PhD Thesis

博士論文

Geometric Theory of Weight Dynamics for Training Hierarchical Neural Networks

（階層型神経回路モデルにおける学習力学の幾何学的理論）

Ryo Karakida

唐木田　亮

Department of Complexity Science and Engineering

The University of Tokyo

February 2017

Abstract

In recent years, machine learning with neural network models has succeeded in the name of deep learning. Novel model architectures and learning algorithms have achieved higher performance than standard machine learning techniques in the various practical application such as visual recognition, speech recognition and so on. However, its performance-oriented development causes many heuristics which have little or no theoretical guarantee on the optimality of the method. For developing and improving the techniques for training the neural network models, it is indispensable to reveal how and why the neural network models and their learning algorithms realize the higher performance. In this thesis, we focus on two learning algorithms proposed in the context of deep learning and theoretically analyze them. That is, Contrastive Divergence learning in restricted Boltzmann machines (RBMs) and weight normalization. On the learning of the RBMs with continuous visible units, we prove that Contrastive Divergence learning, a rough approximation of maximum likelihood learning, has the same solutions as the exact learning algorithm. Besides, we reveal a geometric structure of the solutions and what information they extract from input data. Based on the theoretical insight obtained by the analysis, we also propose a novel efficient algorithm for the RBMs. Next, we analyze a gradient method known as Weight Normalization, which has experimentally made the convergence of learning faster. We identify the mechanisms of the acceleration, that is, an automatic turning of a learning rate and scale invariant gradients. Moreover, we extend the weight Normalization to a natural gradient method and confirm its effectiveness in numerical experiments. Mathematical analysis and novel algorithms shown in this thesis will be expected to advance the studies of deep learning further.

Acknowledgements

First of all, I would like to express sincere appreciation to my supervisor, Professor Masato Okada, who has encouraged me throughout my graduate school days. This thesis is greatly indebted for his enthusiasm, patience, and support. His insightful perspective and practical knowledge through different scientific fields have made me grow up as a researcher with the wider viewpoint. Besides my supervisor, I express my gratitude to the rest of my Ph.D. committee, Professor Koji Tsuda, Professor Akinao Nose, Associative Professor Noboru Kunihiro and Lecturer Issei Sato for providing valuable comments on my drafts and presentation for this thesis.

My sincere appreciation also goes to my collaborator, Dr. Shun-ichi Amari, who has given me opportunities to work together and has invited me to his theoretical studies of learning in brains and machines. His insightful comments on mathematical engineering and warmful personality have got rid of my limit and have matured my philosophy of research.

I am gratefully thankful to all members in Okada laboratory and friends in the Department of Physics, who have frequently given me a fun conversation and stimulating discussion. There have been too many to count, but I was mainly guided by my collaborators, Dr. Kenji Nagata and Dr. Yasuhiko Igarashi. It is also a pleasure to thank Yoshihiro Nagano and Yuki Yoshida for giving me a chance to work together and made my daily life in the lab enjoyable. Thank you.

Let me remark that my entire Ph.D. course was financially supported by Japan Society for the Promotion of Science (JSPS). I was proud of working as a research fellow of the JSPS.

Finally, I also express my profound gratitude to my parents Minoru and Meiko, my brother Takuro, grandfathers Toyoaki and Atsuyoshi, and grandmothers Ritsuko and Fujiko. I think words cannot express how grateful I am to my family. The accomplishment of this thesis would not have been possible without their selfless love. Unfortunately, Atsuyoshi passed away just a month before I finished writing this thesis. I would like to dedicate this milestone to him.

Table of contents

List of figures

Nomenclature

Acronyms / Abbreviations

AE	Auto-Encoder
CD	Contrastive Divergence
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
G-B RBM	Gaussian-Bernoulli Restricted Boltzmann Machine
G-G RBM	Gaussian-Gaussian Restricted Boltzmann Machine
ICA	Independent Component Analysis
ML	Maximum Likelihood
NG	Natural Gradient
PCA	Principal Component Analysis
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RNG	Radial Natural Gradient
SGD	Stochastic Gradient Descent
WN	Weight Normalization

Chapter 1

Introduction

Without a theory the facts are silent.
– Friederich A. von Hayek [1]

In this 15 years, researchers have found that it is possible to train deep neural networks by using plenty of computational resource, a significant amount of data and novel techniques for training. In 2012, the deep neural network outperformed standard machine learning models in ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [60]. Compared to the previous years, the deep neural networks have achieved about a 10 % lower test classification error rate and continues to update the record year after year. In speech recognition, deep neural networks have also achieved about 10 % improvement on a word error rate [95]. The application field of deep learning is not restricted to visual or audio data but has expanded into natural language processing, playing games like Go [96], spectral analysis in high energy physics [12], prediction of molecular electronic properties in chemistry [71] and more [64].

In contrast to the empirical success of deep learning in practical applications, there are limited number of theoretical studies, which explain why and how the learning succeeds and the neural network models can perform better than the traditional machine learning. To understand the mechanism and to develop better algorithms, it is crucial to construct a theory of learning.

In this section, we first explain background knowledge of this thesis: the recent development of neural networks for practical application. Next, we briefly introduce the previous theoretical studies of learning in hierarchical neural networks. Finally, we describe the objective of this thesis, that is, to construct theories of the recently developed learning algorithms for hierarchical neural network models. Because the learning in the neural network models is

[1]This quote is attributed to him by William N. Butos [21].

an optimization of matrices representing synaptic connectivity, it is important to understand the geometry of the matrices.

1.1 Neural network models

There are various types of architectures such as fully-connected networks, convolutional neural networks (CNN) and recurrent neural networks (RNN) [14, 64]. Although we focus on the fully-connected networks as illustrated in Figure 1.1, mathematical foundations of the neural networks are mostly common among the different architectures.

As described below, there are two types of formulation to represent neural networks, that is, discriminative models and generative models.

1.1.1 Discriminative models

The discriminative model is the easiest way to formulate learning in neural network models. It has a cost function, and its learning algorithm is derived from a gradient of the cost function. The gradient method in multi-layer neural networks is known as the backpropagation algorithm [89]. Here, Figure 1.1 illustrates the typical fully-connected hierarchical neural networks. For instance, one of the way to formulate the cost function on the network in Figure 1.1 (b) is a squared error between input data and output data such as

$$E(W^{(1)},W^{(2)},...,W^{(L)}) = \frac{1}{T}\sum_i ||\mathbf{y}^i - g(W^{(L)}\cdots g(W^{(2)}g(W^{(1)}\mathbf{x}^i)))||^2, \qquad (1.1)$$

where we have T input samples $\mathbf{x}^i$ and their labels $\mathbf{y}^i$ $(i = 1,2,...,T)$. The activation function is given by $g(\cdot)$. The weight matrix $W^{(l)}$ represents a strength of synaptic connectivity between the l-th layer and $(l+1)$-th layer. For binary label samples, we usually define the cost function as a cross-entropy.

Sigmoid and Tanh functions are classical and typical activation function, which imitate the firing activity of biological neurons. Recently, rectified linear unit (ReLU) , i.e., $g(x) = 0 \ \ (x \leq 0), \ \ x \ \ (x > 0)$ is widely used and give faster convergence and better generalization errors in practical applications [74]. Some extensions of ReLU have also been proposed, for instance, leaky ReLU which has the finite gradient with negative argument and can further improve the convergence speed of learning [66]. MaxOut unit is a more complicated activation than ReLU, which is defined by a mixture of linear activation functions [40]. When trained with the Dropout algorithm [99], MaxOut is easier to optimize the parameters than ReLU.

Briefly speaking, the particular case of a two-layer network with the output $\mathbf{y}^i$ replaced by the input $\mathbf{x}^i$ is known as auto-encoders (AEs). The AEs perform regression between an input signal and its feedback signal from the hidden layer, and compress the information of input data in an unsupervised manner. There are many types of variants of the auto-encoders [14], for instance, denoising auto-encoder [104]. The denoising AE receives input samples with added noise, and attempts to eliminate the noise by using the feedback signals. Deep networks pre-trained by denoising AE can achieve better discrimination error rates than those pre-trained by conventional AE and comparable discrimination error rates to those pre-trained by RBMs [104].

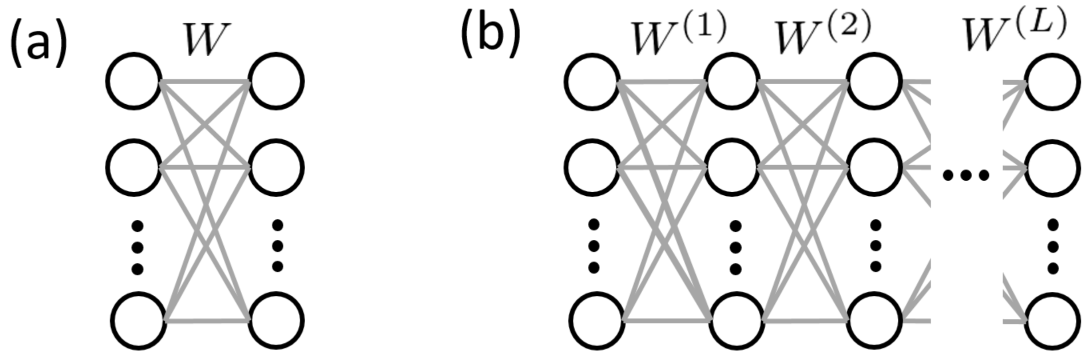


Fig. 1.1 Network architectures studied in this thesis. (a) The two-layer network such as auto-encoder and RBM. (b) The $(L+1)$-layer network with one input layer and L hidden layers.

1.1.2 Generative models

Let us define a random variable for each neuron in a neural network model and the probability distribution of the model. This model is known as the generative model. In the inference with the generative model, we can utilize the framework of probabilistic learning algorithms such as maximum likelihood estimation, Bayesian estimation with a prior distribution, variational Bayesian methods, MCMC sampling and so on [14, 90]. In particular, the generative model is useful for unsupervised tasks because it has its own probability distribution fitted to the input data. The trained generative model enables us to generate artificial samples as used in image generation or compensation of the input data [90].

Boltzmann machines

Boltzmann machine is a typical generative neural network model defined by the following probabilistic distribution [1]:

$$p(\mathbf{x}) = \exp(-\mathbf{x}^T W \mathbf{x})/Z, \tag{1.2}$$

where the random variable x_i for each neuron interacts with each other through a weight matrix W. Note that, to realize exact inference in Boltzmann machines, we need to compute a normalization constant Z. This normalization constant Z makes the Boltzmann machine difficult to use in practice because it takes too much computational time to compute. To overcome this problem, the researchers have developed the following restricted Boltzmann machine.

Restricted Boltzmann machines

The restricted Boltzmann machine (RBM) is a bipartite graphical model of the two-layer neural network illustrated in Figure 1.1 (a) [97, 42]. Its probability distribution is given by

$$p(\mathbf{h}, \mathbf{v}) = \exp(-\mathbf{h}^T W \mathbf{v})/Z, \tag{1.3}$$

where **v** denotes the visible random variable in the first layer and **h** denotes the hidden random variable in the second layer. The visible and hidden units of the RBM are conditionally independent of each other, that is,

$$p(\mathbf{h}|\mathbf{v}) = \prod_i p(h_i|\mathbf{v}), \tag{1.4}$$

$$p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i|\mathbf{h}). \tag{1.5}$$

This conditional independence of the variables in the same layer makes the sampling of variables mix easily in the MCMC process of maximum likelihood (ML) learning [42]. Due to easy mixing, a rough approximation of ML learning, i.e., the contrastive divergence explained in Chapter 2, have empirically succeeded to train RBMs with a finite number of sampling.

There are various types of RBMs as briefly introduced in the following. The most standard RBM holds binary visible and hidden random variables. There is an extension for continuous input data such as images and sounds, that is, Gaussian-Bernoulli RBM, where the visible variables obey continuous Gaussian distribution and the hidden ones follow binary.

3-way RBM is the expansion of Gaussian-Bernoulli RBM and has probabilistic variables to imitate the simple-cell-like and complex-cell-like activities [59]. It can capture the covariance structure of the natural images and obtain even better features by stacking binary RBMs above it. Ranzato et. al. have even further generalized the 3-way RBM and done extensive experiments on natural image datasets [86]. A generalized formulation of RBMs is known as exponential family harmoniums [109]. Although RBMs are likely to be used in unsupervised learning, we can make use of discriminative RBMs in the case of supervised learning [61].

The maximum likelihood learning is a typical learning criterion of the RBMs. It is noteworthy that many studies have also developed the alternative methods to the ML learning in recent years. Contrastive divergence explained in Chapter 2 can been seen as a rough approximation of the ML learning. TAP approximation has succeeded to train RBMs and to make them perform comparable or better than those trained by contrastive divergence learning [36]. Grosse and Salakhutdinov have proposed an approximated natural gradient method with low computational complexity, which is scalable in large RBMs [41]. Training based on Wasserstein distance makes Boltzmann machines more suitable to data completion and denoising [71]. Cho et al. proposed a novel parameterization of variance parameters and a parallel tempering learning algorithm for G-B RBM [25]. Their experiments demonstrated that the proposed improvements overcome several difficulties encountered in training G-B RBMs.

1.1.3 Deep generative models

It is possible to extend the restricted Boltzmann machines to multi-layer models illustrated in 1.1 (b). Directed graph version of the multi-layer RBM is known as deep belief network (DBN), and undirected version is known as deep Boltzmann machine (DBM). To approximately realize ML learning in these deep generative models, we can use the wake-sleep learning algorithm for training DBN [44] and the variational Bayes method for training DBM [90]. In both of DBN and DBM, layerwise pre-training described in the following section has succeeded to make them perform better.

Compared to deep discriminative models such as stacked auto-encoders and convolutional neural networks, it is hard to train deep generative models. It is because marginalized probability distributions $P(\mathbf{v})$ and $P(\mathbf{h}|\mathbf{v})$ have too many modes (peaks) to make the sampling algorithm mix states well. To overcome this difficulty for training the deep generative models, generative stochastic networks (GSNs) [19], variational auto-encoders [57] and generative adversarial networks (GAN) [39] have been developed over the past few years. In unsupervised learning with image data, these models have enabled us to generate sharper artificial images than the previous deep generative models.

1.2 Techniques for learning in neural network models

The recent success of hierarchical neural networks owes a great deal to the development of learning techniques. For instance, we sometimes explicitly or implicitly assume the sparse firing on the activation of the neurons [43]. Regularization techniques such as max-norm and Dropout are also essential to make the models achieve high generalization performance [99]. Researchers have also developed novel algorithms based on a fundamentally new principle like adversarial training [39]. Although there are many important techniques to be explained, we focus on the following two topics in this section: layerwise pre-training and gradient methods.

1.2.1 Layerwise pre-training: deep networks composed by two-layer networks

Layerwise pre-training is one of breakthrough techniques for training hierarchical neural networks. In general, neural network models with millions of parameters are likely to overfit to training samples. To overcome this problem in deep hierarchical neural networks, Hinton and Salakhutdinov have proposed layerwise pre-training [45]. Before training all of the layers by a supervised manner, one can train each neighboring two layers from the bottom layer to the top layer one by one in an unsupervised manner. Experiments demonstrated that this unsupervised layerwise pre-training finds typical features of input data, which make easy to train all layers at the same time after the layerwise pre-training [31, 45]. In other words, layerwise pre-training is a method to find good initial values of the parameters that prevent overfitting. In particular, Erhan and Bengio have experimentally and quantitatively confirmed that the supervised backpropagation after the unsupervised layerwise pre-training converges faster to solutions with better generalization performance than the backpropagation started from random initial values [31]. Bengio et al. speculate that the layerwise pre-training disentangles the input data into more abstract features in deeper layers and that these abstract features will be helpful to describe the unknown test dataset and to realize much lower generalization errors [18].

1.2.2 Gradient methods

1st-order methods

The most primary method of learning is a steepest descent algorithm, which decreases the cost function by using its 1-st order derivatives. In particular, the stochastic gradient decent

(SGD) is widely used. This method computes the steepest descent gradient by using 10-500 training samples, or mini-batch, at each update. The extensions of the SGD such as AdaGrad, RMSprop, and Adam modify the learning rates by normalizing the amplitude of the gradients [56]. Let us remark that normalization techniques such as Batch Normalization [49] and Weight Normalization [91] are also effective to make the gradient methods converge faster. In particular, the Batch Normalization normalizes the input from the previous layer and it stabilizes the learning dynamics. Besides it allows us to use a larger learning rate and to remove the Dropout procedure. The Weight Normalization will be intensively investigated in Chapter 4 and 5.

2nd-order methods

Hessian-Free (HF) method is an efficient approximation of Newton's method [67] and has lower computational time and smaller memory space suitable for large deep networks. The deep auto-encoders trained by HF methods without layerwise pre-training has achieved generalization errors comparable to those trained by SGD with layerwise pre-training. In the recurrent neural networks, HF method also accomplished the state-of-the-art performance [69].

There is an essential work to modify the Newton's method for training hierarchical neural networks, that is, saddle free Newton (SFN) algorithm [28]. The cost functions of the hierarchical models are believed to have many saddle points (See Section 1.3.2). The SFN algorithm corrects the steepest descent direction by an absolute Hessian matrix that makes the learning dynamics escape from the saddle points along the direction with negative eigenvalues of the Hessian. It succeeded to make the convergence of learning faster in a deep auto-encoder and to find better solutions with lower generalization errors.

Note that the study of SFN also experimentally investigated the error landscape in 3-layer nonlinear perceptrons [28]. They reported that most of the fixed points with higher errors are not local minima but saddle points. In addition, they remarked that several statistical models in statistical physics, such as Gaussian random field and spherical spin model, have the same energy landscape. Furthermore, Fukumizu and Amari theoretically proved that the 3-layer perceptrons have bad saddle points where the learning dynamics become very slow [35]. Therefore, the SFN algorithm is believed to be effective in deep learning and even in other high-dimensional non-convex optimization problems.

Natural gradient methods

The natural gradient method was invented to accelerate the 1st-order gradients by using underlying Riemannian parameter space [5, 6]. It corrects the steepest decent direction by using the Riemannian metric of Kullback-Leibler (KL) divergence, i.e., Fisher information matrix, and has succeeded in practical applications. In particular, for training multi-layer perceptrons, the natural gradient has been superior to other methods like 2nd-order optimization because it can avoid or alleviate the plateau phenomena [82].

The natural gradient methods require the inversion of the Fisher information matrix and it takes too much computational time. Because deep networks hold many parameters and require much computational time and memory space, it seems to be difficult to use natural gradient methods. Fortunately, natural gradient algorithms suited to the large-scale neural networks have been developed such as the block diagonal approximation of the Fisher metric [62], metric-free optimization with conjugate gradients [29], or approximation of the metric by sufficient statistics unique to exponential family models [41].

1.3 Theory of neural networks

In contrast to success in practical applications, there are less theoretical studies to answer why and how the neural network models perform well. Let us briefly overview the theoretical studies related to this thesis.

1.3.1 Theory of models: expressive power

It has been known that 3-layer networks, in other words, shallow networks, are universal approximators for continuous input data [13]. Recently, several studies have revealed that the deep networks have more expressive power than the shallow ones [16, 73, 20, 84]. The complexity of functions that the deep network can represent is known to grow exponentially with the number of layers. In contrast, in the shallow network, the complexity cannot grow exponentially but increases with a polynomial of the number of hidden units. Therefore, if the number of hidden units is same between the deep network and the shallow one, the expressive power of the deep network is much bigger than that of shallow one.

With regards to RBMs, several studies clarified that binary RBMs can approximate arbitrarily well any probability distribution and that the representation power does not decrease by stacking the binary RBMs [62, 72]. Moreover, the binary RBN never requires more units than a binary RBM to represent a distribution with a certain accuracy [63]. Under mild assumptions, Krause et al. also analyzed a DBN with continuous visible variables and showed that it approximates mixture of exponential families with an arbitrarily good accuracy [58].

Note that the expressive power of the model is not necessarily equal to the easiness of training or high generalization performance. In general, deep network models are hard to train because of vanishing gradient problems [9, 6, 94]. The magnitude of the gradient becomes much smaller in the lower layer and the learning is difficult to process. In addition, even if the model potentially has a high expressive power, learning algorithms may converge to the solution that corresponds to a poor expressive power or overfitting.

1.3.2 Theory of learning: dynamical analysis

In contrast to the expressive power of the neural network "models", the previous studies have paid little attention to the theoretical perspective in "learning". There are several topics on the theory of learning quite related to this thesis.

Error Landscape of 3-layer linear neural networks

In the neural network models with linear activation functions, we can derive exact solutions by analytical calculation and obtain theoretical insight into the structure of the solution space. Let us consider a 3-layer liner neural network with the following cost function,

$$E(W^{(1)}, W^{(2)}) = \frac{1}{T}\sum_i ||\mathbf{y}^i - W^{(2)}W^{(1)}\mathbf{x}^i||^2. \tag{1.6}$$

We assume that its learning algorithm is derived by the steepest decent update such as

$$\frac{dW^{(l)}}{dt} = -\eta \frac{dE}{dW^{(l)}} \quad (l = 1, 2), \tag{1.7}$$

where η denotes a given constant learning rate.

We can analytically obtain the fixed points of the learning dynamics in this 3-layer network (1.7), where the gradients dW_l/dt become zero. The stable points of eq. (1.7) correspond to perform a singular value decomposition on the correlation matrix between the input samples and the output samples [11]. The number of singular values extracted in the trained network equals to that of hidden units. Therefore, the learning rules (1.7) perform a kind of dimension reduction of the data.

It is remarkable that there exist unstable fixed points where the learning trajectory never converges. Baldi and Hornik have proved that this stable fixed point is the global minimum of the cost function and that there is no local minimum [11]. All of the other fixed points are saddle points. Because the saddle points always have higher training errors than the stable global minimum, the learning trajectory escapes the saddle points and eventually converges to the global minimum.

Note that a linear auto-encoder is a special case of the above 3-layer network, where the target signal is given by the input sample such as $\mathbf{y} \leftarrow \mathbf{x}$ and the network has a tied weight, i.e., $W^{(2)} = W^{(1)T}$. In this linear AE, the global minimum corresponds to perform principal component analysis of the input data and the largest principal eigenvalues are extracted after training [111, 78]. The number of the extracted eigenvalues is equivalent to the number of the hidden units. There are saddle points with higher errors than the global minimum and no local minima. Besides, when the number of the hidden units is less than that of the input units, the linear AE is also known as a reduced rank regression [103]. It is notable that Nakajima formulated the reduced rank regression by a probabilistic model and revealed the exact solution of its Bayesian estimation [75].

Although this linear 3-layer network can only perform relatively simple information processing, its learning dynamics of the weight matrix is nonlinear and a bit complicated. Therefore, it seems to be difficult to analytically solve the differential equation (1.7). Surprisingly, several studies have revealed that the differential equation (1.7) can be reduced to the Riccati matrix differential equation and is solvable for broad initial conditions [113, 34]. Yan et al. have solved the case of linear AE [113] and Fukumizu has generalized the theory into the case of the 3-layer linear perceptrons [34]. In addition, he derived the theoretical values of the generalization error and discussed the occurrence of the overfitting in the linear neural networks.

Error landscape of deep linear neural networks

Here, let us consider a deep linear neural network with more than four layers which has the following cost function,

$$E(W^{(1)}, W^{(2)}, ..., W^{(L)}) = \frac{1}{T} \sum_i ||\mathbf{y}^i - W^{(L)} \cdots W^{(2)} W^{(1)} \mathbf{x}^i||^2. \tag{1.8}$$

In this case, the following studies have revealed theoretical properties of the deep linear neural network model. Kawaguchi proved that the steepest descent gradient of the cost function (1.8) has only one stable global minimum, no local minima and saddle points which have higher training errors than the global minimum [55]. Therefore, the structure of the solution space in the deep linear neural network is similar to that in the 3-layer linear one. Moreover, Saxe et al. found that the learning dynamics in the deep linear neural networks becomes analytically tractable for special initial conditions [92]. Their analytical solution of the learning dynamics shows that the larger singular value of the data correlation matrix is extracted faster. This means that the learning dynamics first captures the typical structure of the data and pick up the detailed structure later. They also theoretically demonstrated that the layerwise pre-training can speed up the convergence of the backpropagation algorithm. This result supports the experimental results of deep learning [45].

Remarkably, some numerical experiments in multi-layer neural network models with nonlinear activation functions also have confirmed that the training errors of local minima are lower than those of global minima and suggested that the obstacle in the learning dynamics is not the local minima but the saddle points [28].

Dynamical analysis in nonlinear neural networks

To analyze the learning dynamics of parameters in nonlinear neural network models, several previous studies have investigated relatively simple models. In a 2-layer auto-encoder model with continuous input variables and binary hidden variables, the stable fixed points of the steepest decent gradient has been derived under some mild assumptions [7, 48]. Their theory proved that those stable fixed points correspond to extract independent components from input data.

In nonlinear hierarchical models, there are some singular regions of the parameter space, where the redundant parameters degenerate. The dynamics of the steepest decent gradient becomes very slow in the singular regions, and this slow transient dynamics is known as plateau phenomena [6]. In 3-layer perceptrons with one output unit and two nonlinear hidden units where the true solution exists in the singular region, dynamical analysis derived the exact solution of the learning differential equation and revealed that the singular regions compose so-called Milnor attractor [26]. Interestingly, the natural gradients avoid or alleviate the plateaus in numerical experiments [82]. Dynamical analysis and statistical mechanical analysis also theoretically confirmed that the natural gradient method accelerates the convergence of the learning trajectory near the singular region [26, 87].

1.4 Contributions of this thesis

As remarked in the previous sections, the recent development of neural network models is due in large part to the empirical success in practical applications. In particular, it is learning algorithms different from the traditional optimization methods that make possible to train the hierarchical neural networks in realistic computational time and to find reasonable solutions. However, there are limited number of theoretical studies, which have revealed how and why the learning algorithms in neural networks perform so well. Understanding the underlying mechanism of the recently developed learning algorithms will be helpful to invent better learning algorithms based on the theory.

In this thesis, we focus on two learning algorithms proposed in the context of deep learning and theoretically analyze them. That is, *contrastive divergence learning in restricted Boltzmann machines* and *weight normalization*. Moreover, the objective of this thesis is not just to analyze the learning algorithms but also to invent better algorithms based on the knowledge obtained in our analyses. The overview of this study is depicted in Figure 1.2.

