temporal difference* and is directly related to the Bellman equation. The temporal difference determines the amount of error between the estimate of the expected return  $r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a})$ , and the current value function  $Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$ . This update rule can also be understood as an instance of gradient-based update (Sutton and Barto, 1998).

It was shown that Q-learning converges to the optimal value function if every state-action pairs are visited for an infinite amount of times (Watkins and Dayan, 1992; Tsitsiklis, 1994; Jaakkola et al., 1994; Bertsekas and Tsitsiklis, 1996). However, such a scenario is most probable only when state and action spaces are discrete. For continuous state and action spaces, we cannot even guarantee that the agent will visit the exactly same state again. Moreover, the update rule in Equation (5.23) requires us to store the value of every possible state-action pair. Such an exact representation of the value function is inapplicable for continuous state and action spaces.

A common approach to tackle continuous state and action spaces in Q-learning is to approximate the value function by a model  $\widehat{Q}$ :

$$Q^\pi(\mathbf{s}, \mathbf{a}) \approx \widehat{Q}^\pi(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}), \quad (5.25)$$

where  $\boldsymbol{\theta}$  is an adjustable parameter of the model. An example is the linear-in-parameter model:

$$\widehat{Q}^\pi(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{\varphi}(\mathbf{s}, \mathbf{a}), \quad (5.26)$$

where  $\boldsymbol{\theta} \in \mathbb{R}^b$  is the parameter vector and  $\boldsymbol{\varphi}(\mathbf{s}, \mathbf{a}) \in \mathbb{R}^b$  is the basis function vector. For Q-learning with function approximation, the update rule for the value function becomes an update rule for the parameter  $\boldsymbol{\theta}$  and is given by

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \left( r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} \widehat{Q}^\pi(\mathbf{s}_{t+1}, \mathbf{a}; \boldsymbol{\theta}) - \widehat{Q}^\pi(\mathbf{s}_t, \mathbf{a}_t; \boldsymbol{\theta}) \right) \nabla_{\boldsymbol{\theta}} \widehat{Q}^\pi(\mathbf{s}_t, \mathbf{a}_t; \boldsymbol{\theta}). \quad (5.27)$$

However, Q-learning with function approximation is not guaranteed to converge, and most of the convergence guarantees are provided only for the linear-in-parameter model under certain conditions (Tsitsiklis and Roy, 1997; Tadić, 2001).

Q-learning is a simple method and researchers have proposed many of its extension such as *Q-learning for multi-agent systems* (Tan, 1993; Tesauro, 2003) and *Bayesian Q-learning* (Dearden et al., 1998). Among these Q-learning-based methods, *deep Q-network* (DQN) (Mnih et al., 2015) is perhaps one of the most successful Q-learning-based methods so far. The main idea of DQN is to model the value function by a deep neural network (Bengio, 2009). As we have briefly discussed in Chapter 1, a deep neural network is a complex structure consisting of multiple layers and it can handle complex and high-dimensional data very well. This allows DQN to solve a highly challenging task of learning to play video games from raw pixel images (Mnih et al., 2015). However, DQN requires a lot of data and network engineering to achieve the desired results.

#### 5.2.4 Least-Squares Policy Iteration

In Q-learning, the value function is updated iteratively where the amount of update depends on the choice of step size  $\alpha$ . In contrast, *least-squares policy iteration* (LSPI) (Lagoudakis and Parr, 2003) computes the value function in a closed form. Below, we briefly review a variant of LSPI called *LSPI based on Bellman residual minimizing approximation*.

The main idea of LSPI based on Bellman residual minimizing approximation lies on the Bellman equation in Equation (5.13) which can be rearranged into

$$r(\mathbf{s}, \mathbf{a}) = Q^\pi(\mathbf{s}, \mathbf{a}) - \gamma \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [Q^\pi(\mathbf{s}', \mathbf{a}')]. \quad (5.28)$$

The error between the left-hand side and the right-hand side is called the *Bellman residual*. In LSPI based on Bellman residual minimizing approximation, the value function is learned such that the Bellman residual is minimized.

Firstly, the value function is modeled by the linear-in-parameter model:

$$\widehat{Q}^\pi(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{\varphi}(\mathbf{s}, \mathbf{a}), \quad (5.29)$$

where  $\boldsymbol{\theta} \in \mathbb{R}^b$  is the parameter vector and  $\boldsymbol{\varphi}(\mathbf{s}, \mathbf{a}) \in \mathbb{R}^b$  is the basis function vector. By substituting this model into the above residual, we obtain

$$\begin{aligned} r(\mathbf{s}, \mathbf{a}) &= \boldsymbol{\theta}^\top \boldsymbol{\varphi}(\mathbf{s}, \mathbf{a}) - \gamma \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [\boldsymbol{\theta}^\top \boldsymbol{\varphi}(\mathbf{s}', \mathbf{a}')] \\ &= \boldsymbol{\theta}^\top \boldsymbol{\psi}(\mathbf{s}, \mathbf{a}), \end{aligned} \quad (5.30)$$

where the basis function  $\boldsymbol{\psi}(\mathbf{s}, \mathbf{a})$  is defined as

$$\boldsymbol{\psi}(\mathbf{s}, \mathbf{a}) = \boldsymbol{\varphi}(\mathbf{s}, \mathbf{a}) - \gamma \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')p(\mathbf{s}'|\mathbf{s}, \mathbf{a})} [\boldsymbol{\varphi}(\mathbf{s}', \mathbf{a}')]. \quad (5.31)$$

Given data  $\{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}_{t=1}^N$ , the parameter  $\boldsymbol{\theta}$  is learned by minimizing the sum of squared Bellman residuals:

$$\sum_{t=1}^N \left( r(\mathbf{s}_t, \mathbf{a}_t) - \boldsymbol{\theta}^\top \widehat{\boldsymbol{\psi}}(\mathbf{s}_t, \mathbf{a}_t) \right)^2. \quad (5.32)$$

The vector  $\widehat{\psi}(\mathbf{s}, \mathbf{a})$  is an approximation of  $\psi(\mathbf{s}, \mathbf{a})$  based on data, i.e.,

$$\widehat{\psi}(\mathbf{s}_t, \mathbf{a}_t) = \varphi(\mathbf{s}_t, \mathbf{a}_t) - \frac{\gamma}{N'} \sum_{\mathbf{s}' \in \{(\mathbf{s}_t, \mathbf{a}_t) \rightarrow \mathbf{s}'\}} \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')} [\varphi(\mathbf{s}', \mathbf{a}')], \quad (5.33)$$

where the set  $\{(\mathbf{s}_t, \mathbf{a}_t) \rightarrow \mathbf{s}'\}$  is a set of states  $\mathbf{s}'$  observed after choosing action  $\mathbf{a}_t$  in state  $\mathbf{s}_t$ , and  $N'$  denotes the cardinality of the set  $\{(\mathbf{s}_t, \mathbf{a}_t) \rightarrow \mathbf{s}'\}$ . Since the current policy  $\pi$  is known, the expectation over  $\pi(\mathbf{a}'|\mathbf{s}')$  can be trivially computed or approximated. The sum of squared Bellman residuals in Equation (5.32) can be compactly represented in a matrix form as

$$\|\mathbf{r} - \Psi\boldsymbol{\theta}\|_2^2, \quad (5.34)$$

where  $\mathbf{r}$  is an  $N$ -dimensional vector and  $\Psi$  is an  $N$ -by- $b$  matrix defined respectively as

$$\mathbf{r} = [r(\mathbf{s}_1, \mathbf{a}_1), \dots, r(\mathbf{s}_N, \mathbf{a}_N)]^\top, \quad (5.35)$$

and

$$\Psi = [\widehat{\psi}(\mathbf{s}_1, \mathbf{a}_1), \dots, \widehat{\psi}(\mathbf{s}_N, \mathbf{a}_N)]^\top. \quad (5.36)$$

The parameter  $\boldsymbol{\theta}^*$  which minimizes the sum of squared Bellman residuals can be computed analytically as

$$\boldsymbol{\theta}^* = (\Psi^\top \Psi)^{-1} \Psi^\top \mathbf{r}. \quad (5.37)$$

Finally, the learned value function is obtained by substituting  $\boldsymbol{\theta}^*$  into the linear-in-parameter model.

Another variant of LSPI is called *LSPI based on least-squares fixed-point approximation* (Lagoudakis and Parr, 2003). In this variant, the value function is learned such that it is a fixed point of the *Bellman operator* (Bertsekas and Tsitsiklis, 1996). Here, we only give its solution without describing the complete details. The learned parameter  $\boldsymbol{\theta}^*$  of LSPI based on least-squares fixed-point approximation is given by

$$\boldsymbol{\theta}^* = (\Phi^\top \Psi)^{-1} \Phi^\top \mathbf{r}, \quad (5.38)$$

where  $\Phi$  is an  $N$ -by- $b$  matrix defined as

$$\Phi = [\varphi(\mathbf{s}_1, \mathbf{a}_1), \dots, \varphi(\mathbf{s}_N, \mathbf{a}_N)]^\top. \quad (5.39)$$

The advantage of LSPI over Q-learning is that the valued function is learned based on many transitions which makes the learned value function of LSPI tends to be more accurate overall (Lagoudakis and Parr, 2003). Moreover, LSPI computes the value function in a closed form and does not need to determine the step size unlike Q-learning. However, LSPI requires inversion of the matrix  $\Psi^\top \Psi$  which can be time consuming.

### 5.2.5 Summary of Policy Iteration

In this section, we have introduced the policy iteration approach which obtains an optimal policy based on learned value functions. While policy iteration was shown to work well and has many theoretical guarantees, they have two major disadvantages. Firstly, a small change in value functions may cause a large change in the policy. This phenomena is not preferable in domains such as robotics where stability of the agent's behavior is important. Secondly, the policy improvement step is often done by finding a maximizer of the state-action value function. For a low-dimensional discrete action space, the maximizer can be found by running through all possible value of actions. However, for a high-dimensional discrete action space or a continuous action space, running through all possible value of actions is improbable and we often require a non-linear optimization method to find a maximizer which can be time consuming. Moreover, the obtained maximizer might be a local maximizer and it will not guarantee that the new policy improves the current policy.

## 5.3 Direct Policy Search

The *direct policy search* approach directly finds an optimal policy  $\pi^*(\mathbf{a}|\mathbf{s})$  which maximizes the expected return:

$$\max_{\pi} \int R(\boldsymbol{\tau})p(\boldsymbol{\tau}; \pi)d\boldsymbol{\tau}, \quad (5.40)$$

without relying on value functions. Instead of finding the optimal policy in the space of functions, we often assume a parameterized policy with an adjustable parameter  $\boldsymbol{\theta}$ , i.e.,

$$\pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta}). \quad (5.41)$$

We further assume that there exists an optimal policy parameter  $\boldsymbol{\theta}^*$  such that  $\pi^*(\mathbf{a}|\mathbf{s}) = \pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta}^*)$ . That is, the optimal policy is obtained by finding the optimal policy parameter:

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \int R(\boldsymbol{\tau})p(\boldsymbol{\tau}; \boldsymbol{\theta})d\boldsymbol{\tau}, \quad (5.42)$$

where the trajectory density  $p(\boldsymbol{\tau}; \boldsymbol{\theta})$  now depends on  $\boldsymbol{\theta}$ . In this section, we review four common direct policy search approaches.

### 5.3.1 Policy Gradient

A common approach to find a maximizer of a function is the *gradient ascent* method (Nocedal and Wright, 2006). Direct policy search methods that are based on gradient ascent are called *policy gradient* methods. Firstly, let us denote the expected return by

$$\mathcal{J}(\boldsymbol{\theta}) = \int R(\boldsymbol{\tau})p(\boldsymbol{\tau}; \boldsymbol{\theta})d\boldsymbol{\tau}. \quad (5.43)$$

The gradient of  $\mathcal{J}(\boldsymbol{\theta})$  is denoted and given by

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \int R(\boldsymbol{\tau})p(\boldsymbol{\tau}; \boldsymbol{\theta})d\boldsymbol{\tau}. \quad (5.44)$$

Based on the gradient ascent method, we may obtain the optimal policy parameter by iteratively updating a policy parameter using the gradient, i.e.,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}), \quad (5.45)$$

where  $\alpha > 0$  denotes the step size. Since the expected return and its gradient are unknown, we need to estimate either of them from data. Below, we firstly review a classical policy gradient method called *REINFORCE* (Williams, 1992) which estimates the gradient from data.

The gradient of the expected return can be expressed as

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left( \int R(\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta}) d\boldsymbol{\tau} \right) \\ &= \int R(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau}; \boldsymbol{\theta}) d\boldsymbol{\tau} \\ &= \int R(\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) d\boldsymbol{\tau}, \end{aligned} \quad (5.46)$$

where in the last line we use the fact that  $\nabla_{\boldsymbol{\theta}} p(\boldsymbol{\tau}; \boldsymbol{\theta}) = p(\boldsymbol{\tau}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta})$ . Here we implicitly assume that the order of integration and differentiation is interchangeable. The gradient of the logarithm of the trajectory density is given by

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \log \left( p_1(\mathbf{s}) \prod_{t=1}^T p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t | \mathbf{s}_t; \boldsymbol{\theta}) \right) \\ &= \nabla_{\boldsymbol{\theta}} \log p_1(\mathbf{s}) + \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) + \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_t | \mathbf{s}_t; \boldsymbol{\theta}) \\ &= \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_t | \mathbf{s}_t; \boldsymbol{\theta}). \end{aligned} \quad (5.47)$$

We can see that  $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta})$  does not depend on the unknown transition probability density function  $p(\mathbf{s}' | \mathbf{s}, \mathbf{a})$ . Combining Equation (5.46) and Equation (5.47) together gives us

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \int R(\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta}) \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_t | \mathbf{s}_t; \boldsymbol{\theta}) d\boldsymbol{\tau}. \quad (5.48)$$

The expectation in the gradient can be approximated from  $N$  trajectory data  $\{\boldsymbol{\tau}_i\}_{i=1}^N$  where each  $\boldsymbol{\tau}_n$  is a trajectory with length  $T$  given by

$$\boldsymbol{\tau}_n = (\mathbf{s}_{1,n}, \mathbf{a}_{1,n}, r_{1,n}, \mathbf{s}_{2,n}, \dots, r_{T,n}, \mathbf{s}_{T+1,n}). \quad (5.49)$$

The *gradient estimate* is then computed by

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N R(\boldsymbol{\tau}_n) \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_{t,n} | \mathbf{s}_{t,n}; \boldsymbol{\theta}). \quad (5.50)$$

Finally, the policy parameter is updated by the gradient ascent rule based on the gradient estimate:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \widehat{\nabla}_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}). \quad (5.51)$$

It should be noted that the derivation in Equation (5.47) is only possible for a stochastic policy  $\pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$ . For a deterministic policy  $\pi(\mathbf{s}; \boldsymbol{\theta})$ , we instead have

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \log \left( p_1(\mathbf{s}) \prod_{t=1}^T p(\mathbf{s}_{t+1}|\mathbf{s}_t, \pi(\mathbf{s}_t; \boldsymbol{\theta})) \right) \\ &= \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \pi(\mathbf{s}_t; \boldsymbol{\theta})).\end{aligned}\quad (5.52)$$

Computing this gradient requires knowledge about the transition probability density  $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$  which is unknown. Also note that Equation (5.47) implicitly implies that the logarithm of the policy is differentiable, i.e.,  $\nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$  exists. Thus, this approach is not applicable to some policy functions such as those defined by the Dirac delta function  $\delta$  in Equation (5.2). Nonetheless, the term  $\nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$  can be trivially computed for many stochastic policy functions. A commonly used policy function is the Gaussian policy with linear-in-parameter mean, i.e.,

$$\pi(a^{(i)}|\mathbf{s}; \boldsymbol{\theta}_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{1}{2\sigma_i^2}(a^{(i)} - \mathbf{w}_i^\top \boldsymbol{\varphi}(\mathbf{s}))^2\right), \quad (5.53)$$

where  $\boldsymbol{\varphi}(\mathbf{s})$  is a basis function and  $\boldsymbol{\theta}_i = (\mathbf{w}_i, \sigma_i)$  are the policy parameters to be learned for the  $i$ -th dimension of the action. The gradient w.r.t.  $\mathbf{w}_i$  and  $\sigma_i$  of the logarithm of the Gaussian policy are simply given by

$$\nabla_{\mathbf{w}_i} \log \pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta}) = \frac{a^{(i)} - \mathbf{w}_i^\top \boldsymbol{\varphi}(\mathbf{s})}{\sigma_i^2} \boldsymbol{\varphi}(\mathbf{s}), \quad (5.54)$$

$$\nabla_{\sigma_i} \log \pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta}) = \frac{(a^{(i)} - \mathbf{w}_i^\top \boldsymbol{\varphi}(\mathbf{s}))^2 - \sigma_i^2}{\sigma_i^3}. \quad (5.55)$$

REINFORCE is considered by many researchers as the standard policy gradient method. It is very simple and very easy to implement. However, it was shown that the variance of gradient estimates of REINFORCE increases as the trajectory length  $T$  increases and this high variance tends to slow down the convergence (Peters and Schaal, 2006).

Fortunately, the variance of gradient estimates can be reduced by using the *baseline subtraction* technique (Williams, 1992). Firstly, from Equation (5.46) we can see that subtracting the return  $R(\boldsymbol{\tau})$  by a scalar  $b$  which is a constant w.r.t.  $\boldsymbol{\theta}$  does not change the gradient, i.e.,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left( \int (R(\boldsymbol{\tau}) - b) p(\boldsymbol{\tau}; \boldsymbol{\theta}) d\boldsymbol{\tau} \right) \\ &= \nabla_{\boldsymbol{\theta}} \left( \int R(\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta}) d\boldsymbol{\tau} \right) - b \nabla_{\boldsymbol{\theta}} \left( \int p(\boldsymbol{\tau}; \boldsymbol{\theta}) d\boldsymbol{\tau} \right) \\ &= \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) - b \nabla_{\boldsymbol{\theta}} 1 \\ &= \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}).\end{aligned}\quad (5.56)$$

Thus, the baseline  $b$  can be any scalar as long as it is a constant w.r.t.  $\boldsymbol{\theta}$ . The optimal choice of a baseline is the baseline that minimizes the variance of gradient estimates (Peters and Schaal, 2006) and it is given by

$$b^* = \frac{\sum_{n=1}^N R(\boldsymbol{\tau}_n) \left\| \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_{t,n}|\mathbf{s}_{t,n}; \boldsymbol{\theta}) \right\|_2^2}{\sum_{n=1}^N \left\| \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_{t,n}|\mathbf{s}_{t,n}; \boldsymbol{\theta}) \right\|_2^2}. \quad (5.57)$$

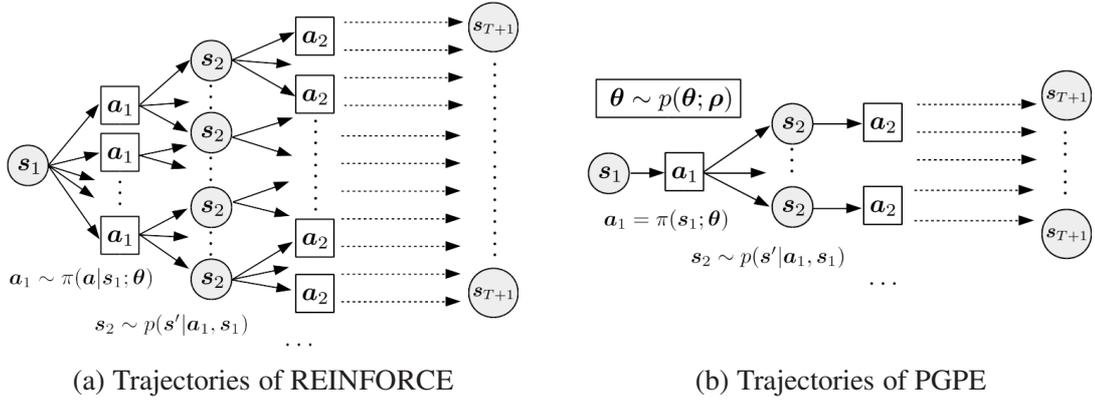


Figure 5.3: Comparison between trajectories of REINFORCE and PGPE. In REINFORCE, even from the same initial state  $s_1$ , there can be many possible trajectories due to the randomness of the stochastic policy  $\pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$ . In contrast, PGPE uses deterministic policy  $\pi(\mathbf{s}; \boldsymbol{\theta})$  and there is a smaller number of possible trajectories.

Then, the gradient estimates with optimal baseline subtraction is simply given by

$$\widehat{\nabla}_{\boldsymbol{\theta}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N (R(\boldsymbol{\tau}_n) - b^*) \sum_{t=1}^T \nabla_{\boldsymbol{\theta}} \log \pi(\mathbf{a}_{t,n} | \mathbf{s}_{t,n}; \boldsymbol{\theta}). \quad (5.58)$$

It was shown that the optimal baseline subtraction significantly improves the performance of REINFORCE (Peters and Schaal, 2006). However, even with the optimal baseline, gradient estimates in REINFORCE may still have relatively high variance.

### 5.3.2 Policy Gradient with Parameter-based Exploration

The main reason that makes the gradient estimate of REINFORCE to have high variance is the randomness caused by the stochastic policy. Since actions are chosen randomly at every time steps, there are many trajectories that can be obtained by a single policy  $\pi(\mathbf{a}|\mathbf{s})$ . This situation is illustrated in Figure 5.3(a). To overcome this issue, *policy gradient with parameter-based exploration* (PGPE) (Sehnke et al., 2010) was proposed.

The main idea of PGPE is to choose actions by a deterministic policy parameterized by a policy parameter  $\boldsymbol{\theta}$ . For example, we may employ a linear-in-parameter policy,

$$\mathbf{a} = \pi(\mathbf{s}; \boldsymbol{\theta}) = \boldsymbol{\theta}^{\top} \boldsymbol{\varphi}(\mathbf{s}). \quad (5.59)$$

Then, PGPE treats  $\boldsymbol{\theta}$  as a random variable and aims to learn a probability density of  $\boldsymbol{\theta}$  denoted by

$$p(\boldsymbol{\theta}; \boldsymbol{\rho}), \quad (5.60)$$

where  $\boldsymbol{\rho}$  is the parameter to be learned. In this setting, the expected return can be expressed as a function of  $\boldsymbol{\rho}$  as

$$\mathcal{J}(\boldsymbol{\rho}) = \iint R(\boldsymbol{\tau}) p(\boldsymbol{\tau} | \boldsymbol{\theta}) p(\boldsymbol{\theta}; \boldsymbol{\rho}) d\boldsymbol{\tau} d\boldsymbol{\theta}. \quad (5.61)$$

Note that we denote the trajectory density by a conditional density  $p(\boldsymbol{\tau}|\boldsymbol{\theta})$  instead of  $p(\boldsymbol{\tau}; \boldsymbol{\theta})$  since we treat  $\boldsymbol{\theta}$  as a random variable. The gradient w.r.t.  $\boldsymbol{\rho}$  of this expected return is given by

$$\begin{aligned}\nabla_{\boldsymbol{\rho}}\mathcal{J}(\boldsymbol{\rho}) &= \nabla_{\boldsymbol{\rho}} \left( \iint R(\boldsymbol{\tau})p(\boldsymbol{\tau}|\boldsymbol{\theta})p(\boldsymbol{\theta}; \boldsymbol{\rho})d\boldsymbol{\tau}d\boldsymbol{\theta} \right) \\ &= \iint R(\boldsymbol{\tau})p(\boldsymbol{\tau}|\boldsymbol{\theta})\nabla_{\boldsymbol{\rho}}p(\boldsymbol{\theta}; \boldsymbol{\rho})d\boldsymbol{\tau}d\boldsymbol{\theta} \\ &= \iint R(\boldsymbol{\tau})p(\boldsymbol{\tau}|\boldsymbol{\theta})p(\boldsymbol{\theta}; \boldsymbol{\rho})\nabla_{\boldsymbol{\rho}}\log p(\boldsymbol{\theta}; \boldsymbol{\rho})d\boldsymbol{\tau}d\boldsymbol{\theta},\end{aligned}\quad (5.62)$$

where in the last line we use the fact that  $\nabla_{\boldsymbol{\rho}}p(\boldsymbol{\theta}; \boldsymbol{\rho}) = p(\boldsymbol{\theta}; \boldsymbol{\rho})\nabla_{\boldsymbol{\rho}}\log p(\boldsymbol{\theta}; \boldsymbol{\rho})$ .

The data for estimating the gradient is collected as follows. For each trajectory  $\boldsymbol{\tau}_n$ , we firstly draw a policy parameter  $\boldsymbol{\theta}_n \sim p(\boldsymbol{\theta}; \boldsymbol{\rho})$  and then use a deterministic policy  $\pi(\boldsymbol{s}; \boldsymbol{\theta}_n)$  to select every action in the trajectory (see Figure 5.3(b)). This data collection process is repeated for  $N$  times to obtain data  $\{(\boldsymbol{\theta}_n, R(\boldsymbol{\tau}_n))\}_{n=1}^N$ . Then, the gradient is estimated from the data by

$$\widehat{\nabla}_{\boldsymbol{\rho}}\mathcal{J}(\boldsymbol{\rho}) = \frac{1}{N} \sum_{n=1}^N R(\boldsymbol{\tau}_n)\nabla_{\boldsymbol{\rho}}\log p(\boldsymbol{\theta}_n; \boldsymbol{\rho}).\quad (5.63)$$

Finally, the parameter  $\boldsymbol{\rho}$  is updated by the gradient ascent rule,

$$\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \alpha \widehat{\nabla}_{\boldsymbol{\rho}}\mathcal{J}(\boldsymbol{\rho}).\quad (5.64)$$

Similarly to REINFORCE, we may apply the baseline subtraction technique to reduce the variance of gradient estimates in PGPE. It is trivial to verify that subtracting the return with a baseline  $b$  does not change the gradient, i.e.,

$$\begin{aligned}\nabla_{\boldsymbol{\rho}}^{(\text{baseline})}\mathcal{J}(\boldsymbol{\rho}) &= \nabla_{\boldsymbol{\rho}} \left( \iint (R(\boldsymbol{\tau}) - b)p(\boldsymbol{\tau}|\boldsymbol{\theta})p(\boldsymbol{\theta}; \boldsymbol{\rho})d\boldsymbol{\tau}d\boldsymbol{\theta} \right) \\ &= \nabla_{\boldsymbol{\rho}} \left( \iint R(\boldsymbol{\tau})p(\boldsymbol{\tau}|\boldsymbol{\theta})p(\boldsymbol{\theta}; \boldsymbol{\rho})d\boldsymbol{\tau}d\boldsymbol{\theta} \right) - b\nabla_{\boldsymbol{\rho}}1 \\ &= \nabla_{\boldsymbol{\rho}}\mathcal{J}(\boldsymbol{\rho}),\end{aligned}\quad (5.65)$$

where  $b$  is a constant w.r.t.  $\boldsymbol{\rho}$ . The optimal baseline which minimizes the variance of gradient estimates in PGPE is given as follows (Zhao et al., 2012):

$$b^* = \frac{\sum_{n=1}^N R(\boldsymbol{\tau}_n)\|\nabla_{\boldsymbol{\rho}}\log p(\boldsymbol{\theta}_n; \boldsymbol{\rho})\|_2^2}{\sum_{n=1}^N \|\nabla_{\boldsymbol{\rho}}\log p(\boldsymbol{\theta}_n; \boldsymbol{\rho})\|_2^2}.\quad (5.66)$$

Using this optimal baseline, the gradient estimate is given by

$$\widehat{\nabla}_{\boldsymbol{\rho}}^{(\text{baseline})}\mathcal{J}(\boldsymbol{\rho}) = \frac{1}{N} \sum_{n=1}^N (R(\boldsymbol{\tau}_n) - b^*)\nabla_{\boldsymbol{\rho}}\log p(\boldsymbol{\theta}_n; \boldsymbol{\rho}).\quad (5.67)$$

The computation of the gradient estimates depends on the choice of  $p(\boldsymbol{\theta}; \boldsymbol{\rho})$ . A common choice is to use the Gaussian distribution for each dimension of  $\boldsymbol{\theta}$ , i.e.,

$$p(\theta^{(i)}; \boldsymbol{\rho}_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\tau_i^2}(\theta^{(i)} - \eta_i)^2\right),\quad (5.68)$$

where  $\rho_i = (\eta_i, \tau_i)$  are the parameters to be learned for the  $i$ -th dimension of  $\theta$ . The gradients w.r.t.  $\eta_i$  and  $\tau_i$  of the logarithm of the Gaussian distribution are given by

$$\nabla_{\eta_i} \log p(\theta; \rho) = \frac{\theta^{(i)} - \eta_i}{\tau_i^2}, \quad (5.69)$$

$$\nabla_{\tau_i} \log p(\theta; \rho) = \frac{(\theta^{(i)} - \eta_i)^2 - \sigma_i^2}{\tau_i^3}. \quad (5.70)$$

Using a deterministic policy to choose actions is advantageous since it removes randomness in the actions. Moreover, it was shown theoretically that the variance of the gradient estimates of PGPE does not increase with the trajectory length  $T$  (Zhao et al., 2012), unlike in the case of REINFORCE. Moreover, since the gradient estimation does not depend on the form of  $\pi(s; \theta)$ , we may use any function for  $\pi(s; \theta)$  including non-differentiable functions.

PGPE was shown to work very well when compared with other policy gradient methods such as REINFORCE (Sehnke et al., 2010; Zhao et al., 2012). However, using PGPE in practice have two important issues. Firstly, the convergence of PGPE heavily depends on the step size  $\alpha$  which is nontrivial to be determined in practice (Nocedal and Wright, 2006). Secondly, accurately estimating the gradients require quite a large amount of data to be collected. In practical domains such as robotics, data is a limited resource and collecting data from the agent can be expensive and need careful considerations. Although the sample reuse technique was shown to significantly improve data efficiency of PGPE (Zhao et al., 2013; Sugimoto et al., 2016), it still needs quite a lot of data for complex problems with many parameters to be learned.

In the followings subsections, we review two direct policy search approaches that are not based on gradient ascent and thus do not have the step size issue.

### 5.3.3 Expectation-Maximization

*Expectation-maximization* (Dempster et al., 1977) is an approach to find a maximum likelihood estimate of a parameter. The general idea of EM is to alternately compute the expectation of a log-likelihood function using the current parameter, and then find a new parameter which maximizes the expected log-likelihood function. The former step is called the *E-step* or the *expectation step*, and the latter step is called the *M-step* or the *maximization step*. In this subsection, we review an application of EM in direct policy search.

For simplicity, we assume the same policy model as that in PGPE. This means that the policy is a deterministic policy parameterized by  $\theta$  and the goal is to learn a parameter  $\rho$  of the probability density  $p(\theta|\rho)$  such that the expected return,

$$\mathcal{J}(\rho) = \iint R(\tau)p(\tau|\theta)p(\theta; \rho)d\tau d\theta, \quad (5.71)$$

is maximized. By assuming that  $R(\tau) \geq 0$ , the logarithm of the expected return is given by

$$\log \mathcal{J}(\rho) = \log \iint R(\tau)p(\tau|\theta)p(\theta; \rho)d\tau d\theta. \quad (5.72)$$

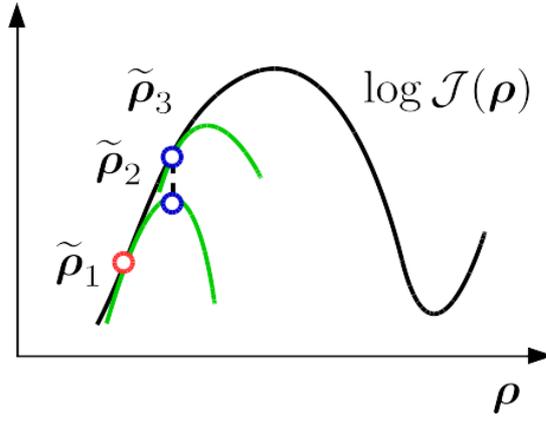


Figure 5.4: A typical behavior of EM-based approach. The black curve denotes the logarithm of the expected return. The green curves denote lower-bounds of the logarithm of the expected return evaluated at points  $\tilde{\rho}$ . In the E-step, EM computes the lower bound at the current parameter, e.g.,  $\tilde{\rho}_1$  at the red point. In the M-step, EM finds a maximizer of the lower-bound, e.g.,  $\tilde{\rho}_2$  at the blue point. Then, the parameter is updated to be the maximizer of the lower-bound and the process is repeated.

Then, by applying the Jensen inequality (Jensen, 1906), we obtain a lower-bound of the logarithm of the expected return as

$$\log \mathcal{J}(\rho) \geq \frac{1}{\mathcal{J}(\tilde{\rho})} \iint p(\boldsymbol{\theta}; \tilde{\rho}) R(\boldsymbol{\tau}) p(\boldsymbol{\tau} | \boldsymbol{\theta}) \log \left( \frac{p(\boldsymbol{\theta}; \rho)}{p(\boldsymbol{\theta}; \tilde{\rho})} \right) d\boldsymbol{\tau} d\boldsymbol{\theta} + \log \mathcal{J}(\tilde{\rho}), \quad (5.73)$$

where  $\tilde{\rho}$  is the current parameter. This inequality suggests that the quantity in the left-hand side can be obtained by maximizing the lower-bound in the right-hand side w.r.t.  $\rho$ . By ignoring terms independent of  $\rho$  in the right-hand side, we obtain an M-step,

$$\rho \leftarrow \operatorname{argmax}_{\rho} \iint p(\boldsymbol{\theta}; \tilde{\rho}) R(\boldsymbol{\tau}) p(\boldsymbol{\tau} | \boldsymbol{\theta}) \log p(\boldsymbol{\theta}; \rho) d\boldsymbol{\tau} d\boldsymbol{\theta}. \quad (5.74)$$

This M-step also guarantees that the expected return is improved (Peters and Schaal, 2007; Hachiya et al., 2011a), i.e.,

$$\mathcal{J}(\rho) \geq \mathcal{J}(\tilde{\rho}). \quad (5.75)$$

In the E-step, the expectation in Equation (5.74) is approximated from data. By using data  $\{(\boldsymbol{\theta}_n, R(\boldsymbol{\theta}_n))\}_{n=1}^N$  collected with parameter  $\tilde{\rho}$ , the E-step and the M-step can be collectively expressed via an update rule,

$$\rho \leftarrow \operatorname{argmax}_{\rho} \frac{1}{N} \sum_{n=1}^N R(\boldsymbol{\tau}_n) \log p(\boldsymbol{\theta}_n; \rho). \quad (5.76)$$

Figure 5.4 illustrates the behavior of the EM-based direct policy search.

The update rule of the EM-based approach mainly depends on the form of  $p(\boldsymbol{\theta}; \rho)$ . The density  $p(\boldsymbol{\theta}; \rho)$  is often chosen so that the maximizer can be computed in a closed form. For example, Kober and Peters (2011) proposed to use the Gaussian distribution and derived an update rule which closely resembles a weighted least-squares solution of a regression problem that can be efficiently computed in a closed form.

While we only introduce the EM-based approach for learning the parameter  $\rho$  of  $p(\theta; \rho)$ , there also exists EM-based approaches which learn the policy parameter  $\theta$  directly (Dayan and Hinton, 1997; Peters and Schaal, 2007; Hachiyu et al., 2011a). In such settings, the same derivation and interpretation of the EM-based approach that we presented here can be applied as well.