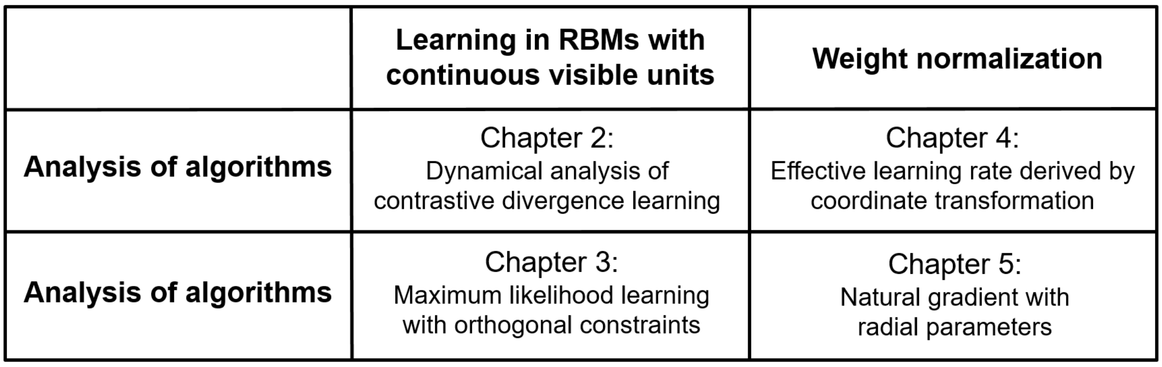
In Chapter 2, we first prove that contrastive divergence learning, a rough approximation of maximum likelihood learning, has the same solution with the exact method in restricted Boltzmann machines with Gaussian visible units. By using dynamical analysis of weight matrices, we analytically reveal the structure of the solution matrices. Our analysis also identifies what specific information of input data RBMs extract at the solutions. The results shown in Chapter 2 are based on our published papers [51, 53].

In Chapter 3, we propose an efficient algorithm for the RBMs based on the theoretical insight into the structure of the solutions obtained in Chapter 2. In details, we reveal that the likelihood and its gradients in the RBMs with Gaussian visible units are analytically tractable when the weight matrix is constrained to the Stiefel manifold. We propose a novel algorithm based on a geodesic flow on the Stiefel manifold. The results shown in Chapter 3 are based on our published proceeding in an international conference [54].

Next, in Chapter 4, we analyze a gradient method for neural networks known as weight normalization. The weight normalization is a kind of coordinate transformation and uses radial and directional parameters to represent the weight matrix. It has experimentally made the convergence of learning faster but the mechanism of the speed up remains unsolved. We show that the weight normalization realizes an automatic turning of a learning rate and scale invariant gradients.

In Chapter 5, we introduce a novel natural gradient algorithm by using the radial parameters of weight normalization, which greatly reduces the computational cost of the inverse of the metric. We confirm the effectiveness of the proposed algorithm by numerical experiments with supervised learning in 3-layer and 4-layer neural networks. In addition, we demonstrate several theoretical results, which uncover the geometric similarities of the Riemannian metric for neural networks between the proposed natural gradient method and the traditional natural gradient method.

Because both algorithms proposed in Chapter 3 and Chapter 5 are based on the geometric structure of the weight matrix, investigation of the geometry seems to be essential for the theory of the learning in neural networks.



	Learning in RBMs with continuous visible units	**Weight normalization**
Analysis of algorithms	Chapter 2: Dynamical analysis of contrastive divergence learning	Chapter 4: Effective learning rate derived by coordinate transformation
Analysis of algorithms	Chapter 3: Maximum likelihood learning with orthogonal constraints	Chapter 5: Natural gradient with radial parameters

Fig. 1.2 Overview of this thesis.

Chapter 2

Dynamical Analysis of Learning in Restricted Boltzmann Machines

The restricted Boltzmann machine (RBM) is an essential constituent of deep learning, but it is hard to train by using maximum likelihood (ML) learning, which minimizes the Kullback-Leibler (KL) divergence. Instead, contrastive divergence (CD) learning has been developed as an approximation of ML learning and widely used in practice. To clarify the performance of CD learning, in this study, we analytically derive the fixed points where ML and CD$_n$ learning rules converge in two types of RBMs: one with Gaussian visible and Gaussian hidden units and the other with Gaussian visible and Bernoulli hidden units. In addition, we analyze the stability of the fixed points. As a result, we find that the stable points of CD$_n$ learning rule coincide with those of ML learning rule in a Gaussian-Gaussian RBM. We also reveal that larger principal components of the input data are extracted at the stable points. Moreover, in a Gaussian-Bernoulli RBM, we find that both ML and CD$_n$ learning can extract independent components at one of stable points. Our analysis demonstrates that the same feature components as those extracted by ML learning are extracted simply by performing CD$_1$ learning. Expanding this study should elucidate the specific solutions obtained by CD learning in other types of RBMs or in deep networks.

2.1 Introduction

The restricted Boltzmann machine (RBM) is a bipartite graphical model widely used as an essential constituent of deep neural networks. The visible and hidden units of the RBM are conditionally independent of each other [97, 42]. Contrastive divergence (CD) learning, an approximate algorithm of maximum likelihood (ML) learning, efficiently uses this conditional independence [42]. If ML learning is used to train an RBM, it requires many iterations of Gibbs sampling at each update step and takes too much computational time. In contrast, CD learning requires only a few iterations of Gibbs sampling, iterated transitions between visible units and hidden units. CD_n learning uses n steps of Gibbs sampling. In particular, CD_1 learning is widely used and can complete the training in a short time. As evidenced empirically, an RBM trained by CD_1 learning achieves solutions close enough to those of an RBM trained by ML learning [22]. Stacked RBMs pre-trained by CD_1 learning have performed well in practical applications such as visual image classification [45] and acoustic modeling [27].

CD learning performs well enough to achieve success in practice, but there is little theoretical evidence that shows that it performs well. The previous theoretical studies demonstrated that the properties of CD learning are quite different from those of ML learning. For instance, there are certain cases where CD learning does not converge because its gradient does not obey any objective function [100]. In a simple case of an RBM with continuous 1-hidden and 1-visible units, Williams and Agakov gained theoretical insights into how the gradients of CD learning are biased in comparison with those of ML learning [110]. In general, the gradient of CD learning is interpreted as a truncated expansion of the log-likelihood gradient [15]. Even if the learning procedure converges to equilibrium solutions, these solutions do not necessarily maximize the likelihood function [22].

The previous studies left a question unanswered: what specific solutions are commonly or differently found by ML and CD learning? Although there are general conditions under which CD learning gives the ML solutions [115, 3], these conditions are loose, and CD solutions are hard to identify. For using CD learning in practice, it is important to identify the specific solutions obtained by CD learning and clarify what features are extracted from input data. A way to identify the solutions obtained by a learning rule is dynamical analysis of equilibrium and its stability [4]. By obtaining fixed points of the learning rule and checking their stability by using the perturbation method, the dynamical analysis has revealed what weight matrix can be extracted as a stable fixed point. For instance, it has clarified principal or minor components extracted in linear neural networks [78, 11, 23], principal components extracted by ML learning in the probabilistic PCA model [102], and independent components extracted by ICA algorithms [7, 48]. If dynamical analysis can be carried out on CD learning, we can understand the features extracted by CD learning.

In this study, we used the dynamical analysis to identify the fixed points of ML and CD_n learning rules in two types of RBMs. First, we derived an exact analytical form of the fixed points in a Gaussian-Gaussian RBM whose visible and hidden units are continuous real values [110, 43]. The ML and CD_n learning rules were explicitly formulated with model parameters. The analytical form demonstrated that ML learning extracts principal components whose eigenvalues are larger than a certain value. In addition, we analyzed the stability of the fixed points by using the perturbation method and revealed that a set of the largest principal components is extracted at stable fixed points. Next, we derived the analytical form for fixed points of CD$_n$ learning rule and found that it coincides with that of ML learning. In addition, their stability also coincides with that of ML learning. We thus concluded that CD$_n$ learning maximizes the likelihood function and extracts the same principal components as ML learning. Moreover, we also apply the same dynamical analysis to a Gaussian-Bernoulli RBM whose hidden units are binary [45, 65]. Under certain conditions, we revealed that both ML and CD$_n$ learning in the Gaussian-Bernoulli RBM have one common stable fixed point, where the Gaussian-Bernoulli RBM decomposes mixed input signals to independent source signals.

Our new paper [53] is a complete version of our previous results [51]. Unlike in the previous results, we generalize the analyses for the stability of the fixed points in Gaussian-Gaussian RBM to the case where there is no constraint on the number of hidden units. In addition, we demonstrate the previously omitted proofs of theories on the stable fixed point in Gaussian-Bernoulli RBM. We also discuss the relationship between our theoretical results and the previous studies such as experiments on natural images [65, 106] and nonlinear PCA [80]. Moreover, in both RBMs, we added the learning rules with bias parameters in Appendix A.

The results for CD$_n$ learning are independent of n. Because CD$_1$ learning can extract the same features as ML learning, CD$_1$ learning seems to be efficient to train RBMs. Expanding our analysis would help to elucidate features that can be extracted in RBMs with binary visible units or stacked RBMs.

2.2 Model

2.2.1 Gaussian-Gaussian RBM

The probability distribution of a Gaussian-Gaussian RBM is defined as follows [110, 43]:

$$p(\mathbf{h},\mathbf{v}) = \exp\left\{ -\sum_{i=1}^{M} \frac{(h_i-c_i)^2}{2s_i^2} - \sum_{j=1}^{N} \frac{(v_j-b_j)^2}{2\sigma_j^2} + \sum_{i,j} W_{ij} \frac{h_i}{s_i} \frac{v_j}{\sigma_j} \right\} / Z, \tag{2.1}$$

where $\mathbf{v}$ and $\mathbf{h}$ are random variables representing the states of the visible and hidden units, respectively. Both hidden h_i and visible v_j take continuous values and obey Gaussian distributions characterized by variances s_i^2 $(i = 1,...,M)$ and σ_j^2 $(j = 1,...,N)$. Let us denote an $M \times N$ weight matrix by W, biases by c_i and b_j, and a normalization constant by Z. The joint probability $p(\mathbf{h}, \mathbf{v})$ yields the following marginal distributions:

$$p(\mathbf{v}) = \mathcal{N}(\mathbf{v}; \Sigma(I - W^T W)^{-1}(W^T S^{-1}\mathbf{c} + \Sigma^{-1}\mathbf{b}), \Sigma(I_N - W^T W)^{-1}\Sigma), \tag{2.2}$$

$$p(\mathbf{h}) = \mathcal{N}(\mathbf{h}; S(I - WW^T)^{-1}(W\Sigma^{-1}\mathbf{b} + S^{-1}\mathbf{c}, S(I_N - WW^T)^{-1}S), \tag{2.3}$$

where we define a multivariate normal distribution with mean μ and variance Σ^2 by $\mathcal{N}\left(\mathbf{v}; \mu, \Sigma^2\right)$. Let us denote the covariance matrix of the hidden units by an $M \times M$ diagonal matrix $S = \mathrm{diag}(s_1, s_2, ..., s_M)$, whose entries satisfy $S_{ii} = s_i$ and $S_{ij} = 0$ $(i \neq j)$. In addition, we denote the covariance matrix of the visible units by an $N \times N$ diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, ..., \sigma_N)$. The conditional probabilities are also given by multivariate normal distributions such as

$$p(\mathbf{h}|\mathbf{v}) = \mathcal{N}\left(\mathbf{h}; SW\Sigma^{-1}\mathbf{v} + \mathbf{c}, S^2\right), \tag{2.4}$$

$$p(\mathbf{v}|\mathbf{h}) = \mathcal{N}\left(\mathbf{v}; \Sigma W^T S^{-1}\mathbf{h} + \mathbf{b}, \Sigma^2\right). \tag{2.5}$$

When training examples of $\mathbf{v}$ are given from the outside, we need to estimate W, $\mathbf{b}$, and $\mathbf{c}$ such that the marginal distribution $p(\mathbf{v})$ is as close as possible to a distribution generating training examples $q(\mathbf{v})$. The model variances Σ^2 and S^2 are given and fixed. For mathematical simplicity, we set the mean of input data to $\mu = \int d\mathbf{v} q(\mathbf{v})\mathbf{v} = \mathbf{0}$ and the bias parameters to $\mathbf{b} = \mathbf{c} = \mathbf{0}$ in the following learning rules. We can formulate a general case in the same way as explained in Appendix A. In Section 2.3, we will also assume that the variances of the visible and hidden units are homogeneous, i.e., $\Sigma = \sigma I_N,\ S = sI_M$, where I_N denote an $N \times N$ identity matrix.

ML learning rule

The learning rule of the maximum likelihood (ML) estimate of W is derived by minimizing the Kullback-Leibler (KL) divergence between the input distribution and the model distribution [42] and is given by

$$\tau \frac{dW}{dt} = S^{-1}\left\{< \mathbf{h}\mathbf{v}^T >_0 - < \mathbf{h}\mathbf{v}^T >_\infty\right\}\Sigma^{-1}, \tag{2.6}$$

where τ is a learning constant. The first term is defined by

$$< \mathbf{h}\mathbf{v}^T >_0 = \int d\mathbf{h} d\mathbf{v} p(\mathbf{h}|\mathbf{v}) q(\mathbf{v}) \mathbf{h}\mathbf{v}^T, \tag{2.7}$$

where $q(\mathbf{v})$ is the input data distribution. In contrast, the second term is defined by

$$< \mathbf{h}\mathbf{v}^T >_\infty = \int d\mathbf{h}d\mathbf{v}p(\mathbf{h},\mathbf{v})\mathbf{h}\mathbf{v}^T, \tag{2.8}$$

which is the expectation with respect to the model distribution $p(\mathbf{h},\mathbf{v})$. In practical application of RBMs, the first term of ML learning is calculated by a finite number of training examples and the second term is calculated by samples of the model distribution obtained by Gibbs sampling. In this study, to analyze average behaviors of the learning rules, we neglect fluctuations caused by the training examples and the Gibbs sampler and try to analytically calculate each term by using its definition. In Gaussian-Gaussian RBM, the ML learning rule (4) becomes

$$\tau\frac{dW}{dt} = W\Sigma^{-1}C\Sigma^{-1} - W(I_N - W^TW)^{-1}. \tag{2.9}$$

Let us denote the data covariance matrix by $C = \int d\mathbf{v}q(\mathbf{v})\mathbf{v}\mathbf{v}^T - \mu\mu^T$, where I_N is an $N \times N$ identity matrix.

CD$_n$ learning rule

In practical application of RBMs, CD$_n$ learning replaces the second term of ML learning with $< \mathbf{h}\mathbf{v}^T >_n$, which is calculated by using samples obtained after n times iteration of alternating Gibbs sampling between the visible and hidden layers [42]:

$$\tau\frac{dW}{dt} = S^{-1}\left\{< \mathbf{h}\mathbf{v}^T >_0 - < \mathbf{h}\mathbf{v}^T >_n\right\}\Sigma^{-1}. \tag{2.10}$$

Figure 2.1 illustrates the alternating Gibbs sampling in CD learning. The point of CD$_n$ learning is that the Gibbs sampling starts not from randomly chosen values but from some training examples. In this study, to analyze average behaviors of CD$_n$ learning, we analytically calculate the second term by an iterated integral of conditional probability distributions, that is,

$$< \mathbf{h}\mathbf{v}^T >_n = \int d\mathbf{h}d\mathbf{v}p(\mathbf{h}|\mathbf{v})p_n(\mathbf{v})\mathbf{h}\mathbf{v}^T, \tag{2.11}$$

$$p_n(\mathbf{v}_n) = \prod_{k=0}^{n-1}\int d\mathbf{h}_k d\mathbf{v}_k p(\mathbf{v}_{k+1}|\mathbf{h}_k)p(\mathbf{h}_k|\mathbf{v}_k)q(\mathbf{v}_0). \tag{2.12}$$

It is noteworthy that CD learning is related to other variants of unsupervised learning methods for generative models. For instance, One can regard Score Matching as the special CD_1 learning with a Langevin Monte Carlo sampling in the limit of infinitesimal noise [47]. It has been also pointed out that the gradient of Minimum Probability Flow (MPF) is closely related to that of CD_1 but a bit different [98].

By using this analytical formulation, the CD$_n$ learning rule for the Gaussian-Gaussian RBM is calculated to give a non-linear differential equation with the $(4n+1)$-th power of W [110]:

$$\tau \frac{dW}{dt} = W\Sigma^{-1}C\Sigma^{-1} - W\left\{(W^TW)^n\Sigma^{-1}C\Sigma^{-1}(W^TW)^n + \sum_{k=0}^{2n-1}(W^TW)^k\right\}. \tag{2.13}$$

We found that the CD learning rule (2.13) corresponds to the ML learning rule (2.9) as follows. Let us assume that the eigenvalues ε_i of W^TW satisfy $0 \le \varepsilon_i < 1$. This is a necessary and sufficient condition in order for the marginal distribution (2.2) to have a positive definite covariance. Under this condition, we can apply a Neumann series expansion to the inverse matrix of the ML learning, that is,

$$(I_N - W^TW)^{-1} = \sum_{k=0}^{\infty}(W^TW)^k. \tag{2.14}$$

The covariance matrix of $p_n(\mathbf{v})$ consists of a Neumann series terminated at the $(2n-1)$-th term and the $4n$-th power of W, i.e., $(W^TW)^n\Sigma^{-1}C\Sigma^{-1}(W^TW)^n$. Therefore, we can recognize the CD$_n$ gradients as a kind of Taylor series approximation of the ML gradient. Note that the CD$_n$ learning rule converges to the ML learning rule when $n \to \infty$. This means that an infinite number of iterations of Gibbs sampling corresponds to ML learning.

In spite of the non-linearity of learning rules (2.9) and (2.13), in Section 2.3, we will derive an analytical form of the fixed points and show their stability under the assumption of homogeneous model variances, i.e., $\Sigma = \sigma I_N$ and $S = sI_M$.

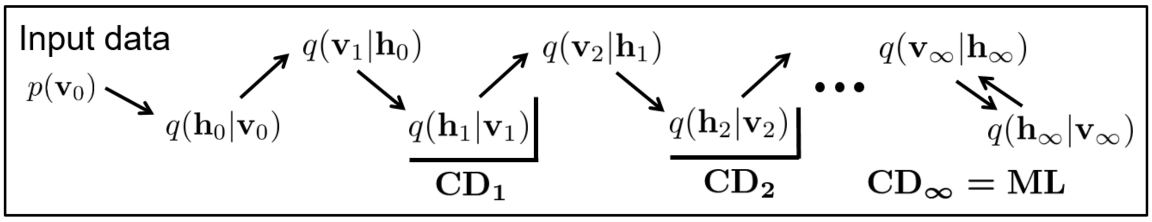

Input data
$p(\mathbf{v}_0)$
$q(\mathbf{h}_0|\mathbf{v}_0)$
$q(\mathbf{v}_1|\mathbf{h}_0)$
$q(\mathbf{h}_1|\mathbf{v}_1)$
$q(\mathbf{v}_2|\mathbf{h}_1)$
$q(\mathbf{h}_2|\mathbf{v}_2)$
$q(\mathbf{v}_\infty|\mathbf{h}_\infty)$
$q(\mathbf{h}_\infty|\mathbf{v}_\infty)$
$\mathbf{CD_1}$
$\mathbf{CD_2}$
$\mathbf{CD_\infty = ML}$

Fig. 2.1 Gibbs sampling in contrastive divergence learning.

2.2.2 Gaussian-Bernoulli RBM

The Gaussian-Bernoulli RBM has binary hidden variables $h_i = \{0,1\}$ and Gaussian visible variables v_i, whose probability distribution is defined as follows [45, 65]:

$$p(\mathbf{h},\mathbf{v}) = \exp\left\{ -\sum_{j=1}^{N} \frac{(v_j - b_j)^2}{2\sigma_j^2} + \sum_{i,j} W_{ij} h_i \frac{v_j}{\sigma_j} + \mathbf{c}^T \mathbf{h} \right\} / Z. \tag{2.15}$$

The joint probability (2.23) yields the following marginalized probabilities:

$$p(\mathbf{v}) = \prod_{i=1}^{M} [1 + \exp(\mathbf{w}_i \Sigma \mathbf{v} + c_i)] \exp\left\{ -\frac{(\mathbf{v}-\mathbf{b})^T \Sigma^{-2} (\mathbf{v}-\mathbf{b})}{2} \right\} / Z, \tag{2.16}$$

$$p(\mathbf{h}) = \exp\left\{ \frac{||W^T \mathbf{h}||^2}{2} + (\Sigma^{-1} W \mathbf{b} + \mathbf{c})^T \mathbf{h} \right\} / Z. \tag{2.17}$$

where we define the i-th row vector of the matrix W by $\mathbf{w}_i$ and the squared norm of a vector by $|| \cdot ||^2$. As easily confirmed, The distribution $p(\mathbf{v})$ is composed of summation over 2^M Gaussian distribution. Therefore, we can recognize the expressive power of the Gaussian-Bernoulli RBM as a subspace of a mixture of Gaussian. It is also noticeable that we can interpret the distribution of hidden states, $p(\mathbf{h})$, as the fully-connected Boltzmann machine with the connectivity WW^T. In addition, The conditional probabilities become

$$p(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^{M} g(\mathbf{w_i} \Sigma^{-1} \mathbf{v} + c_i)^{h_i} \{1 - g(\mathbf{w_i} \Sigma^{-1} \mathbf{v} + c_i)\}^{1-h_i}, \tag{2.18}$$

$$p(\mathbf{v}|\mathbf{h}) = \mathcal{N}\left(\mathbf{v}; \Sigma W^T \mathbf{h} + \mathbf{b}, \Sigma^2\right), \tag{2.19}$$

where we denote a sigmoid function as $g(\cdot)$.

As assumed in Gaussian-Gaussian RBM, we set the model variances Σ^2 to fixed values and the bias parameters to $\mathbf{b} = \mathbf{c} = \mathbf{0}$ in the following learning rules.

ML learning rule

Assuming $\mathbf{b} = \mathbf{c} = \mathbf{0}$ for simplicity, we obtain the ML learning rule,

$$\tau \frac{dW}{dt} = < g(W\Sigma^{-1}\mathbf{v})\mathbf{v}^T >_0 \Sigma^{-1} - KW, \tag{2.20}$$

where $g(\mathbf{x})$, a sigmoid function with a vector argument $\mathbf{x}$, denotes the vector whose i-th element is $g(x_i)$. The first term $< \cdot >_0$ is the average over the input distribution $q(\mathbf{v})$. The matrix K in the second term is the average over $p(\mathbf{h})$ represents the interaction between hidden units, i.e.,

$$K_{ij} = \frac{\sum_{\mathbf{h}} h_i h_j \exp\left(||W^T\mathbf{h}||^2/2\right)}{\sum_{\mathbf{h}} \exp(||W^T\mathbf{h}||^2/2)}. \tag{2.21}$$

To compute K_{ij}, we need to take summation over 2^M hidden states and it takes too much computational time in practice.

CD$_n$ learning rule

Regarding the CD learning rule (2.10), the first term is the same as that in the ML learning rule. The second term is difficult to formulate explicitly in terms of the weight matrix W:

$$\tau\frac{dW}{dt} =< g(W\Sigma^{-1}\mathbf{v})\mathbf{v}^T >_0 \Sigma^{-1} - < g(W\Sigma^{-1}\mathbf{v})\mathbf{v}^T >_n \Sigma^{-1}, \tag{2.22}$$

where $< \cdot >_n$ represents the average over the distribution $p_n(\mathbf{v})$.

Although it is difficult to derive all fixed points of the learning rules (2.20) and (2.22), we will obtain one of the stable fixed points under the assumptions of $\Sigma = \sigma I_N$, $M = N$, and the other certain conditions in Section 2.4.

2.2.3 Bernoulli-Bernoulli RBM

Although we focus on the RBMs with Gaussian visible units in this study, let us remark Bernoulli-Bernoulli RBM (sometimes referred to as "binary RBM" or just "RBM") for comparison. The Bernoulli-Bernoulli RBM has binary visible and hidden variables $v_i, h_i = \{0, 1\}$, whose joint probability distribution is defined as follows [42]:

$$p(\mathbf{h}, \mathbf{v}) = \exp(\mathbf{h}^{\mathbf{T}} W\mathbf{v} + \mathbf{b}^T\mathbf{v} + \mathbf{c}^T\mathbf{h})/Z. \tag{2.23}$$

The joint probability (2.23) yields the following marginalized and conditional probabilities:

$$p(\mathbf{v}) = \prod_{i=1}^{M}[1 + \exp(\mathbf{w}_i\mathbf{v} + c_i)]\exp(\mathbf{b}^T\mathbf{v})/Z, \tag{2.24}$$

$$p(\mathbf{h}) = \prod_{j=1}^{N}[1 + \exp(\mathbf{w}_j^T\mathbf{h} + b_j)]\exp(\mathbf{c}^T\mathbf{h})/Z, \tag{2.25}$$

$$p(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^{M} g(\mathbf{w_i}\mathbf{v} + c_i)^{h_i}\{1 - g(\mathbf{w_i}\mathbf{v} + c_i)\}^{1-h_i}, \tag{2.26}$$

$$p(\mathbf{v}|\mathbf{h}) = \prod_{j=1}^{N} g(\mathbf{w_j}^T\mathbf{v} + b_j)^{v_j}\{1 - g(\mathbf{w_j}^T\mathbf{v} + b_j)\}^{1-v_j}, \tag{2.27}$$

where we define the j-th row vector of the matrix W^T by $\mathbf{w}_j^T$.

The structure of the probability density function $p(\mathbf{v})$ is seemingly similar to that of the Gaussian-Bernoulli RBM. However, they are essentially different from each other as is intuitively described below. Let us relax the domain of the visible random variables from binary numbers to continuous real ones. In this case, we can rewrite $p(\mathbf{v})$ of the Bernoulli-Bernoulli RBM by using delta functions such as

$$p(\mathbf{v}) = \lim_{\sigma \to 0} \prod_{i=1}^{M} [1 + \exp(\mathbf{w}_i \mathbf{v} + c_i)] \exp(\mathbf{b}^T \mathbf{v}) \prod_{j=1}^{N} \left\{ e^{-\frac{v_j^2}{2\sigma^2}} + e^{-\frac{(v_j-1)^2}{2\sigma^2}} \right\} / (\sqrt{2\pi}\sigma Z). \tag{2.28}$$

Before taking the limit for the delta function, we can recognize this distribution as the mixture of two types of Gaussian distributions $\mathcal{N}(0, \sigma^2)$ and $\mathcal{N}(1, \sigma^2)$. In contrast, $p(\mathbf{v})$ of the Gaussian-Bernoulli RBM is composed from only one Gaussian distribution $\mathcal{N}(\mathbf{b}, \Sigma)$ and has a simpler structure than that of the Bernoulli-Bernoulli RBM.