The main advantage of EM-based methods is that we do not need to choose the step size for the update rule, provided that the maximizer in Equation (5.76) can be computed in a closed form. However, the major issue of EM-based methods is that the amount of update can be very large as can be seen from the illustration in Figure 5.4, where the distance from red point to the blue point is quite large. In domains involving physical systems such as robotics, a large amount of updates between iterations is not preferred since they can cause instability to physical systems (Deisenroth et al., 2013).

### 5.3.4 Information-Theoretic Approach

The principal idea of the *information-theoretic* approach for direct policy search is to stay close to data (Peters et al., 2010; Deisenroth et al., 2013). More specifically, probability distributions of data between consecutive policy updates should not be too different. Since data is collected through a policy, this implies that the new policy should not be too different from the current policy as well. Below, we review an information-theoretic direct policy search method that is based on the *relative entropy policy search* (REPS) framework (Peters et al., 2010).

We assume a similar policy model as those in PGPE and the EM-based approach with slight changes in notation. Here, the aim is to find a probability density function  $p(\theta)$  which maximizes the expected return,

$$\int R(\theta)p(\theta)d\theta. \quad (5.77)$$

This problem setting can be made equivalent to those in PGPE and the EM-based approach by setting  $R(\theta) = \int R(\tau)p(\tau|\theta)d\tau$  and finding a parameter  $\rho$  of the density function  $p(\theta; \rho)$ .

The REPS framework implements the idea of the information-theoretic approach by upper-bounding the *Kullback-Leibler* (KL) divergence (Kullback and Leibler, 1951). As previously introduced in Chapter 2, a divergence is a pseudo-distance between two probability distributions and the KL-divergence from  $p(\theta)$  to  $q(\theta)$  is defined as

$$\text{KL}(p(\theta)\|q(\theta)) = \int p(\theta) \log \frac{p(\theta)}{q(\theta)} d\theta. \quad (5.78)$$

The KL-divergence is also known as the *relative entropy*. Based on the KL-divergence, each update iteration of REPS solves the following optimization problem:

$$\begin{aligned} & \max_p \int R(\theta)p(\theta)d\theta \\ & \text{subject to } \text{KL}(p(\theta)\|q(\theta)) \leq \epsilon, \\ & \int p(\theta)d\theta = 1, \end{aligned} \quad (5.79)$$

where  $q(\theta)$  is the current density function. The important characteristic of this optimization problem is that the KL-divergence between two consecutive density functions are upper-bounded by  $\epsilon$ . Thus, the amount of update between two consecutive iterations will not be too large, provided that the upper-bound  $\epsilon$  is appropriately chosen.

The REPS framework solves the constrained optimization problem in Equation (5.79) by the *method of Lagrange multipliers* (Falk, 1967; Boyd and Vandenberghe, 2004). The method of Lagrange multipliers is an important tool in mathematics and machine learning for solving general constrained optimization problems. We briefly describe the main idea of the method of Lagrange multipliers as follows. Assume that our goal is to solve a generic constrained optimization problem:

$$\begin{aligned} & \max_{\mathbf{x}} f(\mathbf{x}) \\ & \text{subject to } g_i(\mathbf{x}) \geq 0, \text{ for } i = 1, \dots, m, \\ & h_j(\mathbf{x}) = 0, \text{ for } j = 1, \dots, n, \end{aligned} \quad (5.80)$$

where  $f(\mathbf{x})$  is the function to be maximized,  $g_i(\mathbf{x}) \geq 0$  are inequality constraints, and  $h_j(\mathbf{x}) = 0$  are equality constraints. This constrained optimization can be reformulated into an equivalent *unconstrained* optimization problem by using the *Lagrange multipliers*  $\{\lambda_i\}_{i=1}^m$ ,  $\lambda_i \geq 0$  for the inequality constraints and  $\{\nu_j\}_{j=1}^n$  for the equality constraints. More specifically, Equation (5.80) is equivalent to the following optimization problem,

$$\max_{\mathbf{x}} \min_{\{\lambda_i\}_{i=1}^m \geq 0, \{\nu_j\}_{j=1}^n} \left[ f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^n \nu_j h_j(\mathbf{x}) \right]. \quad (5.81)$$

To verify that they are equivalent, we can firstly check the fact that

$$\min_{\lambda_i \geq 0} \lambda_i g_i(\mathbf{x}) = \begin{cases} 0 & \text{if } g_i(\mathbf{x}) \geq 0, \\ -\infty & \text{otherwise.} \end{cases} \quad (5.82)$$

This is because if the constraint is satisfied, i.e.,  $g_i(\mathbf{x}) \geq 0$ , then we can set  $\lambda_i = 0$  to achieve the minimum of 0. On the other hand, if the constraint is violated, i.e.,  $g_i(\mathbf{x}) < 0$ , then we can set  $\lambda_i = \infty$  to achieve the minimum of  $-\infty$ . The same interpretation also apply to the equality constraint. Thus, the optimization problem in Equation (5.81) is reduced to  $\max_{\mathbf{x}} f(\mathbf{x})$  when the constraints are satisfied and it does not have a solution when the constraints are violated. Therefore, the optimization problems in Equation (5.80) and Equation (5.81) are equivalent. The objective function in Equation (5.81) is called the *Lagrangian*:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\mathbf{x}) + \sum_{i=1}^m \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^n \nu_j h_j(\mathbf{x}), \quad (5.83)$$

where we abuse the notation and represent  $\{\lambda_i\}_{i=1}^m$  with  $\boldsymbol{\lambda}$  and  $\{\nu_j\}_{j=1}^n$  with  $\boldsymbol{\nu}$ . Next, it can be easily verified that every function satisfies the *min-max inequality* (Boyd and Vandenberghe, 2004), i.e.,

$$\max_{\mathbf{x}} \min_{\boldsymbol{\lambda} \geq 0, \boldsymbol{\nu}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq \min_{\boldsymbol{\lambda} \geq 0, \boldsymbol{\nu}} \max_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}). \quad (5.84)$$

The left-hand side of the inequality is equivalent to the original optimization problem in Equation (5.80) and is called the *primal problem*. The right-hand side of the inequality is called the *dual problem* and the function  $\max_x \mathcal{L}(x, \boldsymbol{\lambda}, \boldsymbol{\nu})$  is called the *dual function*. This dual function is an upper-bound of the primal problem, and the method of Lagrange multipliers aims to find the Lagrange multipliers  $\boldsymbol{\lambda} \geq 0$  and  $\boldsymbol{\nu}$  which minimize this upper-bound. Under certain conditions such as convexity of the functions, the bound is tight and solutions of the primal and dual problems are equivalent (Boyd and Vandenberghe, 2004). However, such conditions may not hold and there will be a gap between the two problems. In such a case, the obtained solution via the method of Lagrange multipliers is only an approximate solution.

By using the method of Lagrange multipliers introduced above, the REPS framework computes the new density function  $p(\boldsymbol{\theta})$  as

$$p(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta}\right), \quad (5.85)$$

where the normalization term is omitted and  $\eta$  is the Lagrange multiplier corresponding to the KL-divergence upper-bound. This multiplier is obtained by minimizing the dual function:

$$g(\eta) = \eta\epsilon + \eta \log \int q(\boldsymbol{\theta}) \exp\left(\frac{R(\boldsymbol{\theta})}{\eta}\right) d\boldsymbol{\theta}. \quad (5.86)$$

The expectation in the dual function can be approximated from data  $\{(\boldsymbol{\theta}_i, R(\boldsymbol{\theta}_i))\}_{i=1}^N$  as

$$\widehat{g}(\eta) = \eta\epsilon + \eta \log \frac{1}{N} \sum_{i=1}^N \exp\left(\frac{R(\boldsymbol{\theta}_i)}{\eta}\right). \quad (5.87)$$

Although the new density function  $p(\boldsymbol{\theta})$  can be computed by Equation (5.85), it is only defined on the data points  $\boldsymbol{\theta}_i$  where we already know their returns  $R(\boldsymbol{\theta}_i)$  and it may not be useful in practice. To obtain a more practical solution, REPS performs a maximum likelihood estimation to determine a parameter  $\boldsymbol{\rho}$  of a parameterized density  $p(\boldsymbol{\theta}; \boldsymbol{\rho})$  such that  $p(\boldsymbol{\theta}; \boldsymbol{\rho})$  matches the solution  $p(\boldsymbol{\theta})$  in Equation (5.85). More specifically, the parameter  $\boldsymbol{\rho}$  is obtained by the weighted maximum log-likelihood estimation,

$$\max_{\boldsymbol{\rho}} \frac{1}{N} \sum_{i=1}^N \exp\left(\frac{R(\boldsymbol{\theta}_i)}{\eta}\right) \log p(\boldsymbol{\theta}_i; \boldsymbol{\rho}). \quad (5.88)$$

At each update iteration in the REPS framework, the dual function  $\widehat{g}(\eta)$  is minimized to obtain the multiplier  $\eta$ . Then, the new parameterized density function  $p(\boldsymbol{\theta}; \boldsymbol{\rho})$  is obtained by the weight maximum log-likelihood estimation. Interestingly, this maximum likelihood estimation closely resembles the update rule of the EM-based approach in Equation (5.76).

The information-theoretic direct policy search method that we reviewed above is an instance of the REPS framework applied to the problem setting in Equation (5.77). The REPS framework has been applied to many settings of direct policy search. The most important instance of REPS would be the *step-based REPS* (Peters et al., 2010) which aims to learn a parameter  $\boldsymbol{\theta}$  of the policy  $\pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$ . This is in fact the same problem setting that we introduced for REINFORCE. Another important instance of

REPS is the *contextual REPS* (Deisenroth et al., 2013) which aims to solve the *direct contextual policy search* problem where the transition probability and the reward function are determined by observable variables called contexts.

It was experimentally shown that the REPS framework yields very stable policy updates and tends to perform better than competitive methods. However, the framework has two critical drawbacks. Firstly, it has a tuning parameter, namely the KL-upper bound  $\epsilon$ , which needs to be appropriately chosen. Secondly, the REPS framework relies on the method of Lagrange multipliers to find a solution in each update iteration. However, as we have shown, the method of Lagrange multipliers requires us to find a minimizer of the dual function. Solving this sub-optimization problem often relies on a non-linear optimization method such as the gradient descent method or the Newton’s method (Nocedal and Wright, 2006), which can be time consuming. Moreover, real-world problems rarely satisfy the conditions to achieve min-max equality in Equation (5.81). Thus, the solutions obtained by the method of Lagrange multipliers are not necessarily good solutions in principal.

### 5.3.5 Summary of Direct Policy Search

In direct policy search, an optimal policy parameter (or its distribution) is optimized so that the expected return is maximized. An important advantage of this approach over the policy iteration approach is that it is naturally applicable to continuous state and action spaces. Moreover, actions can be computed trivially from a policy parameter, unlike the policy iteration approach which requires finding a maximizer of a value function. For these reasons, direct policy search is usually more preferable in domains involving continuous state and actions spaces.

The direct policy search methods we reviewed above use data to estimate some quantities, e.g., policy gradient methods estimate gradients of the expected return from data. In general, accurately estimating these quantities often require a large amount of data. However, collecting data in practical domains such as robotics is often expensive and collecting a large amount of data is considered impractical. This *data efficiency* is in fact an important evaluation in reinforcement learning. That is, the practical usefulness of a reinforcement learning method often depends on how much data the method requires in order to achieve a desirable performance.

There are generally two approaches to improve data efficiency of direct policy search methods. The first approach is the *sample reuse* approach which reuses data collected in previous update iterations to estimate a quantity in the current iteration. Direct policy search methods with sample reuse were shown to perform better than methods without sample reuse (Peshkin and Shelton, 2002; Hachiya et al., 2011b; Zhao et al., 2013). However, even with sample reuse these methods still have a limitation when the budget for collecting data is limited. This is because they need to determine the *sampling schedule*, i.e., how many data points the agent needs to collect in each iteration. This sampling schedule heavily affects the performance of these methods but unfortunately they cannot determine this schedule before hand.

The second approach which improves data efficiency of direct policy search is the model-based approach, which is explained in the next section below.

## 5.4 Model-based Reinforcement Learning

The reinforcement learning methods that we have reviewed so far use the collected data in the policy learning directly by, e.g., approximating the involved expectations using the collected data. This class of reinforcement learning methods is called *model-free* reinforcement learning. Another class of reinforcement learning methods is *model-based* reinforcement learning which uses the collected data to firstly learn about unknown environment. In this section, we focus on model-based reinforcement learning.

### 5.4.1 Learning the Environment Model

As introduced earlier in this chapter, the environment consists of the transition probability density function  $p(s'|s, \mathbf{a})$ , the reward function  $r(s, \mathbf{a}, s')$ , and the initial state probability density function  $p_1(s)$ . In model-free reinforcement learning, the agent interacts with the environment to collect data  $\{\tau_n\}_{n=1}^N$  and then directly uses this data to update the policy by e.g., approximating the involved expectations using the collected data. Now, let us assume that the agent *knows* the transition probability density, the reward function, and the initial state probability density. In such a case, the agent does not need to interact with the environment to collect any data since it can use its knowledge about the environment to learn the optimal policy by, e.g., generating data by itself.

Unfortunately, the above situation is too unrealistic since in practice we rarely have exact knowledge about the environment. However, it suggests the idea that once an accurate model of the environment is obtained, the agent can instead rely on the model to learn the optimal policy. This is the main idea behind model-based reinforcement learning which uses the collected data to firstly learn a model of the environment and then learn an optimal policy based on the model. More specifically, model-based reinforcement learning methods learn all of the unknown transition probability density  $p(s'|s, \mathbf{a})$ , the reward function  $r(s, \mathbf{a}, s')$ , and the initial state probability density  $p_1(s)$ . In many settings, however, the reward function and the initial state probability density are determined by users and are considered known. Thus, most model-based reinforcement learning methods focus on learning a model of the transition probability density and this model is often called the *transition model*.

There are two approaches to use the model. The first approach is the *simulation-based approach* which generates *artificial data* from the model and then uses model-free reinforcement learning methods to learn an optimal policy from this artificial data. The second approach is the *integrated approach* where the model is an essential part of policy learning and the policy is learned without any data generation. There is no clear answer which approach is the best approach for using the model. Experimental evaluations showed that the integrated approach tends to perform better than the simulation-based approach (Deisenroth and Rasmussen, 2011). However, the integrated approach requires strong assumptions about the transition probability density function and the reward function and may not be widely applicable.

The choice of using model-free or model-based reinforcement learning mostly depends on the problems and there is no clear justification to choose between them. However, in many cases, the model-based approach is more preferable when the budget for data collection is limited or when the transition probability density can be learned accurately from a small amount of data.

The important concern of model-based reinforcement learning is how to learn the (transition) model from data. Below, we introduce approaches that were commonly used to learn the transition model. We assume that the trajectory data  $\{\tau_n\}_{n=1}^N$  is converted into transition data  $\{(s_i, \mathbf{a}_i, s'_i)\}_{i=1}^M$  where  $M = N \times T$  is the total number of observed transitions. Note that while we only focus on learning the transition model, these approaches can be straightforwardly applied to learning the reward model as well.

### 5.4.2 Locally Weighted Linear Regression

The problem of learning a transition model  $\hat{p}(s'|s, \mathbf{a})$  from data  $\{(s_i, \mathbf{a}_i, s'_i)\}_{i=1}^M$  is in fact a supervised learning problem where  $(s, \mathbf{a})$  is the input and  $s'$  is the output. Instead of learning a model of the conditional probability density  $\hat{p}(s'|s, \mathbf{a})$ , we may only learn a model that accurately predicts the expected value of  $s'$  given  $(s, \mathbf{a})$ . This learning problem is called *regression* and is the central problem in supervised learning. More specifically, regression aims to find a function  $f(s, \mathbf{a})$  such that

$$s' = f(s, \mathbf{a}) + \epsilon, \quad (5.89)$$

where  $\epsilon$  is independent noise with zero mean and finite variance. In regression we often assume a scalar output. For vector-valued output, we may learn multiple regression models where each model corresponds to a dimension of the output, i.e.,  $s^{(j)'} = f_j(s, \mathbf{a}) + \epsilon_j$ . The index  $j$  is omitted in the remainder of this section for notational convenience and we only consider learning a model with a scalar output  $s'$ . Below, we review a regression method called *locally weighted linear regression* (LWR) (Cleveland and Devlin, 1988; Atkeson et al., 1997) which is widely used for learning the transition model.

For notational convenience, we denote the input  $(s, \mathbf{a})$  by a vector  $\mathbf{x} \in \mathbb{R}^{d_x}$  and the transition probability  $p(s'|s, \mathbf{a})$  by  $p(s'|\mathbf{x})$  where  $d_x = d_s + d_a$ . *Linear regression* (Hastie et al., 2001; Bishop, 2006; Murphy, 2012) is the simplest form of regression which assumes that the function  $f$  is linear in both input and parameter, i.e.,

$$f(\mathbf{x}; \boldsymbol{\beta}) = \boldsymbol{\beta}^\top \mathbf{x}, \quad (5.90)$$

where  $\boldsymbol{\beta} \in \mathbb{R}^{d_x}$  is a parameter vector to be learned. The parameter is often learned by minimizing the sum of squared errors,

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^M (\boldsymbol{\beta}^\top \mathbf{x}_i - s'_i)^2. \quad (5.91)$$

The solution is obtained in a closed form as

$$\hat{\boldsymbol{\beta}} = (\mathbf{X} \mathbf{X}^\top)^{-1} \mathbf{X} \vec{s}', \quad (5.92)$$

where  $\mathbf{X}$  is a  $d_x$ -by- $M$  data matrix and  $\vec{s}'$  is an  $M$ -dimensional vector of the output  $s'$ . Then, given a new input  $\tilde{\mathbf{x}}$ , the predicted output  $\tilde{s}'$  is given by

$$\tilde{s}' = \hat{\boldsymbol{\beta}}^\top \tilde{\mathbf{x}}. \quad (5.93)$$

The main issue of linear regression is that the model is *globally* linear and this is strongly restrictive. To overcome this issue, *locally weighted linear regression*

(LWR) (Cleveland and Devlin, 1988; Atkeson et al., 1997) was proposed. In LWR, each pair of input points is associated with the *weight* that determines similarity between the two points. The commonly used weight is the exponential weight defined as

$$w(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{D} (\mathbf{x}_i - \mathbf{x}_j)\right), \quad (5.94)$$

where  $\mathbf{D}$  is a tuning parameter matrix. Unlike linear regression, LWR does not compute the parameter  $\beta$  during training. Instead, given a new input  $\tilde{\mathbf{x}}$ , LWR finds a parameter  $\beta$  of the linear model by solving the following optimization problem:

$$\min_{\beta} \sum_{i=1}^M w(\mathbf{x}_i, \tilde{\mathbf{x}}) (\beta^\top \mathbf{x}_i - s'_i)^2. \quad (5.95)$$

This means that the influence of a data point  $(\mathbf{x}_i, s'_i)$  for the estimated parameter  $\hat{\beta}$  depends on the similarity between  $\mathbf{x}_i$  and  $\tilde{\mathbf{x}}$ . Thus, the LWR model is linear only on the local region where the new input  $\tilde{\mathbf{x}}$  is located. The solution can be computed in a closed form as

$$\hat{\beta} = (\mathbf{X} \mathbf{W} \mathbf{X}^\top)^{-1} \mathbf{X} \mathbf{W} \vec{s}', \quad (5.96)$$

where  $\mathbf{W}$  is an  $M$ -by- $M$  matrix whose  $(i, j)$ -th entry is the weight  $w(\mathbf{x}_i, \mathbf{x}_j)$ . Finally, the predicted output  $\hat{s}'$  is given by

$$\hat{s}' = \hat{\beta}^\top \tilde{\mathbf{x}}. \quad (5.97)$$

LWR is a popular transition model learning method for many model-based reinforcement learning methods (Schaal and Atkeson, 1994; Schneider, 1996; Atkeson et al., 1997; Schaal et al., 2000). It was applied to learn the value functions in model-free reinforcement learning as well (Boyan and Moore, 1994; Neumann and Peters, 2008). However, the important issue of LWR is that it does not consider the uncertainty of the transition probability. More specifically, regression only learns the conditional mean of the transition probability and does not take into account the randomness of the transition probability. Next, we introduce a method that learns the actual transition probability.

### 5.4.3 Gaussian Process Regression

*Gaussian process regression* (GP) (Rasmussen and Williams, 2006) is another method that is widely used in model-based reinforcement learning. Unlike LWR, GP aims to learn the transition probability density function as a conditional density function and not just the conditional mean function. Below, we briefly review GP for transition model learning.

The important assumption of GP is the assumption that the transition probability is Gaussian, i.e.,

$$p(s'|\mathbf{x}) = \mathcal{N}(s'|m(\mathbf{x}), \sigma(\mathbf{x})), \quad (5.98)$$

where  $m(\mathbf{x})$  and  $\sigma(\mathbf{x})$  denote the mean and the variance, respectively. The goal of GP is to estimate the mean  $m(\tilde{\mathbf{x}})$  and the variance  $\sigma(\tilde{\mathbf{x}})$  of the output  $s'$  for the new

input  $\tilde{\mathbf{x}}$  from data  $\{(\mathbf{x}_i, s'_i)\}_{i=1}^M$ . These estimated mean and variance are given by

$$m(\tilde{\mathbf{x}}) = \mathbf{k}^\top (\mathbf{K} + \lambda \mathbf{I}_M)^{-1} \vec{s}', \quad (5.99)$$

$$\sigma(\tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) - \mathbf{k}^\top (\mathbf{K} + \lambda \mathbf{I}_M)^{-1} \mathbf{k}, \quad (5.100)$$

respectively where  $\lambda$  is a tuning parameter corresponding to the standard deviation of the independent noise in data (Rasmussen and Williams, 2006). The vector  $\mathbf{k}$  is an  $M$ -dimensional vector and  $\mathbf{K}$  is an  $M \times M$  matrix defined as

$$\mathbf{k} = [k(\mathbf{x}_1, \tilde{\mathbf{x}}), \dots, k(\mathbf{x}_M, \tilde{\mathbf{x}})]^\top, \quad (5.101)$$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_M, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_1, \mathbf{x}_M) & \dots & k(\mathbf{x}_M, \mathbf{x}_M) \end{bmatrix}. \quad (5.102)$$

The function  $k(\mathbf{x}_i, \mathbf{x}_j)$  denotes the covariance function which defines similarity between input points  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . The important requirement of the covariance function is that the matrix  $\mathbf{K}$  needs to be positive semi-definite, i.e.,  $\mathbf{a}^\top \mathbf{K} \mathbf{a} \geq 0$  for any  $M$ -dimensional vector  $\mathbf{a}$ . A common choice of the covariance function is the squared exponential function defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{D} (\mathbf{x}_i - \mathbf{x}_j)\right), \quad (5.103)$$

where  $\mathbf{D}$  is a tuning parameter matrix. Note that this covariance function is the same as the exponential weight in Equation (5.94) for LWR. The tuning parameter  $\mathbf{D}$  and  $\lambda$  are usually determined by evidence maximization (Rasmussen and Williams, 2006).

GP has been applied to many model-based reinforcement learning methods with good successes (Deisenroth and Rasmussen, 2011; Kupcsik et al., 2013; Ko et al., 2007). However, the important issue of GP is that the transition probability  $p(s'|\mathbf{x})$  is assumed to be Gaussian. This Gaussian assumption can be easily violated in real-world problems and GP would not be an appropriate method in such problems.

#### 5.4.4 Summary of Model-based Reinforcement Learning

Model-based reinforcement learning was experimentally shown to be more data efficient than the model-free counterpart for both policy iteration methods (Sutton, 1990; Rasmussen and Kuss, 2003) and direct policy search methods (Wang and Dietterich, 2003; Deisenroth and Rasmussen, 2011; Kupcsik et al., 2013). An intuitive explanation is that once the transition model is learned, model-based methods do not need to collect more data in order to get more information. In contrast, model-free methods need to collect more data to get more information. For this reason, model-based methods are very attractive in domains where data efficiency is of primary concern.

While model-based methods are attractive, many researchers often avoid using them because they often suffer from *model bias*. More specifically, model-based methods learn an optimal policy based on the transition model and they implicitly assume that the transition model accurately represents the transition probability. In a case where the transition model is inaccurate, the learned policy will perform very poorly even though it is regarded as an optimal policy for the transition model. Thus, the success of model-based methods heavily depend on the accuracy of the transition

model. However, learning an accurate model is a challenging problem especially for high-dimensional state and action spaces.

The main focus of our contributions in the following sections is to develop model-based reinforcement learning methods which accurately learn a transition model even when the state and action spaces are high-dimensional.

## Chapter 6

# Model-based Policy Gradient with Parameter-based Exploration

In this chapter, we present our third contribution on a model-based policy gradient with parameter-based exploration method. Firstly, we briefly reintroduce reinforcement learning and the policy gradient with parameter-based exploration method. Then, we focus on the transition model estimation problem and review the least-squares conditional density estimation method. Next, we present our method that uses LSCDE to improve data efficiency of the policy gradient with parameter-based exploration method. Lastly, we present our experimental results on benchmark and simulated humanoid robot problems, and then conclude this chapter.

### 6.1 Introduction

Reinforcement learning aims to learn an optimal policy which controls an agent to achieve maximum cumulative rewards. Reinforcement learning methods can be categorized into policy iteration methods and directly policy search methods. Policy iteration methods learn an optimal policy based on value functions which represents expected cumulative reward from a given state and a given policy. Policy iteration methods alternately estimate value functions and improve policy until convergence. However, accurately estimating value functions in continuous state and action spaces is highly challenging. Moreover, a small change in value functions may cause a large change in policy, and this is not preferable in many domains where agent's stable behaviors are desired.

In direct policy search methods, a policy is optimized directly so that it achieve the maximum cumulative rewards. Among many direct policy search methods, policy gradient methods are simple and widely applicable. The most well-known policy gradient method is REINFORCE (Williams, 1992) which estimates the gradient of the expected return using data. However, it was shown that the variance of gradient estimates in REINFORCE increases with the trajectory length and can be very large which results in unreliable policy improvement (Peters and Schaal, 2006). A more recent policy gradient method is the *policy gradient with parameter-based exploration* (PGPE) (Sehnke et al., 2010). It was shown that the variance of gradient estimates in PGPE does not increase with the trajectory length (Zhao et al., 2012) which makes the policy improvement in PGPE tend to be more stable than that in REINFORCE. This property makes PGPE an attractive policy gradient method. However, PGPE is originally a model-free method where the collected data is used to estimate the gradient

of the expected return. In order to accurately estimate the gradient, the agent needs to collect a large amount of data in each iteration which makes PGPE not suitable in domains where a large amount of data is difficult to be obtained.

The *importance weight PGPE* (IW-PGPE) method (Zhao et al., 2013) was proposed to improve data efficiency of PGPE. The key idea of IW-PGPE is to reuse data from previous iterations to estimate gradient in the current iteration. Since naively reusing the data can cause bias in the gradient estimate, IW-PGPE corrects this bias by applying the importance weight technique. It was experimentally shown that IW-PGPE uses data more efficiently than PGPE. However, IW-PGPE is still a model-free method and requires collecting new data at each iteration to efficiently improve the policy.

Differently from the previous work by Zhao et al. (2013), we adopt the model-based reinforcement learning to improve data efficiency of PGPE. More specifically, our contribution is a model-based extension of PGPE called *model-based PGPE* (M-PGPE). Unlike model-free PGPE, M-PGPE uses the collected data to firstly learn a transition model. Then, it uses the transition model to generate a large amount of data for gradient estimation. There are two significant advantages of M-PGPE when compared with model-free PGPE. Firstly, given a fixed budget for data collection, we are not required to determine the sampling schedule in advance for M-PGPE. More specifically, we do not need to determine how many trajectories are collected at each iteration for M-PGPE since we can spend the whole budget at first to learn the transition model.

The second advantage of M-PGPE lies in *baseline subtraction*. As we explain in Chapter 5, baseline subtraction allows us to significantly reduce the variance of the gradient estimates in PGPE. In principal, subtraction by the optimal baseline reduces the variance without introducing the bias (see Equation (5.66)). In practice, model-free PGPE typically estimates this optimal baseline using the same data that is used for gradient estimation. However, this estimated optimal baseline may increase the estimation bias since the two data are not statistically independent. On the other hand, for M-PGPE we can simply generate two set of data and use one set of data for gradient estimation and the other set of data for optimal baseline estimation.

An important step of M-PGPE is to learn a transition model from data. In literature, many methods were proposed and used to learn a transition model. However, as we have reviewed in Chapter 5, regression-based methods such as locally weighted regression (Cleveland and Devlin, 1988; Atkeson et al., 1997) do not consider the randomness of the transition probability, and the Gaussian process (Rasmussen and Williams, 2006) relies on a strong assumption that the transition probability can be accurately represented by the Gaussian distribution. To overcome the limitations of these methods, we propose to learn a transition model by the *least-squares conditional density estimation* (LSCDE) method (Sugiyama et al., 2010). The advantage of LSCDE over existing transition model estimation method is that LSCDE non-parametrically estimates the transition probability and does not rely on strong assumption about the transition probability. In fact, it was shown that LSCDE posses the optimal asymptotic convergence rate, meaning that LSCDE can asymptotically estimate any transition probability (Sugiyama et al., 2010). Moreover, the solution of LSCDE can be efficiently computed in a closed form. Through experiments, we demonstrate that M-PGPE with LSCDE is a promising method.

The remaining of this chapter is organized as follows. In Section 6.2, we briefly recall the transition model estimation problem and review the LSCDE method in details.

Then in Section 6.3, we describe the learning framework of our M-PGPE method. Next, we experimentally evaluate M-PGPE in Section 6.4 and conclude this chapter in Section 6.5.

## 6.2 Transition Model Estimation via Least-Squares Conditional Density Estimation

The goal of transition model estimation is to learn a transition model  $\widehat{p}(s'|\mathbf{s}, \mathbf{a})$  from transition data  $\{(s_i, \mathbf{a}_i, s'_i)\}_{i=1}^P$ . This transition data can be obtained from trajectory data  $\{\tau_i\}_{i=1}^N$ . The transition model estimation problem is a supervised learning problem where  $(\mathbf{s}, \mathbf{a})$  is input and  $s'$  is output. For notational convenience, we denote input  $(\mathbf{s}, \mathbf{a})$  by a vector  $\mathbf{x} \in \mathbb{R}^{d_x}$  where  $d_x = d_s + d_a$ . In this case, the transition probability and transition model are denoted by  $p(s'|\mathbf{x})$  and  $\widehat{p}(s'|\mathbf{x})$ , respectively.

We use LSCDE to learn the transition model in M-PGPE. The goal of LSCDE is to find a transition model  $\widehat{p}(s'|\mathbf{x})$  which minimizes the squared error:

$$\frac{1}{2} \iint (\widehat{p}(s'|\mathbf{x}) - p(s'|\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} ds'. \quad (6.1)$$

This squared error can be expanded as

$$\frac{1}{2} \iint (\widehat{p}(s'|\mathbf{x})^2 p(\mathbf{x}) - 2\widehat{p}(s'|\mathbf{x})p(\mathbf{x}, s') - p(s'|\mathbf{x})^2 p(\mathbf{x})) d\mathbf{x} ds'. \quad (6.2)$$

The third term of the above equation can be ignored since it is a constant w.r.t. the transition model. The minimizer which minimizes the squared-error depends on the form of the transition model. Let us use a linear-in-parameter model for the transition model:

$$\widehat{p}(s'|\mathbf{x}) = \boldsymbol{\alpha}^\top \boldsymbol{\varphi}(\mathbf{x}, s'), \quad (6.3)$$

where  $\boldsymbol{\alpha} \in \mathbb{R}^b$  is a parameter vector to be learned and  $\boldsymbol{\varphi}(\mathbf{x}, s') \in \mathbb{R}^b$  is a basis function vector. In particular, we use the Gaussian basis function for the  $i$ -th dimension of the basis function vector:

$$\varphi^{(i)}(\mathbf{x}, s') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{u}_i\|^2}{2\sigma^2}\right) \exp\left(-\frac{\|s' - v_i\|^2}{2\sigma^2}\right), \quad (6.4)$$

where  $\mathbf{u}_i$  and  $v_i$  are the  $i$ -th Gaussian centers chosen randomly from the data and  $\sigma$  is the Gaussian width. Under this model, we have the squared error:

$$\frac{1}{2} \iint ((\boldsymbol{\alpha}^\top \boldsymbol{\varphi}(\mathbf{x}, s'))^2 p(\mathbf{x}) - \boldsymbol{\alpha}^\top \boldsymbol{\varphi}(\mathbf{x}, s') p(\mathbf{x}, s')) d\mathbf{x} ds'. \quad (6.5)$$

The expectations over  $p(\mathbf{x})$  and  $p(\mathbf{x}, s')$  in the squared error can be approximated from data  $\{(\mathbf{x}_i, s'_i)\}_{i=1}^P$ . This approximation yields the following empirical squared error:

$$\frac{1}{2} \boldsymbol{\alpha}^\top \widehat{\mathbf{H}} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \widehat{\mathbf{h}}, \quad (6.6)$$

where

$$\hat{\mathbf{h}} = \frac{1}{P} \sum_{i=1}^P \varphi(\mathbf{x}_i, \mathbf{s}'_i), \quad (6.7)$$

$$\widehat{\mathbf{H}} = \frac{1}{P} \sum_{i=1}^P \Phi(\mathbf{x}_i), \quad (6.8)$$

$$\Phi(\mathbf{x}) = \int \varphi(\mathbf{x}, \mathbf{s}') \varphi(\mathbf{x}, \mathbf{s}')^\top d\mathbf{s}'. \quad (6.9)$$

For the Gaussian basis function in Equation (6.4), the  $(k, k')$ -th entry of  $\Phi(\mathbf{x})$  can be computed in a closed form as

$$\Phi(\mathbf{x})_{(k,k')} = (\sqrt{\pi}\sigma)^{d_s} \exp\left(-\frac{2\|\mathbf{x} - \mathbf{x}_k\|^2 + 2\|\mathbf{x} - \mathbf{x}_{k'}\|^2 + \|\mathbf{u}_k - \mathbf{u}_{k'}\|^2}{4\sigma^2}\right). \quad (6.10)$$

To control the complexity of the model, an  $\ell_2$  regularization term with a regularization parameter  $\lambda > 0$  is included in the empirical squared error in Equation (6.6) as

$$\frac{1}{2} \boldsymbol{\alpha}^\top \widehat{\mathbf{H}} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \hat{\mathbf{h}} + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \boldsymbol{\alpha}. \quad (6.11)$$

By setting the derivative w.r.t.  $\boldsymbol{\alpha}$  to zero, we obtain the following closed form solution

$$\hat{\boldsymbol{\alpha}} = \left(\widehat{\mathbf{H}} + \lambda \mathbf{I}\right)^{-1} \hat{\mathbf{h}}. \quad (6.12)$$

Then, the transition model can be obtained by plugging in the solution into the model:

$$\hat{p}(\mathbf{s}'|\mathbf{x}) = \hat{\boldsymbol{\alpha}}^\top \varphi(\mathbf{x}, \mathbf{s}'). \quad (6.13)$$

In addition, we also need to ensure that the estimator is a proper conditional density, i.e., non-negative and integrated to one. To do so, the solution is truncated and the transition model is normalized as

$$\hat{p}(\mathbf{s}'|\mathbf{x}) = \frac{\tilde{\boldsymbol{\alpha}}^\top \varphi(\mathbf{x}, \mathbf{s}')}{\int \tilde{\boldsymbol{\alpha}}^\top \varphi(\mathbf{x}, \mathbf{s}') d\mathbf{s}'}, \quad (6.14)$$

where  $\tilde{\boldsymbol{\alpha}} = \max(\hat{\boldsymbol{\alpha}}, \mathbf{0})$  is the truncated solution. For the Gaussian basis function, the normalization term can also be computed in a closed form as

$$\int \tilde{\boldsymbol{\alpha}}^\top \varphi(\mathbf{x}, \mathbf{s}') d\mathbf{s}' = (\sqrt{2\pi}\sigma)^{d_s} \sum_{k=1}^b \tilde{\alpha}^{(b)} \exp\left(-\frac{\|\mathbf{x} - \mathbf{u}_k\|^2}{2\sigma^2}\right). \quad (6.15)$$

### 6.3 Policy Learning Framework

M-PGPE method is a model-based extension of PGPE where the transition model is learned by LSCDE. The policy learning framework of M-PGPE is given by the following steps.

1. Collect trajectory data  $\{\boldsymbol{\tau}_i\}_{i=1}^N$  and convert it to transition data  $\{(s_i, \mathbf{a}_i, s'_i)\}_{i=1}^P = \{(\mathbf{x}_i, s'_i)\}_{i=1}^P$  by, e.g., using a uniform random policy.
2. Learn a transition model  $\hat{p}(s'|\mathbf{x})$  from data  $\{(\mathbf{x}_i, s'_i)\}_{i=1}^P$  by LSCDE.
3. Initialize a parameter  $\boldsymbol{\rho}$  of the the policy parameter distribution  $p(\boldsymbol{\theta}; \boldsymbol{\rho})$ .
4. Draw policy parameters  $\{\boldsymbol{\theta}_i\}_{i=1}^{M+M'}$  from  $p(\boldsymbol{\theta}; \boldsymbol{\rho})$ .
5. Generate two set of artificial trajectory data:  $\{\hat{\boldsymbol{\tau}}_i\}_{i=1}^M$  and  $\{\hat{\boldsymbol{\tau}}_i\}_{i=1}^{M'}$ , using the transition model  $\hat{p}(s'|\mathbf{x})$  and policy parameter  $\{\boldsymbol{\theta}_i\}_{i=1}^{M+M'}$  and then evaluate their return  $R(\hat{\boldsymbol{\tau}})$ .
6. Estimate the optimal baseline  $b^*$  and gradient with baseline subtraction  $\hat{\nabla}_{\boldsymbol{\rho}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\rho})$  using two separate artificial trajectory data (see Chapter 5 for details):

$$b^* = \frac{\sum_{i=1}^M R(\hat{\boldsymbol{\tau}}_i) \|\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_i; \boldsymbol{\rho})\|_2^2}{\sum_{i=1}^M \|\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_i; \boldsymbol{\rho})\|_2^2}, \quad (6.16)$$

$$\hat{\nabla}_{\boldsymbol{\rho}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\rho}) = \frac{1}{M'} \sum_{i=1}^{M'} (R(\hat{\boldsymbol{\tau}}_i) - b^*) \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_i; \boldsymbol{\rho}). \quad (6.17)$$

7. Update parameter  $\boldsymbol{\rho}$  by gradient ascent with step size  $\beta > 0$ :

$$\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \beta \hat{\nabla}_{\boldsymbol{\rho}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\rho}). \quad (6.18)$$

8. Repeat Steps 4 to 7 until  $\boldsymbol{\rho}$  converges.

At Step 5, the artificial trajectory data are generated by sampling sequences of states, actions, and next states from the transition model. For the Gaussian basis function, this sampling procedure can be done similarly to sampling from a mixture of Gaussian distributions (Bishop, 2006). We emphasize that the number of artificial trajectory  $M$  and  $M'$  can be much larger than the number of collected trajectory  $N$ .

## 6.4 Experiment

In this section, we demonstrate the usefulness of the proposed M-PGPE method through experiments.

### 6.4.1 Continuous Chainwalk

For illustration purposes, let us first consider a simple continuous chain-walk task (see Figure 6.1).

**Setup:** Let

$$\begin{aligned} s &\in \mathcal{S} = [0, 10], \\ a &\in \mathcal{A} = [-5, 5], \\ r(s, a, s') &= \begin{cases} 1 & (4 < s' < 6), \\ 0 & (\text{otherwise}). \end{cases} \end{aligned}$$

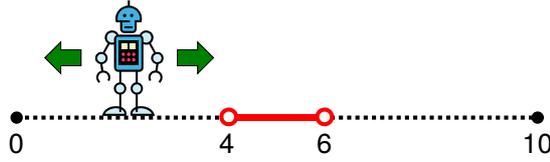


Figure 6.1: Illustration of continuous chain walk.

That is, the agent receives positive reward  $+1$  at the center of the state space. We set the trajectory length at  $T = 10$ , the discount factor at  $\gamma = 0.99$ , and the learning rate at  $\beta = 0.1$ . The initial mean parameter  $\eta$  is set at 0 and initial deviation parameter  $\tau$  is set at 1. We use the following linear-in-parameter policy model in all the compared methods, i.e., M-PGPE and IW-PGPE:

$$\forall s \in \mathcal{S}, \quad a = \sum_{i=1}^6 \theta_i \exp\left(-\frac{(s - c_i)^2}{2}\right),$$

where  $(c_1, \dots, c_6) = (0, 2, 4, 6, 8, 10)$ . If an action determined by the above policy is out of the action space, we pull it back to be confined in the domain. Thus, the action space itself is independent of the current position.

As transition probability, we consider two setups:

**Gaussian:** The true transition probability is given by

$$\forall s_t \in \mathcal{S}, a_t \in \mathcal{A},$$

where  $\epsilon_t$  is the Gaussian noise with mean 0 and standard deviation 0.3. If the next state is out of the state space, then we project it back to the domain.

**Bimodal:** The true transition probability is given by

$$\forall s_t \in \mathcal{S}, a_t \in \mathcal{A}, \quad s_{t+1} = s_t \pm a_t + \epsilon_t,$$

where  $\epsilon_t$  is the Gaussian noise with mean 0 and standard deviation 0.3, and the sign of  $a_t$  is randomly chosen with probability  $1/2$ .

We compare the following three policy search methods:

**M-PGPE(LSCDE):** The model-based PGPE method with transition model estimated by LSCDE.

**M-PGPE(GP):** The model-based PGPE method with transition model estimated by GP.

**IW-PGPE:** The model-free PGPE method with sample reuse by importance weighting<sup>1</sup> (Zhao et al., 2013).

Below, we consider the situation where the budget for data collection is limited to  $N = 20$  trajectory data.

<sup>1</sup>We have also tested the plain PGPE method without importance weighting, but this did not perform well in our preliminary experiments. For this reason, we decided to omit the results.

## LSCDE VS. GP

When the transition model is learned by the M-PGPE methods, all  $N = 20$  trajectory data are gathered randomly in the beginning at once. More specifically, the initial state  $s_1$  and the action  $a_1$  are chosen from the uniform distributions over  $\mathcal{S}$  and  $\mathcal{A}$ , respectively. Then the next state  $s_2$  and the immediate reward  $r_1$  are obtained. Then the action  $a_2$  is chosen from the uniform distribution over  $\mathcal{A}$ , and the next state  $s_3$  and the immediate reward  $r_2$  are obtained. This process is repeated until we obtain  $r_T$ . This gives a trajectory data, and we repeat this data generation process  $N$  times to obtain  $N$  trajectory data. These  $N = 20$  trajectory data give  $P = 200$  transition data for transition-model learning.

In the implementation of LSCDE, the Gaussian width  $\sigma$  and the regularization parameter  $\lambda$  are selected by cross-validation from the following candidate values:

$$\begin{aligned} \kappa &\in \{0.0316, 0.0487, 0.0750, 0.1155, 0.1778, 0.2738, 0.4217, 0.6494, 1.00\}, \\ \lambda &\in \{0.0010, 0.0032, 0.0100, 0.0316, 0.1000, 0.31620, 1.0000, 3.1623, 10.0000\}. \end{aligned}$$

Figure 6.2 and Figure 6.5 illustrate the true transition probability and its estimates obtained by LSCDE and GP in the Gaussian and bimodal cases, respectively. Figure 6.2(b) and Figure 6.5(b) show the conditional density estimation error of the learned transition models. The conditional density estimators are evaluated by the squared error (without constant) computed from test data points  $\{(\mathbf{x}_i, \mathbf{s}'_i)\}_{i=1}^{N_{\text{test}}}$ :

$$\frac{1}{2N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \int \widehat{p}(\mathbf{s}'|\mathbf{x}_i)^2 d\mathbf{s}' - \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \widehat{p}(\mathbf{s}'_i|\mathbf{x}_i). \quad (6.19)$$

In the experiments, we set  $N_{\text{test}} = 10201$ . Note that Figure 6.2(b) and Figure 6.5(b) are different in the scale of error values since a constant is ignored from the squared error. It is also worth mentioning that the integral in the first term of Equation (6.19) can be computed analytically for LSCDE with the Gaussian kernel model and GP.

Figure 6.2 shows that both LSCDE and GP can learn the entire profile of the true transition probability well in the Gaussian case. The squared error of the learned models in Figure 6.2(b) show that GP provides smaller squared error in this Gaussian case. On the other hand, Figure 6.5 shows that LSCDE can still successfully capture the entire profile of the true transition model well even in the bimodal case, but GP fails to capture the bimodal structure. The squared error shown in Figure 6.5(b) further illustrates that LSCDE estimates the transition model more accurately than GP in this bimodal case.

Based on the transition models, we learn policies by the M-PGPE method. More specifically, we generate  $M = 1000$  artificial trajectory data for policy gradient estimation and another  $M' = 1000$  artificial trajectory data for baseline estimation from the transition model. Then policies are updated based on these estimations. We repeat this policy update step 100 times. For evaluating the return of a learned policy, we use 100 additional test trajectory data which are not used for policy learning. Figure 6.3 and Figure 6.6 depict the average performance of learned policies over 100 runs for the Gaussian and bimodal cases, respectively. The results show that the GP-based method performs very well in the Gaussian case, but LSCDE still exhibits reasonably good performance. In the bimodal case, on the other hand, GP performs poorly and LSCDE gives much better policies than GP. Furthermore, the performance of GP

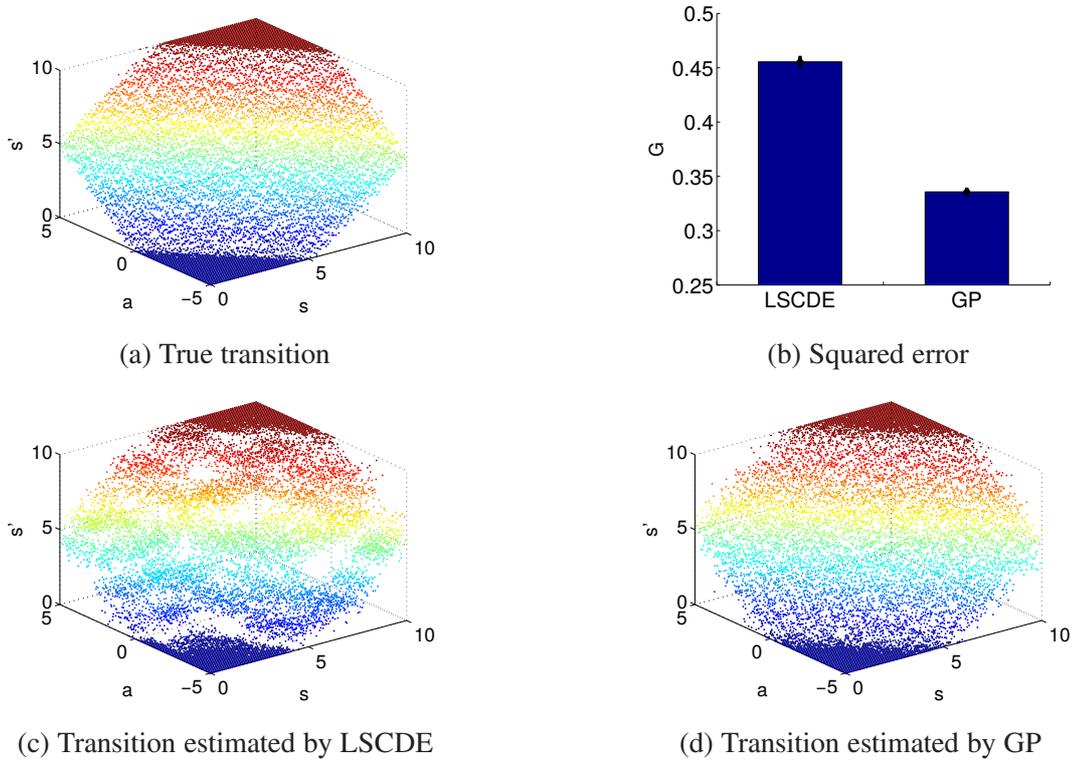


Figure 6.2: Gaussian transition probability and its estimates by LSCDE and GP. The value  $\operatorname{argmax}_{s'} p(s'|s, a)$  is plotted as a function of  $s$  and  $a$ .

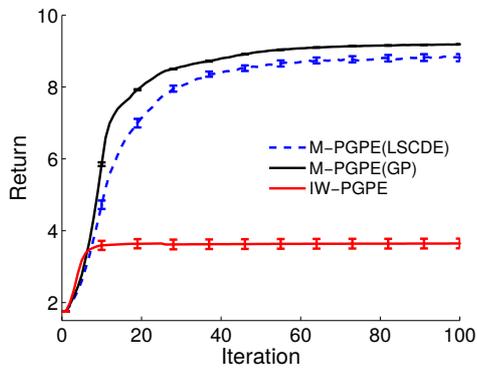


Figure 6.3: Averages and standard errors of returns of the policies over 100 runs obtained by M-PGPE with LSCDE, M-PGPE with GP, and IW-PGPE for Gaussian transition.

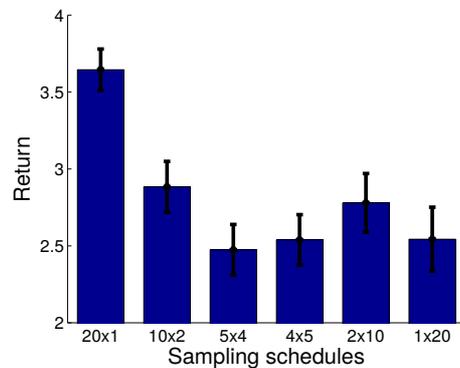


Figure 6.4: Averages and standard errors of returns obtained by IW-PGPE over 100 runs for Gaussian transition probability with different sampling schedules (e.g.,  $5 \times 4$  means gathering  $k = 5$  trajectory data 4 times).

drops after some iterations due to its inaccurate estimation of the bimodal transition model.

Since GP is a Gaussian parametric method as a conditional density estimator, it works very well if the parametric assumption is (approximately) correct (see Figure 6.2(d)). However, if the parametric model is strongly misspecified, it tends to produce a highly biased solution (see Figure 6.5(d)). On the other hand, LSCDE is non-parametric and thus it can flexibly approximate *any* conditional densities (see Figure 6.5(c)). However, its statistical efficiency tends to be lower than parametric approaches; nevertheless, Figure 6.2(c) shows that LSCDE still performs reasonably well even in the Gaussian case.

### Model-Based VS. Model-Free

Next, we compare the performance of M-PGPE with the model-free IW-PGPE method.

For the IW-PGPE method, we need to determine the schedule of collecting 20 trajectory data under the fixed budget scenario. More specifically, the “sampling schedule” indicates that a small batch of  $k$  trajectory data are gathered  $20/k$  times for different  $k$  values.  $k = 20$  means that all 20 trajectory data are gathered following the initial data-collecting policy. If  $k < 20$ , the first batch of  $k$  trajectory data are gathered following the initial data collecting policy; then the parameter of the current policy is updated and this is used as the data collecting policy to gather the next batch of  $k$  trajectory data; these parameter update and data collection steps are iterated until the sampling budget runs out.

First, we illustrate how the choice of sampling schedules affects the performance of IW-PGPE. Figure 6.4 and Figure 6.7 show expected returns averaged over 100 runs under the sampling schedule that a batch of  $k$  trajectory data are gathered  $20/k$  times for different  $k$  values. Figure 6.4 shows that the performance of IW-PGPE depends heavily on the sampling schedule, and gathering  $k = 20$  trajectory data at once is shown to be the best choice in the Gaussian case. Figure 6.7 shows that gathering  $k = 20$  trajectory data at once is also the best choice in the bimodal case.

Although the best sampling schedule is not accessible in practice, we use this optimal sampling schedule for IW-PGPE. Figure 6.3 and Figure 6.6 also include the returns obtained by IW-PGPE averaged over 100 runs as functions of the policy update iterations. These graphs show that IW-PGPE can improve the policies only in the beginning, because all trajectory data are gathered at once in the beginning. The performance of IW-PGPE may be further improved if it is possible to gather more trajectory data, but this is prohibited under the fixed budget scenario. On the other hand, return values of M-PGPE constantly increase throughout iterations, because artificial trajectory data can be kept generated without additional sampling costs. This illustrates a potential advantage of model-based reinforcement learning methods.

Note that, in our implementation of IW-PGPE, policy update is performed 100 times after observing each batch of trajectory data, because we empirically observed that this performs better than the conventional way of performing policy update only once (note that no additional sampling costs are necessary during this policy update process). On the other hand, policies are updated only once for each batch of data in M-PGPE, because we can gather as many data as we want without additional sampling costs.

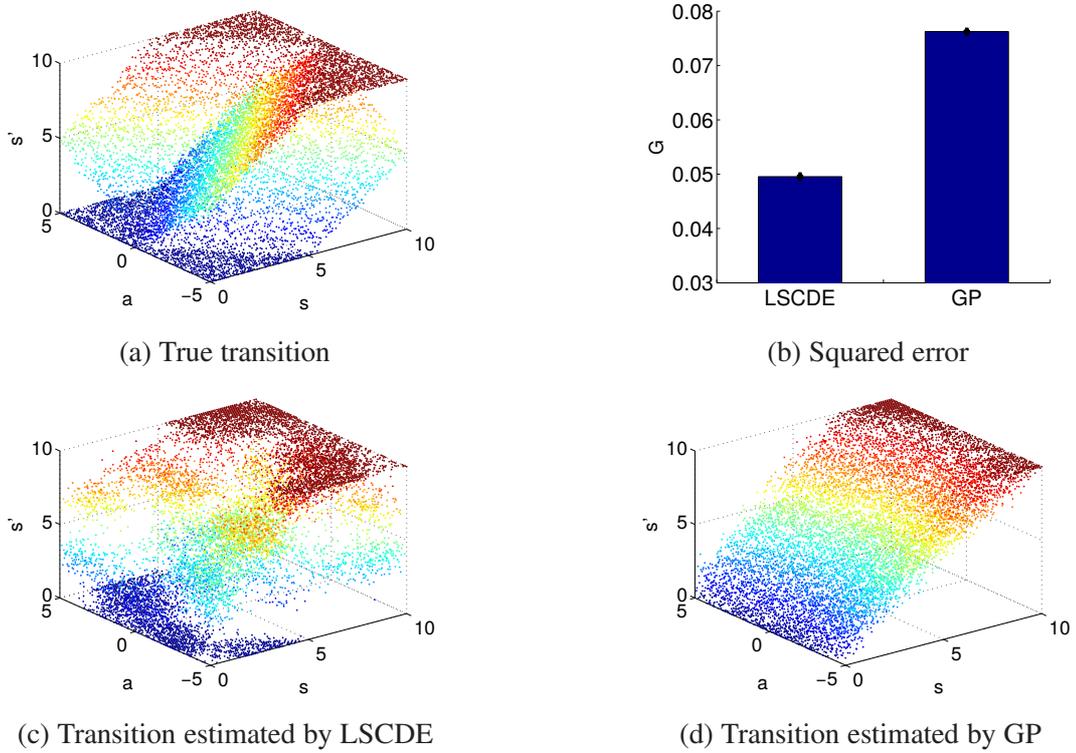


Figure 6.5: Bimodal transition probability and its estimates by LSCDE and GP. The value  $\operatorname{argmax}_{s'} p(s'|s, \mathbf{a})$  is plotted as a function of  $s$  and  $a$ .

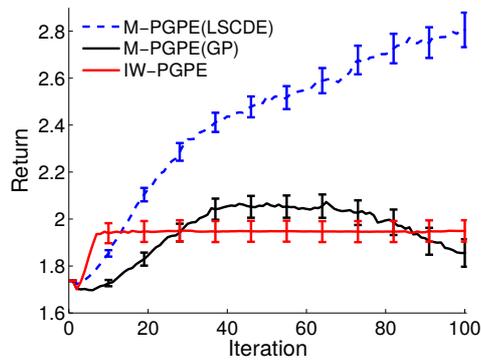


Figure 6.6: Averages and standard errors of returns of the policies over 100 runs obtained by M-PGPE with LSCDE, M-PGPE with GP, and IW-PGPE for bimodal transition.

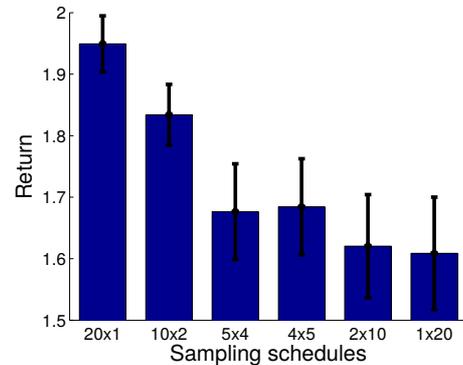
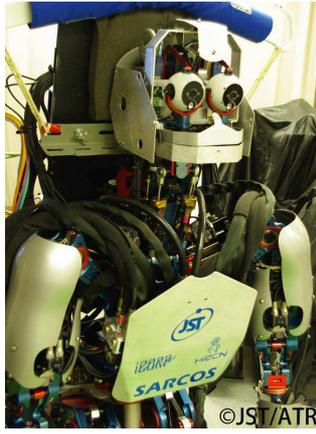
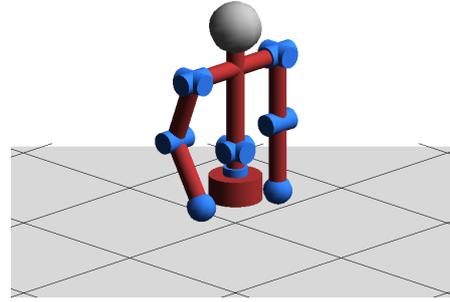


Figure 6.7: Averages and standard errors of returns obtained by IW-PGPE over 100 runs for bimodal transition with different sampling schedules (e.g.,  $5 \times 4$  means gathering  $k = 5$  trajectory data 4 times).



(a) CB-i



(b) Simulator of the upper-body

Figure 6.8: Humanoid robot CB-i and its upper-body model.

## 6.4.2 Humanoid Robot Control

Finally, we evaluate the performance of M-PGPE on a practical control problem of a simulated upper-body model of the humanoid robot *CB-i* Cheng et al. (2007) (see Figure 6.8(a)). We use its simulator for experiments (see Figure 6.8(b)). The goal of the control problem is to lead the end-effector of the right arm (right hand) to the target object.

### Setup

The simulator is based on the upper-body of the *CB-i* humanoid robot, which has 9 joints for shoulder pitch, shoulder roll, elbow pitch of the right arm, shoulder pitch, shoulder roll, elbow pitch of the left arm, waist yaw, torso roll, and torso pitch.

At each time step, the controller receives a state vector from the system and sends out an action vector. The state vector is 18-dimensional and real-valued, which corresponds to the current angle in degree and the current angular velocity for each joint. The action vector is 9-dimensional and real-valued, which corresponds to the target angle of each joint in degree.

At each time step, given a state-action pair, the physical control system calculates torques at each joint by using a proportional-derivative (PD) controller as

$$\tau_i = K_{p_i}(a_i - s_i) - K_{d_i}\dot{s}_i, \quad (6.20)$$

where  $s_i$ ,  $\dot{s}_i$ , and  $a_i$  denote the current angle, current angular velocity, and target angle of the  $i$ th joint, respectively.  $K_{p_i}$  and  $K_{d_i}$  denote the position and velocity gains for the  $i$ th joint, respectively. In the experiments, we set  $K_{p_i} = 2000$  and  $K_{d_i} = 100$  for all joints except the elbow pitch joints for which we set  $K_{p_i} = 200$  and  $K_{d_i} = 10$ .

We simulate a noisy control system by perturbing action vectors with independent bimodal Gaussian noise. More specifically, for each action element, we add Gaussian noise with mean 0 and standard deviation 3 with probability 0.6, and Gaussian noise with mean  $-5$  and standard deviation 3 with probability 0.4.

The initial posture of the robot is fixed to be standing up straight with arms down. The target object is located in front-above of the right hand, which is reachable by

using the controllable joints. The reward function at each time step is defined as

$$r_t = \exp(-10d_t) - 0.000005 \min\{c_t, 1000000\}, \quad (6.21)$$

where  $d_t$  is the distance between the right hand and the target object at time step  $t$ .  $c_t$  is the sum of control costs for each joint, i.e.,  $c_t = \sum_{i=1}^J \tau_i^2$ , where  $J$  is the total number of used joints. The coefficient 0.000005 is multiplied to keep the values of the two terms in the same order of magnitude. Note that, for this multi-dimensional action problem, the policy for each joint is learned independently. We set the trajectory length at  $T = 100$ , the discount factor at  $\gamma = 0.9$ , and the learning rate at  $\beta = 0.1/\|\widehat{\nabla}_{\rho}\mathcal{J}(\rho)\|$ .

The initial mean parameter of the Gaussian prior is randomly chosen from a standard normal distribution, i.e.,  $\eta_0 \sim N(0, 1)$ . The initial deviation parameter of the Gaussian prior is set at 1, i.e.,  $\tau_0 = 1$ . Note that we have no prior knowledge of the choice of the initial parameters. Choosing a good initial parameter is especially important for IW-PGPE, since poor initialization may lead to a significant difference between the data collection distribution and the target distribution, and thus IW-PGPE will produce gradient estimators with large variance. On the other hand, M-PGPE does not suffer this problem. In the implementation of LSCDE, the Gaussian width  $\sigma$  and the regularization parameter  $\lambda$  are chosen by cross-validation from the following candidate values:

$$\sigma, \lambda \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}. \quad (6.22)$$

## Experiment with 2 Joints

First, we only use 2 joints among the 9 joints, i.e., we allow only the right shoulder pitch and right elbow pitch to be controlled, while the other joints are fixed to the initial posture. This means that 4-dimensional states and 2-dimensional actions are considered here. However, since the simulator calculates 18-dimensional humanoid dynamics, the other joints can slightly move due to movements of the controlled joints. Therefore, to some extent, the learning algorithms are evaluated in the partially observable environment. Under this setup, we compare the performance of M-PGPE(LSCDE), M-PGPE(GP), and IW-PGPE.

We suppose that the budget for data collection is limited to  $N = 50$  trajectory data. For the M-PGPE methods, all trajectory data are collected at first using the uniformly random initial states and policy. More specifically, the initial state is chosen from the uniform distribution over  $\mathcal{S}$ . At each time step, the  $i$ th element of the action vector,  $a_i$ , is chosen from the uniform distribution on  $[s_i - 5, s_i + 5]$ . In total, we have  $P = 5000$  transition data for model estimation. Then, we generate  $M' = 1000$  artificial trajectory data for policy gradient estimation and another  $M = 1000$  artificial trajectory data for baseline estimation from the learned transition model and update the control policy based on these artificial trajectory data. For the IW-PGPE method, we performed preliminary experiments to determine the optimal sampling schedule (Figure 6.9), showing that collecting  $k = 5$  trajectory data  $50/k$  times yields the highest average return. We use this sampling schedule for performance comparison with the M-PGPE methods. Moreover, the policy update is performed 100 times after observing each batch of data for the IW-PGPE method, which we confirmed to work better than just updating the policy only once. Thus, policy updates are performed 1000 times in total under this sampling schedule. On the other hand, in M-PGPE,

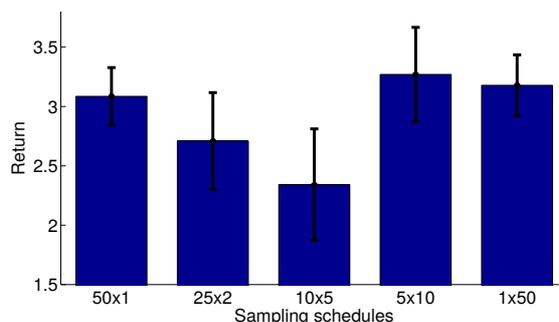


Figure 6.9: Averages and standard errors of returns obtained by IW-PGPE over 10 runs for the 2-joint humanoid robot simulator for different sampling schedules (e.g.,  $5 \times 10$  means gathering  $k = 5$  trajectory data 10 times).

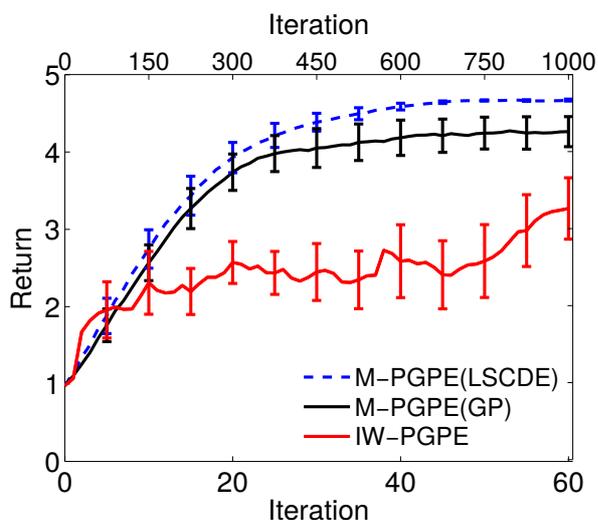


Figure 6.10: Averages and standard errors of obtained returns over 10 runs for the 2-joint humanoid robot simulator. All methods use 50 trajectory data for policy learning. In M-PGPE(LSCDE) and M-PGPE(GP), all 50 trajectory data are gathered in the beginning and the environment model is learned; then 2000 artificial trajectory data are generated in each update iteration. In IW-PGPE, a batch of 5 trajectory data are gathered for 10 iterations, which was shown to be the best sampling scheduling (see Figure 6.9). Note that policy update is performed 100 times after observing each batch of trajectory data, which we confirmed to perform well. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.

policies are updated only once for each batch of data, since no additional sampling costs are necessary to gather data from the learned model.

The returns obtained by each method averaged over 10 runs are plotted in Figure 6.10, showing that M-PGPE(LSCDE) tends to outperform both M-PGPE(GP) and IW-PGPE. In Figure 6.10, the IW-PGPE curve is adjusted to have the same horizontal scale as M-PGPE. The top horizontal axis indicates the policy update iterations for IW-PGPE, while the bottom horizontal axis indicates the policy update iterations for M-PGPE.

Figure 6.11 shows the minimum distance and discounted control cost averaged

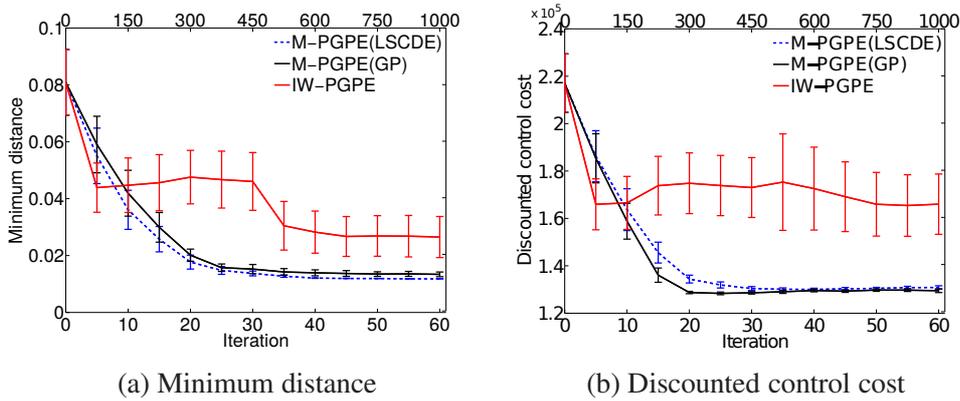


Figure 6.11: Averages and standard errors of the minimum distance and discounted control cost over 10 runs for the 2-joint humanoid robot simulator. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.

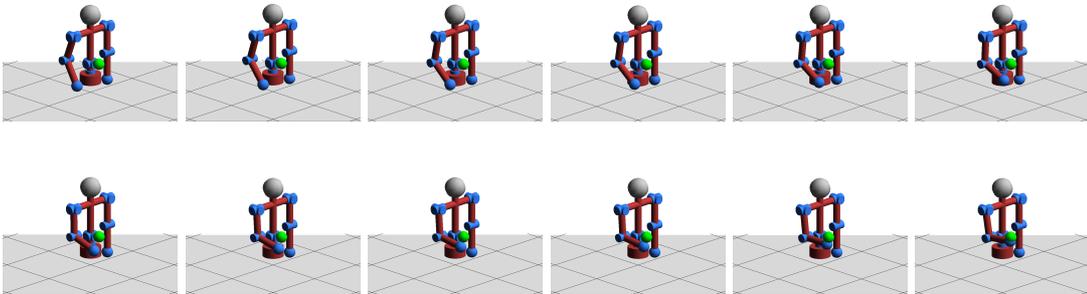


Figure 6.12: Example of arm reaching with 2 joints using a policy obtained by M-PGPE at the 60th iteration (some intermediate steps are not shown here).

over 10 runs. The minimum distance plot in Figure 6.11(a) shows that the learned policies from M-PGPE(LSCDE) and M-PGPE(GP) get closer to the target than that of IW-PGPE, and M-PGPE(LSCDE) gets slightly closer to the target than M-PGPE(GP). The discounted control cost in Figure 6.11(b) shows that M-PGPE(LSCDE) and M-PGPE(GP) use smaller discounted torques than IW-PGPE to reach the target object.

Figure 6.12 illustrates an example of the reaching motion with 2 joints obtained by M-PGPE(LSCDE) at the 60th iteration policy. This shows that the learned policy successfully leads the right hand to the target object within only 13 steps in this noisy control system.

### Experiment with 9 Joints

Finally, we evaluate the performance of the proposed method on the reaching task with 9 joints, i.e., all joints are allowed to move. In this experiment, we compare learning performance between M-PGPE(LSCDE) and IW-PGPE. We do not include M-PGPE(GP) since it is outperformed by M-PGPE(LSCDE) on the previous 2-joint experiments, and furthermore the GP-based method requires an enormous amount of computation time.