2.3 Principal component extraction in Gaussian-Gaussian RBM

The previous section gave the ML learning rule (2.9) and CD$_n$ learning rule (2.13) for the Gaussian-Gaussian RBM. These learning rules are non-linear differential equations of the weight matrix W and are difficult to solve analytically. Here, let us assume that the variances of the visible and hidden units are homogeneous, i.e.,

$$\Sigma = \sigma I_N, \tag{2.29}$$

$$S = s I_M. \tag{2.30}$$

In this case, we demonstrate that both ML and CD$_n$ learning have the same analytical form of stable fixed points. In addition, our analysis reveals that principal components of input data are extracted in the Gaussian-Gaussian RBM at the stable fixed points.

2.3.1 ML Solutions

By setting $dW/dt = O$ in the ML learning rule (2.9), we obtain the equation of the equilibrium state, i.e.,

$$WC = \sigma^2 W (I_N - W^T W)^{-1}. \tag{2.31}$$

We can derive the following lemma about the fixed points W satisfying (2.31). As preparation, let us denote a singular value decomposition of W by $W = UAV$, where U is an $M \times M$ orthogonal matrix, A is an $M \times N$ diagonal matrix, and V is an $N \times N$ orthogonal matrix.

Lemma 1 *Assume that the covariance matrix C has non-degenerate eigenvalues λ_i $(i = 1,...,N)$, which satisfy $\lambda_1 > \lambda_2 > \cdots > \lambda_N$. When there are k eigenvalues larger than σ^2 $(k = 0,...,N)$ such as $\lambda_i > \sigma^2$, a necessary and sufficient condition for $W = UAV$ to satisfy (2.31) is 1) U is an arbitrary $M \times M$ orthogonal matrix, and 2) A is an $M \times N$ diagonal matrix whose diagonal entries are $A_{ll} = \sqrt{1 - \frac{\sigma^2}{\lambda_{i_l}}}$ $(l = 1,...,m)$ and all the other entries are zero. The m distinct eigenvalues $\{\lambda_{i_1},...,\lambda_{i_m}\}$ $(m \le \{k,M\})$ are chosen from a set of the largest eigenvalues $\{\lambda_1,...,\lambda_k\}$, and 3) V diagonalizes the covariance matrix such that $C = V^T \mathrm{diag}(\lambda_1,...,\lambda_N)V$.*

Proof Without loss of generality, we can represent A to be an $M \times N$ diagonal matrix with $A_{ll} = \alpha_l$ $(l = 1,...,m)$ and $A_{ll'} = 0$ for all the other entries, where we denote the rank of W as m. Note that the case $m = 0$ where all diagonal entries satisfy $A_{ll} = 0$ is a trivial fixed point $W = O$. Substituting a singular value decomposition $W = UAR$ into (2.31), we obtain $A\left\{RCR^T(I_N - A^TA) - \sigma^2 I_N\right\} = O$. This equation needs RCR^T to become

$$RCR^T = \begin{bmatrix} \mathrm{diag}\left(\frac{\sigma^2}{1-\alpha_1^2},...,\frac{\sigma^2}{1-\alpha_m^2}\right) & O \\ O & Q \end{bmatrix}. \tag{2.32}$$

Note that Q is an $(N-m) \times (N-m)$ symmetric matrix. Diagonalizing Q by using an orthogonal matrix P and a diagonal matrix D_Q such that $Q = P^T D_Q P$, we obtain

$$C = \left(\begin{bmatrix} I_m & O \\ O & P \end{bmatrix} R\right)^T \begin{bmatrix} \mathrm{diag}\left(\frac{\sigma^2}{1-\alpha_1^2},...,\frac{\sigma^2}{1-\alpha_m^2}\right) & O \\ O & D_Q \end{bmatrix} \left(\begin{bmatrix} I_m & O \\ O & P \end{bmatrix} R\right). \tag{2.33}$$

Here, we diagonalize $C = V^T \mathrm{diag}(\lambda_1,...,\lambda_N)V$ and assume that λ_i $(i = 1,...,N)$ are non-degenerate eigenvalues of C. They can be ordered as $\lambda_1 > \lambda_2 > \cdots > \lambda_N$. It is necessary for R to become $R = \begin{bmatrix} I_m & O \\ O & P^T \end{bmatrix} V$. In addition, α_l must be $\alpha_l = \sqrt{1 - \sigma^2/\lambda_{i_l}}$ $(l = 1,...,m)$ and $\alpha_l = 0$ $(l = m+1,...,M)$. The m eigenvalues λ_{i_l} $(l = 1,...,m)$ are chosen from a set of the largest eigenvalues $\{\lambda_1,...,\lambda_k\}$ and should satisfy $m \le k$. Because m is the rank of W, m is equal to or less than M by definition. Therefore, $m \le \min\{k,M\}$ is necessary. After all this, we obtain

$$W = UAR = U \begin{bmatrix} \mathrm{diag}\left(\sqrt{1-\frac{\sigma^2}{\lambda_{i_1}}}, ..., \sqrt{1-\frac{\sigma^2}{\lambda_{i_m}}}\right) & O \\ O & O \end{bmatrix} \begin{bmatrix} I_m & O \\ O & P^T \end{bmatrix} V = UAV. \tag{2.34}$$

The sufficient condition that the derived $W = UAV$ satisfies (2.31) is easily confirmed. Therefore, $W = UAV$ is a necessary and sufficient condition for W to satisfy (2.31). $\square$

As we can see from the analytical form $W = UAV$ shown in Lemma 1, ML learning extracts principal component vectors corresponding to larger eigenvalues $\lambda_i > \sigma^2$ and reduces the dimension of the input data. The extracted eigenvectors are rescaled by $\sqrt{1-\sigma^2/\lambda_i}$. Remarkably, the principal components extracted by ML learning are characterized by the model variance of the visible units σ^2, not that of the hidden units s^2.

If one substitutes $W = UAV$ into the model distribution, one can obtain the model distribution of the visible units $p(\mathbf{v})$ as the following Gaussian distribution:

$$p(\mathbf{v}) = \mathcal{N}\left(\mathbf{v}; \mathbf{0}, V^T \mathrm{diag}(\lambda_1, ..., \lambda_m, \sigma^2, ..., \sigma^2) V\right). \tag{2.35}$$

Let us define η_i as the eigenvalues of the model covariance $\int d\mathbf{v} p(\mathbf{v}) \mathbf{v}\mathbf{v}^T$. The model distribution (2.35) leads to $\eta_i = \lambda_i$ $(i = 1, ..., m)$, $\eta_i = \sigma^2$ $(i = m+1, ..., N)$. This means that the smaller eigenvalues λ_i of the data covariance are replaced by the model variance σ^2.

Because the whole set of fixed points is represented by $W = UAV$ with an arbitrary orthogonal matrix U, the solution space has rotational degrees of freedom. In addition, there exist other degrees of freedom for the rank of W, that is, m $(0 \leq m \leq \min\{k, M\})$. In the following Theorem 1, we will show that m is limited to a certain value when we consider the stability of W.

2.3.2 Stability of ML Solutions

We derived the analytical form of the fixed points, $W = UAV$, for the ML learning. However, they include not only stable but also unstable points, to which the learning rule never converges. A theoretical perturbation analysis around the fixed points leads to a necessary and sufficient condition for stability.

Theorem 1 *The fixed points $W = UAV$ obtained in Lemma 1 are stable if and only if the diagonal matrix A has a set of the largest eigenvalues, $A_{ll} = \sqrt{1-\frac{\sigma^2}{\lambda_l}}$ $(l = 1, ..., m)$, with $m = \min\{k, M\}$.*

Proof Let us denote the gradient of the ML learning as $F(W) \equiv WC - \sigma^2 W(I_N - W^T W)^{-1}$. To prove that a fixed point is stable, we should show that the inner product between any perturbation ΔW and $\Delta F \equiv F(W + \Delta W)$ is negative around the fixed point. The first-order approximation of ΔF expanded by ΔW becomes

$$\begin{aligned}\Delta F \sim \; & \Delta W\{C - \sigma^2(I_N - W^T W)^{-1}\} \\ & - \sigma^2 W(I_N - W^T W)^{-1}(\Delta W^T W + W^T \Delta W)(I_N - W^T W)^{-1}.\end{aligned} \tag{2.36}$$

Here, we can represent the perturbation as $\Delta W = \sum_{ab} dW_{ab} U E^{(ab)} V$ without loss of generality, where dW_{ab} is an infinitesimal change, and entries of $M \times N$ matrix $E^{(ab)}$ are zero except for the a-th row and the b-th column entry $E^{(ab)}_{ab} = 1$. We set the matrix U to the same orthogonal matrix as that of the fixed point $W = UAV$, whose stability we are now checking. The matrix $UE^{(ab)}V$ means perturbing the a-th row of W by the b-th eigenvector of the covariance matrix C on the coordinate rotated by U. Note that one can calculate the inner product by using a matrix trace such as

$$\mathrm{Tr}(\Delta W^T \Delta F) = \sum_{i=1}^{M} \sum_{j=1}^{N} \Delta W_{ij} \Delta F_{ij}. \tag{2.37}$$

After we substitute the expansion (2.36) and $\Delta W = \sum_{ab} dW_{ab} U E^{(ab)} V$ into the inner product $\mathrm{Tr}(\Delta W^T \Delta F)$, we obtain a quadratic form of dW_{ab} ($a = 1, ..., \min\{M,N\}$, $b = 1, ..., \min\{M,N\}$) as

$$\mathrm{Tr}(\Delta W^T \Delta F) \sim -\sum_{a>b} \begin{bmatrix} dW_{ab} & dW_{ba} \end{bmatrix} Q^{(ab)} \begin{bmatrix} dW_{ab} \\ dW_{ba} \end{bmatrix} - \sum_a r_a (dW_{aa})^2, \tag{2.38}$$

where,

$$Q^{(ab)} \equiv \begin{bmatrix} s^{(ab)}\alpha_a^2 - t^{(b)} & s^{(ab)}\alpha_a\alpha_b \\ s^{(ab)}\alpha_a\alpha_b & s^{(ab)}\alpha_b^2 - t^{(a)} \end{bmatrix}, \tag{2.39}$$

$$r^{(a)} \equiv 2s^{(aa)}\alpha_a^2 - t^{(a)}, \tag{2.40}$$

$$s^{(ab)} \equiv \frac{\sigma^2}{(1-\alpha_a^2)(1-\alpha_b^2)}, \tag{2.41}$$

$$t^{(a)} \equiv \lambda_a - \frac{\sigma^2}{1-\alpha_a^2}. \tag{2.42}$$

Let us repeat that α_l denotes $\alpha_l = \sqrt{1 - \sigma^2/\lambda_{i_l}}$ ($l = 1, ..., m$), 0 ($l = m+1, ..., N$).

A necessary and sufficient condition for W to be stable is that any matrix $Q^{(ab)}$ has non-negative eigenvalues and that any r_a is also non-negative:

$$\begin{cases} Q_{ii}^{(ab)} \geq 0 \ \ (i = 1, 2), \\ \det(Q^{(ab)}) \geq 0, \\ r^{(a)} \geq 0. \end{cases} \tag{2.43}$$

The stable points are slightly different in the following two cases: $k \leq M$ and $M < k$. (i) If $k \leq M$, $r^{(a)}$ becomes negative when we take the index a as $m < a \leq k$. This means that the fixed point with $m < k$ is unstable. Because m must satisfy $m \leq k$ as proved in Lemma 1, $m = k$ is necessary for the stable points. In the case of $m = k$, all the conditions (2.44) are satisfied for any a and b, and the fixed point becomes stable. (ii) If $M < k$, because m is the rank of W, the value of m is limited to $m \leq M$ by definition. In the same manner as case (i), $m = M$ is necessary for the stable points. In addition, let us consider the case where an eigenvalue λ_i such that $\lambda_i < \lambda_M$ is extracted in a singular value $\alpha_l = \sqrt{1 - \sigma^2/\lambda_i}$. In this case, there is an eigenvalue λ_j such that $\lambda_j > \lambda_M$, which does not appear in the other singular values $\alpha_{l'}$ $(l' \neq l)$, and $Q_{11}^{(ab)}$ becomes negative, i.e., $Q_{11}^{(ab)} = \lambda_i - \lambda_j < 0$. Therefore, the stable fixed point requires all the M largest eigenvalues $\{\lambda_1, ..., \lambda_M\}$ to be extracted in α_l $(l = 1, ..., M)$. This fixed point with $m = M$ and the M largest eigenvalues satisfies all the conditions (2.44) for any a and b and becomes stable. □

As indicated by Theorem 1, in the case of $k \leq M$, the ML learning converges to the stable fixed points with rank $m = k$ and extracts all the eigenvalues larger than σ^2. If all of the input eigenvalues are smaller than σ^2, the trivial solution $W = O$ becomes stable. In the case of $M < k$, where the number of hidden units is small enough, the ML learning extracts only the largest M eigenvalues $\{\lambda_1, ..., \lambda_M\}$ among $\{\lambda_1, ..., \lambda_k\}$. This means that the Gaussian-Gaussian RBM can extract only the largest principal components by reducing the model variance or the number of hidden units.

Ir is noteworthy that the threshold of eigenvalues, that is, σ^2, is caused by a spatial resolution of the model over the input space. The Gaussian model distribution $p(\mathbf{v})$ has a covariance matrix $\sigma^2(I - W^T W)$, whose eigenvalues η always satisfy $\eta \geq \sigma^2$. This means that the lowest variance of the distribution $p(\mathbf{v})$ is determined by σ^2. We can also confirm the spatial resolution of the model by the conditional distribution $p(\mathbf{v}|\mathbf{h}) = \mathcal{N}(\sigma W^T \mathbf{h}, \sigma^2)$. This distribution means that the input space observed by a hidden state $\mathbf{h}$ has the constant variance σ^2. This spatial resolution may be regarded as a receptive field of $\mathbf{h}$ in terms of neuroscience. Therefore, the components of the input with eigenvalues larger than σ^2 are crucial to increase the likelihood of the model but those with smaller ones are negligible.

It is also remarkable that, if we ignore the rotational degrees of freedom caused by U, the stable point is unique and is the global minimum of the likelihood function. All the other fixed points are unstable and correspond to saddle points. There are no local minima. Figure 2.2 demonstrates the conceptual diagram of this landscape.

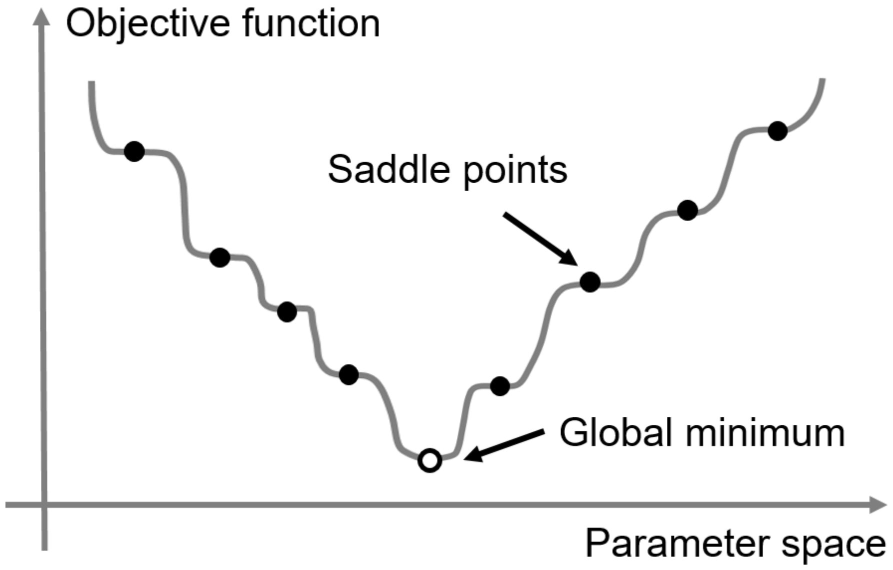


Fig. 2.2 Landscape of Likelihood function in Gaussian-Gaussian RBM.

2.3.3 CD$_n$ solutions

In the same way as shown for ML learning, we can also derive an analytical form for fixed points of CD$_n$ learning rule. We obtain the following equation by setting $dW/dt = 0$ in the CD$_n$ learning rule (2.13):

$$WC = W\left\{(W^TW)^nC(W^TW)^n + \sigma^2\sum_{k=0}^{2n-1}(W^TW)^k\right\}. \tag{2.44}$$

We can derive the following lemma for the fixed points satisfying (2.44).

Lemma 2 *Assume that the covariance matrix C has non-degenerate eigenvalues λ_i $(i = 1,\ldots,N)$, which satisfy $\lambda_1 > \lambda_2 > \cdots > \lambda_N$. When there are k eigenvalues larger than σ^2 $(k = 0,\ldots,N)$ such as $\lambda_i > \sigma^2$, a necessary and sufficient condition for $W = UAV$ to satisfy (2.44) is 1) U is an arbitrary $M \times M$ orthogonal matrix, and 2) A is an $M \times N$*

diagonal matrix whose diagonal entries are $A_{ll} = \sqrt{1 - \frac{\sigma^2}{\lambda_{i_l}}}$ $(l = 1, ..., m)$ *and all the other entries are zero. The m distinct eigenvalues* $\{\lambda_{i_1}, ..., \lambda_{i_m}\}$ $(m \leq \min\{k, M\})$ *are chosen from a set of the largest eigenvalues* $\{\lambda_1, ..., \lambda_k\}$*, and 3) V diagonalizes the covariance matrix such that* $C = V^T \mathrm{diag}(\lambda_1, ..., \lambda_N) V$.

Proof Without loss of generality, we can represent A to be an $M \times N$ diagonal matrix with $A_{ll} = \alpha_l$ $(l = 1, ..., m)$ and $A_{ll'} = 0$ for all the other entries, where we denote the rank of W as m. Substituting a singular value decomposition $W = UAR$ into (2.44), we obtain

$$A \left\{ (1 - A^{2n}) RCR^T (1 - A^{2n}) - \sigma^2 \sum_{k=0}^{2n-1} A^{2k} \right\} = 0, \tag{2.45}$$

for any i and j. This equation needs RCR^T to satisfy (2.32). From here, we can prove Lemma 2 by using the same process as shown in Lemma 1. □

Lemmas 1 and 2 clarify that the fixed points of ML and CD$_n$ learning rules have the same analytical form: $W = UAV$. Namely, the whole set of fixed points including stable and unstable points in CD$_n$ learning coincides with that in ML learning.

2.3.4 Stability of CD$_n$ solutions

We can also derive the following necessary and sufficient condition for the stability of CD$_n$ solutions.

Theorem 2 *The fixed points* $W = UAV$ *obtained in Lemma 2 are stable if and only if the diagonal matrix A has a set of the largest eigenvalues,* $A_{ll} = \sqrt{1 - \frac{\sigma^2}{\lambda_l}}$ $(l = 1, ..., m)$*, with* $m = \min\{k, M\}$.

Proof Let us denote the perturbation of the gradient (2.13) by ΔF, which consists of the following first order approximation of $(W^T W)^k$:

$$(\bar{W}^T \bar{W})^k \sim (W^T W)^k + \sum_{i=0}^{k-1} (W^T W)^{k-i-1} (W^T \Delta W + \Delta W^T W)(W^T W)^i, \tag{2.46}$$

where $\bar{W} = W + \Delta W$ with the fixed point $W = UAV$. Substituting the above expansion to the CD$_n$ gradient, we obtain the following ΔF:

$$\begin{aligned}\Delta F = \Delta W\{C - (X^n C X^n + \sigma^2 \sum_{k=1}^{2n-1} X^k)\} \\ - W \sum_{i=0}^{n-1} \{X^n C X^{n-i-1}(W^T \Delta W + \Delta W^T W) X^i \\ + X^{n-i-1}(W^T \Delta W + \Delta W^T W) X^i C X^n\} \\ - \sigma^2 W \sum_{k=1}^{2n-1} \sum_{i=0}^{k-1} X^{k-i-1}(W^T \Delta W + \Delta W^T W) X^i,\end{aligned} \tag{2.47}$$

where we define $X = W^T W$.

For the inner product $\mathrm{Tr}(\Delta W^T \Delta F)$, we obtain the quadratic form (2.38) with the following $s^{(ab)}$ and $t^{(a)}$:

$$s^{(ab)} \equiv D_n^{(ab)}(\lambda_a \alpha_a^{2n} + \lambda_b \alpha_b^{2n}) + \sigma^2 \sum_{k=0}^{2n-1} D_k^{(ab)}, \tag{2.48}$$

$$t^{(a)} \equiv (1 - \alpha_a^{4n}) \left(\lambda_a - \frac{\sigma^2}{1 - \alpha_a^2} \right), \tag{2.49}$$

where $D_k^{(ab)} = \sum_{i=0}^{k-1} \alpha_a^{2(k-1-i)} \alpha_b^{2i}$. By using these results, we can prove Theorem 2 by using the same process as shown in Theorem 1. □

Interestingly, as we can see from Theorem 2, the stability of CD$_n$ solutions coincides with that of ML solutions. Thus, we can conclude that if the CD$_n$ learning converges to any solution, CD$_n$ learning maximizes the likelihood function. Note that the analytical form of stable CD$_n$ solutions is independent of n, the number of Gibbs sampling. Therefore, the same principal components as in ML learning can be extracted simply by performing CD$_1$ learning.

It is also remarkable that the model variance parameters are likely to be fixed to unit values or the variances of the input data in practice [43]. This is because the learning of σ often causes unstable learning dynamics and the result of estimation becomes poor. In that sense, it is unnecessary to learn σ and our analysis with the fixed σ seems to be rational. Although we have remarked the dynamical analysis of the learning of σ in appendix B, its stable fixed points suggest that fixing σ is a reasonable condition.

2.3.5 Related works

The following two models of principal component analysis (PCA) have similar equilibrium solutions to Gaussian-Gaussian RBM.

Probabilistic PCA (PPCA) is a latent variable model of PCA, which gives the conditional distribution of the visible variables when the hidden variables are known [102]. The ML learning rule for PPCA is

$$\tau\frac{dW}{dt} = W(\sigma^2 I_N + W^T W)^{-1} C(\sigma^2 I_N + W^T W)^{-1} - W(\sigma^2 I_N + W^T W)^{-1}. \tag{2.50}$$

Tipping and Bishop proved that the ML solutions in PPCA model are
$W = U\mathrm{diag}\left(\sqrt{\lambda_1 - \sigma^2}, ..., \sqrt{\lambda_m - \sigma^2}, 0, ..., 0\right) V$, and they also proved the stability of the solutions [102] . ML learning of PPCA is similar to that of Gaussian-Gaussian RBM with regard to extracting principal components whose eigenvalues are larger than the model variance σ^2. However, the singular values extracted by using PPCA are different from those of Gaussian-Gaussian RBM.

Oja's subspace algorithm is a method for carrying out PCA in a single-layer neural network [78, 111]. Its learning rule is

$$\tau\frac{dW}{dt} = WC - WCW^T W. \tag{2.51}$$

The subspace algorithm extracts principal components as stable equilibrium solutions [79]. In this respect, the subspace algorithm is similar to the Gaussian-Gaussian RBM but different in that the singular values of W extracted by the subspace algorithm are always equal to 1.

We can also see that the subspace algorithm corresponds to a CD learning rule used to train the Gaussian-Gaussian RBM as follows. Consider CD learning, which is one less step of Gibbs sampling than CD$_1$, that is, $< \mathbf{h}\mathbf{v}^T >= \int d\mathbf{v}_1 d\mathbf{h}_0 d\mathbf{v}_0 p(\mathbf{v}_1|\mathbf{h}_0) p(\mathbf{h}_0|\mathbf{v}_0) q(\mathbf{v}_0) \mathbf{h}_0 \mathbf{v}_1^T = s(\sigma^{-1} WCW^T W + \sigma W)$. We call this CD$_{1/2}$ learning. This is computationally easier, having the learning rule $\tau dW/dt = WC - WCW^T W - \sigma^2 W$, which corresponds to the subspace algorithm with a weight decay, or L2 regularization. We can prove that this CD$_{1/2}$ learning has the same fixed points and stability as CD$_n$ learning.

2.3.6 Feature extraction realized by stacking Gaussian-Gaussian RBMs

One of the techniques to train deep networks is layerwise pretraining, which stacks RBMs [45]. Here, let us consider stacking Gaussian-Gaussian RBMs as the simplest model of the layerwise pretraining.

Because each G-G RBM reduces the input space by using principal components, stacking G-G RBMs realizes step-by-step noise reduction. First, G-G RBMs implicitly reduce the noise by the hyperparameter σ. Furthermore, if the number of hidden units is limited and smaller than the dimension of the input signals, G-G RBMs reduce the noise again. As you can see in the previous studies of unsupervised layerwise pretraining [45], the deep network composed by RBMs has a pyramidal network structure, where the number of hidden units is gradually decreased in higher layers. Considering stacking G-G RBMs, one can see that this network structure is crucial to reduce the effective dimension of the input data and to capture valuable information of the input.

Certainly, stacking Gaussian-Gaussian RBMs are poor as information processing because it performs only principal component analysis that can be realized by only one Gaussian-Gaussian RBM. However, we can analog the step-by-step noise reduction realized by the stacking Gaussian-Gaussian RBM to a step-by-step projection realized by stacking RBMs with nonlinear units. As known in several numerical experiments, RBMs with binary units project input data to a low dimensional feature space from layer to layer [31, 18]. For instance, after the training with MNIST dataset, strokes of the hand-written numbers are extracted in the first RBM. In the following RBMs at higher layers extract the combination of the strokes. Finally, the RBM in top layer extracts the manifold of each number, which is quite low dimensional compared to the number of pixels of the input samples [31]. Therefore, stacking RBMs with nonlinear units is the step-by-step projection to the low dimensional space, or so-called, *manifold unfolding* [18]. In the case of the stacked G-G RBMs, the counterpart of the projection to the low dimensional space appears as a noise reduction by using the principal components.

2.4 Independent component extraction in Gaussian-Bernoulli RBM

The dynamical analysis shown in the previous section can be extended to the Gaussian-Bernoulli RBM. Here, let us assume that the number of units is $N = M$, having homogeneous variances $\Sigma = \sigma I_N$. In this case, we found the following sufficient condition for stable fixed points of ML and CD$_n$ learning rules in the Gaussian-Bernoulli RBM.