The experimental setup is essentially the same as the 2-joint experiments, but we have a budget for gathering  $N = 1000$  trajectory data for this complex and high-

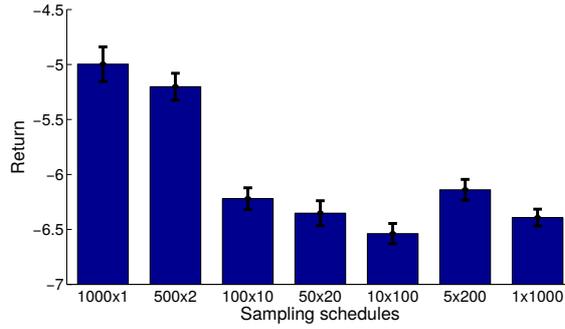


Figure 6.13: Averages and standard errors of returns obtained by IW-PGPE over 30 runs for the 9-joint humanoid robot simulator for different sampling schedules (e.g.,  $100 \times 10$  means gathering  $k = 100$  trajectory data 10 times).

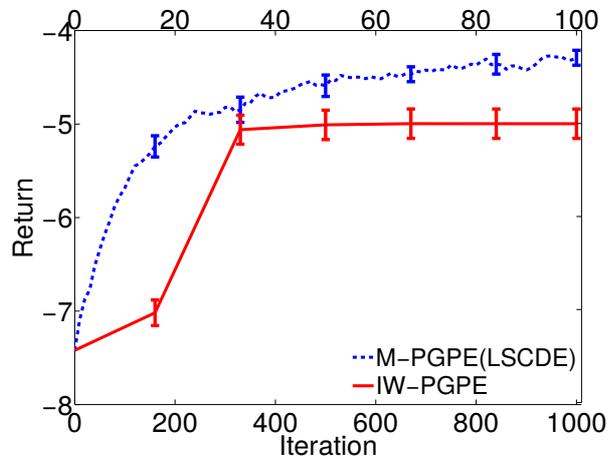


Figure 6.14: Averages and standard errors of obtained returns over 30 runs for the humanoid robot simulator with 9 joints. Both methods use 1000 trajectory data for policy learning. In M-PGPE(LSCDE), all 1000 trajectory data are gathered in the beginning and the environment model is learned; then 2000 artificial trajectory data are generated in each update iteration. In IW-PGPE, a batch of 1000 trajectory data are gathered at once, which was shown to be the best scheduling (see Figure 6.13). Note that policy update is performed 100 times after observing each batch of trajectory data. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.

dimensional task. The position of the target object is moved to far left, which is not reachable only by using 2 joints. Thus, the robot is required to move other joints to reach the object with the right hand. Among  $P = 100000$  transition data points, we randomly choose 5000 transition data points for Gaussian centers for M-PGPE(LSCDE). The sampling schedule for IW-PGPE was set at gathering 1000 trajectory data at once, which is the best sampling schedule according to Figure 6.13. The returns obtained by M-PGPE(LSCDE) and IW-PGPE averaged over 30 runs are plotted in Figure 6.14, where the IW-PGPE curve is elongated to have the same horizontal scale as M-PGPE; the top horizontal axis indicates the policy update iterations for IW-PGPE, while the bottom horizontal axis indicates the policy update iterations for M-PGPE. The graph shows that M-PGPE(LSCDE) tends to outperform IW-PGPE

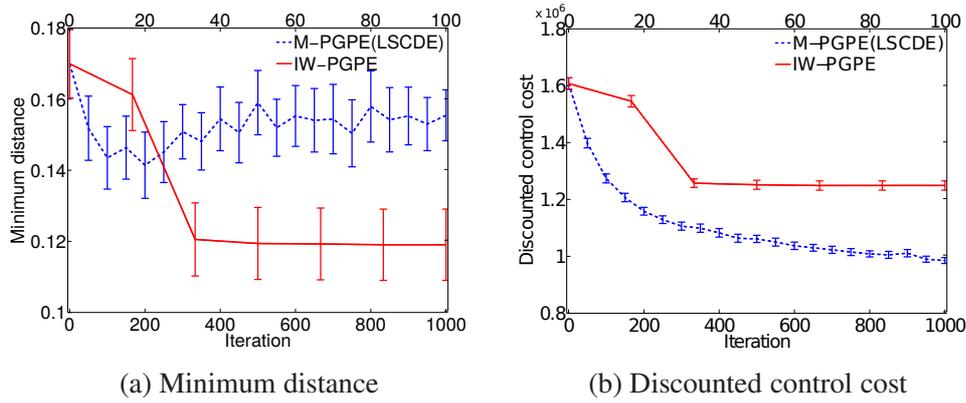


Figure 6.15: Averages and standard errors of the minimum distance and discounted control cost over 30 runs for the 9-joint humanoid robot simulator. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.

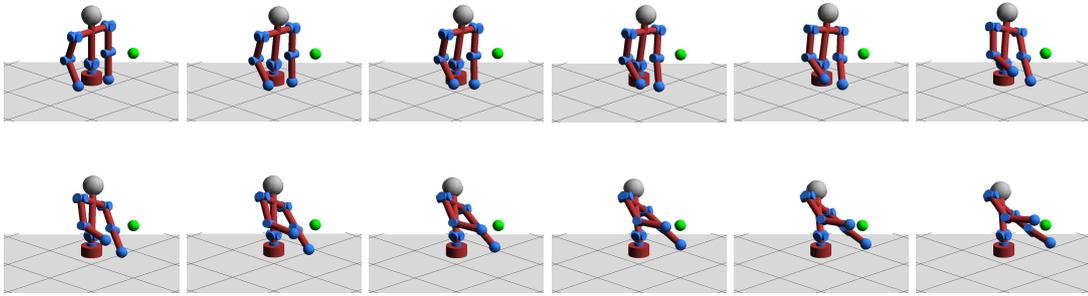


Figure 6.16: Example of arm reaching with 9 joints using a policy obtained by M-PGPE(LSCDE) at the 1000th iteration (some intermediate steps are not shown here).

in this challenging robot control task.

Figure 6.15 shows the minimum distance and discounted control cost averaged over 10 runs. The minimum distance plot in Figure 6.15(a) shows that the learned policies from IW-PGPE get closer to the target object than that of M-PGPE(LSCDE), while the discounted control cost plot in Figure 6.15(b) shows that M-PGPE(LSCDE) uses much smaller discounted torques than IW-PGPE. Note that, as shown in Figure 6.14, the return (the weighted discounted combination of the distance and control cost) for M-PGPE(LSCDE) is better than that for IW-PGPE, and either strongly optimizing the distance or control cost depends on the choice of the balancing constant in the definition of the reward function (see Equation (6.21)).

Figure 6.16 shows a typical example of the reaching motion with 9 joints obtained by M-PGPE(LSCDE) at the 1000th iteration. The images show that the policy learned by M-PGPE(LSCDE) leads the right hand to the distant object successfully within 14 steps.

## 6.5 Conclusion

In this chapter, we extended the model-free PGPE method to a model-based scenario, and proposed to combine it with a conditional density estimation method called least-

squares conditional density estimation (LSCDE). Under the fixed sampling budget, appropriately designing a sampling schedule is critical for the model-free IW-PGPE method, while this is not a problem for the proposed model-based PGPE (M-PGPE) method. Through experiments, we confirmed that an existing GP-based model estimation is not as flexible as the LSCDE-based method when the transition model is not Gaussian, and the proposed M-PGPE based on LSCDE was overall demonstrated to be promising.

As we have demonstrated, the GP-based model estimation does not perform well when the transition probability possess multimodality characteristic. There are many practical situations which involve multimodality. A well-known example is the mobile robot global localization, where the dynamics of the global localization is usually multimodal due to the symmetry of the environment and ambiguity of detected features Liu et al. (2007). For example, the robot is placed somewhere in the environment without the knowledge of its global location. After state transition, the robot finds out that the current position is next to a door according to the information from observations and sensors. However, if there are 2 doors in the environment, the distribution of the sensed position of the robot cannot be uniquely determined in general. Multimodality due to such ambiguity is conceivable in the many real-world problems because of imperfect state information. On the other hand, if the robot is highly certain about its position, it would follow a unimodal distribution centered around the true position of the robot (Fox et al., 1999).

Although LSCDE perform well as a transition model estimation method, it requires a tremendous amount of data for high-dimensional problems (e.g., 100000 transition data for 9 joints humanoid robot in Section 6.4.2). In the next chapter, we propose our contribution which improves the performance of M-PGPE in high-dimensional problems.

## Chapter 7

# Dimension Reduction for Model-based Policy Gradient with Parameter-based Exploration

This chapter presents our fourth contribution on utilizing single-step dimension reduction for our model-based policy gradient with parameter-based exploration method. Firstly, we briefly reintroduce our previous method and explain how can we use dimension reduction to improve its performance. Then, we briefly discuss existing approach to perform dimension reduction for transition model estimation. Next, we present our contribution which combines our least-squares conditional entropy in Chapter 4 and our model-based policy gradient with parameter-based exploration in Chapter 6. Lastly, we present our experimental results and conclude the chapter.

### 7.1 Introduction

Reinforcement learning aims to find an optimal policy using data collected by an agent. In model-free reinforcement learning, the collected data is used to improve the policy directly. In contrast, model-based reinforcement learning firstly learns a transition model from collected data and then improve the policy based on the learned transition model. It has been experimentally shown that the model-based approach is a highly data efficient approach and is suitable for domains where collecting data is expensive (Sutton, 1990; Rasmussen and Kuss, 2003; Wang and Dietterich, 2003; Deisenroth and Rasmussen, 2011; Kupcsik et al., 2013).

However, learning an accurate transition model from high-dimensional data often require a large amount of data due to the curse of dimensionality. This large amount of data requirement may make model-based methods not as data efficient as they should be. A common approach to mitigate the curse of dimensionality is linear dimension reduction. However, naively applying linear dimension reduction to the transition model estimation problem involves a multi-step procedure where the former dimension reduction step is perform independently from the latter transition model estimation step. This is not preferred since an estimation error from the dimension reduction step can be magnified by the latter transition model estimation step.

To overcome the issue of the multi-step approach, we propose to use our least-squares conditional entropy (LSCE) method which is presented in Chapter 4 to learn the transition model. LSCE is a single-step dimension reduction method which simultaneously performs both supervised linear dimension reduction and conditional density estimation. Through experiments in Chapter 4 we have shown that LSCE outperforms multi-step combinations of dimension reduction methods and conditional

density estimation methods. Moreover, we also evaluated LSCE in transition data from a simulated humanoid robot and computer art problems, and showed that LSCE also outperforms other methods on these data as well. In this chapter, we further show that LSCE is indeed suitable as a transition model estimation method for model-based reinforcement learning.

This chapter is organized as follows. In Section 7.2, we briefly discuss existing dimension reduction approaches in reinforcement learning. Then, in Section 7.3 we present our proposed method which combines M-PGPE with LSCE. Section 7.4 presents the experimental evaluations of the proposed method and Section 7.5 concludes this chapter.

## 7.2 Existing Approach to Dimension Reduction in Reinforcement Learning

Dimension reduction is a common technique to mitigate the curse of dimensionality in reinforcement learning. In this section, we discuss existing dimension reduction approaches in reinforcement learning.

### 7.2.1 Feature Selection in Factored Markov Decision Process

The *factored MDP* (Boutilier et al., 1995) is a variant of MDPs formulated for high-dimensional problems. The important characteristic of the factored MDP is its transition probability which is represented by

$$p(\mathbf{s}' | \mathbf{s}, \mathbf{a}) = \prod_{k=1}^{d_s} p(s'_k | f_k(\mathbf{s}), \mathbf{a}), \quad (7.1)$$

where  $f_k(\mathbf{s})$  denotes a subset of feature of  $\mathbf{s}$  corresponds to the  $k$ -th state feature. More specifically, the transition probability of the  $k$ -th state feature does not depend on the transition probability of the other features, and only depends on the  $k$ -th subset of state features and an action. This transition probability is commonly represented by dynamics Bayesian networks (DBNs) (Dean and Kanazawa, 1989).

Based on the factored MDP, Kroon and Whiteson (2009) proposed a feature selection method where a state feature is selected if it either directly or indirectly influences rewards. This method is an online method where feature selection and transition model estimation are performed while the agent is learning the policy. They showed that this feature selection approach improves the performance of *factored-RMAX* (Guestrin et al., 2002). However, learning the DBNs is time consuming. To cope with this problem, Nguyen et al. (2013) proposed an extension of the factored MDP called the *situation calculus MDP* and learned the transition model by logistic regression with group lasso regularization. This approach was shown to be computationally more efficient than learning the DBNs.

These feature selection approaches were shown to perform well on high-dimensional tasks. However, they strictly rely on the factored MDP formulation. Thus, these methods would not be applicable when the problems cannot be formulated as a factored MDP. In contrast, the proposed method does not assume specific structures of the transition probability and is more widely applicable.

## 7.2.2 Supervised Dimension Reduction in Reinforcement Learning

Supervised linear dimension reduction was used for reinforcement learning previously. The KDR method (Fukumizu et al., 2004) was used by Morimoto et al. (2008) to find a low-dimensional state variable which preserves reward information. More precisely, KDR finds a matrix  $\mathbf{W}$  such that  $\mathbf{z} = \mathbf{W}\mathbf{s}$  satisfies the conditional independence

$$((\mathbf{s}, \mathbf{a}) \perp\!\!\!\perp \tilde{\mathbf{s}}') \mid (\mathbf{z}, \mathbf{a}), \quad (7.2)$$

where  $\tilde{\mathbf{s}}'$  denotes state features which influence rewards, i.e.,  $r_t = r(\tilde{\mathbf{s}}_t, \mathbf{a}_t, \tilde{\mathbf{s}}_{t+1})$ . Subsequently,  $\mathbf{z}$  is used instead of  $\mathbf{s}$  in a model-free reinforcement learning method. This approach was shown to find a low-dimensional representation of states which is relevant to a robot walking task. However, the weakness of this approach is that it could not detect implicit relations between state features and rewards. For instance, assume that  $s^{(k)}$  influences the reward, i.e.,  $r_t = r(s_t^{(k)}, \mathbf{a}_t, s_{t+1}^{(k)})$ . Then, consider a case where  $s^{(i)}$  influences  $s^{(j)}$  at the next time step, and  $s^{(j)}$  influences  $s^{(k)}$  at the next-next time step. Thus,  $s^{(i)}$ ,  $s^{(j)}$ , and  $s^{(k)}$  are relevant features. However, the KDR-based approach only chooses  $s^{(k)}$  and  $s^{(j)}$  as relevant features.

To cope with the above problem, Hachiya and Sugiyama (2010) proposed a feature selection procedure which considers the return instead of the reward. Their goal is to find a subset of state features  $\mathbf{z}$  such that it satisfies the conditional independence

$$(R(\mathbf{s}) \perp\!\!\!\perp \mathbf{s}) \mid \mathbf{z}, \quad (7.3)$$

where  $R(\mathbf{s}_t) = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$  is the return along a trajectory from time step  $t$  onward. In this way, the implicit relation of states and rewards is captured through the return. The subset of state features  $\mathbf{z}$  is obtained by maximizing a squared-loss variant of conditional mutual information (Suzuki et al., 2009).

The two methods we discussed above have demonstrated the usefulness of dimension reduction in reinforcement learning where their goal is to find a low-dimensional variable which compactly represents the state. However, this is quite different from our usage of dimension reduction in this chapter which focuses on the transition model estimation problem.

## 7.3 Model-based PGPE with Single-step Dimension Reduction

As we have shown in our previous contributions, M-PGPE with LSCDE is a promising model-based reinforcement learning method and LSCE is the most promising supervised linear dimension method for transition model estimation. In this section, we propose a model-based reinforcement learning method based on this combination. In addition, we also propose an extension to *imitation learning* which is a suitable framework for model-based reinforcement learning.

### 7.3.1 Learning Framework

Figure 7.1 shows a schematic diagram of the proposed model-based reinforcement learning method. In Step 1, the agent collects a trajectory dataset  $\{\tau_n\}_{n=1}^N$  by using either an uninformative random policy or an informative policy. Then the dataset is used in Step 2 to estimate a transition model by LSCE (see Chapter 4). Note that

when an informative policy is used to collect the data, we can use imitation learning to initialize a policy in an informative way. The imitation learning step for the proposed method will be explained in Section 7.3.2.

In Step 3, the agent generates a dataset  $\{(\boldsymbol{\theta}_m, \hat{\boldsymbol{\tau}}_m)\}_{m=1}^M$  by the following procedure. It first draws a parameter  $\boldsymbol{\theta}_m$  from the current policy parameter distribution  $p(\boldsymbol{\theta}|\boldsymbol{\rho})$ . This parameter is used to generate an artificial trajectory:

$$\hat{\boldsymbol{\tau}}_m = [\hat{\mathbf{s}}_1, \mathbf{a}_1, \hat{\mathbf{s}}_2, \dots, \mathbf{a}_T, \hat{\mathbf{s}}_{T+1}], \quad (7.4)$$

where  $\hat{\mathbf{s}}_1 \sim p_1(\mathbf{s})$  and  $\mathbf{a}_t = \boldsymbol{\theta}_m^\top \boldsymbol{\phi}(\hat{\mathbf{s}}_t)$ . The next state  $\hat{\mathbf{s}}_{t+1}$  is obtained from  $\hat{p}(\mathbf{s}'|\mathbf{W}\mathbf{x})$  and  $\mathbf{x}_t = (\hat{\mathbf{s}}_t^\top, \mathbf{a}_t^\top)^\top$  by numerically approximating the mean:

$$\hat{\mathbf{s}}_{t+1} = \frac{1}{K} \sum_{k=1}^K \hat{\mathbf{s}}'_k, \quad (7.5)$$

where  $\hat{\mathbf{s}}'_k \sim \hat{p}(\mathbf{s}'|\mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{x}_t)$ . We set  $K = 100$  in our experiments<sup>1</sup>.

In Step 4, the dataset  $\{(\boldsymbol{\theta}_m, \hat{\boldsymbol{\tau}}_m)\}_{m=1}^M$  is used to update the parameter  $\boldsymbol{\rho}$  by the PGPE method. More specifically, we split the dataset into two disjoint subset  $\{(\boldsymbol{\theta}_m, \hat{\boldsymbol{\tau}}_m)\}_{m=1}^{M'}$  and  $\{(\boldsymbol{\theta}_m, \hat{\boldsymbol{\tau}}_m)\}_{m=1}^{M''}$  and then estimate the gradient and the optimal baseline by

$$\begin{aligned} \hat{\nabla}_{\boldsymbol{\rho}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\rho}) &= \frac{1}{M'} \sum_{i=1}^{M'} (R(\hat{\boldsymbol{\tau}}_i) - b^*) \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_i; \boldsymbol{\rho}), \\ b^* &= \frac{\sum_{i=1}^{M''} R(\hat{\boldsymbol{\tau}}_i) \|\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_i; \boldsymbol{\rho})\|_2^2}{\sum_{i=1}^{M''} \|\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_i; \boldsymbol{\rho})\|_2^2}, \end{aligned} \quad (7.6)$$

We refer to Chapter 5 for details of PGPE. Then parameter  $\boldsymbol{\rho}$  is updated by

$$\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \beta \hat{\nabla}_{\boldsymbol{\rho}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\rho}), \quad (7.7)$$

where  $\beta > 0$  is the step size.

Note that the number of artificial trajectory  $M$  can be much larger than the number of collected trajectory  $N$ . In our experiments, we generate  $M = 1000$  artificial trajectories. The first half is used to estimate the optimal baseline, and the second half is used to estimate the gradient of the expected return, i.e.,  $M' = M'' = 500$ .

For policy gradient methods such as PGPE, the choice of the initial parameter  $\boldsymbol{\rho}$  is very important. Improving a poorly chosen initial parameter hardly yields a good policy even for a highly accurate transition model. Next, we assume that an informative policy is used for collecting the trajectory data. This assumption allows us to use imitation learning to informatively initialize the parameter.

### 7.3.2 Imitation Learning

Imitation learning is a learning framework similar to supervised learning (Schaal, 1999). The goal of imitation learning is to learn a policy which reproduces demonstrated trajectories. Imitation learning has been widely used in robot controls where

<sup>1</sup>We set  $K = 1$  in our previous M-PGPE method that is presented in Chapter 6. However, we observed that for unimodal transition probability using the mean as a predicted state gives better performance overall than using an individual sampled data point as a predicted state.

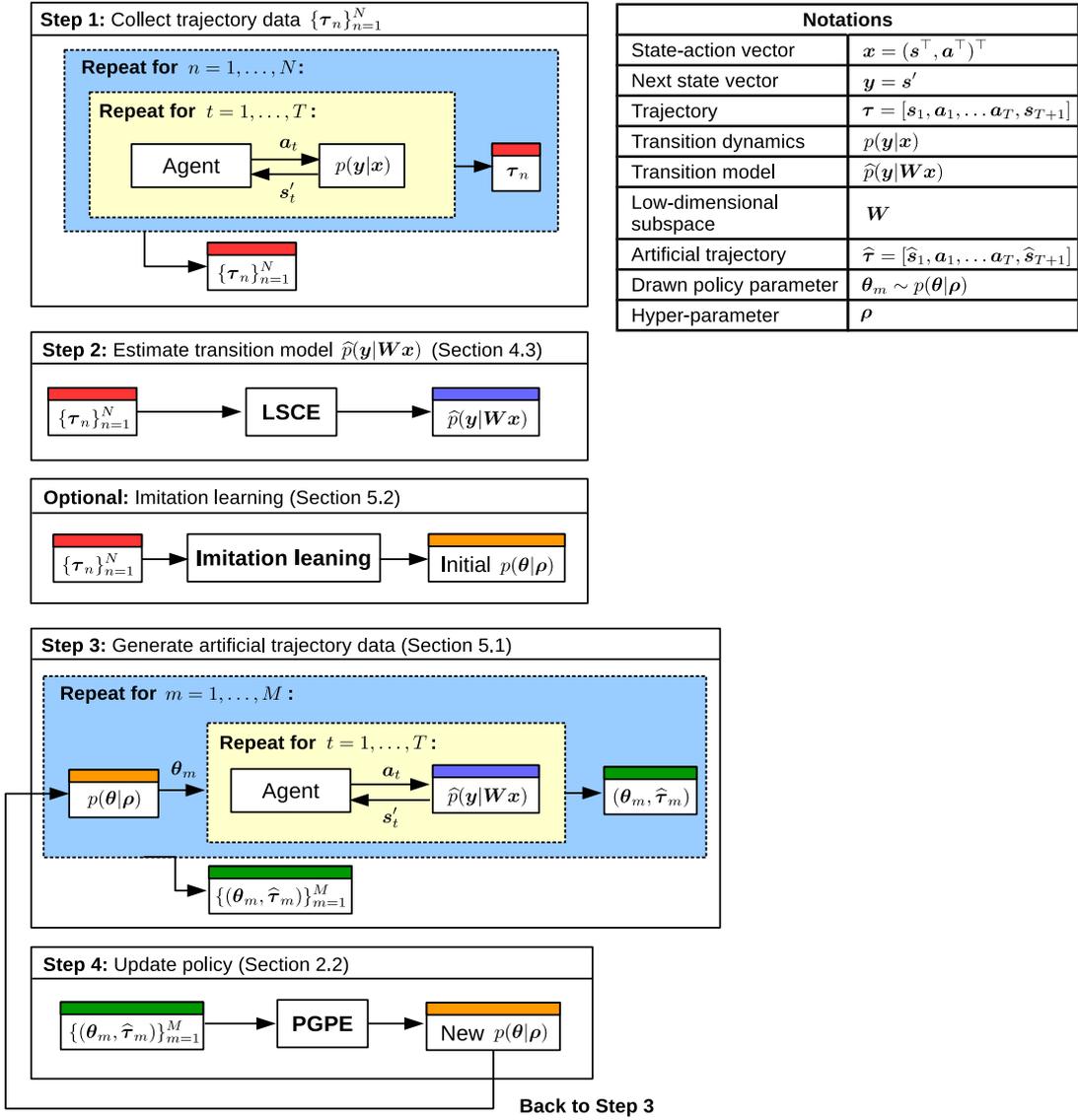


Figure 7.1: A schematic diagram of the proposed model-based reinforcement learning method. The agent firstly collects  $N$  trajectories (Step 1) and uses the collected trajectories to estimate a transition model (Step 2). Then it uses the transition model to generate  $M$  artificial trajectories (Step 3), and use these artificial trajectories in PGPE (Step 4).

a human expert demonstrates task-solving trajectories to the robot by, e.g., recording joint movements of a human expert (Ijspeert et al., 2002b). A problem of imitation learning is that the demonstrated trajectories are not always optimal, and thus the policy mimicking the demonstrated trajectories is not necessary optimal. In such a case, it is common to further improve the mimicked policy by reinforcement learning. From this viewpoint, imitation learning can be regarded as an approach to learn a good initial policy for reinforcement learning.

We formulate the imitation learning problem for PGPE as follows. We assume that  $N$  demonstrated trajectories are given as  $\{\tau_n\}_{n=1}^N$ , where

$$\tau_n = [\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{a}_T, \mathbf{s}_{T+1}]. \quad (7.8)$$

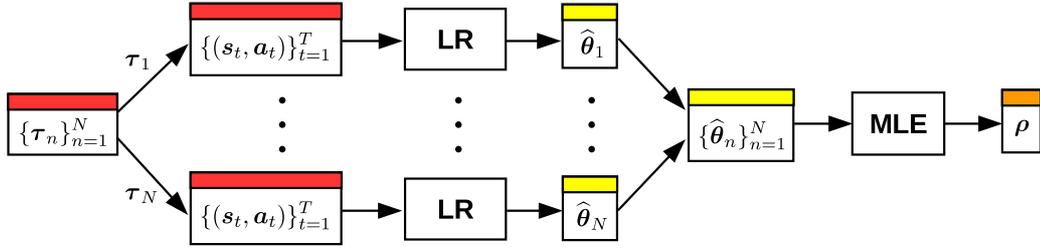


Figure 7.2: A schematic diagram of the proposed imitation learning step. Linear regression (LR) is firstly used to obtain a set of estimated policy parameters  $\{\hat{\theta}_n\}_{n=1}^N$ , then maximum likelihood estimation (MLE) is used to obtain the initial parameter  $\rho$ .

We assume that  $\tau_n$  are obtained by the deterministic policy  $\mathbf{a} = \boldsymbol{\theta}_n^\top \boldsymbol{\phi}(s)$  and thus  $\mathbf{a}_t = \boldsymbol{\theta}_n^\top \boldsymbol{\phi}(s_t)$ . We propose to estimate the policy parameter  $\boldsymbol{\theta}_n$  by minimizing the mean squared error:

$$\min_{\boldsymbol{\theta}_n} \left[ \frac{1}{T} \sum_{t=1}^T \|\mathbf{a}_t - \boldsymbol{\theta}_n^\top \boldsymbol{\phi}(s_t)\|^2 \right], \quad (7.9)$$

where the dataset  $\{(s_t, \mathbf{a}_t)\}_{t=1}^T$  is obtained directly from  $\tau_n$ . This minimization problem corresponds to linear least-squares regression with feature mapping  $\boldsymbol{\phi}$ . The solution to this linear regression problem is

$$\hat{\boldsymbol{\theta}}_n = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{A}, \quad (7.10)$$

where the matrices  $\boldsymbol{\Phi} \in \mathbb{R}^{d \times T}$  and  $\mathbf{A} \in \mathbb{R}^{T \times d_a}$  are given by

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}(s_1), \dots, \boldsymbol{\phi}(s_T)], \quad (7.11)$$

$$\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_T]^\top. \quad (7.12)$$

PGPE assumes that  $\boldsymbol{\theta}_n \sim p(\boldsymbol{\theta}|\boldsymbol{\rho})$ . Therefore, given a set of estimated policy parameters  $\{\hat{\boldsymbol{\theta}}_n\}_{n=1}^N$ , the initial parameter  $\boldsymbol{\rho}$  may be estimated by maximum likelihood estimation. For the independent Gaussian model in Equation (5.68), i.e.,  $p(\theta_{i,j}|\rho_{i,j}) = \mathcal{N}(\theta_{i,j}|\eta_{i,j}, \tau_{i,j}^2)$ , maximum log-likelihood estimation yields the empirical mean and standard deviation:

$$\hat{\eta}_{i,j} = \frac{1}{N} \sum_{n=1}^N (\theta_n)_{i,j}, \quad (7.13)$$

$$\hat{\tau}_{i,j} = \sqrt{\frac{1}{N} \sum_{n=1}^N ((\theta_n)_{i,j} - \hat{\eta}_{i,j})^2}, \quad (7.14)$$

where  $(\theta_n)_{i,j}$  denotes the  $(i, j)$ -th entry of  $\boldsymbol{\theta}_n$ . Figure 7.2 shows a schematic diagram of the proposed imitation learning step.

The above formulation implies that the agent can fully observe states and actions of the demonstrator, and that state and action descriptions of the agent and the demonstrator are identical. This allows us to formulate imitation learning as a regression problem. However, this assumption may not always be satisfied in practice. For such a case, more sophisticated techniques are needed to transfer information from the

demonstrator to the agent (Argall et al., 2009). The choice of these techniques often depends on the tasks and is beyond the scope of this dissertation. For simplicity, we assume the above assumptions and use linear regression to learn a policy from demonstrated trajectories.

In addition to the initial policy, imitation learning is also beneficial to the model-based approach since it provides task-solving trajectories for transition model estimation. This allows transition model estimation to focus on learning task-relevant parts of the transition probability. We use the imitation learning framework in the humanoid robot experiment in Section 7.4.3.

## 7.4 Experiment

In this section, we demonstrate the usefulness of the proposed method through experiments. We perform experiments on three control tasks: a continuous chainwalk, cartpole balancing, and a real humanoid robot.

### 7.4.1 Continuous Chainwalk

First, we perform experiments on a continuous chainwalk.

#### Setup

The state space, action space, and reward function of a chainwalk with  $d_s$  state dimensions are considered:

$$\begin{aligned} \mathcal{S} &= [0, 1]^{d_s}, \\ \mathcal{A} &= [-0.4, 0.4], \\ r(\mathbf{s}, \mathbf{a}, \mathbf{s}') &= \exp\left(-\frac{(s'^{(1)} - 0.5)^2}{2(0.03)^2}\right). \end{aligned}$$

Each dimension  $i \in \{1, \dots, d_s\}$  of the initial state  $\mathbf{s}_1$  is drawn independently as

$$s_1^{(i)} \sim \begin{cases} \frac{1}{2}\text{U}(0, 0.1) + \frac{1}{2}\text{U}(0.9, 1) & \text{if } i = 1, \\ \text{U}(0, 1) & \text{otherwise.} \end{cases} \quad (7.15)$$

The transition dynamics is given independently for each dimension as

$$\begin{aligned} s_{t+1}^{(i)} &= s_t^{(i)} + a_t - g \times \text{sign}(a_t) && \text{if } i = 1, \\ s_{t+1}^{(i)} &\sim \text{U}(0, 1) && \text{otherwise,} \end{aligned} \quad (7.16)$$

where  $g \sim \Gamma(0.5, 0.03)$ . The trajectory length is  $T = 10$  and the discount factor is  $\gamma = 0.9$ . We use the linear deterministic policy model with Gaussian basis functions<sup>2</sup>:

$$\begin{aligned} a &= \boldsymbol{\theta}^\top \boldsymbol{\phi}(s^{(1)}) \\ &= \sum_{i=1}^6 \theta_i \exp\left(-\frac{(s^{(1)} - c_i)^2}{2(0.1)^2}\right), \end{aligned} \quad (7.17)$$

---

<sup>2</sup>To make the experiment simple, we use this policy model which depends only on the first dimension of the state vector and ignores the other dimensions.

where  $(c_1, \dots, c_6) = (0, 0.2, 0.4, 0.6, 0.8, 1)$ . We use the independent Gaussian distribution for PGPE and initialize the mean and the standard deviation by  $\eta_i = 0$  and  $\tau_i \sim \mathcal{N}(0, 1)$ , respectively. The learning rate for policy update is  $\beta = \frac{0.05}{\|\widehat{\nabla}_{\rho}^{(\text{baseline})} \mathcal{J}(\rho)\|}$ .

The transition probability only depends on the action and the first dimension of the state vector. Thus, the irrelevant state dimensions ( $s^{(j)}$  such that  $j \neq 1$ ) can be safely removed from transition model estimation. The optimal subspace for transition model estimation has intrinsic dimension  $d_z = 1$  and we assume that the agent knows this. The corresponding optimal transformation matrix is the 1-by- $(d_s + 1)$  matrix given by

$$\mathbf{W}_{\text{opt}} = \pm \left[ \frac{1}{\sqrt{2}}, 0, \dots, 0, \frac{1}{\sqrt{2}} \right]. \quad (7.18)$$

## Evaluation on Different Supervised Linear Dimension Reduction methods

We firstly evaluate the performance of M-PGPE using different supervised linear dimension reduction methods. We consider chainwalk tasks with  $d_s \in \{3, 5, 8\}$ . We collect  $N = 20$  trajectories using the uniform random policy  $a_t \sim \text{U}(-0.4, 0.4)$ . The dataset  $\{(\mathbf{x}_i, \mathbf{s}'_i)\}_{i=1}^P$  with  $P = 200$  is standardized so that it has zero mean and unit variance. This dataset is used by the following supervised linear dimension reduction methods:

**LSCE:** The matrix  $\mathbf{W}$  and the transition model  $\widehat{p}(s'|\mathbf{W}\mathbf{x})$  are learned by minimizing an estimator of the squared-loss conditional entropy over the Grassmann manifold. This method is used by the proposed model-based reinforcement learning method.

**KDR:** The matrix  $\mathbf{W}$  is learned by minimizing the trace of the estimated conditional covariance operator over the Stiefel manifold (Fukumizu et al., 2004)<sup>3</sup>.

**LSQMI:** The matrix  $\mathbf{W}$  is learned by maximizing an estimator of the quadratic mutual information over the Grassmann manifold (Sainui and Sugiyama, 2014)<sup>4</sup>.

**LSQMID:** The matrix  $\mathbf{W}$  is learned by maximizing quadratic mutual information where the derivative of quadratic mutual information is estimated directly. Recall that this method is our first contribution which we presented in Chapter 3.

**LSDR:** The matrix  $\mathbf{W}$  is learned by maximizing an estimator of the squared-loss mutual information over the Grassmann manifold (Suzuki and Sugiyama, 2013)<sup>5</sup>.

We use the Gaussian basis function for all methods. For LSCE, LSQMI, LSQMID, and LSDR, we restart the optimization with 10 different initial solutions and choose the best solution according to their criteria. For KDR, we use the initial solution provided by the gKDR method proposed by Fukumizu and Leng (2012).

We firstly evaluate the dimension reduction performance of each method by the dimension reduction error:

$$\text{Error}_{\text{DR}} = \|\mathbf{W}^{\top} \mathbf{W} - \mathbf{W}_{\text{opt}}^{\top} \mathbf{W}_{\text{opt}}\|_{\text{Frobenius}}, \quad (7.19)$$

<sup>3</sup>We use the code from <http://www.ism.ac.jp/~fukumizu/software.html>.

<sup>4</sup>The manifold optimization is performed by the manifold optimization toolbox (Boumal et al., 2014).

<sup>5</sup>We use the code from <http://www.ms.k.u-tokyo.ac.jp/software.html#LSDR>.

where  $\mathbf{W}_{\text{opt}}$  is the optimal transformation matrix and  $\|\cdot\|_{\text{Frobenious}}$  denotes the Frobenious norm of a matrix. Note that this dimension reduction error is invariant to rotation within the subspace. The dimension reduction error of each method is given in Table 7.1. For  $d_s = 3$ , LSDR and KDR give the best results and outperform LSCE. However, they perform quite poorly for  $d_s = 5$  and 8 when compared to LSCE. A reason for their failure is that the transition probability is highly skewed due to the Gamma distribution. LSDR estimates the density ratio  $\frac{p(z, s')}{p(z)p(s')}$  contained in SMI. This density ratio tends to be highly fluctuated around the tails of the transition probability density, and thus it is difficult to be estimated when the number of data is relatively small. For the case of KDR, the heuristic parameter tuning of KDR does not work well for the Gamma transition probability. In addition, the gKDR method may provide poor initial solutions which make the result of KDR unstable as can be observed by the large standard error of KDR. Both LSQMI and LSQMID seem to not work well in these tasks.

After the dimension reduction step, we learn the transition model  $\hat{p}(s'|\mathbf{W}\mathbf{x})$  by LSCDE (Sugiyama et al., 2010). Note that without the dimension reduction step, M-PGPE with LSCDE is the same as the original M-PGPE method we presented in Chapter 6. Also note that LSCE does not need this step since the transition model has already been obtained (see Chapter 4). To evaluate the accuracy of the transition model, we compute the root mean square error (RMSE) which is defined by

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left( s'_i{}^{(1)} - \hat{s}_i{}^{(1)} \right)^2}, \quad (7.20)$$

where  $\hat{s}_i{}^{(1)}$  is obtained by Equation (7.5). The test dataset with  $N_{\text{test}} = 500$  is obtained by executing the uniform random policy.

The RMSE of each method is also given in Table 7.1. LSCE is the best method in this evaluation, even outperforming LSDR and KDR in terms of the mean error when  $d_s = 3$ . This result emphasizes the advantage of using LSCE for performing supervised linear dimension reduction in transition model estimation. LSQMID also performs well when  $d_s = 3$  and  $d_s = 8$ , even though it performs very poorly at the dimension reduction step. A reason for this phenomenon can be seen by its significantly large standard errors. This implies that LSQMID is unstable and its dimension reduction results can yield both highly accurate and highly inaccurate transition models. KDR and LSQMI do not work well, and LSCDE without dimension reduction gives the most inaccurate transition models.

Next, we evaluate the reinforcement learning performance of the proposed method. The reinforcement learning performance is measured by the averaged return over 1000 trajectories. As the baseline, we use the optimal deterministic policy which guides the agent to high reward regions:

$$a = \text{sign}(0.5 - s^{(1)}) \times \max(|0.5 - s^{(1)}|, 0.4). \quad (7.21)$$

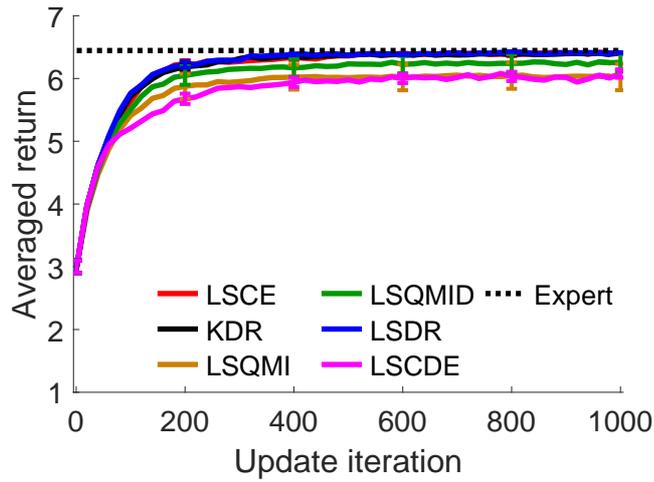
Figure 7.3 shows the averaged return against update iteration and Table 7.1 shows averaged return obtained at the 1000th update iteration. We can see that for  $d_s = 3$ , all supervised linear dimension reduction methods performs equally well, including LSQMI and LSQMID which have very large standard errors. However, only LSCE performs well across different state dimensions. Our previous M-PGPE method with LSCDE performs relatively poorly. This result shows that the proposed method which combines LSCE with M-PGPE is promising.

Table 7.1: Mean and standard error over 30 trials of the dimension reduction error (DR), the root mean square error (RMSE), and the averaged return at the 1000th update iteration on the chainwalk task with different state dimensions  $d_s$ . The best method in terms of the mean error and comparable methods according to the paired  $t$ -test at the significance level 5% are specified by bold face.

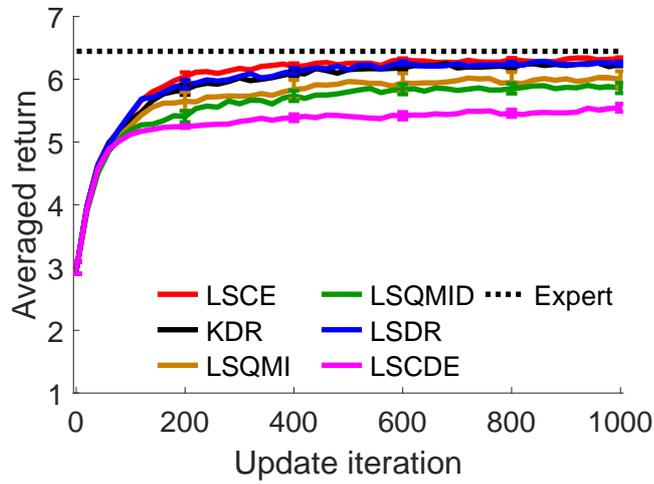
Criterion	$d_s$	LSCE	KDR	LSQMI	LSQMID	LSDR	LSCDE
DR	3	0.383(0.013)	<b>0.382(0.014)</b>	0.499(0.052)	0.442(0.028)	<b>0.359(0.006)</b>	-
	5	<b>0.419(0.012)</b>	0.476(0.021)	0.540(0.050)	0.622(0.027)	0.467(0.021)	-
	8	<b>0.569(0.033)</b>	0.882(0.052)	1.322(0.024)	0.711(0.038)	0.795(0.030)	-
RMSE	3	<b>0.230(0.008)</b>	0.273(0.011)	0.385(0.065)	<b>0.281(0.037)</b>	<b>0.241(0.006)</b>	0.462(0.008)
	5	<b>0.444(0.010)</b>	0.540(0.014)	0.618(0.045)	0.539(0.016)	0.558(0.013)	0.717(0.011)
	8	<b>0.693(0.032)</b>	0.992(0.047)	1.378(0.028)	<b>0.761(0.040)</b>	0.945(0.026)	0.869(0.012)
Averaged return	3	<b>6.405(0.009)</b>	<b>6.406(0.006)</b>	<b>6.024(0.211)</b>	<b>6.257(0.143)</b>	<b>6.410(0.007)</b>	6.079(0.066)
	5	<b>6.331(0.019)</b>	6.241(0.034)	6.001(0.127)	5.861(0.082)	6.252(0.029)	5.546(0.064)
	8	<b>6.028(0.074)</b>	4.976(0.247)	2.936(0.169)	5.469(0.167)	5.550(0.121)	5.261(0.025)

Table 7.2: Mean and standard error over 30 trials of the dimension reduction error (DR), the root mean square error (RMSE), and the averaged return at the 1000th update iteration of the proposed method on the 8-dimensional chainwalk task. The number of trajectories for transition model estimation is indicated by  $N$ . The best method in terms of the mean error and comparable methods according to the paired  $t$ -test at the significance level 5% are specified by bold face.

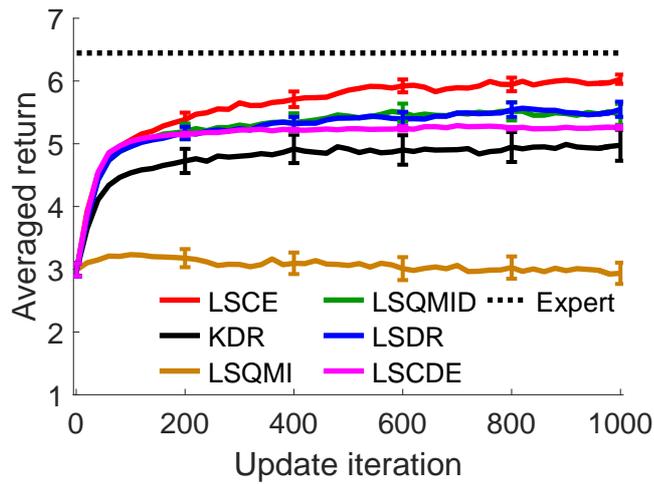
Criterion	$N = 10$	$N = 20$	$N = 50$	$N = 100$
DR	1.006(0.040)	0.569(0.033)	<b>0.481(0.019)</b>	<b>0.448(0.007)</b>
RMSE	1.132(0.042)	0.690(0.030)	<b>0.597(0.014)</b>	<b>0.568(0.005)</b>
Averaged return	4.330(0.225)	<b>6.028(0.074)</b>	<b>6.118(0.061)</b>	<b>6.190(0.040)</b>



(a) 3-dimensional chainwalk task.



(b) 5-dimensional chainwalk task.



(c) 8-dimensional chainwalk task.

Figure 7.3: Averaged return over 30 trials of the chainwalk tasks. The error bar indicates the standard error.

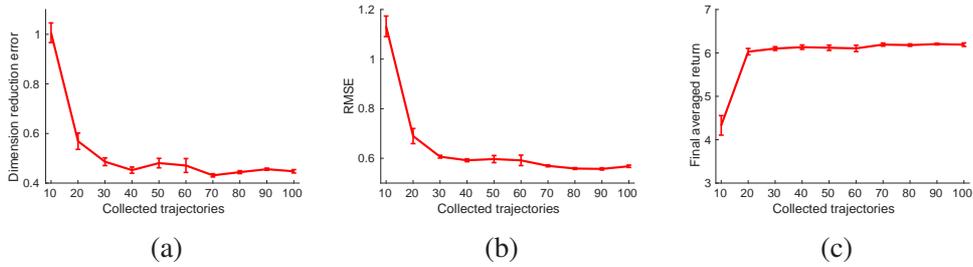


Figure 7.4: Performance of the proposed method on the 8-dimensional chainwalk task in terms of (a) dimension reduction error, (b) root mean square error, and (c) averaged return at the 1000th update iteration. The results are averaged over 30 trials and the error bar indicates the standard error.

### Evaluation on Different Numbers of Collected Trajectories

In this experiment, we evaluate the proposed method on different numbers of collected trajectories. We consider the chainwalk task with  $d_s = 8$ . Figure 7.4 and Table 7.2 show the performance of the proposed method against the number of collected trajectories  $N$ . The performance is measured by the dimension reduction error, the RMSE, and the averaged return at the 1000th update iteration. Figure 7.4 shows that the performance of the proposed method in all criteria can be improved by increasing  $N$ . However, Table 7.2 also shows that there is no significance difference in terms of the averaged return when increasing  $N$  to be more than 20. These results suggest that the performance of the proposed method can be improved by collecting more trajectories. However, there is a soft threshold, e.g.,  $N = 20$ , where collecting more trajectories only slightly improves the performance.

### Comparison with Model-Free Reinforcement Learning

This experiment aims to evaluate the data efficiency of the proposed method. We compare the performance of the proposed method with an existing model-free method called *importance-weighted PGPE* (IW-PGPE) (Zhao et al., 2013). IW-PGPE is an extension of PGPE which reuses previously collected trajectories to estimate the gradient and the optimal baseline. It was shown to outperform the original PGPE in both benchmark tasks and humanoid robot control (Zhao et al., 2013; Sugimoto et al., 2016).

We collect  $N' \in \{1, 5, 10\}$  trajectories at each update iteration for IW-PGPE. The averaged return against update iteration in Figure 7.5(a) shows that the averaged return of IW-PGPE increases as  $N'$  increases. Moreover, IW-PGPE with  $N' = 10$  outperforms the proposed method after 1000 update iterations. However, it collects a total of 10000 trajectories while the proposed method only collects a total of 100 trajectories.

Figure 7.5(b) shows the averaged return against collected trajectories of IW-PGPE and the proposed method when the budget for collecting trajectories is limited to 100 trajectories. In this scenario, IW-PGPE does not perform well when compared with the proposed method. Note that small  $N'$  performs better than large  $N'$  in this evaluation since small  $N'$  means many update iterations, e.g., 100 update iterations for  $N' = 1$ . Figure 7.5(c) shows that even when IW-PGPE collects up to 500 trajectories, it still cannot outperform the proposed method which collects only 100 trajectories.

These results demonstrate the advantage of the proposed model-based approach over the model-free approach in terms of data efficiency.

## 7.4.2 Cartpole Balancing

Next, we evaluate the proposed method on a more challenging task of cartpole balancing. We use a similar setup as the cartpole balancing task in Zhao et al. (2012).

### Setup

We formulate the cartpole balancing task as follows. The state  $\mathbf{s} = [\omega, \dot{\omega}, s^{(3)}, s^{(4)}]$  is a 4-dimensional vector consisting of the angle of the pole  $\omega \in [0, 2\pi]$ , the angular velocity of the pole  $\dot{\omega} \in [-3\pi, 3\pi]$ , and 2-dimensional noise  $s^{(3)}, s^{(4)} \in \mathbb{R}$ . The action  $a \in [-20, 20]$  corresponds to the force applied to the cart. The goal of the agent is to swing the pole to the upright position and balance it. This goal can be achieved by the reward function:

$$r(\mathbf{s}_t, \mathbf{a}, \mathbf{s}_{t+1}) = \cos(\omega_{t+1}). \quad (7.22)$$

The initial states are randomly chosen by

$$\omega_1 \sim \text{U}(0, 2\pi), \quad (7.23)$$

$$\dot{\omega}_1 \sim \text{U}(-3\pi, 3\pi), \quad (7.24)$$

$$s_1^{(3)}, s_1^{(4)} \sim \mathcal{N}(0, 1). \quad (7.25)$$

The angle and angular velocity of the pole are updated by the following transition dynamics:

$$\omega_{t+1} = \omega_t + \dot{\omega}_{t+1}\Delta t + \epsilon_1, \quad (7.26)$$

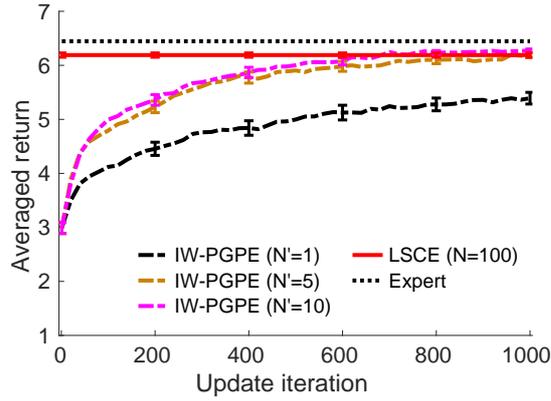
$$\dot{\omega}_{t+1} = \dot{\omega}_t + \frac{9.8 \sin(\omega_t) - \alpha w \dot{\omega}_t^2 \sin(2\omega_t)/2 + 10\alpha \cos(\omega_t) a_t}{4\ell/3 - \alpha w \ell \cos^2(\omega_t)} \Delta t + \epsilon_2, \quad (7.27)$$

where  $\alpha = 1/W + w$  and  $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, 0.25^2)$ . The mass of the cart is  $W = 8$  [kg], the mass of the pole is  $w = 2$  [kg], and the length of the pole is  $\ell = 0.5$  [m]. The update frequency is set to  $\Delta t = 0.1$  [s]. There is no transition dynamics for the noise, i.e.,  $s_{t+1}^{(3)}, s_{t+1}^{(4)} \sim \mathcal{N}(0, 1)$ . The trajectory length is  $T = 40$  and the discount factor is  $\gamma = 0.9$ . We use the following linear deterministic policy:

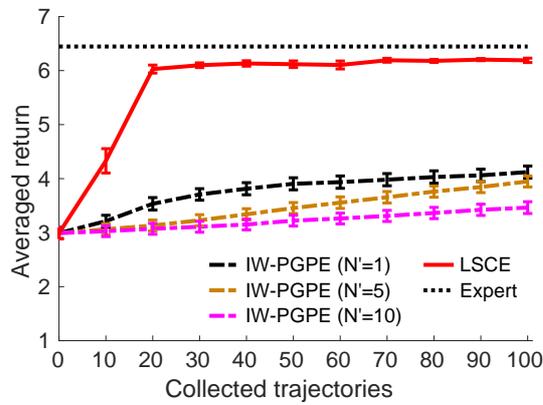
$$\begin{aligned} a &= \boldsymbol{\theta}^\top \boldsymbol{\phi}(\omega, \dot{\omega}) \\ &= \sum_{i=1}^{20} \theta_i \exp\left(-\frac{(\cos(\omega) - \cos(c_i))^2 + (\sin(\omega) - \sin(c_i))^2 + 4(\dot{\omega} - \dot{c}_i)^2/(6\pi)^2}{8(0.5)^2}\right), \end{aligned} \quad (7.28)$$

where  $(c_i, \dot{c}_i) \in \{0, \pi/2, \pi, 3\pi/2\} \times \{-3\pi, -3\pi/2, 0, 3\pi/2, 3\pi\}$ . PGPE uses the independent Gaussian distribution with the initial mean  $\eta_i = 0$  and the initial standard deviation  $\tau_i \sim \mathcal{N}(0, 1)$ . The learning rate is  $\beta = \frac{0.05}{\|\hat{\nabla}_{\boldsymbol{\rho}}^{(\text{baseline})} \mathcal{J}(\boldsymbol{\rho})\|}$ .

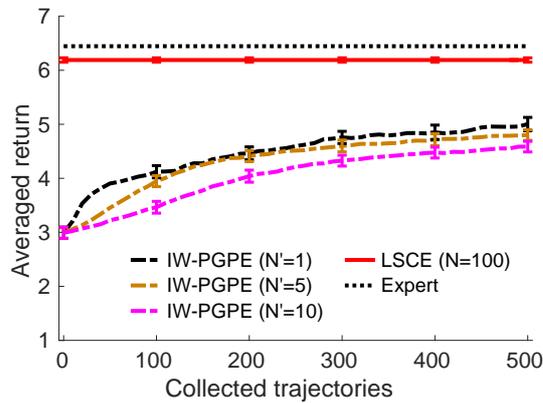
The transition probability only depends on the action and the first and second dimensions of the state vector. The optimal subspace for transition model estimation



(a) Averaged return by model-free IW-PGPE against update iteration.



(b) Averaged return by model-free IW-PGPE against collected trajectories when the number of collected trajectories is limited to 100 trajectories.



(c) Averaged return by model-free IW-PGPE against collected trajectories when the number of collected trajectories is limited to 500 trajectories.

Figure 7.5: Averaged return over 30 trials of the 8-dimensional chainwalk task by IW-PGPE. The error bar indicates the standard error. The number of trajectories collected at each update iteration is indicated by  $N'$ . The averaged return of the proposed method at the 1000th update iteration is included for comparison.

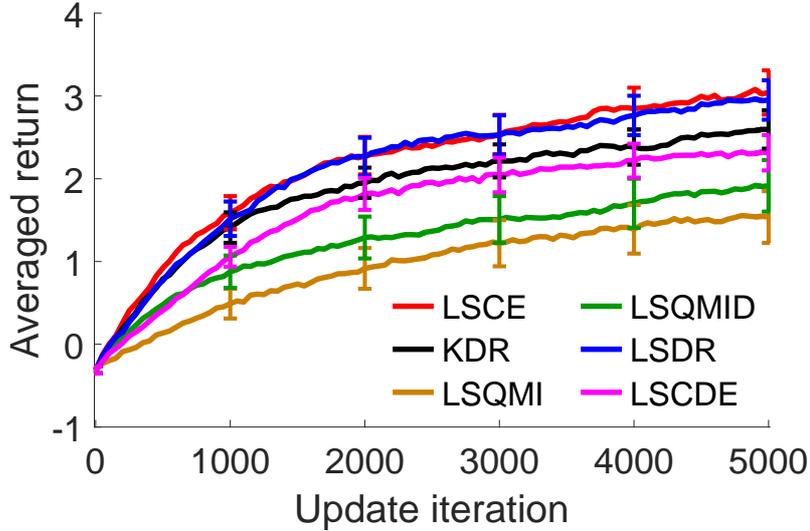


Figure 7.6: Averaged return over 30 trials of the cartpole balancing task. The error bar indicates the standard error.

has intrinsic dimension  $d_z = 3$  and we assume that the agent knows this. The corresponding optimal transformation matrix is given by any rotations of

$$\mathbf{W}_{\text{opt}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (7.29)$$

### Evaluation on Different Supervised Linear Dimension Reduction methods

We collect  $N = 15$  trajectories using the uniform random policy  $a_t = U(-20, 20)$ . The dataset  $\{(\mathbf{x}_i, \mathbf{s}'_i)\}_{i=1}^P$  with  $P = 600$  is standardized so that it has zero mean and unit variance. The setup of each supervised linear dimension reduction method is the same as that in the chainwalk experiment.

Table 7.3 shows the dimension reduction error, the RMSE, and the averaged return obtained at the 5000th update iteration. It shows that LSCE gives the best dimension reduction results in terms of the mean. LSQMID and LSDR are comparable to LSCE, while KDR and LSQMI do not work well.

After the dimension reduction step, we learn the transition model by LSCDE and compute the RMSE by a test dataset with  $N_{\text{test}} = 900$ . The result in Table 7.3 shows that LSCE and LSDR work equally well in terms of the RMSE and are consistent with their dimension reduction performance. However, LSQMID does not work well due to its unstable behavior despite its comparable dimension reduction performance. KDR, LSQMI and LSCDE do not work well in this task. This suggests that LSCE and LSDR should work equally well in the following reinforcement learning evaluation.

We evaluate the reinforcement learning performance by the averaged return over 1000 trajectories. Figure 7.6 the averaged return against update iteration and Table 7.3 shows averaged return obtained at the 5000th update iteration. The results show that LSCE and LSDR work equally well, while both LSQMI and LSQMID perform very poorly and are outperformed by LSCDE without dimension reduction.

Table 7.3: Mean and standard error over 30 trials of the dimension reduction error (DR), the root mean square error (RMSE), and the averaged return at the 5000th update iteration on the cartpole task. The best method in terms of the mean error and comparable methods according to the paired *t-test* at the significance level 5% are specified by bold face.

Criterion	LSCE	KDR	LSQMI	LSQMID	LSDR	LSCDE
DR	<b>0.533(0.083)</b>	0.774(0.032)	1.306(0.079)	<b>0.657(0.097)</b>	<b>0.641(0.052)</b>	-
RMSE	<b>1.789(0.008)</b>	1.807(0.007)	1.894(0.011)	1.811(0.011)	<b>1.786(0.007)</b>	1.858(0.004)
Averaged return	<b>3.043(0.267)</b>	2.595(0.232)	1.537(0.313)	1.914(0.311)	<b>2.951(0.238)</b>	2.315(0.214)

Table 7.4: Mean and standard error over 30 trials of the dimension reduction error (DR), the root mean square error (RMSE), and the averaged return at the 5000th update iteration of the proposed method on the cartpole task. The number of trajectories for transition model estimation is indicated by  $N$ . The best method in terms of the mean error and comparable methods according to the paired *t-test* at the significance level 5% are specified by bold face.

Criterion	$N = 10$	$N = 30$	$N = 60$	$N = 100$
DR	0.789(0.080)	0.193(0.028)	0.252(0.027)	<b>0.128(0.013)</b>
RMSE	1.309(0.011)	<b>1.227(0.004)</b>	<b>1.225(0.002)</b>	<b>1.223(0.003)</b>
Averaged return	2.308(0.275)	<b>2.864(0.267)</b>	<b>3.066(0.279)</b>	<b>3.355(0.267)</b>

Table 7.5: Mean and standard error over 10 trials of the root mean squared error by LSCE on the humanoid robot reaching task.

$d_z = 2$	$d_z = 5$	$d_z = 7$	$d_z = 10$	$d_z = 12$	$d_z = 15$	$d_z = 17$
1.097(0.093)	0.490(0.018)	0.491(0.019)	0.472(0.025)	0.454(0.021)	0.506(0.026)	0.536(0.030)

## Evaluation on Different Numbers of Collected Trajectories

We evaluate the proposed method on different numbers of collected trajectories. Table 7.4 shows the dimension reduction error, the RMSE, and the averaged return of the proposed method with the number of collected trajectory  $N \in \{10, 30, 60, 100\}$ . It shows that the performance of the proposed method can be improved by collecting more trajectories. However, there is a soft threshold, e.g.,  $N = 60$ , where collecting more trajectories only slightly improves the performance in terms of the averaged return. These results are similar to those in the chainwalk experiment.

## Comparison with Model-Free Reinforcement Learning

We compare the performance of the proposed method with IW-PGPE. We collect  $N' \in \{1, 5, 10\}$  trajectories at each iteration for IW-PGPE. Figure 7.7(a) shows that IW-PGPE does not perform well in this experiment. IW-PGPE with  $N' = 10$  requires 5000 update iterations or equivalently 50000 trajectories to be comparable to the proposed method which only uses 100 trajectories. Figure 7.7(b) shows the averaged return against collected trajectories of IW-PGPE when the budget for collecting trajectories is limited to 100 trajectories. In this scenario, the policy improvement by IW-PGPE is almost negligible when compared with the policy improvement by the proposed method. This result emphasizes the advantage of the model-based approach over the model-free approach in terms of data efficiency.

### 7.4.3 Humanoid Robot Control

Finally we evaluate the performance of the proposed method on a real-world control task using the humanoid robot *CB-i* (Cheng et al., 2007) (see Figure 7.8(a)).

#### Setup

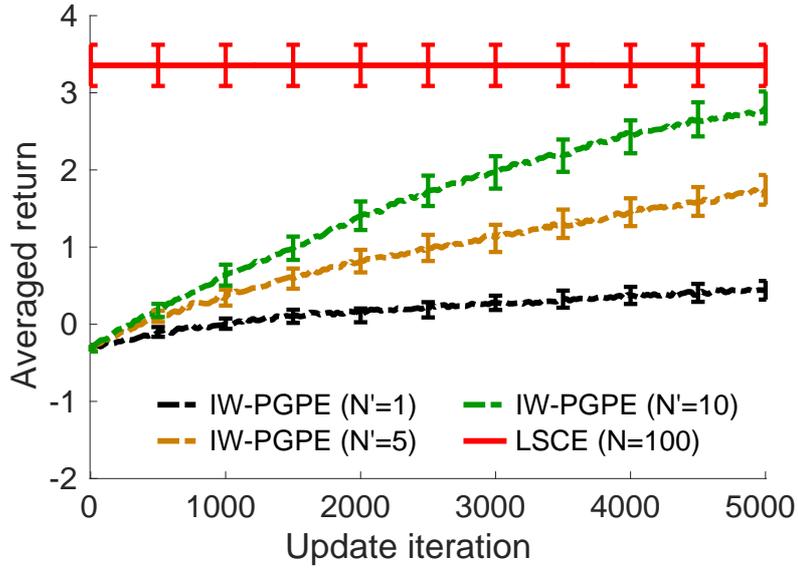
We consider a robot reaching task using 5 degrees of freedom (DoF) of the left arm. They are the torso yaw joint, the three left shoulder joints, and the left elbow joint. The goal of the robot is to move its left end-effector to the target object from the fixed initial position (see Figure 7.8(b)). The trajectory length is  $T = 100$  which equals approximately 1 second. The reward at each time step is

$$r_t = \exp(-d_t) - 0.0005 \min(c_t, 10000), \quad (7.30)$$

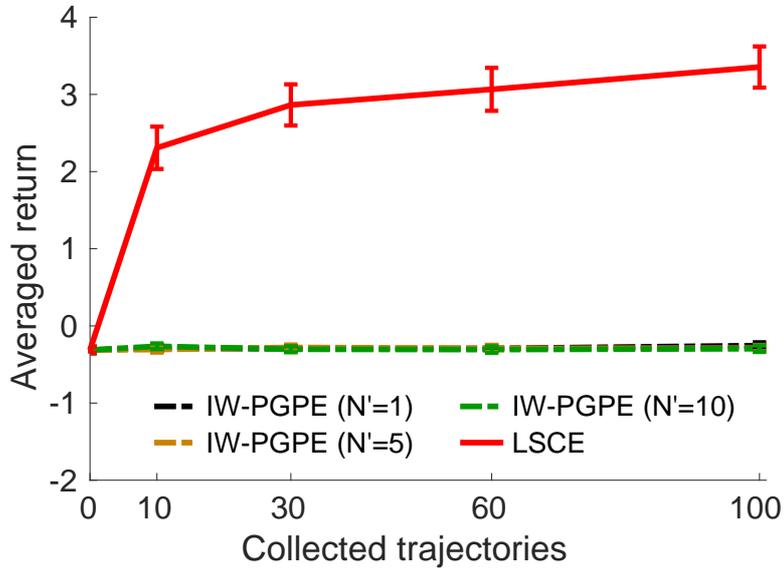
where  $d_t$  denotes the distance between the end effector and the target object, and  $c_t$  denotes the control cost. The discount factor is set at  $\gamma = 0.999$ . The optimal policy needs to reach the target object while keeping the control cost low.

We consider learning the robot transition probability in the joint space as follows. The state is a 10-dimensional vector  $\mathbf{s} = [\boldsymbol{\omega}^\top, \dot{\boldsymbol{\omega}}^\top]^\top$ , where  $\boldsymbol{\omega} \in \mathbb{R}^5$  denotes the joint angles and  $\dot{\boldsymbol{\omega}} \in \mathbb{R}^5$  denotes the joint angular velocities. The action is a 10-dimensional vector  $\mathbf{a} = [\boldsymbol{\omega}_+^\top, \dot{\boldsymbol{\omega}}_+^\top]^\top$ , where  $\boldsymbol{\omega}_+ \in \mathbb{R}^5$  denotes the desired joint angles and  $\dot{\boldsymbol{\omega}}_+ \in \mathbb{R}^5$  denotes the desired joints angular velocities. Thus, the transition probability in this joint space has 20 input dimensions and 10 output dimensions.

We use the time-dependent policy model similarly to that in Sugimoto et al. (2014, 2016). For each DoF, the target angle  $\omega_+$  and the target angular velocity  $\dot{\omega}_+$  at time



(a) Averaged return by model-free IW-PGPE against update iteration.



(b) Averaged return by model-free IW-PGPE against collected trajectories.

Figure 7.7: Averaged return over 30 trials of the cartpole task by IW-PGPE. The error bar indicates the standard error. The number of trajectories collected at each update iteration is indicated by  $N'$ . The averaged return of the proposed method at the 5000th update iteration is included for comparison.

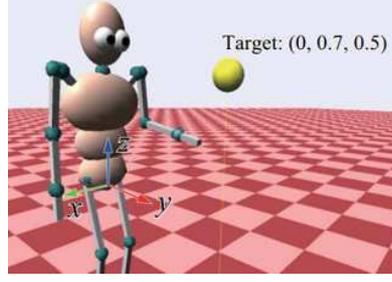
step  $t$  are determined from the augmented state  $\tilde{\mathbf{s}} = [\omega, \dot{\omega}, t]^\top$  by

$$\omega_+ = \omega + 0.01\dot{\omega}_+, \quad (7.31)$$

$$\dot{\omega}_+ = \sum_{i=1}^6 \theta_i \phi_i(t), \quad (7.32)$$



(a) Humanoid robot CB-i



(b) Humanoid robot simulator SL

Figure 7.8: Humanoid robot CB-i and simulator SL. The target object is located in front of the robot (the figure of the simulator is identical to that in Sugimoto et al. (2014)).

where  $\theta = [\theta_1, \dots, \theta_6]^\top$  is the policy parameter vector and

$$\phi_i(t) = \frac{\exp\left(-\frac{(t-c_i)^2}{2\sigma^2}\right)}{\sum_{j=1}^6 \exp\left(-\frac{(t-c_j)^2}{2\sigma^2}\right)}, \quad (7.33)$$

is the time-dependent normalized Gaussian basis function. The motor torque command for each joint is then computed by the proportional-derivative (PD) controller:

$$\tau = -K_P(\omega - \omega_+) - K_D(\dot{\omega} - \dot{\omega}_+), \quad (7.34)$$

where  $K_P > 0$  and  $K_D > 0$  denote the gain constants.

We use the independent Gaussian distribution for PGPE and use the learning rate  $\beta = \frac{0.05}{\|\hat{\varphi}_{\rho}^{(\text{baseline})} \mathcal{J}(\rho)\|}$ . The imitation learning discussed in Section 7.3.2 is used to initialize the mean and the standard deviation of the Gaussian distribution.  $N$  demonstrated trajectories are collected by an inverse kinematics controller as follows. For each demonstrated trajectory, we randomly choose a position  $(x, y, z) = (p_1, 0.7, 0.5)$  around the target object where  $p_1 \sim \text{U}(-0.1, 0.1)$  (see Figure 7.8(b)). Then we use the inverse kinematics controller to move the left end-effector to that  $(x, y, z)$  position. Note that for this setup the demonstrator is fully observable by the agent and they have identical state and action descriptions. Also note that the inverse kinematics controller is a suboptimal policy since it does not take into account the control cost term in the reward function.

### Evaluation on the Simulator SL

Firstly, we evaluate the proposed method on the humanoid robot simulator SL (Schaal, 2009) (see Figure 7.8(b)). The robot collects  $N = 20$  demonstrated trajectories for policy initialization and transition model estimation.

Since the intrinsic dimension  $d_z$  is unknown, we perform the following procedure to find good candidates for the intrinsic dimension. The dataset  $\{(\mathbf{x}_i, \mathbf{s}'_i)\}_{i=1}^{2000}$  is randomly split into a training dataset with 1500 data points and a validation dataset with 500 data points. The training dataset is used to learn transition models and the validation dataset is used to compute the RMSE. Table 7.5 shows the RMSE on

the validation dataset averaged over 10 trials. It shows that the intrinsic dimensions  $d_z \in \{7, 10, 12\}$  give small RMSEs and should be good candidates.

Based on the above model selection, we learn transition models with  $d_z \in \{7, 10, 12\}$  using all 2000 data points. Then, we evaluate the reinforcement learning performance by the averaged return over 10 trajectories. The averaged return against the update iteration in Figure 7.9(a) shows that LSCE with  $d_z = 10$  gives stable improvement as well as the highest averaged return at the 300th update iteration. On the other hand, the performance of LSCE with  $d_z \in \{7, 12\}$  and LSCDE are quite poor and quite unstable. Nonetheless, LSCE still outperforms LSCDE in term of the mean averaged return. This result implies that there exists a low-dimensional subspace of the state-action variable which can describe the transition probability of the 5-DoF humanoid robot in the joint space. Since  $d_z = 10$  gives the best performance we may assume that such a subspace has dimension of approximately 10.

Although the proposed method with  $d_z = 10$  gives a good performance, it still could not outperform the expert, i.e., the inverse kinematics controller. Nonetheless, we expect that the performance of the proposed method can be improved by collecting more trajectories, as suggested by our previous experiments.