2.4.1 ML solution

Theorem 3 *Assume that 1) the input distribution $q(\mathbf{v})$ is a linear mixture of independent source signals, that is, $\mathbf{v} = B\mathbf{s}$ and $q(\mathbf{s}) = q(s_1)q(s_2)\cdots q(s_N)$, where the mixing matrix B is an $N \times N$ orthogonal matrix, and 2) the source signals are positive, $s_i > 0$, and their means satisfy $\mu_i = \int ds_i q(s_i)s_i \gg \sigma$. Under these assumptions, a sufficient condition for stable fixed points of ML learning rule is $W = DB^T$, where $D = \sigma^{-1}\mathrm{diag}(\mu_1, \mu_2, ..., \mu_N)$.*

Proof Let us assume that a fixed point is represented by the product of a diagonal matrix $D = \mathrm{diag}(d_1, d_2, ..., d_N)$ and the unmixing matrix B^T, i.e., $W = DB^T$. Under assumption 1) in Theorem 3, we substitute $W = DB^T$ into the condition $dW/dt = 0$ and then obtain the following equations:

$$< s_i g\left(d_i s_i/\sigma\right) >_{q(s_i)} = \sigma d_i g\left(d_i^2/2\right), \tag{2.52}$$

$$\mu_i < g\left(d_j s_j/\sigma\right) >_{q(s_j)} = \sigma d_i g\left(d_i^2/2\right) g\left(d_j^2/2\right) \quad (i \neq j), \tag{2.53}$$

where $< \cdot >_{q(s_i)}$ represents the average over the distribution $q(s_i)$. Note that the left-hand sides of (2.52, 2.53) correspond to the first term of the ML gradient $< g\left(W\mathbf{v}/\sigma\right)\mathbf{v}^T >_0$, and the right hand sides correspond to the second term σKW.

Here, let us assume that d_i is very large, i.e., $d_i \gg 1$. This assumption allows the sigmoid functions to converge as follows:

$$g(d_i s_i/\sigma) \to 1 \quad (s_i > 0), \tag{2.54}$$

$$g(d_i^2/2) \to 1. \tag{2.55}$$

If we also assume positive source signals $s_i > 0$, the averages in (2.52, 2.53) converge to constant values as follows:

$$< s_i g\left(d_i s_i/\sigma\right) >_{q(s_i)} \to \mu_i, \tag{2.56}$$

$$< g\left(d_j s_j/\sigma\right) >_{q(s_j)} \to 1. \tag{2.57}$$

Therefore, the equations (2.52, 2.53) are reduced into $\mu_i = \sigma d_i \ \ (i = 1, ..., N)$. As a consequence, the weight matrix $W = DB^T$ with $d_i = \mu_i/\sigma \gg 1$ is a fixed point of the ML learning rule.

Next, we prove the stability of this $W = DB^T$. Let us denote the gradient of the ML learning as $F(W) \equiv < g(W\mathbf{v}/\sigma)\mathbf{v}^T >_0 - \sigma KW$. The gradient perturbed around the fixed point is expanded by the first-order of ΔW as follows:

$$\Delta F \sim < \Delta g \mathbf{v}^T >_0 - \sigma K \Delta W - \sigma \Delta K W. \tag{2.58}$$

The first term of ΔF comes from the first term of $F(W)$. The vector Δg denotes a perturbation of the sigmoid function, whose i-th element is given by

$$\Delta g_i \equiv g(\mathbf{w}_i \mathbf{v}/\sigma)\{1 - g(\mathbf{w}_i \mathbf{v}/\sigma)\}\Delta \mathbf{w}_i \mathbf{v}. \tag{2.59}$$

The second and third terms of ΔF come from the second term of $F(W)$. The matrix $\Delta K \equiv K(W + \Delta W) - K(W)$ represents a perturbation of the interaction K. We can represent the perturbation as $\Delta W = \sum_{ab} dW_{ab} E^{(ab)} B^T$ and expand ΔK by the first-order of ΔW as

$$\begin{aligned}\Delta K_{ij} = \sum_{ab} d_b [&< h_i h_j h_a h_b >_{p(\mathbf{h})} \\ &- < h_i h_j >_{p(\mathbf{h})} < h_a h_b >_{p(\mathbf{h})}] (dW_{ab} + dW_{ba}),\end{aligned} \tag{2.60}$$

where $< \cdot >_{p(\mathbf{h})}$ denotes the average over the following model distribution at the fixed point:

$$p(\mathbf{h}; W = DB^T) = \prod_i g(d_i^2/2)^{h_i} \{1 - g(d_i^2/2)\}^{1-h_i}. \tag{2.61}$$

Because $p(\mathbf{h}; W = DB^T)$ is independent among the hidden variables h_i, we can explicitly calculate $< \cdot >_{p(\mathbf{h})}$ as the products of the sigmoid functions $g(d_i^2/2)$.

Here, the convergences (2.54, 2.55) let each term of ΔF converge as follows:

$$< \Delta g_i v_j >_0 \to 0, \tag{2.62}$$

$$\Delta K_{ij} \to 0, \tag{2.63}$$

$$K_{ij} \to 1. \tag{2.64}$$

Therefore, the inner product becomes

$$\begin{aligned}\mathrm{Tr}(\Delta W^T \Delta F) &\to -\sigma \mathrm{Tr}(\Delta W^T \mathbf{1}\mathbf{1}^T \Delta W) \\ &= -\sigma \sum_a (\sum_b dW_{ab})^2.\end{aligned} \tag{2.65}$$

The notation **1** denotes a vector whose elements are all 1. Since we obtain the negative inner product $\mathrm{Tr}(\Delta W^T \Delta F) < 0$ for any perturbation, $W = DB^T$ is a stable point. □

At stable fixed point $W = DB^T$ obtained in Theorem 3, the Gaussian-Bernoulli RBM functions as independent component analysis (ICA). It is equivalent to the unmixing matrix, which is obtained by ICA [7, 48]. Namely, $W = DB^T$ separates the input data $\mathbf{v}$ into independent source signals $\mathbf{s}$ such as $W\mathbf{v} = DB^T B\mathbf{s} = D\mathbf{s}$. Therefore, the Gaussian-Bernoulli RBM decomposes the mixed input signals to the independent source signals. In addition, each hidden unit h_i becomes the detector of each independent source s_i as is confirmed by a conditional distribution:

$$p(h_i = 1|\mathbf{v}) = \text{sigmoid}(\mu_i s_i/\sigma^2). \tag{2.66}$$

Note that the limit $\mu_i \geq \sigma$ assumed in Theorem 3 corresponds to a deterministic situation. Under this situation, the conditional probability $p(h_i = 1|\mathbf{v})$ always gets activated when the information source s_i is given as the input. In addition, another conditional probability $p(\mathbf{v}|\mathbf{h})$, or the receptive field of the input space, also becomes deterministic under the limit of $\sigma \to 0$. This is obvious because $p(\mathbf{v}|\mathbf{h})$ is represented by $\mathbf{v} = \text{diag}(\mu)A^T\mathbf{h} + \sigma\mathcal{N}(0,1)$.

In the case of learning rules including bias parameters, we can also obtain a similar stable fixed point. In addition, we can relax assumption 2) of Theorem 3 as shown in Appendix A.

2.4.2 CD$_n$ solution

Theorem 4 *Under the same assumptions, 1) and 2), in Theorem 3, a sufficient condition for stable fixed points of CD$_n$ learning rule is $W = DB^T$, where $D = \sigma^{-1}\text{diag}(\mu_1, \mu_2, ..., \mu_N)$.*

Proof Let us assume that a fixed point is represented by $W = DB^T$ as assumed in the proof of Theorem 3. Because the first term of CD$_n$ gradient is the same as that of ML gradient, what we should analyze here is the second term $< g(W\mathbf{v}/\sigma)\mathbf{v}^T >_n$. Transforming visible variables such as $\mathbf{v}_k = B\mathbf{s}_k$ $(k = 1, ..., n)$ and substituting $W = DB^T$, the distribution $p_n(\mathbf{v})$ is transformed into the following form:

$$p_n(\mathbf{s}_n) = \prod_{k=0}^{n-1}\prod_{i=1}^{N}\sum_{h_{k,i}}\int ds_{k,i}\, p(s_{k+1,i}|h_{k,i})p(h_{k,i}|s_{k,i})q(s_{0,i}), \tag{2.67}$$

where $s_{k,i}$ denotes the i-th element of $\mathbf{s}_k$, the sample obtained after k iterated Gibbs sampling. It is a remarkable property that the matrix $W = DB^T$ decomposes the chain of Gibbs sampling into the products of each chain for the i-th variables s_i and h_i. The conditional distributions are written by:

$$p(s_{k+1,i}|h_{k,i}) = \mathcal{N}\left(s_{k+1,i}; \sigma d_i h_{k,i}, \sigma^2\right), \tag{2.68}$$

$$p(h_{k,i}|s_{k,i}) = g(d_i s_{k,i}/\sigma)^{h_{k,i}}\{1-g(d_i s_{k,i}/\sigma)\}^{1-h_{k,i}}. \tag{2.69}$$

When we assume that d_i is very large, i.e., $d_i \gg 1$, $p_n(\mathbf{s})$ is analytically tractable and we can obtain the explicit expression of the second term
$< g(W\mathbf{v}/\sigma)\mathbf{v}^T >_n$ as below. The assumption $d_i \gg 1$ lets the conditional distributions $p(h_{k,i}|s_{k,i})$ converge as follows:

$$p(h_{k,i}=1|s_{k,i}) \to 1, \tag{2.70}$$

$$p(h_{k,i}=0|s_{k,i}) \to 0 \;\; (s_{k,i} > 0). \tag{2.71}$$

If the previous sample $s_{k,i}$ is positive, the above convergences let the distribution of the next sample $s_{k+1,i}$ converge to the following Gaussian distribution ($k = 0,...,n-1$):

$$\begin{aligned} p(s_{k+1,i}|s_{k,i}) &= \sum_{h_{k,i}} p(s_{k+1,i}|h_{k,i})p(h_{k,i}|s_{k,i}) \\ &\to \mathcal{N}(s_{k+1,i}; \sigma d_i, \sigma^2). \end{aligned} \tag{2.72}$$

Let us repeat that we assumed positive source signals $s_{0,i} > 0$. In addition, under the assumption that d_i is very large, i.e., $d_i \gg 1$, the Gaussian distribution (2.72) has a large mean value and mostly generates positive samples $s_{k+1,i} > 0$. Therefore, the iterated integral (2.67) with a finite number of n is reduced to the sequential integral of the Gaussian distribution (2.72), and we obtain

$$p_n(s_{n,i}) \to \mathcal{N}(s_{n,i}; \sigma d_i, \sigma^2). \tag{2.73}$$

As a result, we can evaluate the second term of CD$_n$ gradient as follows:

$$\begin{aligned} < g(\mathbf{w}_i\mathbf{v}/\sigma)\mathbf{v} >_n &\to \int d\mathbf{s}_n \mathcal{N}(\mathbf{s}_n; \sigma\mathbf{d}, \sigma^2 I_N) B\mathbf{s}_n \\ &= \sigma B\mathbf{d}, \end{aligned} \tag{2.74}$$

where $\mathbf{d} = (d_1, d_2, ..., d_N)^T$.

Additionally, the first term of CD$_n$ gradient $< g(\mathbf{w}_i\mathbf{v}/\sigma)\mathbf{v} >_0$ converges to $\to B\mu$ by using the convergence (2.54). It coincides with the second term when $\mu = \sigma\mathbf{d}$. Therefore, $W = DB^T$ with $d_i = \mu_i/\sigma \gg 1$ $(i = 1,...,N)$ is a sufficient condition for the fixed point.

Next, we prove the stability. The gradient of CD$_n$ learning perturbed by ΔW becomes

$$\Delta F =< \Delta g \mathbf{v}^T >_0 - < \Delta g \mathbf{v}^T >_n - \int d\mathbf{v} g(W\mathbf{v}/\sigma)\mathbf{v}^T \Delta p_n(\mathbf{v}). \tag{2.75}$$

The first term of ΔF converges to 0 as shown in ML learning (2.62-2.64). In the same way, the second term of ΔF also converges to 0. In the following proof, we will show to what values the third term of ΔF converges.

The third term corresponds to a perturbation of $p_n(\mathbf{v})$ and is decomposed into three terms:

$$\begin{aligned}\Delta p_n(\mathbf{v}_n) = &\sum_{k=0}^{n-1}\sum_{\mathbf{h}_k} p(\mathbf{v}_n|\mathbf{h}_k)\int d\mathbf{v}_k \Delta p(\mathbf{h}_k|\mathbf{v}_k)p_k(\mathbf{v}_k)\\ &+\sum_{k=0}^{n-2}\int d\mathbf{v}_{k+1}d\mathbf{v}_k p(\mathbf{v}_n|\mathbf{v}_{k+1})\sum_{\mathbf{h}_k}\Delta p(\mathbf{v}_{k+1}|\mathbf{h}_k)p(\mathbf{h}_k|\mathbf{v}_k)p_k(\mathbf{v}_k)\\ &+\int d\mathbf{v}_{n-1}\sum_{\mathbf{h}_{n-1}}\Delta p(\mathbf{v}_n|\mathbf{h}_{n-1})p(\mathbf{h}_{n-1}|\mathbf{v}_{n-1})p_{n-1}(\mathbf{v}_{n-1}).\end{aligned} \tag{2.76}$$

The first term of $\Delta p(\mathbf{v}_n)$ consists of the conditional distribution perturbed around the equilibrium $W = DB^T$ such as $\Delta p(\mathbf{h}|\mathbf{v}) \equiv p(\mathbf{h}|\mathbf{v}; W+\Delta W) - p(\mathbf{h}|\mathbf{v}; W)$. The second and third terms of $\Delta p(\mathbf{v}_n)$ consist of another perturbed conditional distribution such as $\Delta p(\mathbf{v}|\mathbf{h}) \equiv p(\mathbf{v}|\mathbf{h}; W+\Delta W) - p(\mathbf{v}|\mathbf{h}; W)$. These conditional distributions are expanded by the first-order of ΔW:

$$\Delta p(\mathbf{h}|\mathbf{v}) \sim p(\mathbf{h}|\mathbf{v})\{\mathbf{h} - g(W\mathbf{v}/\sigma)\}^T \Delta W \mathbf{v}/\sigma, \tag{2.77}$$

$$\Delta p(\mathbf{v}|\mathbf{h}) \sim p(\mathbf{v}|\mathbf{h})(\mathbf{v} - \sigma W^T\mathbf{h})^T \Delta W^T \mathbf{h}/\sigma. \tag{2.78}$$

Let us transform variables such as $\mathbf{v}_k = B\mathbf{s}_k$ and replace ΔW with $\Delta W B^T$. Under the assumption $d_i \gg 1$, the following integral in the first term of $\Delta p_n(\mathbf{v})$ becomes

$$\begin{aligned}\int d\mathbf{s}\Delta p(\mathbf{h}|\mathbf{s})p_k(\mathbf{s}) &= \int d\mathbf{s} p(\mathbf{h}|\mathbf{s})p_k(\mathbf{s})\left\{\mathbf{h} - g(D\mathbf{s}/\sigma)\right\}^T \Delta W \mathbf{s}/\sigma\\ &\to 0.\end{aligned} \tag{2.79}$$

Note that $p_k(\mathbf{s})$ mostly generates positive samples $\mathbf{s}$, as you can see from (2.73). The positive sample $s_i > 0$ lets the conditional distribution $p(\mathbf{h}|\mathbf{s})$ in (2.79) obey the convergences (2.70, 2.71). Therefore, the first term of $\Delta p_n(\mathbf{v})$ converges to 0. For the second term of $\Delta p_n(\mathbf{v})$, by summing over **h** in the same way as shown in the convergence (2.72), we obtain

$$\begin{aligned}\sum_{\mathbf{h}} \Delta p(\mathbf{s}_{k+1}|\mathbf{h})p(\mathbf{h}|\mathbf{s}_k) &= \sum_{\mathbf{h}} p(\mathbf{s}_{k+1}|\mathbf{h})p(\mathbf{h}|\mathbf{s}_k)(\mathbf{s}_{k+1}-\sigma D\mathbf{h})^T \Delta W^T \mathbf{h}/\sigma \\ &\to \mathscr{N}(\mathbf{s}_{k+1};\sigma\mathbf{d},\sigma^2)(\mathbf{s}_{k+1}-\sigma\mathbf{d})^T\Delta W^T \mathbf{1}/\sigma. \end{aligned} \tag{2.80}$$

In addition, the transition probability $p(\mathbf{s}_n|\mathbf{s}_{k+1})$ converges to $\mathscr{N}(\mathbf{s}_n;\sigma\mathbf{d},\sigma^2 I_N)$. Calculating the integral of the product of (2.80) and $p(\mathbf{s}_n|\mathbf{s}_{k+1})$ over $\mathbf{s}_{k+1}$, we find that the second term of $\Delta p_n(\mathbf{v})$ converges to 0. In comparison, the third term of $\Delta p_n(\mathbf{v})$ converges to a nonzero value as follows:

$$\begin{aligned}&\int d\mathbf{s}_n g(D\mathbf{s}_n/\sigma)\mathbf{s}_n^T \int d\mathbf{s}_{n-1} \sum_{\mathbf{h}_{n-1}} \Delta p(\mathbf{s}_n|\mathbf{h}_{n-1})p(\mathbf{h}_{n-1}|\mathbf{s}_{n-1})p_{n-1}(\mathbf{s}_{n-1}) \\ &\to \int d\mathbf{s}_n \mathbf{1}\mathbf{s}_n^T \mathscr{N}(\mathbf{s}_n;\sigma\mathbf{d},\sigma^2)(\mathbf{s}_n-\sigma\mathbf{d})^T\Delta W^T\mathbf{1}/\sigma \\ &= \sigma\mathbf{1}\mathbf{1}^T\Delta W. \end{aligned} \tag{2.81}$$

After all this, the inner product converges to the same quadratic form as that of ML learning (2.65):

$$\begin{aligned}\mathrm{Tr}(\Delta W^T\Delta F) &\to -\sigma\mathrm{Tr}(\Delta W^T\mathbf{1}\mathbf{1}^T\Delta W) \\ &= -\sigma\sum_a(\sum_b dW_{ab})^2. \end{aligned} \tag{2.82}$$

Because the inner product always becomes negative, $W = DB^T$ is a stable point. □

We have thus proven the remarkable fact that CD$_n$ learning can obtain the same stable solution as ML learning and that the mixed input signals are decomposed into independent source signals (See also Appendix A and B).

Note that there are some trivial expansions of the theorems. In the theorems, we assumed that the number of hidden units M is equivalent to that of visible units N. We also required that the number of independent sources K equals M. In the same way as shown in the proof, when $M > N$, $W = DB^T$ with $D = \sigma^{-1}\mathrm{diag}(\mu_1,...,\mu_K,0,....,0)$ becomes a stable fixed point. The case $K < N$ is equivalent to $K = N$ with ($s_{K+1} = \cdots = s_N = 0$) and the $W = DB^T$ also becomes a stable fixed point. Interesting case is overcomplete independent sources, $K > N$ and $M > N$, but we cannot analyze this case as an extension of the above theorems and it remains to be an open problem.

2.4.3 Marginal distribution $p(\mathbf{v})$ at ICA solution

In Gaussian-Bernoulli RBM with the uniform model variance σ^2, the marginal distribution of visible variables becomes

$$p(\mathbf{v}) = \Pi_i[1+\exp(\sigma^{-1}\mathbf{w}_i\mathbf{v})]\exp\left(-\frac{\mathbf{v}^2}{2\sigma^2}\right)/Z, \tag{2.83}$$

where Z in a normalization constant. As described in Section 2.2.2, this distribution is composed of the summation over 2^M Gaussian distributions. Wang, Melchior, and Wiskott experimentally demonstrated that the Gaussian distributions composing $p(\mathbf{v})$ are likely to be allocated to independent components after training [106]. As shown in Fig. 2.3, our ICA solution realizes the same allocation and it means that this allocation is one of the maximum likelihood solutions.

Remarkably, Vinnikov and Shalev-Shwartz theoretically and experimentally demonstrated that the K-means method, one of the unsupervised learning algorithms for clustering, can extract independent components [105]. Since its clusters are allocated along independent components, this allocation seems to be similar to the allocation realized in Gaussian-Bernoulli RBM. Our ICA solution and the study of the K-means suggest that the unsupervised clustering algorithm is likely to extract independent components if the dataset is generated by a mixture of independent sources.

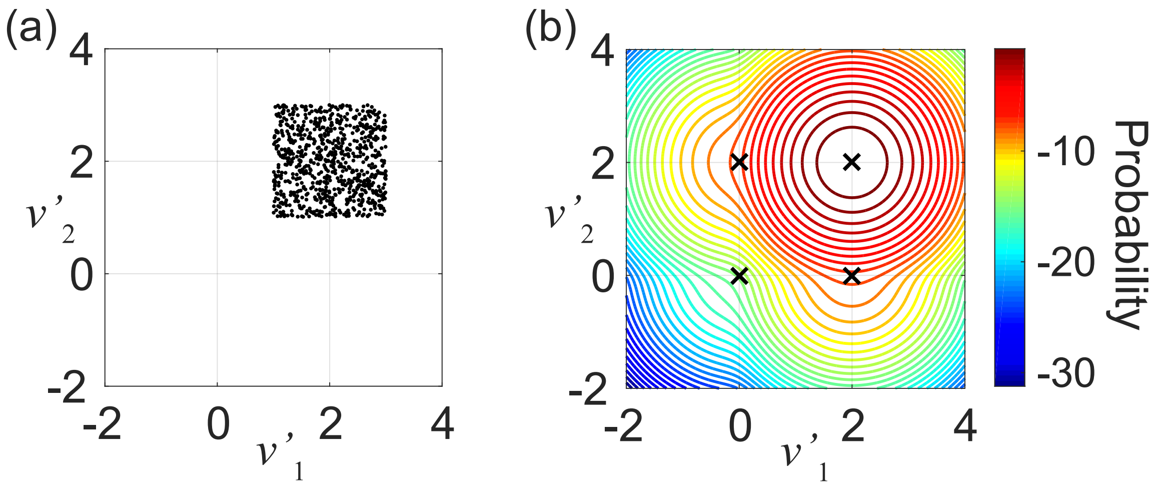


Fig. 2.3 Comparison between input distribution and model distribution in Gaussian-Bernoulli RBM. (a) Input distribution $q(\mathbf{v})$ observed in the coordinate system $\mathbf{v}' = A^T\mathbf{v}$. (b) Model distribution $p(\mathbf{v})$ at ICA solution. The color represents the log probability. Each black cross means the location of the mean of the Gaussian distribution, which composes the marginal distribution $p(\mathbf{v})$.

2.4.4 Related works

Theorem 4 gives a theoretical insight into several experimental results [65, 106]. In particular, Wang et al. trained a Gaussian-Bernoulli RBM by CD_1 learning in modeling natural images [106]. They demonstrated that a trained Gaussian-Bernoulli RBM is likely to obtain solutions close to those obtained by the ICA algorithm.

We can also consider the $\mathrm{CD}_{1/2}$ learning in the Gaussian-Bernoulli RBM. Its learning rule becomes

$$\tau \frac{dW}{dt} = < g(W\mathbf{v}/\sigma)\mathbf{v}^T >_0 - \sigma J W, \tag{2.84}$$

where the matrix J has entries $J_{ij} = < g(\mathbf{w}_i\mathbf{v}/\sigma)g(\mathbf{w}_j\mathbf{v}/\sigma) >_0 \; (i \neq j)$ and $J_{ii} = < g(\mathbf{w}_i\mathbf{v}/\sigma)^2 >_0$. In the case of the $\mathrm{CD}_{1/2}$ learning, we can also prove that it has the same fixed point and stability as CD_n learning. Note that, if we take $\sigma = 1$ and add an additional term $g(\mathbf{w}_i\mathbf{v}/\sigma)(1 - g(\mathbf{w}_i\mathbf{v}/\sigma))$ to the diagonal elements J_{ii}, the $\mathrm{CD}_{1/2}$ learning is equivalent to the nonlinear PCA rule. The nonlinear PCA rule is based on Hebbian learning between linear neurons in an input layer and the nonlinear neurons in another layer [80]. Oja demonstrated that it can extract the independent components of the input data at stable fixed points [80].

2.5 Experimental results

2.5.1 Gaussian-Gaussian RBM

Here, we show that the results of a simulation on the Gaussian-Gaussian RBM agree very well with the theorems described in Section 2.3.

Figure 2.4 (a) shows that the Gaussian-Gaussian RBM trained by ML learning extracted principal components corresponding to $\lambda_i > \sigma^2$. We set the number of units to $N = M = 10$ and the model variance to $\sigma^2 = s^2 = 1/4$. We artificially generated 10-dimensional input data in the following way. We first generated samples x_i $(i = 1, 2, ..., 10)$ from independent uniform distributions on [-1, 1]. To create input variables with small variances such that $\lambda_i < \sigma^2$, we scaled the sample vector $\mathbf{x}$ by diagonal matrix $D = \mathrm{diag}(0.2, 0.4, ..., 2)$. Then, to allow a covariance matrix of input data to have off-diagonal elements, we mixed the scaled samples $D\mathbf{x}$ by a random orthogonal matrix Q, i.e., $\mathbf{v} = QD\mathbf{x}$. We used these samples $\mathbf{v}$ as the input to the Gaussian-Gaussian RBM. The black points in Figure 2.4 (a) represent the principal eigenvalues λ_i of the input. Note that we can theoretically calculate them by $\lambda_i = D_{ii}^2 \mathrm{Var}[x_i]$ $(i = 1, 2, ..., 10)$. The red points in Figure 2.4 (a) represent the mean and standard deviation of the principal eigenvalues extracted in the weight matrix, η_i, in 20 simulation trials. Note that we estimated η_i, where batch input consisted of 1000 training examples

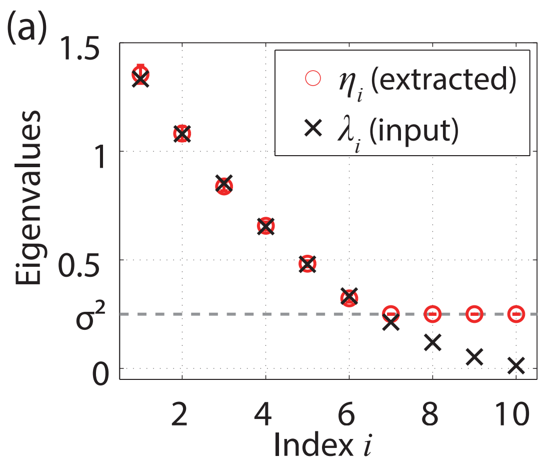


Fig. 2.4 Extraction of principal components in Gaussian-Gaussian RBM. (a) The red points are the principal eigenvalues extracted in the weight matrix, η_i ($N = M = 10$, model variance $\sigma^2 = s^2 = 1/4$). The black points are the principal eigenvalues of input data, λ_i. The gray dashed line represents the value of the model variance $\sigma^2 = 1/4$. (b) The eigenvalues η_i extracted by CD$_1$ learning with the same model and input data as used in the case of ML learning.

$\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(1000)}\}$ artificially generated from the above procedure. As is claimed in Theorem 1, ML learning extracted only larger eigenvalues $\eta_i = \lambda_i > \sigma^2$. In addition, the smaller eigenvalues corresponding to $\lambda_i < \sigma^2$ were replaced with the model variance such as $\eta_i = \sigma^2$, as shown in the model distribution (2.35).