### Comparison with Model-Free Reinforcement Learning

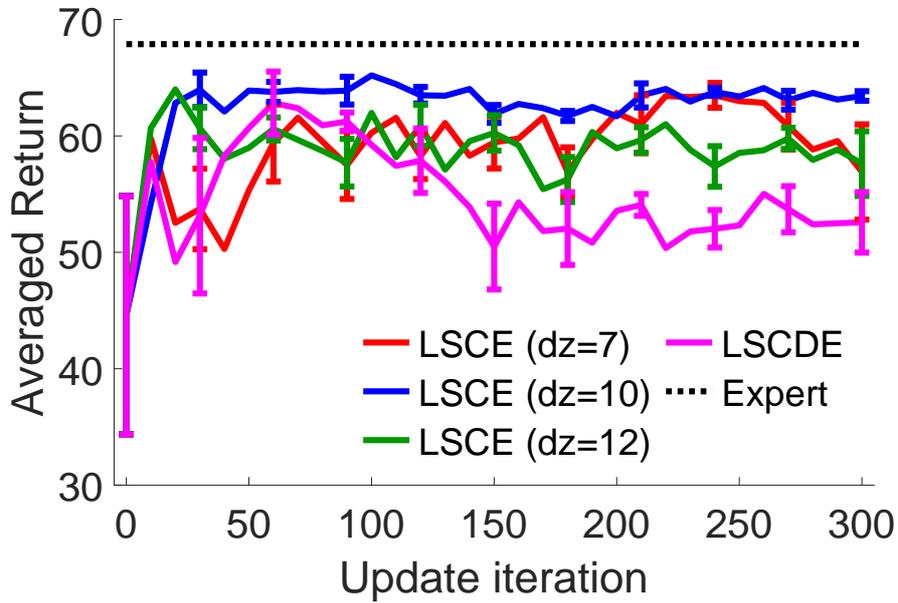
We compare the performance of the proposed method with IW-PGPE. The initial policy is obtained by imitation learning identically to the proposed method. We collect  $N' \in \{1, 5\}$  trajectories at each update iteration. Figure 7.9(b) shows the averaged return against collected trajectories of IW-PGPE. IW-PGPE improves the policy as it collects more trajectories and it eventually outperforms the proposed method and the expert. However, IW-PGPE requires approximately 100 to 150 trajectories to be comparable to the proposed method which only uses the 20 demonstrated trajectories. Thus, if we assume the scenario that the budget for collecting trajectories is limited to 20 trajectories, the proposed method would be more preferable.

### Evaluation on the Humanoid Robot CB-i

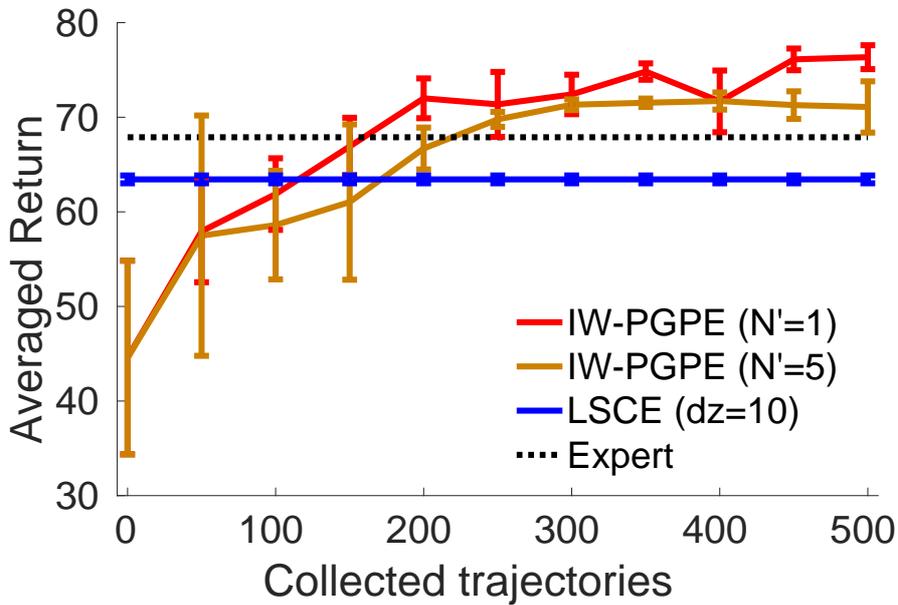
Lastly, we evaluate the proposed method on the real humanoid robot CB-i. The experimental setup is almost identical to that on the simulator SL, but with only  $N = 10$  demonstrated trajectories. We only consider the proposed method with  $d_z = 10$ . Figure 7.10 shows the averaged return against the update iteration on the CB-i. The policy improves gradually and converge to a stable policy. Figure 7.11 shows a typical reaching trajectory obtained by the proposed method using the learned policy at the 150th update iteration.

## 7.5 Conclusion

The performance of model-based reinforcement learning depends on the accuracy of the transition model. Existing transition model estimation methods tend to perform poorly when the input has high dimensionality. In this chapter, we propose to learn a transition model by LSCE which performs both transition model estimation and supervised linear dimension reduction in an integrated manner. We showed through experiments that LSCE is a promising transition model estimation method for high-dimensional reinforcement learning problems.



(a) Averaged return by M-PGPE with different transition models.



(b) Averaged return by model-free IW-PGPE with different data collection scheduling. The result of LSCE ( $d_z = 10$ ) at the 300th update iteration is included for comparison.

Figure 7.9: Averaged return over 10 test trajectories on the simulator SL. The error bar indicates the standard error.

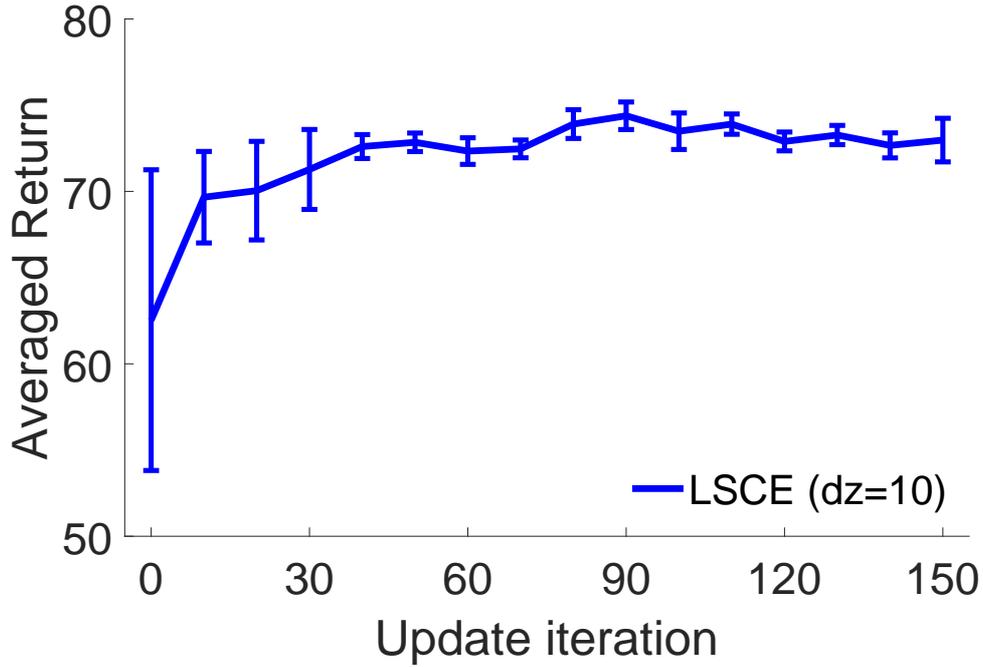


Figure 7.10: Averaged return over 10 trajectories by the proposed method on the humanoid robot CB-i. The error bar indicates the standard error.

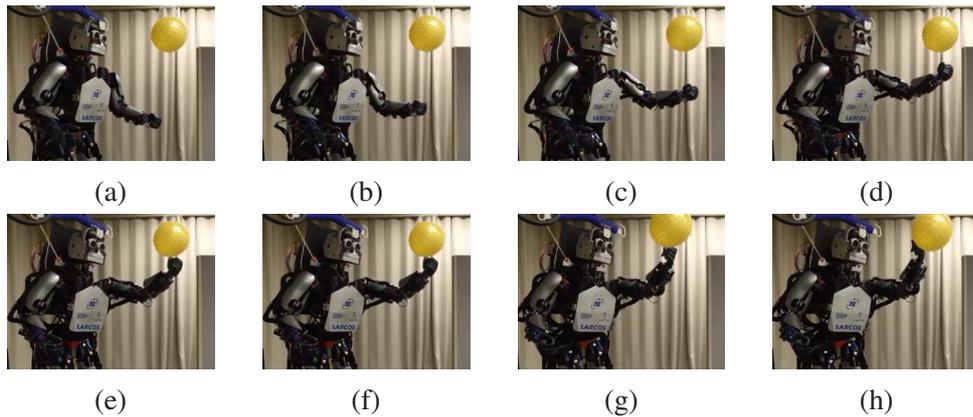


Figure 7.11: A typical reaching trajectory obtained by the proposed method on the CB-i.

We focused on the transition model estimation in a batch learning setting, i.e., all data points are use at once for learning. On the other hand, an online learning setting is also important since the agent may collect more trajectory data during learning. LSCE can be straightforwardly used in such an online scenario by learning the transition model from scratch every time a new data is observed. However, this naive usage is computationally expensive and may not be suitable for real-world control tasks which requires real-time operation. A computationally efficient online extension of LSCE will be considered in our future work.

We proposed to predict the next state vector from the transition model by numerically approximating the mean of the transition model (see Equation (7.5)). Although this prediction approach performs well in our experiments, it may not perform well when the transition model has multi-modality. For instance, the mean of a bi-modal

Gaussian transition model does not give an informative prediction of the next state. A possible approach to cope with this problem is to use the *mode* of the transition model, i.e.,

$$\hat{\mathbf{s}}_{t+1} = \operatorname{argmax}_{\mathbf{s}'} [\hat{p}(\mathbf{s}' | \mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{x}_t)]. \quad (7.35)$$

However, for the LSCE transition model we need to solve

$$\hat{\mathbf{s}}_{t+1} = \operatorname{argmax}_{\mathbf{s}'} \left[ \frac{\tilde{\alpha}^\top \varphi(\mathbf{W}\mathbf{x}_t, \mathbf{s}')}{\int \tilde{\alpha}^\top \varphi(\mathbf{W}\mathbf{x}_t, \tilde{\mathbf{s}}') d\tilde{\mathbf{s}}'} \right] = \operatorname{argmax}_{\mathbf{s}'} \left[ \tilde{\alpha}^\top \varphi(\mathbf{W}\mathbf{x}_t, \mathbf{s}') \right]. \quad (7.36)$$

For the Gaussian basis function, Equation (7.36) is a non-concave maximization problem which requires a non-linear optimization method and its solution can be a local solution. A more efficient prediction approach will be considered in our future work.

So far, we only focused on mitigating the curse of dimensionality for transition model estimation. However, the direct policy search approach also suffers from the curse of dimensionality. The high dimensionality in direct policy search refers to the high number of policy parameters, which is often a result of highly complex policy function. In our experiments, we avoid using highly complex policy function by using the handcrafted policy functions which rely only on small subset of feature of state. However, these handcrafted policy functions require prior knowledge about the tasks and are not always available. Thus, it is important to develop a policy model selection method to choose an appropriate policy model without prior knowledge. A policy gradient extension of the policy model selection approach proposed by Ueno et al. (2012) would be promising.

## Chapter 8

# Contextual Policy Search with Single-step Dimension Reduction

In this chapter, we present our fifth contribution on the contextual policy search with single-step dimension reduction. We firstly introduce the contextual policy search problem in Section 8.1. Then, in Section 8.2 we mathematically formulate the contextual policy search problem and briefly discuss existing methods and their limitation when the context variables have high dimensionality. Next, in Section 8.3 we present our method on the contextual policy search method with single-step dimension reduction. Lastly, we present our experimental evaluations in Section 8.4 and conclude the chapter in Section 8.5.

### 8.1 Introduction

In our previous contributions on model-based reinforcement learning, we considered a situation where an agent has to solve only one task, e.g., the robot learns a policy that reaches an object at a fixed position. However, in practice an autonomous agent often requires different policies for solving tasks with different contexts. For instance, in an object hitting task the robot has to adapt his controller according to the object position, i.e., the context. The direct policy search approach that we considered so far allows the agent to learn a separate policy for each context. However, learning optimal policies for many large contexts, such as in the presence of continuous context variables, is impractical. On the other hand, direct *contextual* policy search approaches (Kober et al., 2011; Neumann, 2011; da Silva et al., 2012) represent the contexts by real-valued vectors and are able to learn a context-dependent distribution over the policy parameters. Such a distribution can generalize across context values and therefore the agent is able to adapt to unseen contexts.

Yet, direct policy search methods (both contextual and plain) usually require a lot of evaluations of the objective and may converge prematurely. To alleviate these issues, Abdolmaleki et al. (2015) recently proposed a stochastic search framework called *model-based relative entropy stochastic search (MORE)*. In this framework, the new search distribution can be computed efficiently in a closed form using a learned model of the objective function. MORE outperformed state-of-the-art methods in stochastic optimization problems and single-context policy search problems, but its application to contextual policy search has not been explored yet. The first part of our contribution in this chapter is a novel contextual policy search method in the MORE framework.

However, a naive extension of the original MORE would still suffer from high-dimensional contexts. Learning from high-dimensional variables, in fact, is still an important problem in statistics and machine learning (Hastie et al., 2001; Bishop, 2006; Murphy, 2012). Nowadays, high-dimensional data (e.g., camera images) can often be obtained quite easily, but obtaining informative low-dimensional variables (e.g., objects positions) is non-trivial and requires prior knowledge and/or human guidance.

In this chapter, we propose a method to handle high-dimensional context variables by learning a low-rank representation of the objective function. We show that learning a low-rank representation corresponds to simultaneously performing supervised linear dimension reduction on the context variables. Since optimization with a rank constraint is generally NP-hard, we minimize the *nuclear norm* (also called trace norm), which is a *convex* surrogate of the rank function (Recht et al., 2010). This minimization allows us to learn a low-rank representation in a fully supervised manner by just solving a convex optimization problem. We evaluate the proposed method on a synthetic task with known ground truth and on robotic ball hitting tasks based on camera images. The evaluation shows that the proposed method with nuclear norm minimization outperforms the methods that naively perform principal component analysis to reduce the dimensionality of context variables in a multi-step manner.

In the next section, we give a mathematical problem formulation of the contextual policy search and briefly discuss related work.

## 8.2 Contextual Policy Search

In this section, we formulate the direct contextual policy search problem and briefly discuss existing methods.

### 8.2.1 Problem Formulation

Recall that in the standard direct policy search, the goal is to find a parameter  $\theta$  which maximizes the expected return:

$$R(\theta) = \int \tilde{R}(\tau) p(\tau|\theta) d\tau. \quad (8.1)$$

where  $\tilde{R}(\tau)$  is the return along a trajectory  $\tau$  and  $p(\tau|\theta)$  is the trajectory density function. Note that we use slightly different notations from those in Chapter 5 in the remainder of this chapter. The return  $\tilde{R}(\tau)$  is given by

$$\tilde{R}(\tau) = \sum_{t=1}^T \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}), \quad (8.2)$$

where  $r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  is the reward function. Also, the trajectory density function  $p(\tau|\theta)$  is given by

$$p(\tau|\theta) = p_1(\mathbf{s}_1) \prod_{t=1}^T p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \pi(\mathbf{a}_t|\mathbf{s}_t; \theta), \quad (8.3)$$

where  $p_1(\mathbf{s})$  is the initial state density function,  $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$  is the transition probability density function, and  $\pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$  is a parameterized policy function with policy parameter  $\boldsymbol{\theta}$ .

In direct contextual policy search, we assume that the reward function and the transition probability density function are determined by the *context variables*  $\mathbf{c}$ . To emphasize dependency on  $\mathbf{c}$ , we use  $r(\mathbf{s}, \mathbf{a}, \mathbf{s}'; \mathbf{c})$  and  $p(\mathbf{s}'|\mathbf{s}, \mathbf{a}; \mathbf{c})$  to denote the context-dependent reward functions and the context-dependent transition probability density function, respectively. Then, the expected return now depends on the context variables as well, i.e.,

$$R(\boldsymbol{\theta}, \mathbf{c}) = \int \tilde{R}(\boldsymbol{\tau}, \mathbf{c}) p(\boldsymbol{\tau}|\boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\tau}, \quad (8.4)$$

where  $\tilde{R}(\boldsymbol{\tau}, \mathbf{c})$  denotes the context-dependent return along a trajectory  $\boldsymbol{\tau}$ :

$$\tilde{R}(\boldsymbol{\tau}, \mathbf{c}) = \sum_{t=1}^T \gamma^{t-1} r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}; \mathbf{c}), \quad (8.5)$$

and  $p(\boldsymbol{\tau}|\boldsymbol{\theta}, \mathbf{c})$  denotes the context-dependent trajectory probability density:

$$p(\boldsymbol{\tau}|\boldsymbol{\theta}, \mathbf{c}) = p_1(\mathbf{s}_1) \prod_{t=1}^T p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t; \mathbf{c}) \pi(\mathbf{a}_t|\mathbf{s}_t; \boldsymbol{\theta}). \quad (8.6)$$

The goal of direct contextual policy search is to find a search distribution  $p(\boldsymbol{\theta}|\mathbf{c})$  maximizing the expected return

$$\iint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\theta} d\mathbf{c}, \quad (8.7)$$

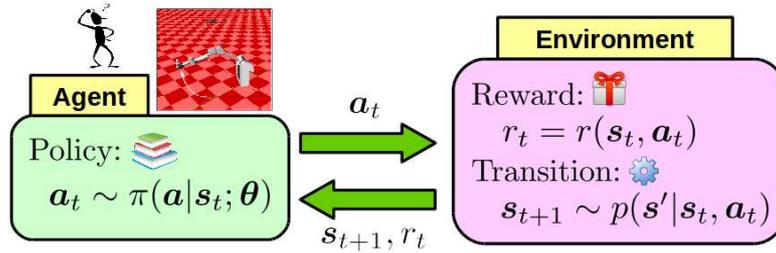
where  $\mu(\mathbf{c})$  denotes the context distribution. We assume that the context dimensionality  $d_c$  is very large. Figure 8.1 illustrates comparison between standard policy search and contextual policy search.

In general, the functional form of  $R(\boldsymbol{\theta}, \mathbf{c})$  is unknown since the transition probability is unknown. However, we assume that the agent can always access its value by the following data collection process. An agent observes the context variable  $\mathbf{c} \in \mathbb{R}^{d_c}$  and draws a parameter  $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$  from a search distribution  $p(\boldsymbol{\theta}|\mathbf{c})$ . Subsequently, the agent executes a policy with the parameter  $\boldsymbol{\theta}$  and observes a return computed by  $R(\boldsymbol{\theta}, \mathbf{c})$ .

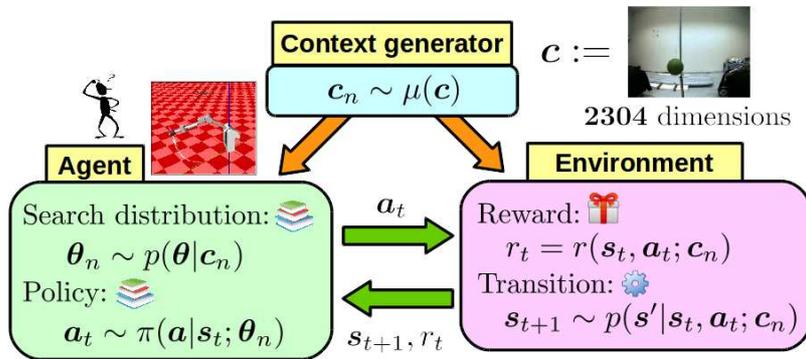
It is also convenient to treat the trajectory execution of the agent as a *black box* system and then simplify the problem into maximizing output of a black box function. That is, contextual policy search methods query the context variable  $\mathbf{c}$  and the drawn policy parameter  $\boldsymbol{\theta}$  to this black box, and then this black box outputs  $R(\boldsymbol{\theta}, \mathbf{c})$ . In this case, the goal of contextual policy search can be understood as finding a search distribution  $p(\boldsymbol{\theta}|\mathbf{c})$  which maximizes the expected output of this black box. This black box treatment is illustrated in Figure 8.2.

Note that  $R(\boldsymbol{\theta}, \mathbf{c})$  is an expected return over the trajectory distribution while the objective function in Equation (8.7) is also an expected return over the trajectory distribution, the context distribution, and the search distribution, i.e.,

$$\iint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\theta} d\mathbf{c} = \iiint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) \tilde{R}(\boldsymbol{\tau}) p(\boldsymbol{\tau}|\boldsymbol{\theta}) d\boldsymbol{\theta} d\mathbf{c} d\boldsymbol{\tau}. \quad (8.8)$$



(a) Standard policy search



(b) Contextual policy search

Figure 8.1: Comparison between standard policy search and contextual policy search. In contextual policy search, an agent observes the context variable  $c$  which determines the reward function and the transition probability of the environment. In the ball hitting task, context variable can be camera images which encode information about the position of the object.

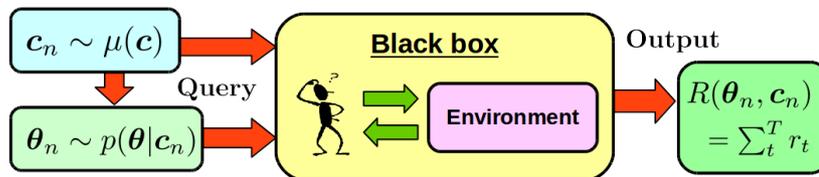


Figure 8.2: A simplified contextual policy search problem. The trajectory execution of the agent is treated as a black box system which we can query  $c$  and  $\theta$  to obtain  $R(\theta, c)$ .

To avoid confusion, in this chapter we simply refer to  $R(\boldsymbol{\theta}, \mathbf{c})$  as *reward* instead, and we refer to Equation (8.7) as the expected return. We also stress that context variables are fixed during task execution and they are drawn independently from  $\mu(\mathbf{c})$ . Thus, context variables are different from state variables in standard policy search.

## 8.2.2 Related Work

In the basic direct contextual policy search framework, the agent iteratively collects data  $\{(\boldsymbol{\theta}_n, \mathbf{c}_n, R(\boldsymbol{\theta}_n, \mathbf{c}_n))\}_{n=1}^N$  using a sampling distribution  $q(\boldsymbol{\theta}|\mathbf{c})$ . Subsequently, it computes a new search distribution  $p(\boldsymbol{\theta}|\mathbf{c})$  such that the expected reward increases or is maximized. In literature, different approaches have been used to compute the new search distribution, e.g., evolutionary strategies (Hansen et al., 2003), expectation-maximization algorithms (Kober et al., 2011), or information theoretic approaches (Deisenroth et al., 2013).

Most of the existing direct contextual policy search methods focus on tasks with low-dimensional context variables. To learn from high-dimensional context variables, usually the problem of learning a low-dimensional context representation is separated from the direct policy search by preprocessing the context space. However, unsupervised linear dimension reduction techniques are insufficient in problems where the latent representation contains distractor dimensions that do not influence the reward. A prominent example is principal component analysis (PCA) (Jolliffe, 1986), that does not take the supervisory signal into account and therefore cannot discriminate between relevant and irrelevant latent dimensions. On the other hand, supervised linear dimension reduction techniques such as our previous contributions require a suitable response variable. However, manually defining such variables is nontrivial for many problems.

In the last years, non-linear dimension reduction techniques based on deep learning have gained popularity (Bengio, 2009). For instance, Watter et al. (2015b) proposed a generative deep network to learn low-dimensional representations of images in order to capture information about the system transition dynamics and allow optimal control problems to be solved in low-dimensional spaces. More recently, Silver et al. (2016) successfully trained a machine to play a high-level game of *go* using a deep convolutional network. Although their work does not directly focus on dimension reduction, deep convolutional networks are known to be able to extract meaningful data representations. Thus, the effect of dimension reduction is achieved.

However, deep learning approaches generally require large datasets that are difficult to obtain in real-world scenarios (e.g., robotics). Furthermore, they involve solving non-convex optimization, which can suffer from local optima.

In this chapter, we tackle the issues raised above. First, the proposed approach integrates supervised linear dimension reduction on the context variables by learning a *low-rank representation* for the reward model. Second, the model learning problem is formalized as a *convex* optimization problem and is therefore guaranteed to converge to a global optimum.

## 8.3 Contextual Model-based Relative Entropy Stochastic Search

The original MORE (Abdolmaleki et al., 2015) finds a search distribution (without context) maximizing the expected reward while upper-bounding the Kullback-Leibler

(KL) divergence (Kullback and Leibler, 1951) between successive search distributions and lower-bounding the entropy of the new search distribution. The KL and the entropy bounds control the exploration-exploitation trade-off. The key insight of MORE is to learn a reward model to efficiently compute a new search distribution in closed form. Below, we propose our method called *contextual model-based relative entropy stochastic search* (C-MORE), which is a direct contextual policy search method in the MORE framework.

### 8.3.1 Learning the Search Distribution

The goal of C-MORE is to find a search distribution  $p(\boldsymbol{\theta}|\mathbf{c})$  that maximizes the expected reward while upper-bounding the expected KL divergence between  $p(\boldsymbol{\theta}|\mathbf{c})$  and  $q(\boldsymbol{\theta}|\mathbf{c})$ , and lower-bounding the expected entropy of  $p(\boldsymbol{\theta}|\mathbf{c})$ . Formally,

$$\begin{aligned} \max_p \quad & \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c})R(\boldsymbol{\theta}, \mathbf{c})d\boldsymbol{\theta}d\mathbf{c}, \\ \text{s.t.} \quad & \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c}) \log \frac{p(\boldsymbol{\theta}|\mathbf{c})}{q(\boldsymbol{\theta}|\mathbf{c})}d\boldsymbol{\theta}d\mathbf{c} \leq \epsilon, \\ & - \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c}) \log p(\boldsymbol{\theta}|\mathbf{c})d\boldsymbol{\theta}d\mathbf{c} \geq \beta, \\ & \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c})d\boldsymbol{\theta}d\mathbf{c} = 1, \end{aligned} \tag{8.9}$$

where the KL upper-bound  $\epsilon$  and the entropy lower-bound  $\beta$  are parameters specified by the user. The former is fixed for the whole learning process. The latter is adaptively changed according to the percentage of the relative difference between the sampling policy's expected entropy and the minimal entropy, as described by Abdolmaleki et al. (2015), i.e.,

$$\beta = \gamma(\mathbb{E}[H(q)] - H_0) + H_0, \tag{8.10}$$

where  $\mathbb{E}[H(q)] = - \int \int \mu(\mathbf{c})q(\boldsymbol{\theta}|\mathbf{c}) \log q(\boldsymbol{\theta}|\mathbf{c})d\boldsymbol{\theta}d\mathbf{c}$  is the sampling policy's expected entropy and  $H_0$  is the minimal entropy. In the experiments, we set  $\gamma = 0.99$  and  $H_0 = -150$ . The above optimization problem can be solved as follows by using the method of Lagrange multipliers which we introduced in Chapter 5.

Firstly, we write the Lagrangian  $\mathcal{L}$  with the Lagrange multipliers  $\eta > 0, \omega > 0$ , and  $\gamma$ , which correspond to the first, second, and third constraints, respectively:

$$\begin{aligned} \mathcal{L}(p, \eta, \omega, \gamma) = & \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c})R(\boldsymbol{\theta}, \mathbf{c})d\boldsymbol{\theta}d\mathbf{c} \\ & + \eta \left( \epsilon - \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c}) \log \frac{p(\boldsymbol{\theta}|\mathbf{c})}{q(\boldsymbol{\theta}|\mathbf{c})}d\boldsymbol{\theta}d\mathbf{c} \right) \\ & + \omega \left( - \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c}) \log p(\boldsymbol{\theta}|\mathbf{c})d\boldsymbol{\theta}d\mathbf{c} - \beta \right) \\ & + \gamma \left( \int \int \mu(\mathbf{c})p(\boldsymbol{\theta}|\mathbf{c})d\boldsymbol{\theta}d\mathbf{c} - 1 \right). \end{aligned} \tag{8.11}$$

Then, we maximize the Lagrangian  $\mathcal{L}(p, \eta, \omega, \gamma)$  w.r.t. the primal variable  $p$ . The

derivative of the Lagrangian w.r.t.  $p$  is

$$\begin{aligned} \partial_p \mathcal{L}(p, \eta, \omega, \gamma) &= \iint \mu(\mathbf{c}) \left( R(\boldsymbol{\theta}, \mathbf{c}) - (\eta + \omega) \log p(\boldsymbol{\theta}|\mathbf{c}) + \eta \log q(\boldsymbol{\theta}|\mathbf{c}) \right) d\boldsymbol{\theta} d\mathbf{c} \\ &\quad - (\eta + \omega - \gamma). \end{aligned} \quad (8.12)$$

By setting this derivative to zero, we have

$$\begin{aligned} 0 &= \iint \mu(\mathbf{c}) \left( R(\boldsymbol{\theta}, \mathbf{c}) - (\eta + \omega) \log p(\boldsymbol{\theta}|\mathbf{c}) + \eta \log q(\boldsymbol{\theta}|\mathbf{c}) \right) d\boldsymbol{\theta} d\mathbf{c} - (\eta + \omega - \gamma) \\ &= R(\boldsymbol{\theta}, \mathbf{c}) - (\eta + \omega) \log p(\boldsymbol{\theta}|\mathbf{c}) + \eta \log q(\boldsymbol{\theta}|\mathbf{c}) - (\eta + \omega - \gamma). \end{aligned} \quad (8.13)$$

This gives us

$$\log p(\boldsymbol{\theta}|\mathbf{c}) = \frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega} + \frac{\eta}{\eta + \omega} \log q(\boldsymbol{\theta}|\mathbf{c}) - \frac{\eta + \omega - \gamma}{\eta + \omega}, \quad (8.14)$$

$$p(\boldsymbol{\theta}|\mathbf{c}) = q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta + \omega}} \exp\left(\frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) \exp\left(-\frac{\eta + \omega - \gamma}{\eta + \omega}\right). \quad (8.15)$$

The last exponential term in Equation (8.15) is the normalization constant for the search distribution  $p(\boldsymbol{\theta}|\mathbf{c})$  since it does not depend on  $\boldsymbol{\theta}$  or  $\mathbf{c}$ . Thus, we have

$$\exp\left(\frac{\eta + \omega - \gamma}{\eta + \omega}\right) = \int q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta + \omega}} \exp\left(\frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) d\boldsymbol{\theta}, \quad (8.16)$$

$$\eta + \omega - \gamma = (\eta + \omega) \log\left(\int q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta + \omega}} \exp\left(\frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) d\boldsymbol{\theta}\right). \quad (8.17)$$

(The minus sign in the exponent in Equation (8.15) becomes the inverse operator and cancels out). This normalization term will be used to derive the dual function. Next, we substitute the term  $\log p(\boldsymbol{\theta}|\mathbf{c})$  back to the Lagrangian

$$\begin{aligned} \mathcal{L}(p^*, \eta, \omega, \gamma) &= \iint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) R(\boldsymbol{\theta}, \mathbf{c}) d\boldsymbol{\theta} d\mathbf{c} \\ &\quad - \eta \left( \iint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) \left[ \frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega} + \frac{\eta}{\eta + \omega} \log q(\boldsymbol{\theta}|\mathbf{c}) - \frac{\eta + \omega - \gamma}{\eta + \omega} \right] d\boldsymbol{\theta} d\mathbf{c} \right) \\ &\quad - \omega \left( \iint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) \left[ \frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega} + \frac{\eta}{\eta + \omega} \log q(\boldsymbol{\theta}|\mathbf{c}) - \frac{\eta + \omega - \gamma}{\eta + \omega} \right] d\boldsymbol{\theta} d\mathbf{c} \right) \\ &\quad + \eta \iint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) \log q(\boldsymbol{\theta}|\mathbf{c}) d\boldsymbol{\theta} d\mathbf{c} + \gamma \left( \iint \mu(\mathbf{c}) p(\boldsymbol{\theta}|\mathbf{c}) d\boldsymbol{\theta} d\mathbf{c} - 1 \right) \\ &\quad + \eta\epsilon - \omega\beta. \end{aligned} \quad (8.18)$$

Most terms cancel out and we only have

$$\begin{aligned} \mathcal{L}(p^*, \eta, \omega, \gamma) &= \eta\epsilon - \omega\beta - \gamma + \int \mu(\mathbf{c}) (\eta + \omega) d\mathbf{c} \\ &= \eta\epsilon - \omega\beta + \int \mu(\mathbf{c}) (\eta + \omega - \gamma) d\mathbf{c} \\ &= \eta\epsilon - \omega\beta + (\eta + \omega) \int \mu(\mathbf{c}) \log\left(\int q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta + \omega}} \exp\left(\frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) d\boldsymbol{\theta}\right) d\mathbf{c} \\ &= g(\eta, \omega). \end{aligned} \quad (8.19)$$

The Lagrange multipliers  $\eta > 0$  and  $\omega > 0$  are obtained by minimizing the dual function  $g(\eta, \omega)$ . Then, the search distribution in Equation (8.15) can be computed using these Lagrange multipliers. However, evaluating  $g(\eta, \omega)$  is not trivial due to the integration over  $q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta+\omega}}$ , that cannot be approximated straightforwardly by sample averages. Below, we describe how to solve this issue and evaluate the dual function from data.

### 8.3.2 Dual Function Evaluation via the Quadratic Model

We assume that the reward function  $R(\boldsymbol{\theta}, \mathbf{c})$  can be approximated by a quadratic model

$$\widehat{R}(\boldsymbol{\theta}, \mathbf{c}) = \boldsymbol{\theta}^\top \mathbf{A} \boldsymbol{\theta} + \mathbf{c}^\top \mathbf{B} \mathbf{c} + 2\boldsymbol{\theta}^\top \mathbf{D} \mathbf{c} + \boldsymbol{\theta}^\top \mathbf{r}_1 + \mathbf{c}^\top \mathbf{r}_2 + r_0, \quad (8.20)$$

where  $\mathbf{A} \in \mathbb{R}^{d_\theta \times d_\theta}$ ,  $\mathbf{B} \in \mathbb{R}^{d_c \times d_c}$ ,  $\mathbf{D} \in \mathbb{R}^{d_\theta \times d_c}$ ,  $\mathbf{r}_1 \in \mathbb{R}^{d_\theta}$ ,  $\mathbf{r}_2 \in \mathbb{R}^{d_c}$ , and  $r_0 \in \mathbb{R}$  are the model parameters. Matrices  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric. We also assume the sampling distribution  $q(\boldsymbol{\theta}|\mathbf{c})$  to be Gaussian of the form

$$q(\boldsymbol{\theta}|\mathbf{c}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{b} + \mathbf{K} \mathbf{c}, \mathbf{Q}). \quad (8.21)$$

Under these assumptions, the dual function and the new search distribution can be computed as follows. Firstly, we consider the problematic term in the dual function and the new search distribution:

$$q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{R(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right). \quad (8.22)$$

Using the Gaussian distribution  $q(\boldsymbol{\theta}|\mathbf{c})$  and replacing  $R(\boldsymbol{\theta}, \mathbf{c})$  with  $\widehat{R}(\boldsymbol{\theta}, \mathbf{c})$  yield

$$\begin{aligned} & q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{\widehat{R}(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) \\ &= \frac{1}{|2\pi\mathbf{Q}|^{\frac{\eta}{2(\eta+\omega)}}} \exp\left(-\frac{\eta}{2(\eta+\omega)}[\boldsymbol{\theta} - (\mathbf{b} + \mathbf{K} \mathbf{c})]^\top \mathbf{Q}^{-1}[\boldsymbol{\theta} - (\mathbf{b} + \mathbf{K} \mathbf{c})]\right) \\ & \quad \times \exp\left(\frac{\boldsymbol{\theta}^\top \mathbf{A} \boldsymbol{\theta} + \mathbf{c}^\top \mathbf{B} \mathbf{c} + 2\boldsymbol{\theta}^\top \mathbf{D} \mathbf{c} + \boldsymbol{\theta}^\top \mathbf{r}_1 + \mathbf{c}^\top \mathbf{r}_2 + r_0}{\eta + \omega}\right) \\ &= \frac{1}{|2\pi\mathbf{Q}|^{\frac{\eta}{2(\eta+\omega)}}} \exp\left(\frac{1}{2(\eta+\omega)}\left(-\eta\boldsymbol{\theta}^\top \mathbf{Q}^{-1}\boldsymbol{\theta} + 2\eta\boldsymbol{\theta}^\top \mathbf{Q}^{-1}\mathbf{b} + 2\eta\boldsymbol{\theta}^\top \mathbf{Q}^{-1}\mathbf{K} \mathbf{c} \right. \right. \\ & \quad \left. \left. - \eta\mathbf{b}^\top \mathbf{Q}^{-1}\mathbf{b} - 2\eta\mathbf{b}^\top \mathbf{Q}^{-1}\mathbf{K} \mathbf{c} - \eta\mathbf{c}^\top \mathbf{K}^\top \mathbf{Q}^{-1}\mathbf{K} \mathbf{c} + 2\boldsymbol{\theta}^\top \mathbf{A} \boldsymbol{\theta} + 2\mathbf{c}^\top \mathbf{B} \mathbf{c} \right. \right. \\ & \quad \left. \left. + 4\boldsymbol{\theta}^\top \mathbf{D} \mathbf{c} + 2\boldsymbol{\theta}^\top \mathbf{r}_1 + 2\mathbf{c}^\top \mathbf{r}_2 + 2r_0\right)\right) \\ &= \frac{1}{|2\pi\mathbf{Q}|^{\frac{\eta}{2(\eta+\omega)}}} \exp\left(\frac{1}{2(\eta+\omega)}\left(\boldsymbol{\theta}^\top (-\eta\mathbf{Q}^{-1} + 2\mathbf{A}) \boldsymbol{\theta} + \boldsymbol{\theta}^\top (2\eta\mathbf{Q}^{-1}\mathbf{b} + 2\mathbf{r}_1) \right. \right. \\ & \quad \left. \left. + \boldsymbol{\theta}^\top (2\eta\mathbf{Q}^{-1}\mathbf{K} + 4\mathbf{D}) \mathbf{c} + G\right)\right) \\ &= \frac{1}{|2\pi\mathbf{Q}|^{\frac{\eta}{2(\eta+\omega)}}} \exp\left(\frac{1}{2(\eta+\omega)}\left(-(\boldsymbol{\theta}^\top \mathbf{F} \boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \mathbf{f} - 2\boldsymbol{\theta}^\top \mathbf{L} \mathbf{c})\right) + \frac{G}{2(\eta+\omega)}\right), \end{aligned} \quad (8.23)$$

where

$$\mathbf{F} = \eta \mathbf{Q}^{-1} - 2\mathbf{A}, \quad (8.24)$$

$$\mathbf{f} = \eta \mathbf{Q}^{-1} \mathbf{b} + \mathbf{r}_1, \quad (8.25)$$

$$\mathbf{L} = \eta \mathbf{Q}^{-1} \mathbf{K} + 2\mathbf{D}, \quad (8.26)$$

$$G = -\eta \mathbf{b}^\top \mathbf{Q}^{-1} \mathbf{b} - 2\eta \mathbf{b}^\top \mathbf{Q}^{-1} \mathbf{K} \mathbf{c} - \eta \mathbf{c}^\top \mathbf{K}^\top \mathbf{Q}^{-1} \mathbf{K} \mathbf{c} + 2\mathbf{c}^\top \mathbf{B} \mathbf{c} + 2\mathbf{c}^\top \mathbf{r}_2 + 2r_0. \quad (8.27)$$

Next, we “complete the square” by considering the following quadratic term

$$\begin{aligned} & [\boldsymbol{\theta} - (\mathbf{F}^{-1} \mathbf{f} + \mathbf{F}^{-1} \mathbf{L} \mathbf{c})]^\top \mathbf{F} [\boldsymbol{\theta} - (\mathbf{F}^{-1} \mathbf{f} + \mathbf{F}^{-1} \mathbf{L} \mathbf{c})] \\ &= \boldsymbol{\theta}^\top \mathbf{F} \boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \mathbf{f} - 2\boldsymbol{\theta}^\top \mathbf{L} \mathbf{c} + (\mathbf{F}^{-1} \mathbf{f} + \mathbf{F}^{-1} \mathbf{L} \mathbf{c})^\top \mathbf{F} (\mathbf{F}^{-1} \mathbf{f} + \mathbf{F}^{-1} \mathbf{L} \mathbf{c}) \\ &= (\boldsymbol{\theta}^\top \mathbf{F} \boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \mathbf{f} - 2\boldsymbol{\theta}^\top \mathbf{L} \mathbf{c}) + \mathbf{f}^\top \mathbf{F}^{-1} \mathbf{f} + 2\mathbf{f}^\top \mathbf{F}^{-1} \mathbf{L} \mathbf{c} + \mathbf{c}^\top \mathbf{L}^\top \mathbf{F}^{-1} \mathbf{L} \mathbf{c}. \end{aligned} \quad (8.28)$$

Therefore, we have

$$\begin{aligned} & q(\boldsymbol{\theta} | \mathbf{c})^{\frac{\eta}{\eta + \omega}} \exp\left(\frac{\widehat{R}(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) \\ &= \frac{1}{|2\pi \mathbf{Q}|^{\frac{\eta}{2(\eta + \omega)}}} \exp\left(\frac{1}{2(\eta + \omega)} \left( \mathbf{f}^\top \mathbf{F}^{-1} \mathbf{f} + 2\mathbf{f}^\top \mathbf{F}^{-1} \mathbf{L} \mathbf{c} + \mathbf{c}^\top \mathbf{L}^\top \mathbf{F}^{-1} \mathbf{L} \mathbf{c} \right. \right. \\ & \quad \left. \left. - [\boldsymbol{\theta} - (\mathbf{F}^{-1} \mathbf{f} + \mathbf{F}^{-1} \mathbf{L} \mathbf{c})]^\top \mathbf{F} [\boldsymbol{\theta} - (\mathbf{F}^{-1} \mathbf{f} + \mathbf{F}^{-1} \mathbf{L} \mathbf{c})] \right) + \frac{G}{2(\eta + \omega)}\right). \end{aligned} \quad (8.29)$$

Using the above result, the inner integral term in the dual function is

$$\begin{aligned} & \int q(\boldsymbol{\theta} | \mathbf{c})^{\frac{\eta}{\eta + \omega}} \exp\left(\frac{\widehat{R}(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) d\boldsymbol{\theta} \\ &= \frac{|2\pi \mathbf{F}^{-1}(\eta + \omega)|^{\frac{1}{2}}}{|2\pi \mathbf{Q}|^{\frac{\eta}{2(\eta + \omega)}}} \exp\left(\frac{1}{2(\eta + \omega)} \left( \mathbf{f}^\top \mathbf{F}^{-1} \mathbf{f} + 2\mathbf{f}^\top \mathbf{F}^{-1} \mathbf{L} \mathbf{c} + \mathbf{c}^\top \mathbf{L}^\top \mathbf{F}^{-1} \mathbf{L} \mathbf{c} \right) \right) \\ & \quad \exp\left(\frac{G}{2(\eta + \omega)}\right), \end{aligned} \quad (8.30)$$

where the squared exponential term in Equation (8.29) depending on  $\boldsymbol{\theta}$  is “integrated out” and becomes the inverted normalization term  $|2\pi \mathbf{F}^{-1}(\eta + \omega)|^{\frac{1}{2}}$ . Plugging this term back to the dual function yields

$$\begin{aligned} g(\eta, \omega) &= \eta \epsilon - \omega \beta + \frac{1}{2} \left( \mathbf{f}^\top \mathbf{F}^{-1} \mathbf{f} - \eta \mathbf{b}^\top \mathbf{Q}^{-1} \mathbf{b} \right. \\ & \quad \left. + (\eta + \omega) \log |2\pi \mathbf{F}^{-1}(\eta + \omega)| - \eta \log |2\pi \mathbf{Q}| \right) \\ & \quad + \int \mu(\mathbf{c}) \left( \mathbf{c}^\top \mathbf{m} + \frac{1}{2} \mathbf{c}^\top \mathbf{M} \mathbf{c} \right) d\mathbf{c}, \end{aligned} \quad (8.31)$$

where

$$\mathbf{m} = \mathbf{L}^\top \mathbf{F}^{-1} \mathbf{f} - \eta \mathbf{K}^\top \mathbf{Q}^{-1} \mathbf{b}, \quad (8.32)$$

$$\mathbf{M} = \mathbf{L}^\top \mathbf{F}^{-1} \mathbf{L} - \eta \mathbf{K}^\top \mathbf{Q}^{-1} \mathbf{K}. \quad (8.33)$$

---

**Algorithm 1: C-MORE**

---

**Input:** Parameters  $\epsilon$  and  $\beta$ , initial distribution  $p(\boldsymbol{\theta}|\mathbf{c})$

- 1 **for**  $k = 1, \dots, K$  **do**
- 2     **for**  $n = 1, \dots, N$  **do**
- 3         Observe context  $\mathbf{c}_n \sim \mu(\mathbf{c})$
- 4         Draw parameter  $\boldsymbol{\theta}_n \sim p(\boldsymbol{\theta}|\mathbf{c}_n)$
- 5         Execute task with  $\boldsymbol{\theta}_n$  and receive  $R(\boldsymbol{\theta}_n, \mathbf{c}_n)$
- 6     Learn the quadratic model  $\widehat{R}(\boldsymbol{\theta}, \mathbf{c})$
- 7     Solve  $\operatorname{argmin}_{\eta>0, \omega>0} g(\eta, \omega)$  using Equation (8.31)
- 8     Set new search distribution  $p(\boldsymbol{\theta}|\mathbf{c})$  using Equation (8.34)

---

This dual function can be evaluated much more conveniently. Since the context distribution  $\mu(\mathbf{c})$  is unknown, we approximate the expectation in Equation (8.31) by sample averages. The dual function can be minimized by standard non-linear optimization routines such as IPOPT (Wächter and Biegler, 2006).

Similarly to the dual function, by using Equation (8.15) and Equation (8.29) we compute the new search distribution in closed form as

$$\begin{aligned} p(\boldsymbol{\theta}|\mathbf{c}) &\propto q(\boldsymbol{\theta}|\mathbf{c})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{\widehat{R}(\boldsymbol{\theta}, \mathbf{c})}{\eta + \omega}\right) \\ &= \mathcal{N}\left(\boldsymbol{\theta}|\mathbf{F}^{-1}\mathbf{f} + \mathbf{F}^{-1}\mathbf{L}\mathbf{c}, \mathbf{F}^{-1}(\eta + \omega)\right), \end{aligned} \quad (8.34)$$

where terms independent on  $\boldsymbol{\theta}$  and  $\mathbf{c}$  are subsumed as a normalization constant. To ensure that the covariance  $\mathbf{F}^{-1}(\eta + \omega)$  is positive definite, the matrix  $\mathbf{A}$  of the quadratic model is constrained to be negative definite. C-MORE is summarized in Algorithm 1.

### 8.3.3 Learning the Quadratic Model

The performance of C-MORE depends on the accuracy of the quadratic model. For many problems, the reward function  $R(\boldsymbol{\theta}, \mathbf{c})$  is not quadratic and the quadratic model is not suitable to approximate the entire reward function. However, the reward function is often smooth and it can *locally* be approximated by a quadratic model. Therefore, we locally approximate the reward function by learning a new quadratic model for each policy update. The quadratic model can be learned by regression methods such as ridge regression<sup>1</sup> (Bishop, 2006). However, ridge regression is prone to error when the context is high-dimensional. Below, we address this issue by firstly showing that performing linear dimension reduction on the context variables yields a low-rank matrix of parameters. Secondly, we propose a nuclear norm minimization approach to learn a low-rank matrix without explicitly performing dimension reduction.

### 8.3.4 Dimension Reduction and Low-Rank Representation

Linear dimension reduction learns a low-rank matrix  $\mathbf{W}$  and projects the data onto a lower dimensional subspace. Performing linear dimension reduction on the context

---

<sup>1</sup>After learning the parameters,  $\mathbf{A}$  is enforced to be negative definite by truncating its positive eigenvalues. Subsequently, we re-learn the remainder parameters. An alternative approach is projected gradient descend, but it is more computationally demanding and requires step size tuning.

variables yields the following quadratic model

$$\begin{aligned} \widehat{R}(\boldsymbol{\theta}, \mathbf{c}) &= \boldsymbol{\theta}^\top \mathbf{A} \boldsymbol{\theta} + \mathbf{c}^\top \mathbf{W}^\top \widetilde{\mathbf{B}} \mathbf{W} \mathbf{c} + 2\boldsymbol{\theta}^\top \widetilde{\mathbf{D}} \mathbf{W} \mathbf{c} \\ &\quad + \boldsymbol{\theta}^\top \mathbf{r}_1 + \mathbf{c}^\top \mathbf{W}^\top \widetilde{\mathbf{r}}_2 + r_0, \end{aligned} \quad (8.35)$$

where  $\mathbf{W} \in \mathbb{R}^{d_z \times d_c}$  denotes a rank- $d_z$  matrix with  $d_z < d_c$ . The model parameters  $\mathbf{A}$ ,  $\widetilde{\mathbf{B}}$ ,  $\widetilde{\mathbf{D}}$ ,  $\mathbf{r}_1$ ,  $\widetilde{\mathbf{r}}_2$  and  $r_0$  can be learned by ridge regression. However, the matrix  $\mathbf{B} = \mathbf{W}^\top \widetilde{\mathbf{B}} \mathbf{W}$  is low-rank, i.e.,  $\text{rank}(\mathbf{B}) = d_z < d_c$ . Thus, performing linear dimension reduction on the contexts makes  $\mathbf{B}$  low-rank. In other words, we may perform dimension reduction in a single-step manner by learning the quadratic model such that  $\mathbf{B}$  has low-rank. Note that the rank of  $\mathbf{D} = \widetilde{\mathbf{D}} \mathbf{W}$  depends on  $\boldsymbol{\theta}$  and is problem dependent. Hence, we do not consider the rank of  $\mathbf{D}$  for dimension reduction.

There are several linear dimension reduction methods that can be applied to learn  $\mathbf{W}$ . Principal component analysis (PCA) (Jolliffe, 1986) is a common method used in statistics and machine learning. However, being unsupervised, it does not take the regression targets into account, i.e., the reward. Alternative supervised techniques such as our previous contributions do not take the regression model, i.e., the quadratic model, into account. Moreover, performing dimension reduction before model learning is a multi-step approach and should be avoided.

On the contrary, in projection regression (Friedman and Stuetzle, 1981; Vijayakumar and Schaal, 2000) the model parameters and the projection matrix are learned simultaneously. Although this is a single-step approach, applying this approach to the model in Equation (8.35) requires alternately optimizing for the model parameters and the projection matrix and is computationally expensive.

In the original MORE, Bayesian dimensionality reduction (Gönen, 2013) is applied to perform linear supervised dimension reduction on  $\boldsymbol{\theta}$ , i.e., the algorithm considers a projection  $\mathbf{W}\boldsymbol{\theta}$ . The matrix  $\mathbf{W}$  is sampled from a prior distribution and the algorithm learns the model parameters using weighted average over the sampled  $\mathbf{W}$ . However, for high-dimensional  $\mathbf{W}$ , this approach requires an impractically large amount of samples  $\mathbf{W}$  to obtain an accurate model, leading to computationally expensive updates.

Next, we propose our single-step approach to dimension reduction in this scenario. Our key idea is to learn a low-rank representation via nuclear norm regularization.

### 8.3.5 Learning a Low-Rank Matrix with Nuclear Norm Regularization

The quadratic model in Equation (8.20) can be re-written as

$$\widehat{R}(\mathbf{x}) = \mathbf{x}^\top \mathbf{H} \mathbf{x}, \quad (8.36)$$

where the input vector  $\mathbf{x}$  and the parameter matrix  $\mathbf{H}$  are defined as

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\theta} \\ \mathbf{c} \\ 1 \end{bmatrix}, \quad (8.37)$$

and

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{D} & 0.5\mathbf{r}_1 \\ \mathbf{D}^\top & \mathbf{B} & 0.5\mathbf{r}_2 \\ 0.5\mathbf{r}_1^\top & 0.5\mathbf{r}_2^\top & r_0 \end{bmatrix}. \quad (8.38)$$

Note that  $\mathbf{H}$  is symmetric since both  $\mathbf{A}$  and  $\mathbf{B}$  are symmetric. As discussed in the previous section, we desire  $\mathbf{B}$  to be low-rank. Unlike Equation (8.35), we do not consider dimension reduction for the linear terms in  $\mathbf{c}$ , i.e.,  $2\boldsymbol{\theta}^\top \mathbf{D}\mathbf{c}$  and  $\mathbf{c}^\top \mathbf{r}_2$ . Instead, we learn  $\mathbf{H}$  by solving the following convex optimization problem

$$\begin{aligned} \min_{\mathbf{H}} [\mathcal{J}(\mathbf{H}) + \lambda_* \|\mathbf{B}\|_*], \\ \text{s.t. } \mathbf{A} \text{ is negative definite,} \end{aligned} \quad (8.39)$$

where  $\mathcal{J}(\mathbf{H})$  denotes the differentiable part

$$\mathcal{J}(\mathbf{H}) = \frac{1}{2N} \sum_{n=1}^N (\mathbf{x}_n^\top \mathbf{H} \mathbf{x}_n - R(\boldsymbol{\theta}_n, \mathbf{c}_n))^2 + \frac{\lambda}{2} \|\mathbf{H}\|_F^2, \quad (8.40)$$

where  $\lambda > 0$  and  $\lambda_* > 0$  are regularization parameters. The Frobenius norm  $\|\cdot\|_F$  is defined as  $\|\mathbf{H}\|_F = \sqrt{\text{tr}(\mathbf{H}\mathbf{H}^\top)}$ . The nuclear norm of a matrix  $\|\cdot\|_*$  is defined as the  $\ell_1$ -norm of its singular values. This optimization problem can be explained as follows. The term  $\mathcal{J}(\mathbf{H})$  consists of the mean squared error and the  $\ell_2$ -regularization term. Thus, minimizing  $\mathcal{J}(\mathbf{H})$  corresponds to ridge regression. Minimizing the nuclear norm  $\|\mathbf{B}\|_*$  shrinks the singular values of  $\mathbf{B}$ . Thus, the solution tends to have sparse singular values and to be low-rank. The negative definite constraint further ensures that the covariance matrix in Equation (8.34) is positive definite.

The convexity of this optimization problem can be verified by checking the following conditions. First, the convexity of the mean squared error can be proven following Boyd and Vandenberghe (2004) (page 74). Let  $g(t) = \widehat{\mathcal{J}}(\mathbf{Z} + t\mathbf{V})$  be the mean squared error and  $\mathbf{Z}$  and  $\mathbf{V}$  are symmetric matrices. Then we have that  $\nabla^2 g(t) = \frac{1}{N} \sum (\mathbf{x}_n^\top \mathbf{V} \mathbf{x}_n)^2 \geq 0$ . Thus, the mean squared error is convex. Since the Frobenius norm is convex,  $\mathcal{J}(\mathbf{H})$  is convex as well. Second, a set of negative definite matrices is convex since  $\mathbf{y}^\top (a\mathbf{X} + (1-a)\mathbf{Y})\mathbf{y} < 0$  for any negative definite matrices  $\mathbf{X}$  and  $\mathbf{Y}$ ,  $0 \leq a \leq 1$ , and any vector  $\mathbf{y}$  (Boyd and Vandenberghe, 2004). Third, the nuclear norm is a convex function (Recht et al., 2010). Note that, since the gradient  $\nabla \mathcal{J}(\mathbf{H})$  is symmetric,  $\mathbf{H}$  is guaranteed to be symmetric as well given that the initial solution is also symmetric.

It is also possible to enforce the matrix  $\mathbf{H}$  (rather than  $\mathbf{B}$ ) to be low-rank, implying that both  $\boldsymbol{\theta}$  and  $\mathbf{c}$  can be projected onto a common low-dimensional subspace. However, this is often not the case, and regularizing by the nuclear norm of  $\mathbf{H}$  did not perform well in our experiments. We may also directly constrain  $\text{rank}(\mathbf{B}) = d_z$  in Equation (8.39) instead of performing nuclear norm regularization. However, minimization problems with rank constraints are NP-hard. On the contrary, the nuclear norm is the convex envelop of the rank function and can be optimized more efficiently (Recht et al., 2010). For this reason, the nuclear norm has been a popular surrogate to a low-rank constraint in many applications, such as matrix completion (Candès and Tao, 2010) and multi-task learning (Pong et al., 2010).

Since the optimization problem in Equation (8.39) is convex, any convex optimization method can be used (Boyd and Vandenberghe, 2004). For our experiments, we use the *accelerated proximal gradient descend (APG)* (Toh and Yun, 2009). The pseudocode of our implementation of APG for solving Equation (8.39) is given in Algorithm 2. Note that APG requires computing the SVD of the matrix  $\mathbf{B}$ . Since computing the exact SVD of a high-dimensional matrix can be computationally expensive, we approximate it by randomized SVD (Halko et al., 2011).

---

**Algorithm 2:** APG for solving the nuclear norm minimization problem

---

**Input:** Parameters  $\lambda$  and  $\lambda_*$ , gradient step size  $\tau$ , maximum number of iteration  $K$ , initial solution  $\mathbf{H}_0$

1 Initialize  $\mathbf{H}_{-1} = \mathbf{H}_0$  and  $t_{-1} = t_0 = 1$

2 **for**  $k = 1, \dots, K$  **do**

3     Set intermediate point

$$\mathbf{Y}_k = \mathbf{H}_k + \frac{t_{k-1} - 1}{t_k} (\mathbf{H}_k - \mathbf{H}_{k-1})$$

4     Do gradient descent using the differentiable term

$$\mathbf{Y}_+ = \mathbf{Y}_k + \tau \nabla \mathcal{J}(\mathbf{Y}_k),$$

where

$$\mathbf{Y}_+ = \begin{bmatrix} \mathbf{A}_+ & \mathbf{D}_+ & 0.5\mathbf{r}_{1+} \\ \mathbf{D}_+^\top & \mathbf{B}_+ & 0.5\mathbf{r}_{2+} \\ 0.5\mathbf{r}_{1+}^\top & 0.5\mathbf{r}_{2+}^\top & r_{0+} \end{bmatrix}$$

5     Shrink singular values of  $\mathbf{B}_+$

$$\mathbf{B}_* = \mathbf{U} \max(\boldsymbol{\Sigma} - \lambda_* \mathbf{I}, \mathbf{0}) \mathbf{V}^\top,$$

where  $\mathbf{B}_+ = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top$  is the SVD of  $\mathbf{B}_+$

6     Truncate positive eigenvalues of  $\mathbf{A}_+$

$$\mathbf{A}_* = \mathbf{P} \min(\boldsymbol{\Lambda}, \mathbf{0}) \mathbf{P}^\top,$$

where  $\mathbf{A}_+ = \mathbf{P} \boldsymbol{\Lambda} \mathbf{P}^\top$  is the eigendecomposition of  $\mathbf{A}_+$

7     Update solution

$$\mathbf{H}_{k+1} = \begin{bmatrix} \mathbf{A}_* & \mathbf{D}_+ & 0.5\mathbf{r}_{1+} \\ \mathbf{D}_+^\top & \mathbf{B}_* & 0.5\mathbf{r}_{2+} \\ 0.5\mathbf{r}_{1+}^\top & 0.5\mathbf{r}_{2+}^\top & r_{0+} \end{bmatrix}$$

8     Update parameter  $t_{k+1} = \frac{1 + \sqrt{1 + 4(t_k)^2}}{2}$

9     **if** *stopping criterion is met* **then**

10     | **return**

---

## 8.4 Experiment

We evaluate the proposed method on three problems. We start by studying C-MORE behavior in a scenario where we know the true reward model and the true low-dimensional context. Subsequently, we focus our attention on two simulated robotic ball hitting tasks. In the first task, a toy 2-DoF planar robot arm has to hit a ball placed on a plane. In the second task, a simulated 6-DoF robot arm has to hit a ball placed in a three-dimensional space. In both cases, the robots accomplish their task by using raw camera images as context variables. However, in the latter case we have limited data and therefore data efficiency is of primary importance.

The evaluation is performed on three different versions of C-MORE, according to the model learning approach: using only ridge regression (*C-MORE Ridge*), aided by a low-dimensional context variables learned by PCA (*C-MORE Ridge+PCA*) and aided by nuclear norm regularization (*C-MORE Nuc. Norm*). We also use *C-REPS* (Deisenroth et al., 2013) with PCA as baseline. For the ball hitting task with 2-DoF robot arm, we additionally evaluate C-MORE with model learned by LASSO (*C-MORE LASSO*), and ridge regression with low-dimensional context variables learned by supervised PCA (Li et al., 2016) (*C-MORE Ridge+SuPCA*). We also tried to pre-process the context space with an autoencoder. However, the learned representation performed poorly, possibly due to the limited amount of data, and therefore this method is not reported.

For each case study, first, the experiments are presented and then the results are reported and discussed.

### 8.4.1 Quadratic Cost Function Optimization

In the first experiment, we want to study the performance of the algorithms in a setup where we are able to analytically compute both the reward and the true low-dimensional context. To this aim, we define the following problem

$$\begin{aligned} R(\boldsymbol{\theta}, \mathbf{c}) &= -(\|\boldsymbol{\theta} - \mathbf{T}_1 \tilde{\mathbf{c}}\|_2)^2, \\ \tilde{\mathbf{c}} &= \tilde{\mathbf{I}} \mathbf{T}_2^{-1} \mathbf{c}, \end{aligned} \quad (8.41)$$

where  $\mathbf{T}_2 \in \mathbb{R}^{d_c \times d_c}$ ,  $d_{\tilde{\mathbf{c}}} < d_c$ ,  $\tilde{\mathbf{I}} \in \mathbb{R}^{d_{\tilde{\mathbf{c}}} \times d_c}$  is a rectangular matrix with ones in its main diagonal and zeros otherwise,  $\tilde{\mathbf{c}}$  is the true low-dimensional context, and  $\mathbf{T}_1 \in \mathbb{R}^{d_{\boldsymbol{\theta}} \times d_{\tilde{\mathbf{c}}}}$  is to match the dimension of the true context and the parameter  $\boldsymbol{\theta}$  in order to compute the reward. This setup is particularly interesting because only a subset of the observed context influences the reward. First, the observed context  $\mathbf{c}$  is linearly transformed by  $\mathbf{T}_2^{-1}$ . Subsequently, thanks to the matrix  $\tilde{\mathbf{I}}$ , only the first  $d_{\tilde{\mathbf{c}}}$  elements are kept to compose the true context, while the remainder is treated as noise. Finally, the reward is computed by linearly transforming the true context by  $\mathbf{T}_1$ .

We set  $d_{\tilde{\mathbf{c}}} = 3$ ,  $d_{\boldsymbol{\theta}} = 10$ ,  $d_c = 25$ , while the elements of  $\mathbf{T}_1, \mathbf{T}_2$  are chosen uniformly randomly in  $[0, 1]$ . The sampling Gaussian distribution is initialized with random mean and covariance  $\mathbf{Q} = 10,000\mathbf{I}$ . For learning, we collect 35 new data points and keeps track of the data collected during the last 20 iterations to stabilize the policy update. The evaluation is performed at each iteration over 1,000 contexts. Each context element is drawn from a uniform random distribution in  $[-10, 10]$ . Since we can generate a large amount of data in this setting, we pre-train PCA using 10,000 random context data points and fixed the dimensionality to  $d_z = 20$  (chosen by cross-validation). The cross-validation results are given in Figure 8.3. It shows that the

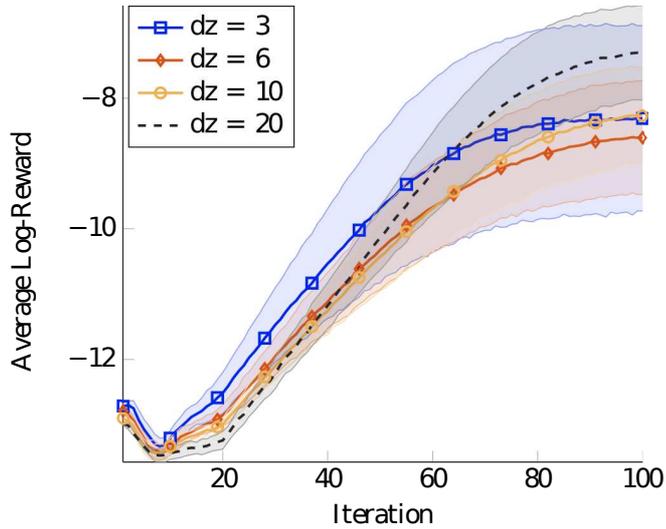


Figure 8.3: Average reward for the quadratic cost function problem with C-MORE PCA using different dimensionality  $d_z$ . Shaded area denotes standard deviation (results are averaged over ten trials). Results do not differ much, since PCA always fails in reducing the dimensionality of the context variables. Clearly, the more principal components we keep, the better results we obtain.

target dimensionality  $d_z$  does not affect the result by much since every direction of the context variables have identical variance. Clearly, the more directions we keep, the better results we obtain by PCA.

The learning is performed for a maximum of 100 iterations. If the KL divergence is lower than 0.1, then the learning is considered to be converged and the policy is not updated anymore.

As shown in Figure 8.4, C-MORE Nuc. Norm clearly outperforms all the competitors, learning an almost optimal policy and being the only one to converge within the maximum number of iterations. It is also the only algorithm correctly learning the true context dimensionality, as nuclear norm successfully regularizes  $\mathbf{B}$  to have rank three. On the contrary, PCA does not help C-MORE much and yields only slightly better results than plain ridge regression. PCA cannot in fact determine task-relevant dimensions as non-relevant dimensions have equally-high variance.

#### 8.4.2 Ball Hitting with a 2-DoF Robot Arm

In this task, a simulated planar robot arm (Figure 8.6) has to hit a green virtual ball placed on RGB camera images of size  $32 \times 24$ . The observed pixels define the context, for a total of 2304 context variables. The ball is randomly and uniformly placed in the robot workspace. Noise drawn from a uniform random distribution in  $[-30, 30]$  is added to the context to simulate different light conditions. The robot controls the joint accelerations at each time step by a linear-in-parameter controller with Gaussian basis functions, for a total of 32 parameters  $\theta$  to be learned. The reward  $R(\theta, c)$  is the negative cumulative joint accelerations plus the negative distance between the end-effector and the ball at the final time step.

For learning, the agent collects 50 data points at each iteration and keeps data points from the last four previous iterations. The evaluation is performed at each iteration over 500 contexts. Pixel values are normalized in  $[-1, 1]$ . The sampling

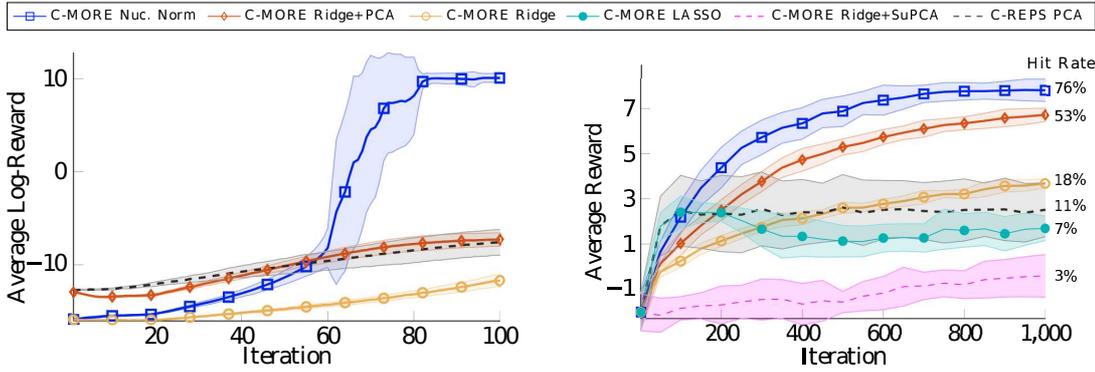


Figure 8.4: Average reward for the quadratic cost function problem. Shaded area denotes standard deviation (results are averaged over ten trials). Only C-MORE Nuc. Norm converges within 100 iterations to an almost optimal policy.

Figure 8.5: Averaged reward for the 2-DoF hitting task. C-REPS outperforms C-MORE early on. However, it prematurely converges to suboptimal solutions, while C-MORE continues to improve and soon outperforms C-REPS.

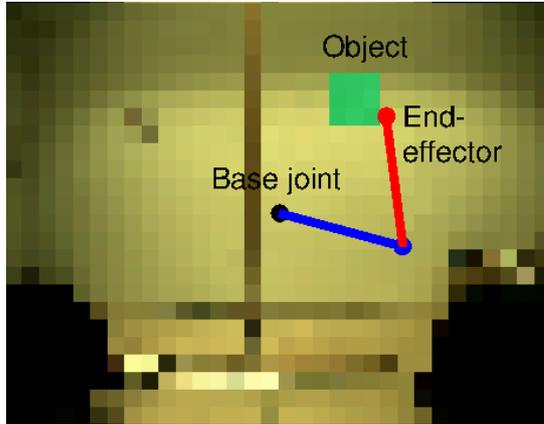


Figure 8.6: 2-DoF hitting task. The robot (blue and red lines) observes the context which consists of a virtual green ball and the background image.

Gaussian distribution is initialized with random mean and identity covariance. For C-MORE Nuc. Norm, C-MORE LASSO and C-MORE PCA, we perform 5-fold cross-validation every 100 policy updates to choose the values of regularization parameter for nuclear norm, regularization parameter for  $\ell_1$  norm, and dimension  $d_z$ , respectively. The decision is based on the mean squared error between the collected returns and the model-predicted ones. Due to high computation time of C-MORE SuPCA for high values of  $d_z$ , we tried different values of  $d_z \in \{5, 7, 10\}$  and selected  $d_z = 10$  which gave the best result<sup>2</sup>. Similarly for C-REPS PCA, we tried different values of  $d_z \in \{10, 20, 30, 40\}$  and selected  $d_z = 10$  which gave the best result.

Figure 8.5 shows the averaged reward against the number of iterations. Once again, C-MORE aided by nuclear norm regularization performs the best, achieving the highest average reward. At the 1000th iteration, the learned controller hits the ball with 76% accuracy. The rank of its learned matrix  $\mathbf{B}$  is approximately 31, which shows that the algorithm successfully learns a low-rank model representation. The

<sup>2</sup>SuPCA with  $d_z = 15$  took approximately 5 minutes/iteration.