Figure 2.4 (b) presents the results of CD$_1$ learning on the same model and input data as used on ML learning. In the simulation, to observe the averaged behaviors of the learning, we used Gibbs sampling for the CD$_1$ learning on the batch input. At each learning step, we generated samples for CD$_1$, $\mathbf{h}_1^{(i)}$ and $\mathbf{v}_1^{(i)}$, by using transition probabilities $p(\mathbf{h}_1^{(i)}|\mathbf{v}_1^{(i)})$, $p(\mathbf{v}_1^{(i)}|\mathbf{h}_0^{(i)})$ and $p(\mathbf{h}_0^{(i)}|\mathbf{v}^{(i)})$. Then, we calculated the second term $< \mathbf{h}\mathbf{v}^T >_1$ by the average $\sum_{i=1}^{1000} \mathbf{h}_1^{(i)} \mathbf{v}_1^{(i)T} / 1000$. CD$_1$ learning in the simulation extracted the principal components as expected from Theorem 2. Remarkably, the CD$_1$ learning results shown in Figure 2.4 (b) extracted the same principal components as ML learning did in Figure 2.4 (a). The second term of ML learning takes a long time to run many iterations of Gibbs sampling. In contrast, CD$_1$ learning only needs one step of Gibbs sampling and therefore saves computational time. In the simulation, CD$_1$ learning efficiently obtained the same principal components and model distribution $p(\mathbf{v})$ as ML learning did.

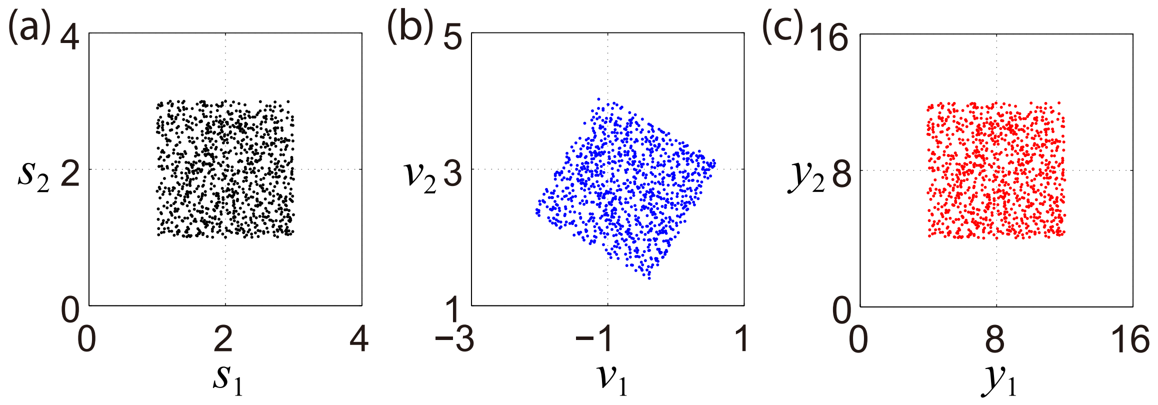


Fig. 2.5 Extraction of independent components by CD$_1$ learning in Gaussian-Bernoulli RBM. (a) Uniform distribution of independent sources s_1 and s_2 ($N = M = 2$, $\sigma^2 = 1/4$). (b) Distribution of input data $q(\mathbf{v})$ generated from $\mathbf{v} = B\mathbf{s}$. (c) Distribution of output $p(\mathbf{y})$ defined by $\mathbf{y} = W\mathbf{v}$, where W is obtained by CD$_1$ learning.

2.5.2 Gaussian-Bernoulli RBM

We show simulation results demonstrating that the Gaussian-Bernoulli RBM extracted independent components as is claimed in Section 2.4.

Figure 2 presents the results of CD$_1$ learning in the Gaussian-Bernoulli RBM with $N = M = 2$ and $\sigma^2 = 1/4$. We set the independent source signals as non-negative uniform distributions $q(s_1)$ and $q(s_2)$ in Figure 2.5 (a). The input distribution $q(\mathbf{v})$ in Figure 2.5 (b) was generated from $\mathbf{v} = B\mathbf{s}$. In the same way as shown in the experiments on the Gaussian-Gaussian RBM, we used batch input and calculated the second term of the CD$_1$ gradient by the average $\sum_{i=1}^{1000} \mathbf{h}_1^{(i)} \mathbf{v}_1^{(i)T}/1000$. Note that the distribution $q(\mathbf{v})$ corresponds to the source distribution rotated by the orthogonal matrix B. After CD$_1$ learning, we obtained the stable solution W as is described in Theorem 4. Figure 2.5 (c) shows the result of the CD$_1$ learning by using an output distribution of $\mathbf{y} = W\mathbf{v} = D\mathbf{s}$. The output y_i became the source signal s_i scaled by $d_i = \mu_i/\sigma$. Because the output decides the activation probability of the hidden unit by $p(h_i = 1|\mathbf{v}) = \mathrm{sigmoid}(y_i/\sigma)$, the i-th hidden unit detects the i-th independent source.

Note that the stable solution $W = DB^T$ proved in Theorems 3 and 4 is not a necessary condition for the stable solutions but a sufficient one. There could be stable solutions that differ from $W = DB^T$. If we began the training from initial values of W close enough to DB^T, the matrix W converged to the solution $W = DB^T$ in the simulation.

2.6 Conclusion

We analytically derived the fixed points of the CD$_n$ learning rule and proved their stability by using the perturbation method. In the Gaussian-Gaussian RBM, we revealed that the fixed points of CD$_n$ learning rule have the same analytical form and stability as those of ML learning rule. We also clarified that principal components whose eigenvalues are larger than a certain value are extracted at the fixed points. These results mean that a trained Gaussian-Gaussian RBM reduces the dimension of input data on the basis of its principal components. Moreover, in the Gaussian-Bernoulli RBM, we revealed that both ML and CD$_n$ learning can extract independent components from input data at one of their stable fixed points. Because our analytical results are independent of the number of Gibbs sampling n, it was demonstrated that the same feature components extracted by ML learning are obtained simply by performing CD$_1$ learning. Note that we assumed there are infinity number of input samples.

By expanding our analytical method, we expect to identify the features that are commonly or differently found by ML and CD learning in other types of models such as RBMs with binary visible units and deep generative models. It would also be practical to investigate how sparsity regularization [65] and rectified linear units [74] affect the stable points and change the extraction of features.

In this year, some studies [112, 50] have reported that there exist relatively general sufficient conditions where the parameters estimated by CD$_n$ learning can coincide with maximum likelihood estimators in some of exponential family. It is noteworthy that our sufficient conditions shown in the RBMs with Gaussian visible units are different from the conditions proposed by them. Building a bridge between our study and theirs will elucidate more general conditions where both CD$_n$ and ML learning coincide with each other.

A further direction of this study will be to investigate the dynamics of CD$_n$ learning rules. The previous theoretical studies on the Oja's learning rule (2.51) obtained an explicit expression of its time course [113, 24]. It is also worth remarking that the dynamics for special initial conditions are exactly solved in unsupervised pre-training and backpropagation learning of deep linear neural networks [92]. It remains to be explored whether we can apply the techniques of the previous works to the dynamics of CD$_n$ learning and clarify its convergence properties.

Chapter 3

Maximum Likelihood Learning of RBMs with Gaussian Visible Units on the Stiefel Manifold

The restricted Boltzmann machine (RBM) is a bipartite graphical model that is widely used as a building block of deep neural networks [45], but it is hard to train by using maximum likelihood (ML) learning. Computation of the likelihood is analytically intractable and requires many iterations of Gibbs sampling, which entails a lengthy computation. Here, contrastive divergence (CD) learning has been developed as an approximate way of computing the gradient of ML learning and is commonly used in practice [45, 101]. CD learning computes the ML gradient with samples obtained by a limited number of Gibbs samplings and empirically converges close enough to the ML solutions in a short time [22]. However, in general, there are almost no theoretical guarantees of convergence or maximization of the likelihood in CD learning [100].

In this study, we reveal that the likelihood and its gradients in RBMs with Gaussian visible units are analytically tractable when the weight matrix is constrained to the Stiefel manifold. We propose a novel algorithm based on geodesic flow on the Stiefel manifold for Gaussian-Bernoulli RBM and demonstrate its effectiveness in experiments on natural image patches. Because our algorithm has a tractable likelihood, there are advantages to using it to monitor the convergence and maximization of the likelihood. Moreover, we prove theoretically that the proposed method arrives at essentially the same solution as standard ML learning in Gaussian-Gaussian RBM.

3.1 Geodesic flow of ML learning on Stiefel Manifold

3.1.1 Definition of geodesic flow on Stiefel manifold

Let us consider the problem of minimizing a cost function L with regards to a parameter matrix $A \in \mathbb{R}^{M\times N}$. We assume $M \leq N$ and that M row vectors of A are mutually orthogonal N-dimensional unit vectors satisfying $AA^T = I_M$. The set of all such matrices is known as the Stiefel manifold. In particular, the Stiefel manifold with $N = M$ reduces to the orthogonal group. When one minimizes L by using the steepest descent algorithm, the update rule is given by

$$A_{t+1} = A_t - \varepsilon\Delta A, \tag{3.1}$$

where ε is a small learning constant and $\Delta A = dL/dA$. It should be noted that, because A_{t+1} is not in the manifold, it is necessary to project A_{t+1} to the manifold in each iteration [77].

In contrast, by considering an extension of the natural gradient method, one can minimize L along geodesic flows on the Stiefel manifold as follows [32]:

$$A_{t+1} = A_t \exp\left(\varepsilon(A_t^T\Delta A - \Delta A^T A_t)\right). \tag{3.2}$$

As a background knowledge, let us briefly describe how to derive the above update rule. Figure 3.1 shows the overview of the derivation. First, we need to project the gradient ΔA to the tangent space of the Stiefel manifold. Let us denote the Stiefel manifold as $V_M(\mathbb{R}^N)$ and its tangent space as $T_A V_M(\mathbb{R}^N)$. The projection to $T_A V_M(\mathbb{R}^N)$ is given by $\Delta A - A\Delta A^T A$ [32, 5]. Next, we transform the coordinate system of A so that the A_t becomes an identity matrix. In that coordinate system, we denote the tangent space as $T_I V_M(\mathbb{R}^N)$. By the coordinate transformation, the projection to the tangent space is also transformed into $\Delta_{//}A = (\Delta A - A\Delta A^T A)A^T = (\Delta AA^T - A\Delta A^T)$. In the Stiefel manifold, it is known that the geodesic from the identity with a velocity X is given by an exponential map $\exp(-\varepsilon X)$, where t is an infinitesimal time step. Therefore, the update along the geodesic is $\exp(-\varepsilon\Delta_{//}A)$. Note that we need to pull back the coordinate system into the original space with the tangent space $T_A V_M(\mathbb{R}^N)$. That is given by A$\exp(-\varepsilon\Delta_{//}A)$ and we thus obtain the update rule (3.2).

Because A_t being on the Stiefel manifold ensures that A_{t+1} will also be on it, we can omit the projection to the manifold in each iteration. In practical applications such as ICA, this geodesic algorithm has outperformed the standard steepest decent algorithms [77, 32]. It is also remarkable that the effective dimension of the matrix on the Stiefel manifold is $MN - M(M+1)/2$ because of the constraints. The effective dimension of the $M \times N$ matrix without any constraint is MN. For instance, in the case of $M = N$, the effective dimension of the Stiefel manifold becomes $N(N-1)/2$, which is almost half of the matrix without the constraints. Therefore, optimization on the Stiefel manifold reduces the number of parameters and can act as a regularization for better generalization [32].

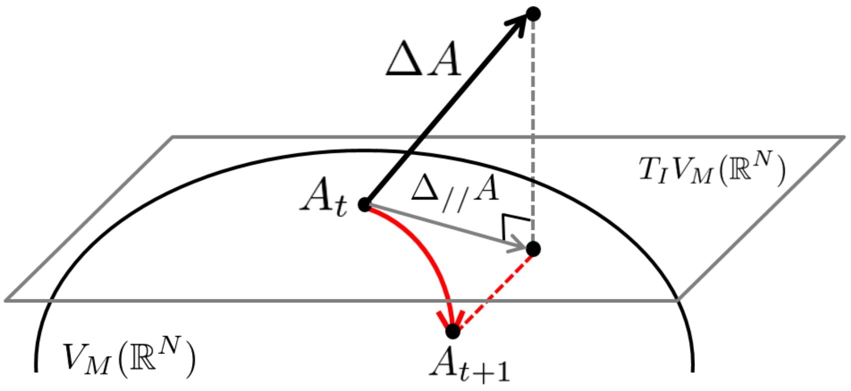


Fig. 3.1 Update along the geodesic flow on the Stiefel manifold.

3.1.2 Gaussian-Bernoulli RBM

In the following, we apply the above geodesic algorithm to two types of RBM with Gaussian visible units. First, we consider the following model distribution of a Gaussian-Bernoulli RBM with an uniform model variance [45]:

$$p(\mathbf{h},\mathbf{v}) = \exp\left(-\frac{1}{2\sigma^2}||\mathbf{v}-\mathbf{b}||^2 + \frac{1}{\sigma}\mathbf{h}^T W\mathbf{v} + \mathbf{c}^T\mathbf{h}\right)/Z. \tag{3.3}$$

Let us denote binary hidden variables as $h_i = \{0,1\}$ $(i=1,...,M)$ and continuous visible variables as v_i $(i=1,...,N)$. Moreover, we consider the variance of the visible units by σ^2 and the normalization constant by Z. We estimate the weight matrix $W \in \mathbb{R}^{M\times N}$ and bias vectors $\mathbf{b}$ and $\mathbf{c}$.

The maximum likelihood (ML) estimate is obtained by minimizing the negative log-likelihood $L = -\int q(\mathbf{v})\ln p(\mathbf{v})d\mathbf{v}$, where $q(\mathbf{v})$ denotes the input distribution and $p(\mathbf{v})$ denotes the marginal model distribution. The steepest gradient of the negative log-likelihood is given by $W_{t+1} = W_t + \varepsilon\Delta W$ with $\Delta W =< \mathbf{h}\mathbf{v}^T >_q - < \mathbf{h}\mathbf{v}^T >_p$ [45]. Let us denote the average over training examples generated from $q(\mathbf{v})$ as $< \cdot >_q$ and the average over the model distribution as $< \mathbf{h}\mathbf{v}^T >_p$. After marginalizing $\mathbf{h}$ in the first term and $\mathbf{v}$ in the second term, the update ΔW can be transformed into

$$\Delta W =< g(W\mathbf{v}/\sigma + \mathbf{c})\mathbf{v}^T >_q -(\sigma < \mathbf{h}\mathbf{h}^T >_{p(\mathbf{h})} W + < \mathbf{h} >_{p(\mathbf{h})} \mathbf{b}^T), \tag{3.4}$$

where $g(\mathbf{x})$, a function with a vector argument $\mathbf{x}$, denotes a vector whose i-th element is a sigmoid function $g(x_i)$. The second term is analytically intractable because one needs to take a summation over an exponential number of hidden states obeying the following model distribution:

$$p(\mathbf{h}) = \exp\left(\frac{||W^T\mathbf{h}||^2}{2} + (W\mathbf{b}/\sigma + \mathbf{c})^T\mathbf{h}\right)/Z. \tag{3.5}$$

This distribution is equivalent to a full-connected Boltzmann machine with the connectivity WW^T and it takes too much computational time to compute its statistic without any assumption or approximation.

Surprisingly, we can avoid this analytically intractable summation by constraining the parameter space of W. Let us assume that the weight matrix W is constrained to $W = DA$, where A is included in the Stiefel manifold and D is a diagonal matrix $D = \text{diag}(d_1, d_2, ..., d_M)$. Substituting $W = DA$ into (3.5), the model distribution $p(\mathbf{h})$ becomes independent among the hidden variables:

$$p(\mathbf{h}) = \prod_{i=1}^{M} g(y_i)^{h_i}(1 - g(y_i))^{1-h_i}, \tag{3.6}$$

where we define $y_i = d_i^2/2 + d_i\mathbf{a}_i\mathbf{b}/\sigma + c_i$ and denote the i-th row vector of A by $\mathbf{a}_i$. The constraint of $W = DA$ corresponds not only to reducing the dimension of the search space W but also to giving a prior that hidden units act independently.

Using the independent distribution (3.6), we can compute the update rule (3.4) analytically as follows:

$$\Delta A = D[< g(W\mathbf{v}/\sigma + \mathbf{c})\mathbf{v}^T >_q - (\sigma KW + g(\mathbf{y})\mathbf{b}^T)], \tag{3.7}$$

where $K_{ii} = g(y_i)$ $(i = 1, ..., M)$ and $K_{ij} = g(y_i)g(y_j)$ $(i \neq j)$. The update rule of A is given by (3.2) with (3.7). In addition, the update rules of the other parameters are given by the ordinary steepest directions:

$$d_i \leftarrow d_i + \varepsilon[< g(d_i\mathbf{a}_i\mathbf{v}/\sigma + c_i)\mathbf{a}_i\mathbf{v} >_q - g(y_i)(\mathbf{a}_i\mathbf{b} + \sigma d_i)], \tag{3.8}$$

$$\mathbf{b} \leftarrow \mathbf{b} + \varepsilon[< \mathbf{v} >_q - (\mathbf{b} + \sigma W^T g(\mathbf{y}))], \tag{3.9}$$

$$\mathbf{c} \leftarrow \mathbf{c} + \varepsilon[< g(W\mathbf{v}/\sigma + \mathbf{c}) >_q - g(\mathbf{y})]. \tag{3.10}$$

Moreover, the constraint $W = DA$ makes the negative log-likelihood analytically tractable as follows:

$$L =< \frac{1}{2\sigma^2}||\mathbf{v} - \mathbf{b}||^2 - \Sigma_i \ln(1 + e^{d_i\mathbf{a}_i\mathbf{v}/\sigma + c_i}) >_q + \Sigma_i \ln(1 + e^{y_i}) + N\ln(\sqrt{2\pi}\sigma). \tag{3.11}$$

In general, L is analytically intractable and difficult to use to monitor the progress of learning in the Gaussian-Bernoulli RBM. Although there are several methods to approximately estimate the likelihood function, its accuracy is controversial [43]. In addition, even such approximated estimation methods require Gibbs sampling and take much computational time to compute the likelihood at each time step on the way of learning. In contrast, our method requires rather low computational complexity $O(M^2N)$ to compute the likelihood and can monitor how well and determine where the learning trajectory converges on the way of learning. This property enables us to easily choose optimal hyperparameter and better initial conditions.

3.1.3 Gaussian-Gaussian RBM

We can also obtain the geodesic ML learning rule in Gaussian-Gaussian RBM, whose model distribution is defined as follows [53, 110]:

$$p(\mathbf{h},\mathbf{v}) = \exp\left(-\frac{1}{2s^2}||\mathbf{h}-\mathbf{c}||^2 - \frac{1}{2\sigma^2}||\mathbf{v}-\mathbf{b}||^2 + \frac{1}{s\sigma}\mathbf{h}^T W\mathbf{v}\right)/Z, \tag{3.12}$$

where both visible and hidden units take continuous real values. Note that the likelihood and its gradient in the standard ML learning without any constraint are analytically tractable in Gaussian-Gaussian RBM [53]. In this study, we consider ML learning with $W = DA$ in order to obtain an insight into how this constraint changes the solution compared with that of standard ML learning. The geodesic ML learning on the Stiefel manifold is given by

$$A \leftarrow A\exp\left(\varepsilon(A^T D^2 AC - CA^T D^2 A)/2\right), \tag{3.13}$$

$$d_i \leftarrow d_i + \varepsilon d_i[<(\mathbf{a}_i\mathbf{v})^2>_q - \sigma^2/(1-d_i^2)], \tag{3.14}$$

where C is the data covariance matrix of the input distribution $q(\mathbf{v})$. We set the mean values of the input data to $\int \mathbf{v}q(\mathbf{v})d\mathbf{v} = 0$ and set the bias parameters to $\mathbf{b} = \mathbf{c} = 0$ for simplicity, but we can also formulate and analyze the general case in the same way.

3.1.4 Other variants

As a simple extension, we can generalize the proposed learning algorithms with orthogonal constraints to the following model;

$$q(\mathbf{h},\mathbf{v}) = \exp\left(-\sum_j \frac{\mathbf{v}_j^2}{2\sigma_j^2} + \sum_{i,j} W_{ij}\phi_i(h_i)\frac{v_j}{\sigma_j}\right)/Z, \tag{3.15}$$

where $\phi(\cdot)$ is an arbitrary function and each h_i represents a continuous or a discrete random variable not limited to a binary one. The notation σ_j represents the model variance of the j-th visible variable. The G-G and G-B RBMs are included in this model as the special cases. Because the analytical tractability under the orthogonal constraint requires either visible units or hidden units obey continuous Gaussian variable, a model distribution $q(\mathbf{v}, \mathbf{h})$, where $\mathbf{v}$ and $\mathbf{h}$ are replaced with each other in the model definition (3.15), is also analytically tractable.

As is pointed out in the footnote of the previous study [108], products of student-T (PoT) model is reduced to a generative model of ICA under the orthogonal constraints. Because the generative model of ICA is analytically tractable, we can also apply the geodesic algorithm with orthogonal constraints as is often used in the studies of ICA.

3.2 Analysis of Gaussian-Gaussian RBM

Here, we provide a theoretical guarantee that ML learning on the Stiefel manifold arrives at essentially the same ML solution as standard ML learning in Gaussian-Gaussian RBM. We consider only the case of $M = N$ in the following analysis, but we can also analyze general situations in the same way.

As assumed in the analysis of Gaussian-Gaussian RBM in Chapter 2, let us assume that the data covariance matrix C has non-degenerate eigenvalues λ_i $(i = 1, ..., N)$ satisfying $\lambda_1 > \cdots > \lambda_k > \sigma^2 > \lambda_{k+1} > \cdots > \lambda_N$ and is diagonalized such that $C = V^T \mathrm{diag}(\lambda_1, ..., \lambda_N)V$, where V is an $N \times N$ orthogonal matrix. Under these assumptions, the previous study found that the stable solution of standard ML learning is limited to the following $\bar{W}$ [53]:

$$\bar{W} = U\mathrm{diag}\left(\sqrt{1-\sigma^2/\lambda_1}, ..., \sqrt{1-\sigma^2/\lambda_k}, 0, ..., 0\right)V, \tag{3.16}$$

where U is an arbitrary $N \times N$ orthogonal matrix. At the stable solution $\bar{W}$, the model distribution becomes a Gaussian distribution: $p(\mathbf{v}) = \mathcal{N}(\mathbf{v}; \mathbf{0}, V^T \mathrm{diag}(\lambda_1$ $, ..., \lambda_k, \sigma^2, ..., \sigma^2)V)$. This mode distribution means that the trained Gaussian-Gaussian RBM extracts only the largest k principal components, whose eigenvalues are larger than the model variance σ^2.

Under the assumptions of the previous study, we find that the ML learning on the Stiefel manifold (3.13, 3.14) has the following analytical solutions:

Theorem 5 *The stable equilibrium solution of ML learning on the Stiefel manifold (3.13, 3.14) is* $\bar{W} = \mathrm{diag}\left(\sqrt{1-\sigma^2/\lambda_1}, ..., \sqrt{1-\sigma^2/\lambda_k}, 0, ..., 0\right)V$.

Proof The update rule (3.13) stops at $A^T D^2 AC = CA^T D^2 A$. Because $A^T D^2 A$ and C are commutative, these matrices are simultaneously diagonalizable, and we obtain an equilibrium $\bar{A} = V$. In addition, substituting $\bar{A} = V$ into (3.14), we get $\bar{d}_i = 0$ or $\bar{d}_i = \sqrt{1-\sigma^2/\lambda_i}$ with $\lambda_i > \sigma^2$.

Next, we check the stability of the equilibrium solution in a similar process as shown in the standard ML learning [53]. We can represent the perturbation of A along the Stiefel manifold as $\Delta A \equiv A\exp(A^T \Delta X A) - A$, where ΔX is an $N \times N$ alternative matrix whose entries satisfy $\Delta X_{ij} = -\Delta X_{ji}$. Because the perturbation ΔX_{ij} takes an infinitesimal value $|\Delta X_{ij}| \ll 1$, we get $\Delta A \sim \Delta X A$. The stable solution requires the following inner product to become negative:

$$\begin{aligned}
&\mathrm{Tr}\left(\Delta A^T \Delta F_A\right) + \Sigma_i \Delta d_i \Delta F_{d_i} \\
&\sim \sum_{a<b}^{N} \Delta X_{ab}^2 (d_a^2 - d_b^2)(\lambda_b - \lambda_a) + \sum_{i=1}^{N} \Delta d_i^2 \left\{\lambda_i - (1+d_i^2)/(1-d_i^2)^2\sigma^2\right\}, \qquad (3.17)
\end{aligned}$$

where Δd_i denotes the perturbation of d_i. When one transforms the update rule (3.13) into the form $A_{t+1} - A_t = F_A(A_t, d_{i,t})$ and the update rule (3.14) into $d_{i,t+1} - d_{i,t} = F_{d_i}(A_t, d_{i,t})$, the perturbations of the gradients, ΔF_A and ΔF_{d_i}, are given by $\Delta F_A = F_A(\bar{A} + \Delta A, \bar{d}_i + \Delta d_i)$ and $\Delta F_{d_i} = F_{d_i}(\bar{A} + \Delta A, \bar{d}_i + \Delta d_i)$. We can easily confirm that the inner product (3.17) becomes negative if and only if $\bar{D} = \mathrm{diag}\left(\sqrt{1-\sigma^2/\lambda_1}, ..., \sqrt{1-\sigma^2/\lambda_k}, 0, ..., 0\right)$. Therefore, the ML learning on the Stiefel manifold has the global minimum $\bar{W} = \bar{D}\bar{A}$. □

We have thus proven the remarkable fact that constraining the weight space to $W = DA$ corresponds to eliminating the rotational degrees of freedom caused by U in the standard ML stable solution. In addition, the model distribution $p(\mathbf{v})$ coincides with that of standard ML learning. Therefore, ML learning on the Stiefel manifold gives essentially the same solution as standard ML learning. In that sense, the orthogonal constraint is a natural assumption to eliminate the redundancy of the solution space.

In the case of $M < N$, we can also analytically obtain the ML solutions. Although some local minima appear in the ML learning with $W = DA$, the global minima coincide with those of the standard ML solutions, as is shown with $M = N$.

3.3 Experiments on Gaussian-Bernoulli RBM

To confirm the effectiveness of our method, we trained a Gaussian-Bernoulli RBM with the algorithm (3.7-3.10) on natural image patches sampled from the van Hateren natural image database [65]. In the preprocessing, we applied global contrast normalization and ZCA whitening to 50,000 image patches of 14x14 pixels. The data set consisted of 40,000 training cases and 10,000 test cases. We set σ^2 to be on the same scale as the variance of the data.