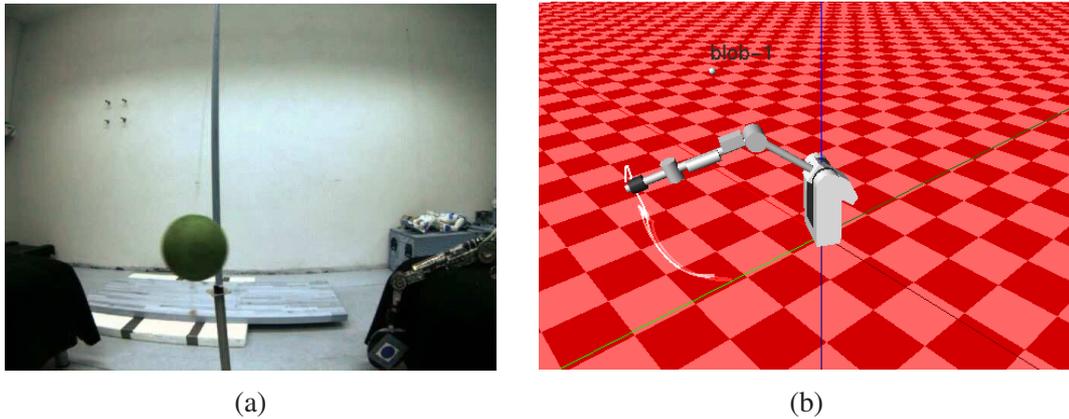


Figure 8.7: The 6-DoF robot as seen from the camera (Figure 8.7(a), bottom right) and in simulation (Figure 8.7(b)). The goal is to control the robot to hit the green ball according to camera images, resized to  $32 \times 24$ .

model learned by LASSO performs very poorly and it is even outperformed by plain ridge regression. However, this is unsurprising since the context variables are highly correlated and LASSO is known to not work well for such variables. On the contrary, preprocessing the context space through PCA still helps C-MORE (the rank of its learned  $B$  is approximately 25), but yields poor results for C-REPS, which suffers of premature convergence. Lastly, preprocessing the context space through SuPCA does not seem work well. This may be due to  $d_z$ , which could be too small for this task.

### 8.4.3 Ball Hitting with a 6-DoF Robot Arm

Similarly to the previous task, here a 6-DoF robotic arm has to hit a ball placed on a three-dimensional space, as shown in Figure 8.7. The context is once again defined by the vectorized pixels of RGB images of size  $32 \times 24$ , for a total of 2304 context variables. Note that Figure 8.7(a) shows an image before we rescale it to size  $32 \times 24$ . However, unlike the 2-DoF task, the ball is directly recorded by a real camera placed near the physical robot, and it is not virtually generated on the images. Furthermore, the robot is controlled by dynamic motor primitives (Ijspeert et al., 2002a) (DMPs), which are non-linear dynamical systems. We use one DMP per joint, with five basis functions per DMP. We also learn the goal attractor of the DMPs, for a total of 36 parameters  $\theta$  to be learned. The reward  $R(\theta, c)$  is computed as the negative cumulative joint accelerations and minimum distance between the end-effector and the ball as well.

The image dataset is collected by taking pictures with the ball placed at 50 different positions. To increase the number of data points, we add a uniform random noise in  $[-30, 30]$  to the context to simulate different light conditions. Therefore, although some data points determine the same ball position, they are considered different due to the added noise. The search distribution is initialized by imitation learning using 50 demonstration data points. For learning, the agent collects 50 data points at each iteration and always keeps data points from the last four previous iterations.

We only evaluate C-MORE with nuclear norm and PCA since they performed well in the previous evaluation. Figure 8.8 shows that nuclear norm again outperforms PCA. At the 500th iteration, the robot hits the ball with 80% accuracy. Considering that the robot is not able to hit the ball in some contexts due to physical constraints and

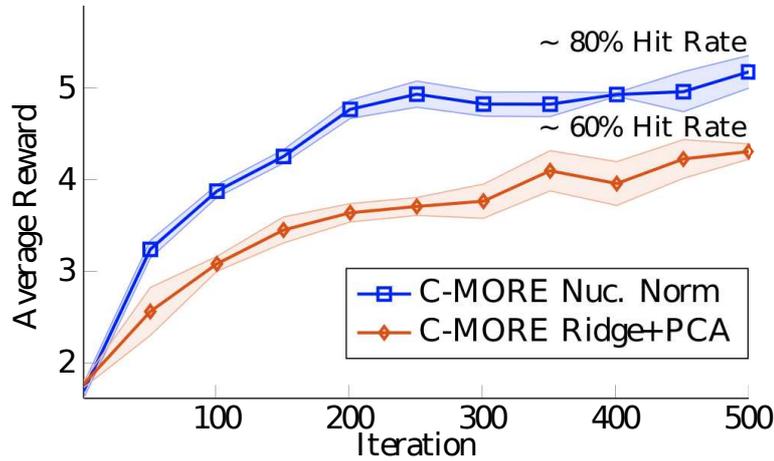


Figure 8.8: 6-DoF hitting task results (averaged over three trials). Nuclear norm regularization outperforms PCA, both in terms of reward and accuracy.

can achieve a maximum accuracy of 90%, this accuracy is impressive for the task. The averaged rank of matrix  $B$  learned by the nuclear norm is approximately 25, which shows that minimizing the nuclear norm successfully learns a low-rank matrix. For PCA, the averaged rank of  $B$  is approximately 30.

## 8.5 Conclusion

Learning with high-dimensional context variables is a challenging and prominent problem in reinforcement learning. In this chapter, we proposed C-MORE, a novel contextual policy search method with single-step dimension reduction. C-MORE learns a reward model that is locally quadratic in the policy parameters and the context variables. By enforcing the model representation to be low-rank, we perform supervised linear dimension reduction in a single-step manner. Unlike existing techniques relying on non-convex formulations, the nuclear norm allows us to learn the low-rank representation by solving a *convex* optimization problem, thus guaranteeing convergence to a global optimum.

The main disadvantage of the proposed method is that it demands more computation time due to the nuclear norm regularization. Although we did not encounter severe problems in our experiments, for very large dimensional tasks this issue can be mitigated by using more efficient optimization techniques, such as active subspace selection (Hsieh and Olsen, 2014).

# Chapter 9

## Conclusion and Future Work

In this chapter, we conclude our contributions in this dissertation and then close this dissertation with discussions on future research directions.

### 9.1 Conclusion

Machine learning has recently gain a lot of interest as a tool for solving complex problems whose traditional methods cannot solve them well. However, one of the most prominent challenges in machine learning is to learn from high-dimensional data. This dissertation contributes to developing and utilizing *single-step* dimension reduction to solve high-dimensional machine learning problems especially reinforcement learning problems.

We presented five contributions in this dissertation and they are divided into two parts. The first part of this dissertation focused on the development of single-step linear dimension reduction methods. We presented two contributions in this first part and they are summarized below.

- **Dimension reduction via single-step estimation of the derivative of quadratic mutual information**

In Chapter 3, we interested in performing linear dimension reduction by maximizing the *quadratic mutual information* (QMI) which is a robust statistical dependence measure. Unlike an existing multi-step approach, our QMI-based linear dimension reduction method directly estimates the derivative of QMI without estimating the QMI itself. The experimental evaluations on artificial and benchmark data showed that the proposed method performs better than existing methods in the presence of outliers.

- **Single-step dimension reduction for conditional density estimation**

In Chapter 4, we focused on solving high-dimensional conditional density estimation problems. We proposed the *least-squares conditional entropy* (LSCE) method which simultaneously performs dimension reduction and conditional density estimation in an integrated manner. We evaluated our proposed method on artificial data, benchmark data, humanoid robot data, and computer art data. Experimental results showed that LSCE gives more accurate estimated conditional densities than existing methods based on a multi-step approach.

The second part of this dissertation focused on utilizing single-step dimension reduction to solve high-dimensional reinforcement learning problems. We presented three contributions in this second part and they are summarized below.

- **Model-based policy gradient with parameter-based exploration**

In Chapter 6, we focused on improving data efficiency of reinforcement learning. We proposed the *model-based policy gradient with parameter-based exploration* (M-PGPE) method which learns a transition model by least-squares conditional density estimation (LSCDE) (Sugiyama et al., 2010). We firstly showed through benchmark experiments that LSCDE is a more flexible method for transition model learning than an existing method. Then, through experiments on a simulated humanoid robot we showed that M-PGPE gives better performances than existing model-free reinforcement learning methods when the budget for collecting data is limited.

- **Dimension reduction for model-based policy gradient with parameter-based exploration**

In Chapter 7, we improved the performance of M-PGPE in high-dimensional reinforcement learning problems by using our LSCE method to learn a transition model. Experimental evaluations on benchmark problems and real humanoid robot control problems showed that M-PGPE with LSCE performs better than M-PGPE with naive multi-step combinations of LSCDE and existing dimension reduction methods.

- **Contextual Policy Search with Single-step Dimension Reduction**

In Chapter 8, we focused on contextual reinforcement learning problems with high-dimensional contexts. We proposed a novel method called *contextual model-based relative entropy stochastic search* (C-MORE) which finds optimal policies based on a learned quadratic model. Then, we proposed to learn a low-rank representation of the model which corresponds to simultaneously perform dimension reduction and model learning. We evaluated C-MORE on a benchmark problem and robot ball hitting problems based on camera images. The results showed that our single-step approach to dimension reduction in model learning performs better than multi-step approaches.

From the experimental evaluations, we conclude that single-step dimension reduction is a promising approach to tackle high-dimensional machine learning problems and should be further investigated in the future.

## 9.2 Future Work

In this dissertation, we focused on an important issue of learning from high-dimensional data, and we have approaches which effectively alleviate this issue. However, our contributions still have rooms for improvement. Moreover, machine learning still faces with many challenges which should be overcome in order to make machine learning more practical.

Below, we discuss approaches which can further improve our contributions as well as new research directions which we will pursue in our future research.

- **More applications of the direct estimation of the derivative of QMI**

In Chapter 3 we focused on maximizing statistical dependence to perform dimension reduction. On the other hand, researchers have shown that problems such as *image registration* (Atif et al., 2003; Pluim et al., 2003) and *independent component analysis* (Hyvärinen and Oja, 2000; Suzuki and Sugiyama, 2011)

can be solved by maximizing or minimizing a statistical dependence as well. However, existing methods solve these problems through a multi-step approach which first estimates a statistical dependence measure from data and then computes the derivative of the estimated measure in order to find a maximizer. As we have shown, such a multi-step approach is not appropriate. In our future work, we will utilize the QMI derivative estimation method we presented in Chapter 3 to solve these problems. We expect that a single-step approach based on our derivative estimation method would work better than existing multi-step methods.

- **Dimension reduction based on other divergences**

Dimension reduction can be performed by maximizing a statistical dependence measure which is defined based on a divergence of two probability distributions. In Chapter 3, we focused on developing QMI-based dimension reduction method mainly because we are interested in the robustness property of QMI. However, in the statistics literature there are many divergences such as the *Hellinger distance* (Hellinger, 1909) which have interesting properties but their usefulness for dimension reduction has not been fully studied and demonstrated. In our future work, we will firstly explore these divergences to study their properties for dimension reduction and then develop dimension reduction methods based on them.

- **Direct estimation of the derivative of SCE**

LSCE is a single-step method when the aim is to perform dimension reduction for conditional density estimation. However, if the aim is only to perform dimension reduction, LSCE is in fact a multi-step method since it firstly estimates SCE from data and then computes derivative of the estimated SCE. Therefore, an ideal single-step method to perform dimension reduction for conditional density estimation via SCE is a method which directly estimates the derivatives of SCE while simultaneously performs conditional density estimation. Investigating such a single-step method is one of our future work.

- **Dimension reduction in model-free reinforcement learning**

In this dissertation, we focused on model-based reinforcement learning and proposed to incorporate dimension reduction into a model learning step in order to mitigate the curse of dimensionality. On the other hand, model-free reinforcement learning still suffers from the curse of dimensionality. Researchers have shown that applying dimension reduction to model-free reinforcement learning can greatly improve the performance (Morimoto et al., 2008; Bitzer et al., 2010; Hachiya and Sugiyama, 2010). However, existing approaches independently perform dimension reduction and reinforcement learning, and as we have shown in this dissertation, such a multi-step approach is not appropriate. A better alternative would be to directly integrate dimension reduction into reinforcement learning. For example, for direct policy search approach we may include an  $\ell_1$  regularization into the expected return in order to simultaneously perform feature selection on the policy parameter space while learning the optimal policy parameter. We will further investigate such an integrated approach in our future work.

We believe that these future works are worth investigating and would be beneficial to future machine learning research.

## References

- Abdolmaleki, A., Lioutikov, R., Peters, J., Lau, N., Reis, L. P., and Neumann, G. (2015). Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems 28*, pages 3537–3545.
- Abe, N., Melville, P., Pendus, C., Reddy, C. K., Jensen, D. L., Thomas, V. P., Bennett, J. J., Anderson, G. F., Cooley, B. R., Kowalczyk, M., Domick, M., and Gardinier, T. (2010). Optimizing debt collections using constrained reinforcement learning. In *The 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 75–84.
- Absil, P.-A., Mahony, R., and Sepulchre, R. (2008). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ.
- Agakov, F. V. and Barber, D. (2005). Kernelized infomax clustering. In *Advances in Neural Information Processing Systems 18*, pages 17–24, Cambridge, MA, USA. MIT Press.
- Ali, S. M. and Silvey, S. D. (1966). A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, 28(1):131–142.
- Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.
- Argall, B., Chernova, S., Veloso, M. M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.
- Armijo, L. (1966). Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal Mathematics*, 16(1):1–3.
- Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.
- Atif, J., Ripoche, X., Coussinet, C., and Osorio, A. (2003). Non rigid medical image registration based on the maximization of quadratic mutual information. In *IEEE 29th Bioengineering Conference*, pages 71–72.
- Atkeson, C. G., Moore, A. W., and Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11(1):11–73.
- Au, T., Chin, M.-L. I., and Ma, G. (2010). *Mining Rare Events Data by Sampling and Boosting: A Case Study*, pages 373–379. Springer Berlin Heidelberg, Berlin, Heidelberg.

- Bach, F. R. and Jordan, M. I. (2002). Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48.
- Bach, F. R. and Jordan, M. I. (2003). Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48.
- Bache, K. and Lichman, M. (2013). UCI machine learning repository.
- Bai, Z., Demmel, J., Dongarra, J., Ruhe, A., and van der Vorst, H. (2000). *Templates for the Solution of Algebraic Eigenvalue Problems*. Society for Industrial and Applied Mathematics.
- Basu, A., Harris, I. R., Hjort, N. L., and Jones, M. C. (1998). Robust and efficient estimation by minimising a density power divergence. *Biometrika*, 85(3).
- Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.
- Bellman, R. E. (1957a). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition.
- Bellman, R. E. (1957b). A markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684.
- Bellman, R. E. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press.
- Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.
- Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.
- Bernard-Michel, C., Gardes, L., and Girard, S. (2009). Gaussian regularized sliced inverse regression. *Statistics and Computing*, 19(1):85–98.
- Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, 1st edition.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- Bitzer, S., Howard, M., and Vijayakumar, S. (2010). Using dimensionality reduction to exploit constraints in reinforcement learning. In *The IEEE International Conference on Intelligent Robotics Systems*, pages 3219–3225.
- Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. (2014). Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459.
- Boutillier, C., Dearden, R., and Goldszmidt, M. (1995). Exploiting structure in policy construction. In *The 14th International Joint Conference on Artificial Intelligence*, pages 1104–1113.

- Boyan, J. A. and Moore, A. W. (1994). Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, pages 369–376.
- Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.
- Brown, G. (2009). A new perspective for information theoretic feature selection. In *The 12th International Conference on Artificial Intelligence and Statistics*, volume 5, pages 49–56. *Journal of Machine Learning Research - Proceedings Track*.
- Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM*, 58(3):11:1–11:37.
- Candès, E. J. and Tao, T. (2010). The power of convex relaxation: near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080.
- Castelvecchi, D. (2016). Can we open the black box of AI? *Nature*, 538(7623):20–23.
- Cawley, G. C. and Talbot, N. L. C. (2007). Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861.
- Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Cheng, G., Hyon, S., Morimoto, J., Ude, A., Joshua, G., Colvin, G., Scroggin, W., and Stephen, C. J. (2007). Cb: A humanoid research platform for exploring neuroscience. *Advanced Robotics*, 21(10):1097–1114.
- Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610.
- Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *The 25th International Conference on Machine Learning*, pages 160–167.
- Cook, R. D. (2000). SAVE: a method for dimension reduction and graphics in regression. *Communications in Statistics - Theory and Methods*, 29(9-10):2109–2121.
- Cook, R. D. and Ni, L. (2005). Sufficient dimension reduction via inverse regression. *Journal of the American Statistical Association*, 100(470):410–428.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.
- Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA.
- Csiszár, I. (1967). Information-type Measures of Difference of Probability Distributions and Indirect Observations. *Studia Scientiarum Mathematicarum Hungarica*, 2:299–318.

- Cunningham, J. P. and Ghahramani, Z. (2015). Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1):2859–2900.
- da Silva, B. C., Konidaris, G., and Barto, A. G. (2012). Learning parameterized skills. In *The 29th International Conference on Machine Learning*.
- Dayan, P. and Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Comput.*, 9(2):271–278.
- Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5(2):142–150.
- Dearden, R., Friedman, N., and Russell, S. (1998). Bayesian q-learning. In *Proceedings of the 15th International Conference on Artificial Intelligence*, pages 761–768.
- Deisenroth, M. P., Neumann, G., and Peters, J. (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142.
- Deisenroth, M. P. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *The 28th International Conference on Machine Learning*, pages 465–472. Omnipress.
- Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.
- Edelman, A., Arias, T. A., and Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353.
- Faivishevsky, L. and Goldberger, J. (2012). Dimensionality reduction based on non-parametric mutual information. *Neurocomputing*, 80:31–37.
- Falk, J. E. (1967). Lagrange multipliers and nonlinear programming. *Journal of Mathematical Analysis and Applications*, 19(1):141 – 159.
- Fan, J., Yao, Q., and Tong, H. (1996). Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika*, 83(1).
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188.
- Fodor, I. (2002). A survey of dimension reduction techniques. Technical report.
- Fox, D., Burgard, W., and Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11.
- Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175.
- Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823.

- Friedman, N. and Russell, S. (1997). Image segmentation in video sequences: A probabilistic approach. In *The 13th Conference on Uncertainty in Artificial Intelligence*, pages 175–181.
- Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5:73–99.
- Fukumizu, K., Bach, F. R., and Jordan, M. I. (2009). Kernel dimension reduction in regression. *The Annals of Statistics*, 37(4):1871–1905.
- Fukumizu, K. and Leng, C. (2012). Gradient-based kernel method for feature extraction and variable selection. In *Advances in Neural Information Processing Systems 25*, pages 2123–2131.
- Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition (2nd Edition)*. Academic Press Professional, Inc., San Diego, CA, USA.
- Galton, F. (1886). Regression Towards Mediocrity in Hereditary Stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263.
- Gao, S., Steeg, G. V., and Galstyan, A. (2015). Efficient estimation of mutual information for strongly dependent variables. In *The 18th International Conference on Artificial Intelligence and Statistics*.
- Gomes, R., Krause, A., and Perona, P. (2010). Discriminative clustering by regularized information maximization. In *Advances in Neural Information Processing Systems 23*, pages 775–783.
- Gönen, M. (2013). Bayesian supervised dimensionality reduction. *IEEE Transactions on Cybernetics*, 43(6):2179–2189.
- Gretton, A., Herbrich, R., Smola, A., Bousquet, O., and Schölkopf, B. (2005). Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6:2075–2129.
- Guestrin, C., Patrascu, R., and Schuurmans, D. (2002). Algorithm-directed exploration for model-based reinforcement learning in factored MDPs. In *The 19th International Conference on Machine Learning*, pages 235–242.
- Hachiyama, H., Peters, J., and Sugiyama, M. (2011a). Reward-weighted regression with sample reuse for direct policy search in reinforcement learning. *Neural Computation*, 23(11):2798–2832.
- Hachiyama, H., Peters, J., and Sugiyama, M. (2011b). Reward-weighted regression with sample reuse for direct policy search in reinforcement learning. *Neural Computation*, 23(11):2798–2832.
- Hachiyama, H. and Sugiyama, M. (2010). Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *Machine Learning and Knowledge Discovery in Databases, Part I, the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*, pages 474–489.

- Halko, N., Martinsson, P., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288.
- Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18.
- Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, New York, NY, USA.
- He, X. and Niyogi, P. (2003). Locality preserving projections. In *Advances in Neural Information Processing Systems 16*. MIT Press.
- Hellinger, E. (1909). Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271.
- Hilas, C. S. and Mastorocostas, P. A. (2008). An application of supervised and unsupervised learning approaches to telecommunications fraud detection. *Knowledge-Based Systems*, 21(7):721–726.
- Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.
- Hsieh, C. and Olsen, P. A. (2014). Nuclear norm minimization via active subspace selection. In *The 31st International Conference on Machine Learning*, pages 575–583.
- Hyvärinen, A. and Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4-5):411–430.
- Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002a). Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems 15*, pages 1523–1530.
- Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002b). Movement imitation with non-linear dynamical systems in humanoid robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA*, pages 1398–1403.
- Ioffe, S. (2006). *Probabilistic Linear Discriminant Analysis*.
- Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201.
- Jensen, J. L. W. V. (1906). Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1):175–193.
- Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *The 10th European Conference on Machine Learning, ECML '98*, pages 137–142.
- Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer Verlag.

- Kasube, H. E. (1983). A technique for integration by parts. *The American Mathematical Monthly*, 90(3):pp. 210–211.
- Ko, J., Klein, D. J., Fox, D., and Hähnel, D. (2007). Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *2007 IEEE International Conference on Robotics and Automation, ICRA*, pages 742–747.
- Kober, J., Bagnell, J. A., and Peters, J. (2013a). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*.
- Kober, J., Bagnell, J. A., and Peters, J. (2013b). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274.
- Kober, J., Oztop, E., and Peters, J. (2011). Reinforcement learning to adjust robot movements to new situations. In *The 22nd International Joint Conference on Artificial Intelligence*, pages 2650–2655.
- Kober, J. and Peters, J. (2011). Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2):171–203.
- Kormushev, P., Calinon, S., and Caldwell, D. G. (2013). Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148.
- Kraskov, A., Stögbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Physical Review E*, 69:066138.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114.
- Kroon, M. and Whiteson, S. (2009). Automatic feature selection for model-based reinforcement learning in factored MDPs. In *International Conference on Machine Learning and Applications, ICMLA*, pages 324–330.
- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.
- Kupcsik, A. G., Deisenroth, M. P., Peters, J., and Neumann, G. (2013). Data-efficient generalization of robot skills with contextual policy search. In *The 27th AAAI Conference on Artificial Intelligence*.
- Lagoudakis, M. G. and Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149.
- Lee, J. A. and Verleysen, M. (2007). *Nonlinear Dimensionality Reduction*. Springer Publishing Company, Incorporated, 1st edition.
- Li, B. and Wang, S. (2007). On directional regression for dimension reduction. *Journal of the American Statistical Association*, 102:997–1008.
- Li, G., Yang, D., Nobel, A. B., and Shen, H. (2016). Supervised singular value decomposition and its asymptotic properties. *Journal of Multivariate Analysis*, 146:7–17.

- Li, K.-C. (1991). Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–342.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.
- Liu, T. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.
- Liu, Z., Shi, Z., Zhao, M., and Xu, W. (2007). Mobile robots global localization using adaptive dynamic clustered particle filters. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007*, pages 1059–1064.
- Mahadevan, S. and Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(2):311 – 365.
- Mitchell, T. (2006). The discipline of machine learning. Technical Report CMU ML-06 108.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., , and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.
- Mollah, M. N. H., Sultana, N., Minami, M., and Eguchi, S. (2010). Robust extraction of local structures by the minimum beta-divergence method. *Neural Networks*, 23(2):226–238.
- Morimoto, J., Hyon, S., Atkeson, C. G., and Cheng, G. (2008). Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 2711–2716.
- Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.
- Murray, J. F., Hughes, G. F., and Kreutz-Delgado, K. (2005). Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research*, 6:783–816.
- Neumann, G. (2011). Variational inference for policy search in changing situations. In *The 28th International Conference on Machine Learning*, pages 817–824.
- Neumann, G. and Peters, J. (2008). Fitted q-iteration by advantage weighted regression. In *Advances in Neural Information Processing Systems 21*, pages 1177–1184.
- Ng, A. T. and Jordan, M. (2000). Pegasus: A policy search method for large mdps and pomdps. In *The 16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415.

- Nguyen, T. T., Li, Z., Silander, T., and Leong, T. (2013). Online feature selection for model-based reinforcement learning. In *The 30th International Conference on Machine Learning*, pages 498–506.
- Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization, second edition*. World Scientific.
- Pál, D., Póczos, B., and Szepesvári, C. (2010). Estimation of renyi entropy and mutual information based on generalized nearest-neighbor graphs. In *Advances in Neural Information Processing Systems 23*, pages 1849–1857.
- Paninski, L. (2003). Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253.
- Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242.
- Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine Series 5*, 50(302):157–175.
- Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238.
- Peshkin, L. and Shelton, C. R. (2002). Learning from scarce experience. In *The 19th International Conference on Machine Learning*, pages 498–505.
- Peters, J., Mülling, K., and Altun, Y. (2010). Relative entropy policy search. In *The 24th AAAI Conference on Artificial Intelligence*, pages 1607–1612.
- Peters, J. and Schaal, S. (2006). Policy gradient methods for robotics. In *The IEEE International Conference on Intelligent Robotics Systems*, pages 2219–2225.
- Peters, J. and Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. In *The 24th International Conference on Machine Learning*, pages 745–750.
- Pluim, J. P. W., Maintz, J. B. A., and Viergever, M. A. (2003). Mutual information based registration of medical images: A survey. *IEEE Trans. Med. Imaging*, 22(8):986–1004.
- Pong, T. K., Tseng, P., Ji, S., and Ye, J. (2010). Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489.
- Principe, J. C., Xu, D., Zhao, Q., and Fisher, J. W. (2000). Learning from examples with information theoretic criteria. *Journal of VLSI Signal Processing System*, 26(1-2):61–77.
- Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.

- Rasmussen, C. E. and Kuss, M. (2003). Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*, pages 751–758.
- Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.
- Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501.
- Reich, B. J., Bondell, H. D., and Li, L. (2011). Sufficient dimension reduction via bayesian mixture modeling. *Biometrics*, 67(3):886–895.
- Romano, S., Bailey, J., Vinh, N. X., and Verspoor, K. (2014). Standardized mutual information for clustering comparisons: One step further in adjustment for chance. In *The 31th International Conference on Machine Learning*, pages 1143–1151.
- Rombokas, E., Malhotra, M., Theodorou, E., Matsuoka, Y., and Todorov, E. (2012). Tendon-driven variable impedance control using reinforcement learning. In *Robotics: Science and Systems VIII, University of Sydney, Sydney, NSW, Australia, July 9-13, 2012*.
- Roweis, S. T. and Saul, L. K. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326.
- Sainui, J. and Sugiyama, M. (2013). Direct approximation of quadratic mutual information and its application to dependence-maximization clustering. *IEICE Transactions on Information and Systems*, 96-D(10):2282–2285.
- Sainui, J. and Sugiyama, M. (2014). Unsupervised dimension reduction via least-squares quadratic mutual information. *IEICE Transactions on Information and Systems*, 97-D(10):2806–2809.
- Samarov, A. M. (1993). Exploring regression structure using nonparametric functional estimation. *Journal of the American Statistical Association*, 88(423):836–847.
- Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.
- Sasaki, H., Noh, Y., and Sugiyama, M. (2015a). Direct density-derivative estimation and its application in kl-divergence approximation. In *The 18th International Conference on Artificial Intelligence and Statistics*.
- Sasaki, H., Tangkaratt, V., and Sugiyama, M. (2015b). Sufficient dimension reduction via direct estimation of the gradients of logarithmic conditional densities. In *The 7th Asian Conference on Machine Learning, ACML*, pages 33–48.
- Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242.
- Schaal, S. (2009). The sl simulation and real-time control software package. Technical report.

- Schaal, S. and Atkeson, C. G. (1994). Robot juggling: An implementation of memory-based learning. *14(1):57–71*.
- Schaal, S., Atkeson, C. G., and Vijayakumar, S. (2000). Real-time robot learning with locally weighted statistical learning. In *The 2000 IEEE International Conference on Robotics and Automation*, pages 288–293.
- Schneider, J. G. (1996). Exploiting model uncertainty estimates for safe dynamic control learning. In *Advances in Neural Information Processing Systems 9*, pages 1047–1053. The MIT Press.
- Schölkopf, B., Smola, A., and Müller, K.-R. (1997). *Kernel principal component analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.
- Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559.
- Silva, V. D. and Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 721–728. MIT Press.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.
- Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. London.
- Stiefel, E. (1935). Richtungsfelder und fernparallelismus in n-dimensionalen mannigfaltigkeiten. *Commentarii mathematici Helvetici*, 8:305–353.
- Stigler, S. M. (1981). Gauss and the invention of least squares. *The Annals of Statistics*, 9(3):465–474.
- Sugimoto, N., Tangkaratt, V., Wensveen, T., Zhao, T., Sugiyama, M., and Morimoto, J. (2014). Efficient reuse of previous experiences in humanoid motor learning. In *14th IEEE-RAS International Conference on Humanoid Robots*, pages 554–559.
- Sugimoto, N., Tangkaratt, V., Wensveen, T., Zhao, T., Sugiyama, M., and Morimoto, J. (2016). Trial and error: Using previous experiences as simulation models in humanoid motor learning. *IEEE Robot Automatation Magazine*, 23(1):96–105.
- Sugiyama, M. (2007). Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of Machine Learning Research*, 8:1027–1061.
- Sugiyama, M., Idé, T., Nakajima, S., and Sese, J. (2008). Semi-supervised local fisher discriminant analysis for dimensionality reduction. In *Advances in Knowledge Discovery and Data Mining, 12th Pacific-Asia Conference, PAKDD 2008*, pages 333–344.

- Sugiyama, M., Kanamori, T., Suzuki, T., du Plessis, M. C., Liu, S., and Takeuchi, I. (2013). Density-difference estimation. *Neural Computation*, 25(10):2734–2775.
- Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., and Okanohara, D. (2010). Conditional density estimation via least-squares density ratio estimation. In *The 13th International Conference on Artificial Intelligence and Statistics*, pages 781–788.
- Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *The 7th International Conference on Machine Learning*, pages 216–224.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.
- Suzuki, T. and Sugiyama, M. (2011). Least-squares independent component analysis. *Neural Computation*, 23(1):284–301.
- Suzuki, T. and Sugiyama, M. (2013). Sufficient dimension reduction via squared-loss mutual information estimation. *Neural Computation*, 25:725–758.
- Suzuki, T., Sugiyama, M., Kanamori, T., and Sese, J. (2009). Mutual information estimation reveals global associations between stimuli and biological processes. *BMC Bioinformatics*, 10(S-1).
- Suzuki, T., Sugiyama, M., Sese, J., and Kanamori, T. (2008). Approximating mutual information by maximum likelihood density ratio estimation. In *Third Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery, FSDM 2008, held at ECML-PKDD 2008*, pages 5–20.
- Tadić, V. (2001). On the convergence of temporal-difference learning with linear function approximation. *Machine Learning*, 42(3):241–267.
- Tan, M. (1993). Multi-agent reinforcement learning: Independent versus cooperative agents. In *The 10th International Conference on Machine Learning*, pages 330–337.
- Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319.
- Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.
- Tesauro, G. (2003). Extending q-learning to general adaptive multi-agent systems. In *Advances in Neural Information Processing Systems 16*, pages 871–878.
- Toh, K. and Yun, S. (2009). An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. In *International Symposium on Mathematical Programming*.
- Torkkola, K. (2003). Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438.

- Tretyakov, K. (2004). Machine learning techniques in spam filtering. Technical report, Institute of Computer Science, University of Tartu.
- Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16(3):185–202.
- Tsitsiklis, J. N. and Roy, B. V. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.
- Udriste, C. (1994). *Convex Functions and Optimization Methods on Riemannian Manifolds*. Springer Netherlands.
- Ueno, T., Hayashi, K., Washio, T., and Kawahara, Y. (2012). Weighted likelihood policy search with model selection. In *Advances in Neural Information Processing Systems 25*, pages 2366–2374.
- Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.
- Vijayakumar, S. and Schaal, S. (2000). Locally weighted projection regression: Incremental real time learning in high dimensional space. In *The 17th International Conference on Machine Learning*, pages 1079–1086.
- Viola, P. A. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154.
- Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.
- Wang, M., Sha, F., and Jordan, M. I. (2010). Unsupervised kernel dimension reduction. In *Advances in Neural Information Processing Systems 23*, pages 2379–2387.
- Wang, X. and Dietterich, T. G. (2003). Model-based policy gradient reinforcement learning. In *The 20th International Conference on Machine Learning*, pages 776–783.
- Watkins, C. J. and Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, 8(3):279–292.
- Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. A. (2015a). Embed to control: A locally linear latent dynamics model for control from raw images. *CoRR*, abs/1506.07365.
- Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. A. (2015b). Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems 28*, pages 2746–2754.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.
- Wu, Q., Mukherjee, S., and Liang, F. (2008). Localized sliced inverse regression. In *Advances in Neural Information Processing Systems 21*, pages 1785–1792.

- Xia, Y. (2007). A constructive approach to the estimation of dimension reduction directions. *The Annals of Statistics*, 35(6):2654–2690.
- Xie, N., Hachiya, H., and Sugiyama, M. (2012). Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. In *The 29th International Conference on Machine Learning*, pages 153–160.
- Yang, W.-Y., Liang, W., Xin, L., and Zhang, S.-W. (2009). Subspace semi-supervised fisher discriminant analysis. *Acta Automatica Sinica*, 35(12):1513 – 1519.
- Yu, D., Seide, F., and Li, G. (2012). Conversational speech transcription using context-dependent deep neural networks. In *The 29th International Conference on Machine Learning*.
- Zhang, L., Zhu, J., and Yao, T. (2004). An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing*, 3(4):243–269.
- Zhang, W. and Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. In *The 14th International Joint Conference on Artificial Intelligence*, pages 1114–1120.
- Zhang, Y. and Yeung, D. (2009). Heteroscedastic probabilistic linear discriminant analysis with semi-supervised extension. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009*, pages 602–616.
- Zhang, Z., Dai, G., Xu, C., and Jordan, M. I. (2010). Regularized discriminant analysis, ridge regression and beyond. *Journal of Machine Learning Research*, 11:2199–2228.
- Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. (2012). Analysis and improvement of policy gradient estimation. *Neural Network*, 26:118–129.
- Zhao, T., Hachiya, H., Tangkaratt, V., Morimoto, J., and Sugiyama, M. (2013). Efficient sample reuse in policy gradients with parameter-based exploration. *Neural Computation*, 25(6):1512–1547.
- Zhong, W., Zeng, P., Ma, P., Liu, J. S., and Zhu, M. Y. (2005). RSIR: regularized sliced inverse regression for motif discovery. *Bioinformatics*, 21(22):4169–4175.