As is shown in Figure 3.2 (a), our method achieved cost values comparable to those of the persistent contrastive divergence (CD) algorithm [101] in the Gaussian-Bernoulli RBM with $M = 16$ and $N = 196$. The thick line in Figure 3.2 (a) represent the mean over ten training runs with different random initializations, and the filled areas represent the standard deviation. The CD algorithm with no constraint on W has an intractable likelihood, but we set the number of hidden units to a small value and computed the exact value of the likelihood. Our method was constrained to $W = DA$ and had fewer free parameters than the CD algorithm. Nevertheless, it achieved the same cost values as the CD algorithm.

We also trained a Gaussian-Bernoulli RBM with $N = M = 196$ with our method. Because we can compute the likelihood function by using (3.11) even in the case of large M, we can

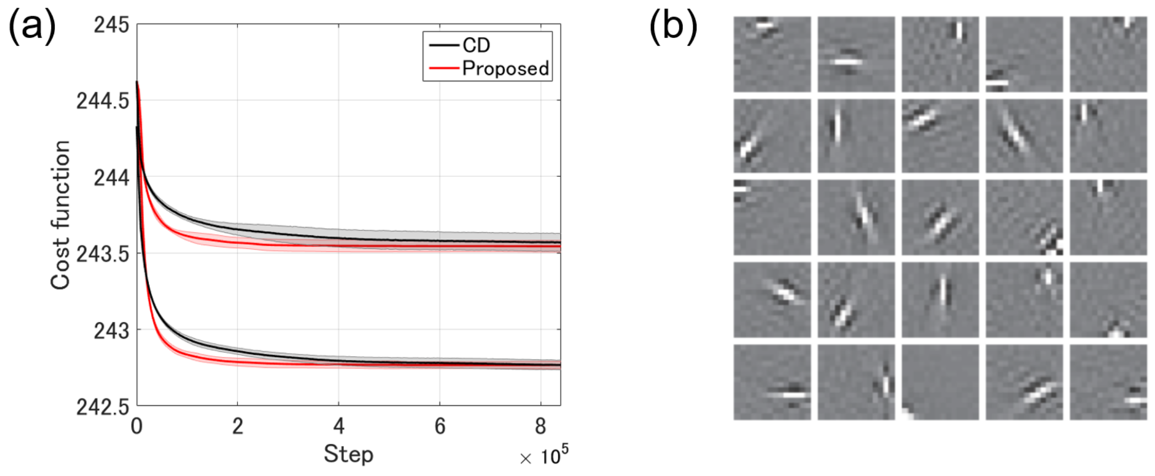


Fig. 3.2 Gaussian-Bernoulli RBM trained by the proposed method on natural image dataset. (a) Comparison of proposed method and CD algorithm. Negative log-likelihoods are shown as the cost function (we set $M = 16$, $N = 196$). Red lines represent the results obtained by the proposed algorithm and black one show those by the conventional CD algorithm. Note that upper lines correspond to test errors, the cost functions with 1000 test samples, and that lower ones are training errors, the cost functions with 10000 training samples. (b) Gabor-like filters obtained by proposed method (we set $M = N = 196$).

easily monitor the convergence of the cost function. We randomly selected the learned filters and the reshaped rows of W with $d_i \neq 0$, and show them in Figure 3.2 (b). As Gabor-like filters are extracted, our method would seem to be useful in feature extraction.

3.4 Conclusion and future work

We proposed a novel algorithm to train RBMs with continuous visible units, where the constraint on the Stiefel manifold enables us to compute analytical values of the likelihood and its gradients. In the experiments on Gaussian-Bernoulli RBM, our method achieved comparable performance with the CD algorithm. For Gaussian-Gaussian RBM, we provided a theoretical guarantee that the proposed method obtains the essentially same solution as that of standard ML learning.

A further direction of study is to apply a similar constraint on the parameter space to more general forms of RBMs, such as exponential family harmoniums and stacked RBMs. In deep networks, it has been suggested that orthogonal weight matrices obtained by layerwise pre-training accelerate the convergence of the supervised learning through all layers [92]. It remains to be explored how to apply our method to the pre-training of the deep networks.

Chapter 4

Analysis of Learning Dynamics in Weight Normalization

Developments of the gradient method have provided essential contributions to the training of deep neural networks. The stochastic gradient descent (SGD) has been developed for optimizing large-scale networks with a large amount of data. Improvements of the SGD such as Adagrad, RMSprop, and Adam [56] have succeeded in achieving the state-of-the-art performances.

In particular, Salimans & Kingma have proposed weight normalization (WN), which separates a weight vector to radial directional and parameters and then optimizes them [91]. Weight normalization is easy to implement and has negligible computational complexity compared to the conventional SGD. From the experimental perspective, the weight normalization has succeeded to speed up the convergence of learning in such deep networks as the convolutional neural network (CNN), variational auto-encoder, the recurrent neural network composed from LSTM and the deep Q-network [91]. Numerical experiments also reported that the effect of weight normalization is even better than batch normalization [49] on the convergence speed. However, it remains theoretically uncovered why and how the weight normalization facilitates the convergence speed of learning. It will be convenient to understand the mechanism of the acceleration to develop better optimization methods in deep learning.

In this chapter, we briefly analyze the weight normalization in the context of a coordinate transformation in a dynamical system. Explicitly evaluating the Jacobian matrix of the coordinate transformation from the parameterization of the weight normalization to the conventional weight matrix, we derive an effective learning rate of the weight normalization. We reveal that the effective learning rate is automatically tuned like the annealing of a learning rate. In addition, we show that it realizes scale invariant gradients against the scale transformation of the weight matrix in multi-layer ReLU networks.

4.1 Definition of weight normalization

The conventional steepest decent gradient is given by

$$\Delta W = -\eta \nabla_W l, \tag{4.1}$$

where W is a weight matrix and l is a given cost function. The stochastic gradient decent (SGD) compute this steepest decent gradient by using mini-batch data. In contrast, weight normalization reparameterizes the weight matrix by

$$\mathbf{w}_i = r_i \frac{\mathbf{v}_i}{||\mathbf{v}_i||}, \tag{4.2}$$

where $\mathbf{w}_i$ is an $N \times 1$ vertical vector and its vector transposition $\mathbf{w}_i^T$ denotes the i-th row vector of the weight matrix W $(i = 1, 2, ..., N)$ [91]. The parameter r_i means a radial parameter of the weight vector and an $N \times 1$ vertical vector $\mathbf{v}_i$ represents the direction of the weight vector. Note that $||\mathbf{x}||$ means L2 norm of the vector $\sqrt{\sum_i x_i^2}$. We reparameterize all of the weight vectors $\mathbf{w}_i$ in the network to the parameters of the weight normalization.

In the weight normalization, the update rule at the time step t is given by the following steepest descent gradient [91]:

$$r_i(t+1) = r_i(t) + \Delta r_i, \tag{4.3}$$

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \Delta \mathbf{v}_i, \tag{4.4}$$

with

$$\Delta r_i = -\eta \nabla_{r_i} l, \tag{4.5}$$

$$\Delta \mathbf{v}_i = -\eta \nabla_{\mathbf{v}_i} l, \tag{4.6}$$

where η is a constant learning rate. We can compute this update of the weight normalization by using the gradient of the conventional SGD, i.e., $\nabla_{\mathbf{w}_i} l$ as follows:

$$\Delta r_i = -\eta \hat{\mathbf{v}}_i^T \nabla_{\mathbf{w}_i} l, \tag{4.7}$$

$$\Delta \mathbf{v}_i = -\eta \frac{r_i}{||\mathbf{v}_i||} (I - \hat{\mathbf{v}}_i \hat{\mathbf{v}}_i^T) \nabla_{\mathbf{w}_i} l, \tag{4.8}$$

where we denote the unit vector composed by $\mathbf{v}_i$ as $\hat{\mathbf{v}}_i = \mathbf{v}_i/||\mathbf{v}_i||$. Weight normalization is easy to use in practice because one can implement weight normalization by a minor change of the conventional SGD program code and its computational complexity is negligible.

Empirically, Salimans and Kingma found that one can set the learning rate η to a relatively large value because $||\mathbf{v}_i||$ in the update (4.8) monotonically increases with the time step [91]. In the following section, we give more sophisticated insight into this phenomena by analysis.

4.2 Learning dynamics projected onto the Cartesian coordinate W

To know the difference between the update rule of the weight normalization and that of the conventional SGD, we consider pulling back the update rule of the weight normalization into the parameter space of the conventional SGD learning. Let us denote the coordinate of the weight normalization as

$$\mathbf{q}_i = \begin{bmatrix} r_i \\ \mathbf{v}_i \end{bmatrix}. \tag{4.9}$$

To avoid complicated notation, let us omit the index of the weight vector i in the following formulation.

Let us represent the update rule of the weight normalization by the following form:

$$\Delta\mathbf{q} = -\eta \nabla_{\mathbf{q}} l. \tag{4.10}$$

In general, if a learning dynamics is transformed from a coordinate system $\mathbf{q}$ into a coordinate system $\mathbf{w}$, the equation of the dynamics should be transformed as follows:

$$\frac{\partial q}{\partial w}\Delta\mathbf{w} = -\eta \frac{\partial w}{\partial q}\nabla_{\mathbf{w}} l, \tag{4.11}$$

$$\frac{\partial w}{\partial q} = \begin{bmatrix} \hat{\mathbf{v}}^T \\ \frac{r_i}{||\mathbf{v}||}(I - \hat{\mathbf{v}}\hat{\mathbf{v}}^T) \end{bmatrix} \equiv J. \tag{4.12}$$

The matrix J means an $(N+1)\times N$ Jacobian matrix whose entry J_{ab} is given by $\frac{\partial w_b}{\partial q_a}$. Because $\frac{\partial w}{\partial q}\frac{\partial q}{\partial w}$ becomes an $N \times N$ identity matrix, we obtain the following learning dynamics by multiplying J^T to eq. (4.11):

$$\Delta \mathbf{w} = -\eta J^T J \nabla_{\mathbf{w}} l \tag{4.13}$$

$$= -\eta \left\{ \hat{\mathbf{v}}\hat{\mathbf{v}}^T + \frac{r^2}{||\mathbf{v}||^2}(I - \hat{\mathbf{v}}\hat{\mathbf{v}}^T) \right\} \nabla_{\mathbf{w}} l. \tag{4.14}$$

This update represents a learning dynamics of the weight normalization observed in the Cartesian coordinate system W. Compared to the conventional SGD, the gradient of the weight normalization is modified by $J^T J$. In other words, we can recognize $J^T J$ as an effective learning rate for the conventional SGD. In detailed, because the matrix $\hat{\mathbf{v}}\hat{\mathbf{v}}^T$ is a projection operator onto the direction of the weight vector $\hat{\mathbf{v}}$, the first term in the update (4.14) is a gradient along $\hat{\mathbf{v}}$. In contrast, the second term is a gradient orthogonal to $\hat{\mathbf{v}}$. Figure 4.1 (a) shows the relationship among each component.

Furthermore, in our numerical experiments, the gradient along the weight vector, i.e., $\hat{\mathbf{v}}^T \nabla_{\mathbf{w}} l$, is likely to become quite smaller compared to the other component of the gradient as shown in Figure 4.1 (b). Therefore, neglecting the terms including $\hat{\mathbf{v}}^T \nabla_{\mathbf{w}} l$, we may consider the following approximation of the gradient (4.14):

$$\Delta \mathbf{w} \sim -\eta \frac{r^2}{||\mathbf{v}||^2} \nabla_{\mathbf{w}} l. \tag{4.15}$$

If we assume that each component of $\hat{\mathbf{v}}$ and $\nabla_{\mathbf{w}} l$ is given by mutually independent random variable $N(0, 1/N)$ and that the number of the units N is large enough, the central limit theorem gives us $\hat{\mathbf{v}}^T \nabla_{\mathbf{w}} l \sim N(0, 1/N)$. Therefore, under this assumption, the gradient along the weight vector becomes small enough to be neglected and the approximation (4.15) becomes reasonable.

4.3 Automatic tuning of learning rate in weight normalization

As shown in (4.14), we found that weight normalization is equivalent to assume the effective learning rate JJ^T in SGD. This effective learning rate functions as an automatic tuning of a given learning rate η as follows.

The component of the gradient orthogonal the direction of the weight vector, i.e., $(I - \hat{\mathbf{v}}\hat{\mathbf{v}}^T)\nabla_{\mathbf{v}} l$, has an effective learning rate $r^2/||\mathbf{v}||^2$ in the update (4.14). Here, let us remark that $||\mathbf{v}||^2$ monotonically increases [91]. Because the gradient of $\mathbf{v}$ always satisfies $\mathbf{v}^T \Delta \mathbf{v} = 0$, we

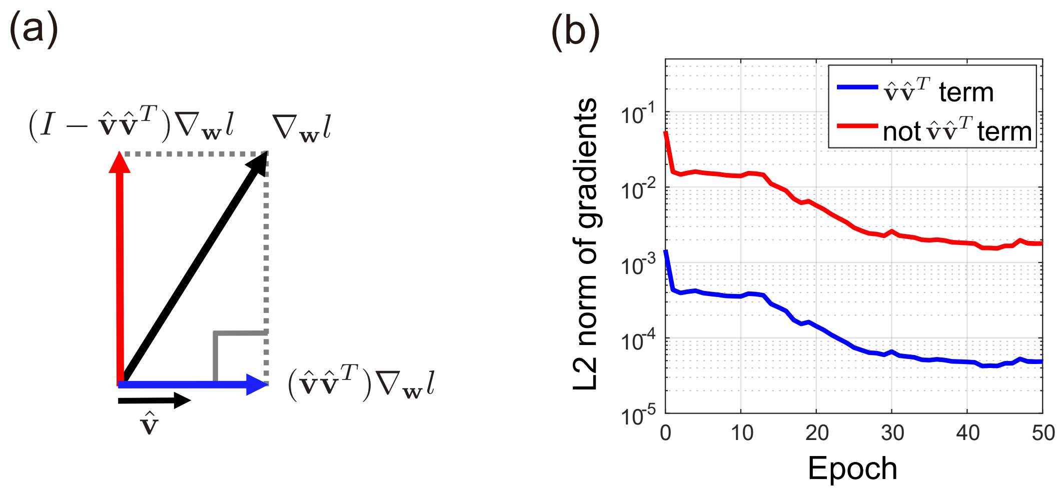


Fig. 4.1 Learning dynamics of weight normalization projected into the Cartesian coordinate system W. (a) Projection of the SGD gradient $\nabla_{\mathbf{w}} l$ to the weight vector. (b) The magnitude of the projected components of the SGD gradient. The blue line shows the magnitude of the gradient along the weight vector. The red one shows the magnitude of the gradient orthogonal to the weight vector (4-layer Tanh network with 500 hidden units in each layer, input data: MNIST, $\eta = 0.1$).

get

$$||\mathbf{v}(t+1)||^2 - ||\mathbf{v}(t)||^2 = ||\Delta\mathbf{v}||^2 \geq 0, \tag{4.16}$$

when the update is given by $\mathbf{v}(t+1) = \mathbf{v}(t) + \Delta\mathbf{v}$. This means that the magnitude $||\mathbf{v}(t)||^2$ increases with the time step t. Therefore, the effective learning rate $r^2/||\mathbf{v}||^2$ monotonically decreases. Therefore, the weight normalization (WN) implicitly realizes an annealing of the learning rate. As shown in Figure 4.2 (a), even if the given learning rate η took a large value, the training cost of WN converged smoothly. In contrast, the learning trajectory of conventional SGD became disturbed in the same situation and slowly converged to the higher training cost than WN. Besides, the SGD frequently diverged for larger learning rates but the WN converged. Therefore, we may set a large learning rate in weight normalization, and this leads to the fast convergence of the learning. Although the previous paper has empirically reported the similar automatic turning in the weight normalization [91], we have theoretically clarified it by deriving the effective learning rate in the Cartesian coordinate system. The magnitude $||\mathbf{v}||$ seemingly affects the SGD by the form of $1/||\mathbf{v}||$ as one can see in eq. (4.8). However, when one properly considers the projection into the Cartesian coordinate system, it actually affects the SGD by the form of the square, i.e., $1/||\mathbf{v}||^2$. It is also noteworthy that, if the approximation (4.15) is valid, the automatic turning of the effective learning rate appears more remarkably.

In Appendix C, we consider another parameterization with the radial parameters, where the directional parameters are given by a spherical coordinate system. As revealed by the coordinate transformation, the automatic turning does not appear in the spherical coordinate system. The automatic turning seems to be specific to the weight normalization method.

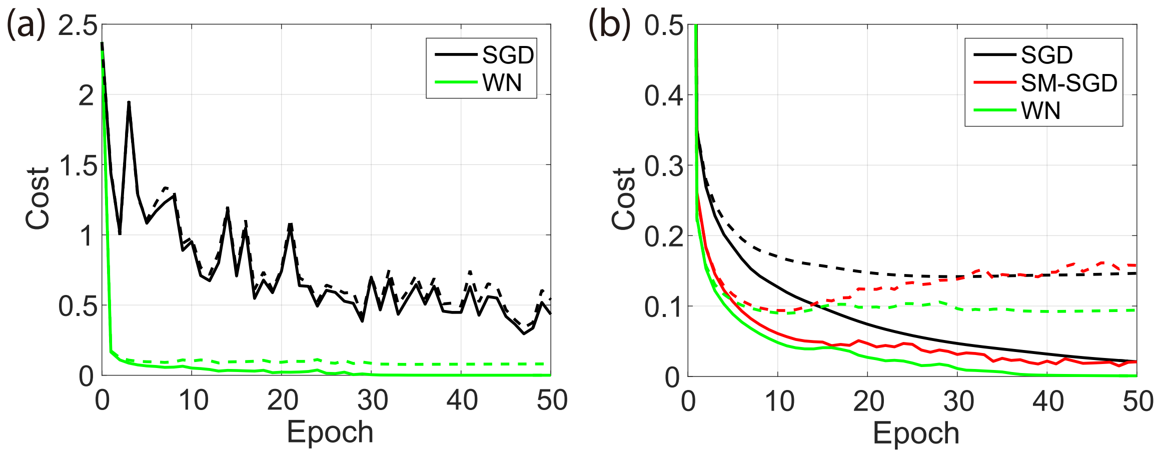


Fig. 4.2 The effects of weight normalization. (a) Learning dynamics with a large learning rate (4-layer Tanh network with 100 hidden units in each layer, input data: MNIST, $\eta = 1.5$). The thick lines show the values of the cost function with training samples and dashed ones show those with test samples. The black lines show the learning trajectories of SGD and the green lines show those of WN from the same initial condition. (b) Learning dynamics when the initial conditions are given by an unbalanced allocation (4-layer ReLU network with 100 hidden units in each layer, input data: MNIST, $\eta = 0.02$). All lines show the expectation among 10 different random initialization seeds. The red line shows the SGD with the scaled metric.

4.4 Scale invariance of weight normalization

We also found that the effective learning rate $J^T J$ contributes to preserving scale invariance of the weight normalization gradients in multi-layer ReLU networks. The cost function of the ReLU networks include $W^{(L+1)}\mathrm{ReLU}(W^{(L)}\mathbf{h}_L)$, where $W^{(L)}$ denotes the weight matrix between the L-th layer and $(L+1)$-th layer and $\mathbf{h}_L$ represents the output from the previous layer. Even if the scales of the weight matrices change with a constant value a such as

$$\mathbf{w}^{(L)} \to a\mathbf{w}^{(L)}, \tag{4.17}$$

$$\mathbf{w}^{(L+1)} \to a^{-1}\mathbf{w}^{(L+1)}, \tag{4.18}$$

the value of the cost function is invariant.

However, the values of SGD gradients change such as

$$\nabla_{\mathbf{w}^{(L)}} l \to a^{-2}\nabla_{\mathbf{w}^{(L)}} l, \tag{4.19}$$

$$\nabla_{\mathbf{w}^{(L+1)}} l \to a^{2}\nabla_{\mathbf{w}^{(L+1)}} l. \tag{4.20}$$

This means that the gradients are scale-dependent. When the scale of the weight matrix becomes large on the way of learning, the gradients in one layer vanishes and those in another layer diverges. Several studies have also pointed out that the scale-dependent gradients slow down the convergence of learning in the ReLU networks [76, 91].

Te effective learning rate of the weight normalization is helpful to overcome the scale dependence of the SGD gradients. Under the scale transformation (4.17, 4.18), the gradients of the weight normalization are transformed as follows:

$$\frac{r^2}{||\mathbf{v}||^2}\nabla_{\mathbf{w}^{(L)}} l \to \frac{(ar)^2}{||\mathbf{v}||^2}a^{-2}\nabla_{\mathbf{w}^{(L)}} l = \frac{r^2}{||\mathbf{v}||^2}\nabla_{\mathbf{w}^{(L)}} l, \tag{4.21}$$

$$\frac{r^2}{||\mathbf{v}||^2}\nabla_{\mathbf{w}^{(L+1)}} l \to \frac{(r/a)^2}{||\mathbf{v}||^2}a^{2}\nabla_{\mathbf{w}^{(L+1)}} L = \frac{r^2}{||\mathbf{v}||^2}\nabla_{\mathbf{w}^{(L+1)}} l. \tag{4.22}$$

Therefore, the weight normalization gradients are scale invariant. In addition, under the approximation (4.15), the component of the gradient vertical to the weight vector is likely to become dominant and then the scale invariance appears more strongly.

Here, let us remark that Badrinarayanan et al. have proposed a SGD gradient with a learning rate proportional to $||\mathbf{w}_i||^2$ such as

$$\Delta\mathbf{w} = -\eta||\mathbf{w}||^2\nabla_{\mathbf{w}} l, \tag{4.23}$$

which is scale invariant [10]. Note that one can regard this gradient as a natural gradient with a metric $G = \mathrm{daig}(WW^T)^{-1}$. They referred to this metric as *scaled metric*. Because the effective learning rate of the weight normalization $r^2/||\mathbf{v}||^2$ includes scaled metric $||\mathbf{w}||^2 = r^2$, weight normalization partially functions as the scaled metric. As shown in Fig. 4.2 (b), the numerical experiments demonstrated that the weight normalization and the SGD with the scaled metric (SM-SGD) converge faster than the SGD in a 4-layer ReLU network. In the experiments, we have set the initial values of the weight in an unbalanced condition to explicitly observe the effect of the scale invariance [76]. In detailed, we generated the initial values of the weight $W^{(1)}$, $W^{(2)}$ and $W^{(3)}$ from uniform random variables [38] and then scaled them such as $5W^{(1)}$, $5W^{(2)}$ and $W^{(3)}/25$. Because of the automatic tuning of the effective learning rate, the WN converged more smoothly and faster than the SM-SGD.

4.5 Discussion

In this chapter, we have analyzed the gradient of the weight normalization. As a result, we found that it acts as an automatic turning of the given learning rate, which monotonically decreases with the magnitudes of the directional parameters. Therefore, one can set a larger learning rate in the weight normalization. We also revealed that the gradients of the weight normalization are scale invariant and this invariance is related to the scaled metric. These two properties seem to make the weight normalization converge faster than SGD.

Recently, Yoshida et al. have challenged to theoretically analyze the learning dynamics of the weight normalization in simple perceptrons [114]. They have derived the statistical mechanical formulation of the learning dynamics and showed that the weight normalization can converge faster than SGD. Their theory has also pointed out that automatic turning of a learning rate is essential to speed up the convergence.

The scale invariance of the weight normalization appears in the widely-used networks with ReLU units. However, it does not appear in general nonlinear networks such as those with sigmoid units. In Chapter 5, we improve the weight normalization to make the gradients invariant against any nonlinear transformation, that is, natural gradients.

Chapter 5

Radial Natural Gradient

The natural gradient is a powerful method that improves the transient dynamics of learning because it utilizes the Riemannian structure of the parameter space. In particular, for training multi-layer perceptrons, the natural gradient is superior to other methods such as second-order optimization because it can avoid or alleviate the plateaus where the learning dynamics becomes very slow [82]. However, the main drawback of the natural gradient method is the high computational cost for the inverse of the metric. To facilitate the learning in hierarchical models in practical applications, it is necessary to develop a method to reduce the Fisher metric to smaller one, which preserves the essential geometries of the original space [81].

In this chapter, we introduce radial natural gradient learning, an efficient method with a simplified version of the natural gradient that accelerates the transient dynamics of learning. We use weight normalization, which separates the weight vectors in a multi-layer network into their radial parameters and direction parameters, and then propose a radial Fisher metric in the subspace of radial parameters. The computation of this metric requires much less computational cost than that of the standard Fisher metric. Our theoretical analysis indicates that the radial Fisher metric captures some of the geometry of the original Fisher metric. As is known in the case of the standard Fisher metric, the natural gradient with the radial Fisher metric is also expected to alleviate plateau phenomena caused by a singularity of the parameter space and hence accelerates the convergence of the learning. We confirmed the effectiveness of our method in several numerical experiments with multi-layer perceptrons and benchmark datasets. Expansion of this study should be an easy way to reduce the computational complexity and accelerate the convergence of learning in large-scale neural network models for practical applications.

5.1 Natural gradient

We first briefly overview the natural gradient method [5]. The natural gradient update of the parameter θ is written by

$$\theta_{t+1} = \theta_t - \eta_t G^{-1} \nabla_\theta l(\theta_t), \tag{5.1}$$

where ∇_θ is the 1st-order differentiation and G denotes a Riemannian metric matrix. When the objective function $l(\theta)$ is given by the negative log likelihood of a probabilistic model, we use the following Fisher information matrix as the metric:

$$G = E\left[\nabla_\theta \log p(\mathbf{x};\theta) \nabla_\theta \log p(\mathbf{x};\theta)^T\right]. \tag{5.2}$$

The natural gradient takes the Riemannian structure of the parameter space into account. It is invariant under arbitrary coordinate transformation and does not depend on a specific coordinate system. In contrast, conventional SGD is the steepest descent under the Euclidean space and is not invariant under the transformation. Even if the parameter space has an ill-shaped error landscape, the natural gradient provides isotropic convergence properties. A more intuitive interpretation of the natural gradient is that it gives the steepest direction of the object function under the constraint of

$$d\theta^T G d\theta = const. \tag{5.3}$$

In particular, the natural gradient in an online regime can achieve Fisher efficient estimation asymptotically if the learning rate is chosen correctly [5]. The natural gradient learning has performed well in various fields of machine learning such as neural networks, independent component analysis, reinforcement learning and MCMC methods [6].

However, the naive implementation of natural gradient learning requires the inversion of the metric, which takes too much computational time. In the following section, we propose another novel metric, whose dimension is much less than the traditional Fisher information metric, and it is suitable for practical applications.

5.1.1 Fisher metric of radial parameters

If one can find a subspace of the whole parameter space in which its metric captures most of the original geometric structure, we can develop a natural gradient method that takes less computational time. To generate such a subspace, in this study, we use weight normalization explained in Chapter 4, $\mathbf{w}_i = r_i \mathbf{v}_i / ||\mathbf{v}_i||$, where r_i is a radial parameter and $\mathbf{v}_i / ||\mathbf{v}_i||$ is a vector with unit norm. In weight normalization, each radial parameter r_i determines the scale of the weight vector of the i-th unit and is responsible for its averaged activity. If r_i becomes zero, the i-th unit receives no input and gets eliminated from the network. Therefore, the learning of the radial parameters seems to be crucial to train the network. However, as described in the update (4.8) and (4.14), the effective learning rate appears only for the directional parameters. The learning rate of the radial parameters is not directly tuned in the standard weight normalization. If we tune it by such gradient methods as natural gradient, the convergence speed of the learning may become further improved.

Let us consider a neural network model with each unit represented by $g(\mathbf{w}_i\mathbf{x}+b_i)$ with $\mathbf{w}_i = r_i\mathbf{v}_i/||\mathbf{v}_i||$, where $g(\cdot)$ is an activation function and $\mathbf{x}$ denotes the input from the previous layer. The whole parameter set θ is given by all of the units $\{r_i, \mathbf{v}_i, b_i\}$ in the network. For this parameterizaton, we define a novel Riemannian metric for weight normalized parameters,

$$G_{rad} = \mathrm{E}\begin{bmatrix} \nabla_{\mathbf{r}}l(\mathbf{x};\theta)\nabla_{\mathbf{r}}l(\mathbf{x};\theta)^T & \nabla_{\mathbf{r}}l(\mathbf{x};\theta)\nabla_{\mathbf{b}}l(\mathbf{x};\theta)^T & O \\ \nabla_{\mathbf{b}}l(\mathbf{x};\theta)\nabla_{\mathbf{r}}l(\mathbf{x};\theta)^T & \nabla_{\mathbf{b}}l(\mathbf{x};\theta)\nabla_{\mathbf{b}}l(\mathbf{x};\theta)^T & \\ O & & I_V \end{bmatrix}, \tag{5.4}$$

where $\mathbf{r}$ denotes the vector composed of all radial parameters r_i and $\mathbf{b}$ denotes that of b_i. The matrix entries for radial parameters and bias terms are prat of the original Fisher information matrix. The directional part of the matrix is given by a $\mathrm{dim.}(V) \times \mathrm{dim.}(V)$ identity matrix I_V, where V means a set of the all directional entries v_{ij}. We refer to the above metric as *radial Fisher metric* and the following natural gradient update as *Radial Natural Gradient (RNG) learning* :

$$\theta^{(t+1)} = \theta^{(t)} - \varepsilon G_{rad}^{-1}\nabla l(W^{(t)}). \tag{5.5}$$

The RNG learning is different from the conventional natural gradient learning with the exact Fisher information matrix, G_{full}, because G_{rad} neglects the non-diagonal components including the directional parameters such as $\nabla_V l\nabla_V l^T$, $\nabla_V l\nabla_{\mathbf{r}} l^T$ and $\nabla_V l\nabla_{\mathbf{b}} l^T$. Therefore, the RNG learning is not necessarily invariant under arbitrary coordinate transformation, but it is invariant under any transformation of the radial and bias parameters $(\mathbf{r}', \mathbf{b}') = f(\mathbf{r}, \mathbf{b})$. This property makes the gradient invariant by the non-linear transformation of the inside of the activation function, whose scale $\mathbf{w}_i\mathbf{x}$ is changed or the bias b_i is shifted.

Because the metric of the direction parameters $\mathbf{v}$ is assumed to be Euclidean having only diagonal elements, the computational complexity of the inverse of G_{rad} is much less than that of G_{full}. The computation cost of the inverse of G_{full} depends on the number of parameters K and is $O(K^3)$. In contrast, that of G_{rad} depends on the number of hidden units N and is $O(N^3)$. For example, let us consider a multi-layer network with L layers and M units in each layer. The computational complexity of RNG learning is $O(L^3N^3)$ at each update step. This is much less than that of exact natural gradient $O(L^3N^6)$. In addition, RNG is more efficient than the adaptive natural gradient (ANG) learning proposed by Park et al. [82]. In ANG learning we estimate the inverse of the Fisher information matrix G_{full}^{-1} by using the Sherman–Morrison formula, that is, $G_{t+1}^{-1} = (1+\eta_t)G_t^{-1} + \eta_t G_t^{-1}\nabla_\theta \log p \nabla_\theta \log p^T G_t^{-1}$. Its computational complexity is given by $O(L^2N^4)$. Because most of the deep networks satisfy $N \gg L$, the computational complexity of RNG learning is much less than that of ANG learning, and RNG learning seems to be suitable for large-scale network models.

Furthermore, in Section 5.2, our theories reveal that the radial metric G_{rad} has a similar singular structure as the Fisher metric G_{full}. The radial metric is expected to inherit the essential geometry of G_{full}. In numerical experiments, we confirmed that the radial natural gradient performs well around such singular regions.

5.1.2 Robustness against vanishing gradients

In a multi-layer model with a sigmoidal nonlinearity ϕ such as

$$f(\mathbf{x};\theta) = \phi(W_L\phi(W_{L-1}(\cdots\phi(W_1\mathbf{x}))), \tag{5.6}$$

whose gradient is likely to become close to 0 because the differentiation of a nonlinearity becomes $\nabla_\theta\phi(z;\theta) \simeq 0$ at large $|z|$. Therefore, the gradient almost vanishes and the learning becomes very slow. The natural gradient with the metric G_{full} is known to be robust against this vanishing gradient problem [6]. When the Riemannian metric of the parameter space is given by a matrix F, the natural gradient vector is given by $\tilde{\nabla} l = F^{-1}\nabla l$. The Riemannian magnitude of the natural gradient becomes $||\tilde{\nabla}_\theta l||^2 = \tilde{\nabla}_\theta l^T F \tilde{\nabla}_\theta l = \mathrm{Tr}(\nabla_\theta l \nabla_\theta l^T F^{-1})$.

Concerning the radial natural gradient, we can prove that its magnitude has a finite lower bound so that it effectively overcomes the vanishing gradient problem.

Theorem 6 *The magnitude of the radial natural gradient has a finite bound such that* $\mathrm{E}[||\tilde{\nabla} l||^2] \geq N$ *where N is the number of radial parameters.*

Proof In this proof, we denote bias parameters as 0 for simplicity, but the general case can be easily proved. By taking the average over the data samples, the magnitude of the radial natural gradient is given by $\mathrm{E}[||\tilde{\nabla} l||^2] = \mathrm{Tr}\left(E\begin{bmatrix} \nabla_{\mathbf{r}} l \nabla_{\mathbf{r}} l^T & \nabla_{\mathbf{r}} l \nabla_V l^T \\ \nabla_V l \nabla_{\mathbf{r}} l^T & \nabla_V l \nabla_V l^T \end{bmatrix} G_{rad}^{-1}\right)$, where we denote the differentiation with regard to direction coordinates by ∇_V. Because of

$G_{rad} = \begin{bmatrix} E[\nabla_{\mathbf{r}} l \nabla_{\mathbf{r}} l^T] & O \\ O & I \end{bmatrix}$, we obtain $\mathrm{E}[||\tilde{\nabla} l||^2] = N + \mathrm{Tr}\, E\left[\nabla_V l \nabla_V l^T\right]$. Since the matrix $E\left[\nabla_V l \nabla_V l^T\right]$ is positive semi-definite by definition, the magnitude is equal to or larger than N. In the case including bias parameters, this bound becomes $2N$. □

Note that the conventional natural gradient learning with G_{full} has $\mathrm{E}[||\tilde{\nabla} l||^2] = K$, where K denotes the number of all parameters. Compared to the conventional natural gradient learning, the magnitude may become smaller in RNG learning, but the magnitude has a lower bound and is guaranteed to stay finite.

5.2 Singularity of radial Fisher metric

The Fisher information matrix is always semi-positive definite by definition but is not necessarily regular or positive definite. The parameter spaces of hierarchical systems such as multi-layer perceptrons include singularities due to the symmetry and degeneration of hidden units. Such singular structure is ubiquitous not only in multi-layer perceptrons but also in Gaussian mixture models, Bayesian network, and many other cases [8].

Problems arise in such singular models. Numerical experiments and Dynamical analyses have revealed that these singular regions cause plateaus and the 1st-order gradients like SGD become very slow here [26]. To overcome this problem, natural gradient methods is effective [82, 87, 6]. If the learning approaches a singular region, some eigenvalues of the metric become close to 0 and the inverse of the metric diverges. On the other hand, the value of the gradient ∇l is close to 0, and the natural gradient $G^{-1}\nabla l$ can take finite values. Therefore, the natural gradient can escape from singular region with finite velocity but SGD cannot.

In this section, we theoretically demonstrate that the geometric structure of the Fisher information matrix is preserved in the radial metric. Also, our numerical experiments show that the radial natural gradient can rapidly escape from a singular region where SGD learning becomes slow.

5.2.1 Singularity of radial Fisher metric: the general case

The singular points of the radial parameter space always capture those of the whole weight parameter space as briefly proved in the following theorem:

Theorem 7 *When the radial metric G_{rad} is singular, then the Fisher information matrix G_{full} is singular.*

Proof The radial metric G_{rad} is singular if and only if there exists a nonzero vector $\mathbf{z}$ such that $\mathbf{z}^T G_{rad}\mathbf{z} = 0$. Because of $\mathbf{z}^T G_{rad}\mathbf{z} = \mathrm{E}[(\mathbf{z}^T\nabla_\theta l)(\mathbf{z}^T\nabla_\theta l)^T] = 0$, the condition of singularity is equivalent to the linear dependence among the differentiation of the object function $\nabla_\theta l$, that is, $\mathbf{v}^T\nabla_\theta l = 0$ with nonzero $\mathbf{v}$.

Here, the radial parameters are given by $r_i = \sqrt{\sum_j W_{ij}^2}$ and the chain rule of differentiation becomes $\nabla_r l = J\nabla_W l$, where $J = \frac{dW}{d\mathbf{r}^T}$ is a Jacobian matrix. This Jacobian matrix becomes $J_{ij} = V_{ij}/||\mathbf{v}_i||$ $((i-1)N+1 \le j \le iN)$, otherwise it is 0. We can easily confirm $JJ^T = I$, where I is an $N\times N$ identity matrix.

Therefore, we obtain $\mathbf{z}^T\nabla_\mathbf{r} l = \mathbf{z}^T \begin{bmatrix} J & O \\ O & I \end{bmatrix} \begin{bmatrix} \nabla_W l \\ \nabla_b l \end{bmatrix} = 0$ at the singular regions of G_{rad}. This means that the differentiation $[\nabla_W l \;\; \nabla_b l]^T$ is linearly dependent with a vector $\bar{\mathbf{z}} = \left(\mathbf{z}^T \begin{bmatrix} J & O \\ O & I \end{bmatrix}\right)$. In other words, $\bar{z}$ is the singular region of the whole parameter space W such that $\bar{\mathbf{z}}^T G_{full}\bar{\mathbf{z}} = 0$. Therefore, the Fisher information matrix G_{full} is singular. $\square$

5.2.2 Singularity of radial Fisher metric: 3-layer perceptron network

Theorem 7 gives a general insight into the singular regions of the radial metric, but it is unclear at which parameter G_{rad} and G_{full} become singular simultaneously. As shown in the following propositions, we theoretically found common parameter space where both of G_{rad} and G_{full} become singular in a 3-layer perceptron networks [35].

Let us define the 3-layer perceptron network with N input units x_k $(k = 1,...,N)$, H hidden units, and M output units $f_i(\mathbf{x})$ $(i = 1,..,M)$ by

$$f_i(\mathbf{x};\theta) = \sum_{j=1}^{H} U_{ij}\phi\left(\sum_{k=1}^{N} W_{jk}x_k + b_j\right) + c_i. \tag{5.7}$$

We assume that the conditional probability distribution of output $\mathbf{y}$ given input $\mathbf{x}$ is given by a Gaussian distribution such as $p(y_i|\mathbf{x}) = \exp(||y_i - f_i(\mathbf{x};\theta)||^2/\sigma^2)/\sqrt{2\pi\sigma}$ [33]. Its log-likelihood function is given by $\log p(y|x;\theta)q(x)$, where $q(\mathbf{x})$ is input data distribution. Maximization of the negative log-likelihood is equivalent to minimization of a square error $||y_i - f_i(\mathbf{x};\theta)||^2$. In this case, the Fisher information matrix becomes

$$G_{full} = \sum_i \mathbf{E}_q[\nabla_\theta f_i \nabla_\theta f_i^T]. \tag{5.8}$$

In practice, the expectation over input distribution is computed by the empirical expectation over training samples.

A previous study [33] has revealed that the singular regions of G_{full} (5.8) is limited to the following cases:

Theorem 8 *(proved in [33]) The Fisher information matrix G_{full} in the 3-layer perceptron network is singular if and only if : (A) $\mathbf{u}_j = 0$ for $^\exists j$ or (B) $\mathbf{w}_i = 0$ for $^\exists i$ or (C) $(\mathbf{w}_i, b_i) = \pm(\mathbf{w}_j, b_j)$ for some two different indices i and j.*

Note that we demote $\mathbf{u}_j = (U_{1j}, U_{2j}, ..., U_{Mj})^T$ and $\mathbf{w}_i = (W_{i1}, W_{i2}, ..., W_{iN})$. The all of cases mean the singular regions where the redundant parameters are eliminated. Such singular regions in multi-layer neural networks are known to form saddle structures [35, 87]. The natural gradient method is helpful to avoid them.

The condition (A) corresponds to the case where the j-th unit in the output layer is eliminated. The condition (B) is the case where the i-th unit in the hidden layer is eliminated. The condition (C) is a more non-trivial case where the i-th and j-th units becomes unidentifiable viewed from the output units. The dynamical analysis in the 3-layer perceptron has theoretically revealed that these singular regions cause plateaus and SGD learning becomes very slow here [26]. In particular, dynamical analysis with one output unit and two hidden units has revealed that the singular regions compose so-called Milnor attractor.

At least, with regard to the singular region (A), we found the following fact:

Theorem 9 *Under the condition of singularity (A), the radial metric G_{rad} becomes singular.*

Proof The condition of singularity is equivalent to the linear dependence among the differentiation of the model function $\nabla_\theta \mathbf{f}$, that is, $\mathbf{z}^T \mathbf{f} = \mathbf{0}$ with nonzero $\mathbf{z}$. Note that we denote $\mathbf{f} = (f_1, f_2, ..., f_M)^T$. Without loss of generality, we may assume the index of the singular condition as $j = 1$. Because of $\nabla_{r_1} \mathbf{f} = 0$ and $\nabla_{b_1} \mathbf{f} = 0$, we can get a nonzero vector $\mathbf{z}$, whose entries z_{r_j} and z_{b_j} satisfy $z_{r_j} \nabla_{r_j} \mathbf{f} + z_{b_j} \nabla_{b_j} \mathbf{f} = 0$. Therefore, G_{rad} is singular. $\square$

Furthermore, regarding the singularity (C), we found,

Theorem 10 *Under the assumption of* $\mathbf{u}_i = \pm\mathbf{u}_j$ *and the condition of singularity (C), the radial metric* G_{rad} *becomes singular.*

Proof We denote the radial expression of W as $W_{kl} = r_k^{(W)} V_{kl}^{(W)}$. Without loss of generality, we may assume the index of the singular condition as $i = 1$ and $j = 1$. Under the condition (C), we get $\nabla_{r_1^{(W)}}\mathbf{f} = \phi'\mathbf{u}_1$ and $\nabla_{r_2^{(W)}}\mathbf{f} = \phi'\mathbf{u}_2$, where $\phi' = \nabla_{r_1^{(W)}}(\mathbf{v}_1\mathbf{x})\cdot\phi(\mathbf{w}_1\mathbf{x}+b_1)$. Here let us assume that the weight matrix U, which is written by $U_{kl} = r_k^{(U)} V_{kl}^{(U)}$ in weight normalization, satisfies $\mathbf{u}_i = \pm\mathbf{u}_j$. Under this assumption, we can compose a nonzero vector $\mathbf{z}$ such that $z_{r_1^{(W)}}\nabla_{r_1^{(W)}}\mathbf{f} + z_{r_2^{(W)}}\nabla_{r_2^{(W)}}\mathbf{f} = 0$. Therefore, G_{rad} is singular. □

Although we need additional degenerate weights $\mathbf{u}_i = \pm\mathbf{u}_j$, the singularity (C) also becomes the singularity of the radial metric. Note that, if we train neural network models with tied weight, i.e. $U^T = W$, $\mathbf{u}_i = \pm\mathbf{u}_j$ is naturally derived from the condition (C). The tied weight is widely used in training of auto-encoders or restricted Boltzmann machines.

The singularity (A) is known as the eliminating singularity and the singularity (C) as overlapping singularity [107]. In the natural gradient learning with the Fisher metric, dynamical analysis has theoretically revealed that when the number of hidden units is redundant for representing input data, the learning converges to these singular regions rapidly and makes the convergence of learning faster [26]. On the RNG learning, We can expect the similar effect because the radial metric inherits the singularity of the Fisher metric. In section 5, we empirically evaluate that RNG learning converges faster than SGD learning and avoids the plateaus.

Note that we can extend the above propositions to the case where the activation function at output units becomes nonlinear such as $\phi(f_i)$. We omit the detail, but the necessary techniques for this extension are described in [33].

In the following experiments, we will confirm that our radial natural gradient learning avoids singular regions, which causes plateaus, and converges faster than SGD learning.

5.2.3 Numerical experiments with a toy model

First, to explicitly evaluate the transient dynamics of learning, we trained a simple 3-layer perceptron with 1 output unit, 2 hidden units, and 8 input units. We artificially generated 1000 training samples $\mathbf{x}$ from Gaussian distribution $N(0,1)$ and their teacher signals by $f(\mathbf{x};\theta^*) = \sum_{i=1}^{2}\phi(\mathbf{w}_i\mathbf{x})$ with a true parameter θ^*. We set a mini batch size of 100 in all of training algorithms.

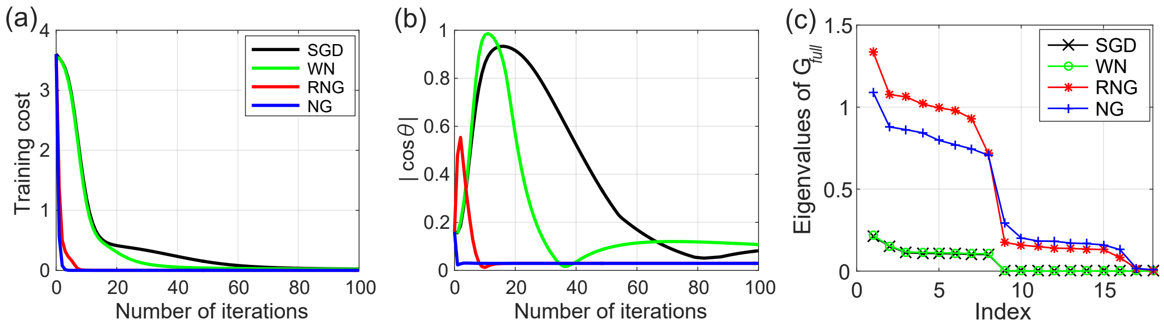


Fig. 5.1 Transient dynamics of supervised learning in simple 3-layer perceptron network with artificial input data. (a) The value of cost function with training samples. (b) Difference of orientation between $\mathbf{w}_1$ and $\mathbf{w}_2$ measured by $|\cos\theta|$. (c) Eigenvalues of the Fisher information matrix G_{full} at $t = 10$.

Figure 5.1 (a) reveals that RNG learning converges much faster than SGD learning and weight normalization (WN) learning. The lines in figures represent expectation among 20 different random initialization seeds, and the training cost means the cost function on training data samples. Because the number of model parameters is small enough, we can compute the exact natural gradient (NG) learning with G_{full}. The dynamics of RNG learning was more similar to that of NG than those of SGD and WN learning.

We can also confirm that RNG learning inherits important properties of NG learning as shown in Figures 5.1 (b) and (c). In Figure 5.1 (b), we show the angle between $\mathbf{w}_1$ and $\mathbf{w}_2$ measured by $\cos\theta = \mathbf{w}_1 \cdot \mathbf{w}_2 / ||\mathbf{w}_1||||\mathbf{w}_2||$. The transient dynamics of SGD and WN learning took long time to approach the singular regions $\mathbf{w}_1 = \pm\mathbf{w}_2$ and to get rid of them. Therefore, even when the true solution exists at regular regions, the singular structure affects the dynamics of learning. Moreover, Figure 5.1 (c) shows that the eigenvalues of SGD and those of WN took near zero values on the way of learning, where the magnitude of the gradient became small. In contrast, the eigenvalues of RNG and those of NG were larger than those of the 1st-order gradient methods. From the above, we can conclude that RNG learning can rapidly escape from the singular regions.

5.3 Related works: efficient approximations of natural gradient

Various kinds of studies have challenged to find a tractable approximation to the Fisher metric G_{full}. The simplest one is a diagonal approximation, but it does not work well. Recently, more sophisticated methods have been developed for the approximation. The unitwise

Fisher metric is an excellent way of reducing computational cost, keeping the performance of the natural gradient high [81]. Topmoumoute online natural gradient (TONGA) is an approximation to the natural gradient that assumes block diagonal structure [88], and low-rank structure within each block. Kronecker-factored approximate curvature (K-FAC) uses a Kronecker product of two much smaller matrices which approximates the coarse structure of the Fisher metric [68]. Factorized natural Gradient (FANG) projects the Fisher metric to a low-dimensional space of sufficient statistics and reduces the size of the metric [41] .

Another approach is to avoid the computation of inverse of the metric. Hessian-free (H-F) optimization uses an iterative method to implicitly compute the inverse of the Fisher metric [67]. It solves the linear system with a series of matrix-vector products.

It is also noteworthy that several studies have proposed the methods to approximately realize natural gradients just by transforming the parameters into the coordinate system where the Fisher matrix becomes an identity [85, 30].

The above works approximate the Fischer metric to some degrees, but it has not been clarified how the singularity changes from the exact Fisher metric. In contrast, we have confirmed that our radial metric inherits some of the fundamental singularities from the exact Fisher metric. It may be possible to combine the above-related works with radial metric and further improve the computational cost and performance.

5.4 Experiments with benchmark dataset

5.4.1 MNIST dataset

To empirically evaluate the performance of the radial natural gradient learning in practical applications, we conducted numerical experiments using multi-layer perceptrons and an MNIST digit recognition dataset (28×28). The dataset is composed of 60000 training samples and 10000 test samples. We normalized the intensity of the MNIST dataset to [0 1] and subtracted the mean activation of each pixel from the data [93].

Figures 5.2 show that the radial natural gradient (RNG) learning converged faster than the stochastic gradient descent (SGD) methods and the conventional weight normalization (WN) in a 4-layer perceptron with tanh nonlinearity. We used cross-entropy as the cost function for multiple classes classification and softmax function as the output $p(y|x)$. Its Fisher metric has been given by Park et al. [82] and we utilized it. The lines in the figures represent the expectation and standard deviation among five different random initialization seeds. The initialization of parameters was given by uniform distributions [38]. Note that we selected the hyperparameters of SGD and WN from the following grid specifications: learning rates in $[1, 0.5, 0.1, 0.05, 0.01]$, the coefficient of L2 regularization in $[10^{-3}, 10^{-4}, 10^{-5}]$, and mini-batch size in $[250, 500]$. We avoided using structure-dependent learning rates such as annealing with step because we want to check unalloyed effects of our proposed algorithms rather than to achieve the state-of-the-art performance. We only used L2 normalization to prevent overfitting in all experiments. In RNG learning, we further searched a learning rate of the radial parameters in $[0.01, 0.005, 0.001]$ to prevent the natural gradient from diverging. The Figure 5.2 shows the best learning trajectories with the hyperparameters which achieved the lowest test classification errors. In both of the update step number and the CPU time, the training cost and the test classification error of RNG converged faster than those of SGD and NG. Besides, the RNG has achieved better generalization errors than the other methods.

Let us look at some technical issues to improve natural gradient learning. First, to avoid numerical instability, we added a damping parameter η in the inverse of the metric such as $(G_{rad} + \eta I)^{-1}$ [68]. we also searched the damping term in $[10^{-3}, 10^{-4}, 10^{-5}]$. Second, we adaptively updated the metric by

$$G_{rad}^{(t+1)} = \beta G_{rad}^{(t)} + (1-\beta) \sum_{i}^{t} \nabla f(x_i) \nabla f(x_i)^T / T, \tag{5.9}$$

with $\beta = 0.9$ to reduce the effect of noise caused by mini-batch input [82, 70]. Note that the most expensive operation regarding computational cost in RNG learning is the computation of the inverse of G_{rad}. Fortunately, the metric does not need to be updated in every iteration, since it is unlikely to change very quickly during the training. In our experiments, we update the metric and computed its inversion once every 100 mini-batches.

As shown in Figure 5.2 (c) and (d), We also confirmed the training of 4-layer networks with ReLU units. We set the network size and the hyperparameters in the same way as in the case of Tanh. Compared to the Tanh network in Figure 5.2 (a) and (b), our RNG gave less improvement of the WN learning on the speed of convergence and the generalization error. Because the WN learning performs scale invariance as investigated in Chapter 4.4, the effect of the radial natural gradient seems to be limited in the ReLU networks. From the above, we can conclude that RNG performs better than SGD and WN, especially in the multi-layer Tanh networks.

5.4.2 CIFAR-10 dataset

Next, we also conducted numerical experiments using multi-layer perceptrons and a CIFAR-10 dataset. The CIFAR-10 dataset consists of (32×32) pixel color images classified to 10 different classes with 50000 training samples and 10000 test samples. We normalized each channel of each pixel to [0 1] and to zero mean. We set the network size and the hyperparameters in the same way as the experiments with MNIST.

The Figure 5.3 (a) and (b) show the best learning trajectories with the hyperparameters which achieved the lowest test classification errors. The all gradient methods were more likely to overfit than the training with MNIST. Therefore, we estimated the lowest test error by early stopping. The training cost and the test classification error of RNG converged faster than those of SGD and NG in both 3-layer and 4-layer perceptrons with the tanh nonlinearity. The RNG achieved the lowest test classification errors. than the other methods.

5.5 Discussion

We have introduced a novel natural gradient algorithm by using radial parameters of weight matrices, which greatly reduces the computational cost of the inverse of the metric. In this study, although we focused on the supervised learning in multi-layer perceptrons, our method is applicable and very suitable for various kinds of training in hierarchical models with weight matrices such as convolutional neural networks and variational autoencoders. In such models, our natural gradient method will be helpful for accelerating the convergence of the learning.

In our experiments, because the radial metric is small enough for computing the inverse of the metric matrix, we have directly computed the inversion and have not applied any approximation for avoiding the computation of the inverted matrix. In the case of a very large network or where the computational time and memory space are limited, it will be helpful to combine our method with the adaptive natural gradient (ANG) or algorithms for more large-scale problems such as Hessian-Free methods or block diagonal approximation of the matrix [69, 88]. In addition, Martens remarked that the on-line optimization of a damping parameter considerably improves the performance of natural gradients [70]. Applying such technical issues to radial natural gradients will further accelerate the convergence of learning.

In this study, we considered the Riemannian structure only for radial parameters and assumed the Euclidean space for directional parameters. If one can find a metric with the low computational cost for the directional parameters, it may further improve the convergence of learning. Because the directional parameters are constrained to the high-dimensional sphere, it may be helpful to use the projection methods to the sphere as is used in the algorithms for independent component analysis [48].

It is also interesting to apply the dynamical analysis of learning into the radial natural gradient [26, 107]. It will elucidate how the learning trajectory avoids the plateau phenomena caused by the singular regions and theoretically prove the effectiveness of the radial parameterization.

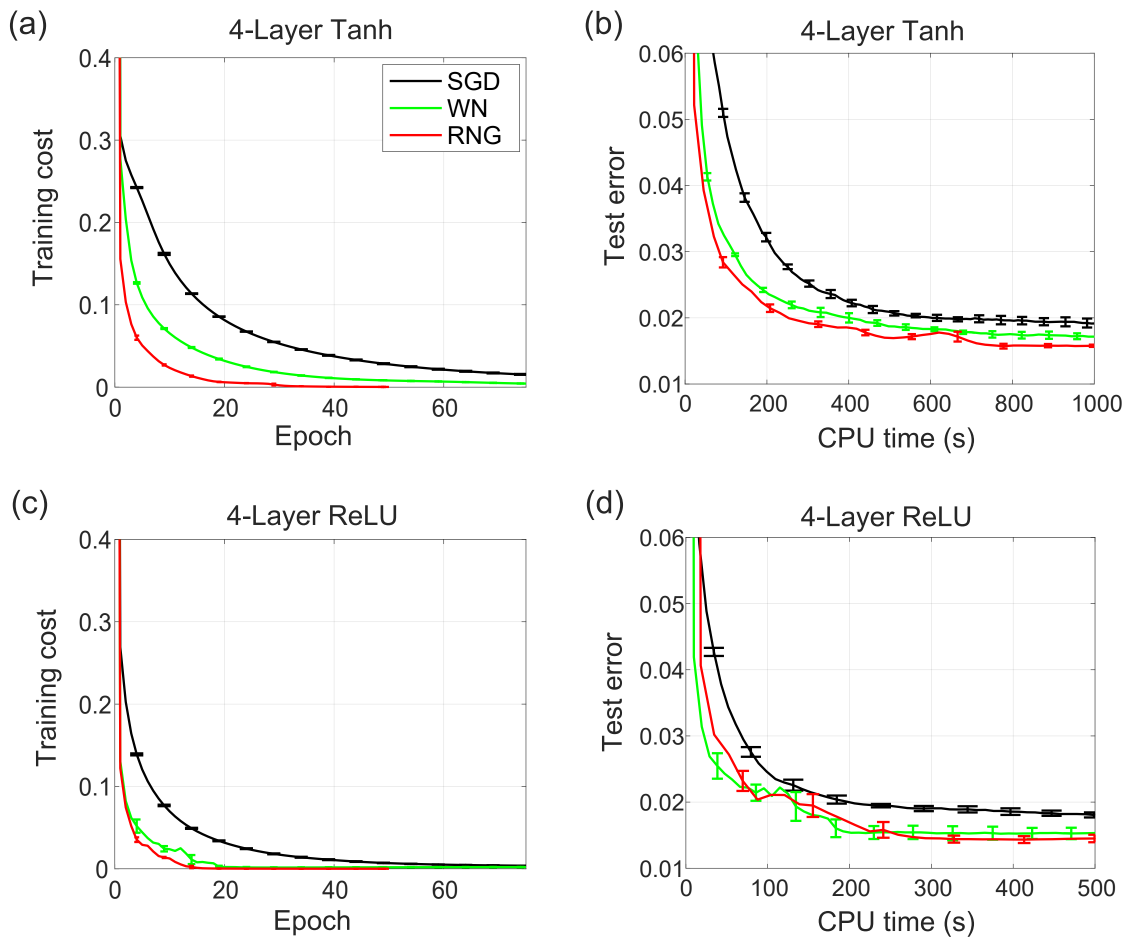


Fig. 5.2 Transient dynamics of supervised learning on MNIST. We trained 4-layer networks with 1000 units in each hidden layer. (a) Cost function on training data samples. The cost function consists of the cross-entropy and the activation function is given by Tanh. We represented the SGD learning as the black line, the weight normalization learning as the green line, and our proposed method as the red line. (b) Classification error rate on test data samples. (c) Cost function on training data samples with the ReLU network. (d) Classification error rate on test data samples with the ReLU network.

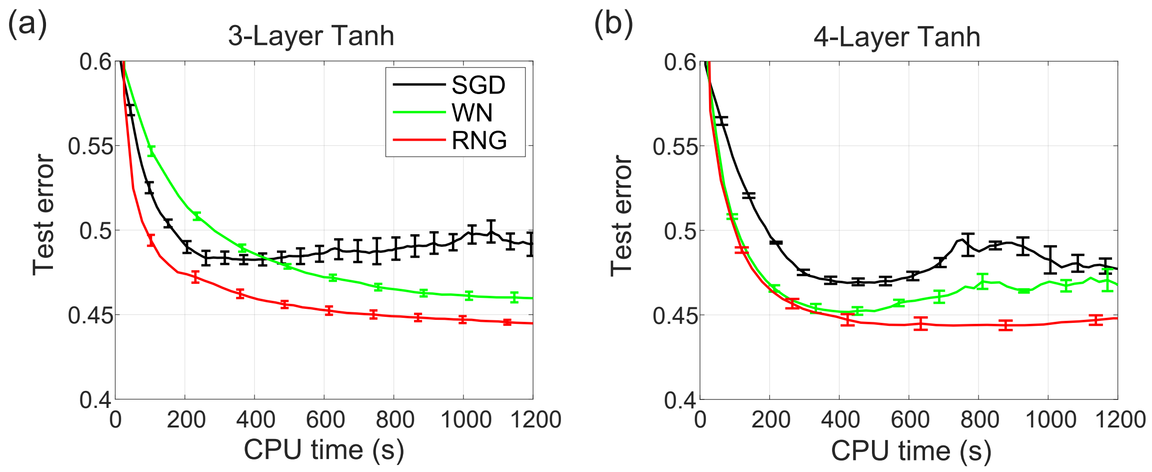


Fig. 5.3 Transient dynamics of supervised learning on CIFAR-10. (a) Classification error rate on test data samples in the 3-layer network. 1000 hidden units are given by Tanh function. (b) Classification error rate on test data samples in the 4-layer network. 1000 hidden units are given by Tanh function.

Chapter 6

Conclusion

In this thesis, we demonstrated analyses of the learning algorithms for deep learning and proposed novel algorithms based on the analyses. Especially, we focused on two types of heuristic learning algorithms: contrastive divergence learning in RBMs and a gradient method known as weight normalization.

In Chapter 2, we performed the dynamical analysis on the contrastive divergence learning in Gaussian-Gaussian RBM and proved that it has the same solutions with maximum likelihood learning. In the case of Gaussian-Bernoulli RBM, we found that at least one of the contrastive divergence solutions coincides with one of the maximum likelihood solutions. Our theoretical results support the experimental knowledge that maximum likelihood solutions are obtained simply by performing CD_1 learning. In Chapter 4, we analyzed the weight normalization, which has been empirically known as a method to accelerate the convergence of learning. By investigating Jacobian matrix of the coordinate transformation, we revealed that the acceleration is attributed to the effective learning rate, which realizes the automatic turning of the given learning rate and scale invariant gradients.

As remarked in Chapter 1, we have worked on not only the analyses of learning algorithms but also the propositions of novel efficient learning algorithms. In Chapter 3, based on our theoretical insight obtained by the analysis in Chapter 2, we derived analytically tractable maximum likelihood learning for the RBMs with Gaussian visible units. The orthogonal constraint, which is a natural assumption derived from the structure of the weight matrix, enabled us to propose the novel efficient algorithm along the geodesic flow. In Chapter 5, we also proposed a natural gradient algorithm for the weight normalization and investigated its effectiveness in the numerical experiments.

The heuristic algorithms investigated here may lead to the extension of the traditional techniques in machine learning. The contrastive divergence has been suggested to be efficient for the maximum likelihood estimation of exponential family models [112, 50]. Such reparameterization approach as the weight normalization and natural neural networks [30] will be applied to other hierarchical models similar to the neural networks such as mixture models and factor analysis [6, 107]. Besides, the mathematical analyses and improvement of the heuristics shown in this thesis will be expected to enable the theory of deep learning to advance further. As a final remark, we discuss the possible directions of the theoretical advances related to this study as bellow.

6.1 Efficient algorithms based on geometry

As shown in Chapter 2, the maximum-likelihood learning in Gaussian-Gaussian RBM has the cost function without local minima. There is one global minimum and then the other fixed points are saddle points. Interestingly, several theories have revealed that the cost function of the multi-layer discriminative models with a linear activation function also has the same geometric structure [55, 11]. Besides, experiments with nonlinear activation function have also observed similar phenomena [28]. Therefore, this geometric structure of the landscape seems to be an universal property in neural network models or even in other non-convex optimization problems [28, 37]. It will be helpful for speeding up the convergence of learning to develop the algorithms that easily escape from the saddle points, such as the saddle-free Newton's method.

We expect that it will also be one of the promising direction to investigate the effectiveness of the natural gradient methods. Because the conventional optimization methods in neural networks have based on the 1st-order gradients, the annoying adjustment of the learning coefficient is inevitable to make the learning converge. Although the recently developed 1st-order heuristics like Adam [56] can partially avoid the adjustment, geometric optimization methods like the natural gradient seems to be more efficient because they automatically tune the learning rate by using the curvature of the parameters. In addition, as we introduced the geodesic flow of the weight matrix on the Stiefel manifold in Chapter 3, the natural gradient method with the orthogonal constraints may be effective in other neural network models. Saxe et al. have pointed out that the orthogonal constraints on weight matrices raise the acceleration of the convergence of the backpropagation algorithm because the orthogonality prevents the vanishing gradient phenomena [92]. Similar approaches also succeeded in the optimization of the recurrent neural networks [9]. Furthermore, there are other types of natural gradient methods, for instance, the natural gradient constraining the weight matrix to be regular [7]. This constraint may enable us to accelerate the convergence of learning because the irregularity of the weight matrix causes the plateau phenomena as discussed in Chapter 6. It is also noteworthy that the information of the curvature can prevent the overfitting in some experiments with natural gradients [83, 70].

In this study, we concentrate on the specific geometric structure observed in learning with RBMs and weight normalization. With regards to more general geometric structure, information geometry gives mathematical insight [6]. In the information geometry, various kinds of learning algorithms are described by universal frameworks, for instance, the dual projections between input data and model distributions. The framework of information geometry may be helpful to unify or to give a better understanding of the recent algorithms developed for neural network models such as score matching [46], minimum probability flow [98], variational auto-encoder [57] and generative adversarial network [39]. Let us remark that our recent work has already given elementary results about the information geometry of the score matching [52]. Using the Riemannian metric of the score functions, it derived the natural gradient algorithm for score matching methods.

6.2 Residual subjects for theory of learning

In Chapter 2 and 3, we have investigated the RBMs with continuous Gaussian units. In contrast, analysis of RBMs with discrete, in particular, binary visible and hidden units remain as a matter to be discussed. The difficulty of analysis in the discrete cases seems not to be peculiar to the generative models like RBMs but common with discriminative models with nonlinear activation functions. As is described in Chapter 1, several studies have recently investigated the learning dynamics and its solution space in deep linear networks. However, this network performs only linear transformation, that is, principal component analysis regardless of the number of layers. This is too simple information processing compared to those observed in practical neural network models with realistic datasets. It will be necessary to construct the theory of learning in the models with the discrete random variables or the nonlinear activation functions. To attack this problem, analytical methods developed in statistical physics will be helpful such as the dynamical mean-field theory in random neural network models [84] and the replica method developed in associative memory models [2].

One more essential question which remains to be uncovered is what kind of information is extracted in a hierarchical structure of the network. Some numerical experiments have reported that a higher layer is likely to extract higher level abstraction of input information, which may allow much easier generalization performance and transfer learning [18]. However, this tendency has not been confirmed by any theory. In addition, the relevance between the extracted features and the landscape of the error function with no local minima is also unclear. We expect that expanding the dynamical analysis shown in Chapter 3 will be a promising way to solve these problems.

It is also a challenge for the future to bridge the gap between learning in machine learning and that in biology. The most basic technique in training neural networks, that is, the backpropagation algorithm, is biologically implausible [17]. It seems to be non-trivial where the cost function and its derivatives are represented in the realistic neural networks and how the derivatives are transmitted thorough synaptic weights. Bridging the gap may inspire heuristic algorithms for machine learning based on Hebbian learning rule or other synaptic learning rules in neuroscience.

As known from the above, there are still many problems to overcome for constructing the theory of learning in neural networks. Attacking the remaining problems will enable us to establish mathematical foundations of the neural algorithms and to build a better system of artificial intelligence based on them.

Appendix A

Dynamical analysis including bias parameters

Gaussian-Gaussian RBM

As shown in Chapter 2, we have derived stable fixed points of ML and CD$_n$ learning rules of the weight matrix W. Although We assumed that bias parameters are zero in the main claims, We can also derive stable fixed points of learning rules including bias parameters and generalize our theoretical results.

The ML learning rules of **b** and **c** are defined as follows:

$$\tau\frac{d\mathbf{b}}{dt} = \Sigma^{-2}\{<\mathbf{v}>_0 - <\mathbf{v}>_\infty\}, \tag{A.1}$$

$$\tau\frac{d\mathbf{c}}{dt} = S^{-2}\{<\mathbf{h}>_0 - <\mathbf{h}>_\infty\}. \tag{A.2}$$

The simultaneous learning rules of the parameters $\{W,\mathbf{b},\mathbf{c}\}$ become the following equations:

$$\tau\frac{dW}{dt} = F + W\Sigma^{-1}(\mu\mu^T - \eta\eta^T)\Sigma^{-1} + S^{-1}\mathbf{c}(\mu-\eta)^T\Sigma^{-1}, \tag{A.3}$$

$$\tau\frac{d\mathbf{b}}{dt} = \Sigma^{-2}(\mu-\eta), \tag{A.4}$$

$$\tau\frac{d\mathbf{c}}{dt} = S^{-1}W\Sigma^{-1}(\mu-\eta), \tag{A.5}$$

where we define $F \equiv W\Sigma^{-1}C\Sigma^{-1} - W(I_N - W^TW)^{-1}$ and $\eta \equiv \Sigma(I_N - W^TW)^{-1}\left(W^TS^{-1}\mathbf{c} + \Sigma^{-1}\mathbf{b}\right)$. Let us define the mean value of input data as $\mu = \int d\mathbf{v}q(\mathbf{v})\mathbf{v}$ and the data covariance as $C = \int d\mathbf{v}q(\mathbf{v})\mathbf{v}\mathbf{v}^T - \mu\mu^T$.

In the following dynamical analysis, we assume homogeneous variances $\Sigma = \sigma I_N$ and $S = sI_M$ as in Section 2.3. The condition $d\mathbf{b}/dt = \mathbf{0}$ gives $\mu = \eta$, that is,

$$\mu = (I_N - W^TW)^{-1}\left(\frac{\sigma}{s}W^T\mathbf{c} + \mathbf{b}\right). \tag{A.6}$$

This equation satisfies $d\mathbf{c}/dt = \mathbf{0}$ and requires $\tau dW/dt = F$. Because $\tau dW/dt = F$ is equivalent to a learning rule with no bias parameter, the fixed point is $W = UAV$ obtained in

Lemma 1. Therefore, (A.6) with the $W = UAV$ is a necessary and sufficient condition that the fixed points of $\mathbf{b}$ and $\mathbf{c}$ must satisfy.

It is a necessary and sufficient condition for stable points that the following inner product is negative for any perturbation:

$$\mathrm{Tr}\left(\Delta W^T \frac{dW}{dt}\right) + \Delta\mathbf{b}^T \frac{d\mathbf{b}}{dt} + \Delta\mathbf{c}^T \frac{d\mathbf{c}}{dt}, \tag{A.7}$$

where we denote perturbations around the fixed point by $W + \Delta W, \mathbf{b} + \Delta\mathbf{b}$, and $\mathbf{c} + \Delta\mathbf{c}$. After straightforward calculation, the inner product (A.7) can be reduced to the following form:

$$\mathrm{Tr}\left(\Delta W^T \Delta F\right) - \frac{1}{\sigma^2}\Delta\eta^T (I_N - W^T W)\Delta\eta. \tag{A.8}$$

The notation $\Delta\eta$ represents the first-order expansion of η:

$$\begin{aligned}\Delta\eta = &(I_N - W^T W)^{-1}\left(\frac{\sigma}{s}\Delta W^T \mathbf{c} + \frac{\sigma}{s} W^T \Delta\mathbf{c} + \Delta\mathbf{b}\right)\\ &+ (I_N - W^T W)^{-1}(W^T \Delta W + \Delta W^T W)\eta.\end{aligned} \tag{A.9}$$

Because $(I_N - W^T W)$ is a positive semidefinite matrix, the second term of the inner product (A.8) is negative for any perturbation. In addition, $\mathrm{Tr}\left(\Delta W^T \Delta F\right)$ is negative if and only if W becomes the stable fixed point as obtained in Theorem 1. Consequently, the stable points are the W obtained in Theorem 1 and such $\mathbf{b}$ and $\mathbf{c}$ that satisfy the relation (A.6).

We can also show that the above fixed points are stable points of CD$_n$ learning rule. The CD$_n$ learning rules including bias parameters correspond to (A.3 - A.5) with $F \equiv W\Sigma^{-1}C\Sigma^{-1} - W\left\{(W^T W)^n \Sigma^{-1} C \Sigma^{-1} (W^T W)^n + \sum_{k=0}^{2n-1}(W^T W)^k\right\}$ and $\eta \equiv \Sigma(W^T W)^n \Sigma^{-1}\mu + \Sigma\sum_{k=0}^{n-1}(W^T W)^k\left(W^T S^{-1}\mathbf{c} + \Sigma^{-1}\mathbf{b}\right)$. In a similar process as shown in ML learning, fixed points become $W = UAV$ obtained in Lemma 2 and $\mu = \eta$. Using an identity, $(I_N - (W^T W)^n)(I_N - W^T W)^{-1} = \sum_{k=0}^{n-1}(W^T W)^k$, we can reduce $\mu = \eta$ of CD$_n$ learning to that of ML learning (A.6). We can also prove the stability of the fixed points in the same manner as ML learning.

Gaussian-Bernoulli RBM

We can extend the stable points of ML learning rule obtained in Section 2.4 into those of ML learning rule including bias parameters. We use assumption 1) shown in Theorem 3 and assume the fixed point to be represented by $W = DB^T$. First, we substitute $W = DB^T$ into $dW/dt = O$ and obtain

$$< s_i g\left(d_i s_i/\sigma + c_i\right) >_{p(s_i)} = (\sigma d_i + z_i) g\left(a_i\right), \tag{A.10}$$

$$\mu_i < g\left(d_j s_j/\sigma + c_j\right) >_{p(s_j)} = (\sigma d_i g(a_i) + z_i) g(a_j) \quad (i \neq j), \tag{A.11}$$

where we define $\mathbf{z} \equiv B^T\mathbf{b}$ and $a_i \equiv d_i^2/2 + d_i z_i/\sigma + c_i$. Second, the condition of $d\mathbf{b}/dt = \mathbf{0}$ gives

$$\mu_i = \sigma d_i g\left(a_i\right) + z_i. \tag{A.12}$$

Last, the condition of $d\mathbf{c}/dt = \mathbf{0}$ gives

$$< g\,(d_i s_i/\sigma + c_i) >_{p(s_i)} = g(a_i). \tag{A.13}$$

Because we can reproduce the condition (A.11) by multiplying conditions (A.12) and (A.13), the conditions (A.10), (A.12), and (A.13) are necessary and sufficient for $W = DB^T$ to become the equilibrium.

Here, we assume c_i to be expanded by d_i such that $c_i = \gamma_i d_i^2/\sigma$, where γ_i is a constant coefficient. Under this assumption, we can relax the non-negativity, $s_i \geq 0$, assumed in Theorem 3 into $s_i \geq -\gamma_i d_i$ because the parameter c_i shifts the argument of the sigmoid function from $d_i s_i/\sigma$ to $d_i s_i/\sigma + c_i = d_i/\sigma(s_i + \gamma d_i)$. In addition, if we also assume $d_i \gg 1$ and $a_i \gg 1$ in (A.10), (A.12) and (A.13), the sigmoid function converges to 1, and then, we obtain the relation that z_i and d_i should satisfy

$$z_i = \sigma d_i - \mu_i. \tag{A.14}$$

Substituting this equation of z_i into a_i, we obtain $a_i = (1/2 - \gamma_i/\sigma)d_i^2 + \mu_i d_i/\sigma$. Because we assumed $d_i \gg 1$ and $a_i \gg 1$ to obtain (A.14), γ_i needs to satisfy $\gamma_i > 2\sigma$. As a result, we obtain the extended ML solutions $W = DB^T$ and $c_i = \gamma_i d_i^2/\sigma \quad (\gamma_i > 2\sigma)$, where $d_i \gg 1$ and z_i satisfy (A.14).

In a process shown in Gaussian-Gaussian RBM, we can also prove the stability of this fixed point. Also, we can obtain the same stable point in the case of CD$_n$ learning rules including the bias parameters.

Appendix B

Learning dynamics of model variance σ^2

In Chapter 2, we fixed the model variance σ^2 to a constant value for simplicity. We can also derive the stable fixed points of learning rules including learning of σ.

Gaussian-Gaussian RBM

In Gaussian-Gaussian RBM, ML learning rule of σ is given by

$$\tau\frac{d\sigma}{dt} = -\frac{dKL(q(\mathbf{v})||p(\mathbf{v}))}{dt} \tag{B.1}$$

$$= \frac{\sum_{i=1}^{N}\lambda_i - \sigma \mathrm{Tr}(W^TWC)}{\sigma^3} - \frac{N}{\sigma}. \tag{B.2}$$

The fixed points σ and W are derived by solving simultaneous equations $d\sigma/dt = 0$ and $dW/dt = O$. As described below, we can conclude that the learning of σ is unnecessary.

The fixed point W becomes the same one obtained in Theorem 1. When $M \geq N$, the stable fixed point becomes $\sigma^2 = \lambda_N$. The G-G RBM extracts all of the input eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_N = \sigma^2$ and cannot perform any dimension reduction of the input. Therefore, the learning of σ seems to be unnecessary for the RBM to realize the noise reduction. When $M < N$, it becomes $\sigma^2 = \sum_{i=M+1}^{N}\lambda_i/(N-M)$. In this case, $N-M$ eigenvalues, i.e., $\lambda_{M+1} > \cdots > \lambda_N$, are not extracted. In other words, the principal eigenvalues are extracted in the G-G RBM as many as the number of hidden units M. Because we can obtain the same solution without training σ, the learning of σ is unnecessary.

CD_n learning rule of σ is given by

$$\tau\frac{d\sigma}{dt} = -\frac{dKL(q(\mathbf{v})||p_n(\mathbf{v}))}{dt} \tag{B.3}$$

$$= \frac{\sum_{i=1}^{N}\lambda_i - \sigma \mathrm{Tr}((I-W^TW)\left\{\sigma^{-2}(W^TW)^nC(W^TW)^n + \sum_{k=0}^{2n-1}(W^TW)^k\right\})}{\sigma^3} - \frac{N}{\sigma}. \tag{B.4}$$

In the same process as ML learning of σ, we can derive the stable fixed points of CD_n method including the learning of σ and find that they coincide with those obtained by ML learning.

Gaussian-Bernoulli RBM

ML learning rule of σ is given by

$$\tau \frac{d\sigma}{dt} = < \frac{||\mathbf{v}||^2 - \sigma g(\sigma^{-1} W\mathbf{v})^T W\mathbf{v}}{\sigma^3} >_0 - \frac{N}{\sigma}. \tag{B.5}$$

As assumed in Chapter 2.4, let us assume that the input data are generated from the independent information sources. The fixed point W becomes the same one obtained in Theorem 4. By using the similar deformation of formula as shown in the proof of Theorem 4, we can obtain one of stable fixed points, that is, $\sigma^2 = \sum_{k=1}^{N} \mathrm{Var}[s_k]$. Because σ only affects the coefficient of the ICA solution, the learning of σ does not change independent feature extraction realized in G-B RBM. In the same process, we can also derive the stable fixed points of CD_n learning, which are the same W and σ as obtained in ML learning.

Appendix C

Steepest descent direction with a spherical coordinate system

Instead of weight normalization, let us consider a reparameterization on the N dimensional sphere, that is, $\mathbf{w} = r\mathbf{e}$, where r denotes a radial length and $\mathbf{e}$ denotes a unit vector. In this case, the steepest decent direction at time step t is given by

$$r(t+1) = r(t) - \eta \nabla_r l, \tag{C.1}$$
$$\tilde{\mathbf{e}}(t+1) = \mathbf{e}(t) - \eta \nabla_{\mathbf{e}} l, \tag{C.2}$$
$$\mathbf{e}(t+1) = \tilde{\mathbf{e}}(t+1)/||\tilde{\mathbf{e}}(t+1)||. \tag{C.3}$$

Note that update (C.3) means the projection onto the sphere.

When we assume the learning rate is small enough and take a first order approximation of the update (C.3), we obtain

$$\mathbf{e}(t+1) \sim \mathbf{e}(t) - \eta r (I - \mathbf{e}(t)\mathbf{e}(t)^T) \nabla_{\mathbf{w}} l. \tag{C.4}$$

Compared to the update of weight normalization (4.8), the effective learning rate of the directional parameters replaced from $r/||\mathbf{v}||$ to r. From the viewpoint of the parameter transformation, the Jacobian matrix becomes

$$J = \frac{\partial w}{\partial e} = \begin{bmatrix} \mathbf{e}^T \\ r(I - \mathbf{e}\mathbf{e}^T) \end{bmatrix}. \tag{C.5}$$

Then, the learning dynamics in the Cartesian coordinate system W follows

$$\Delta \mathbf{w} = -\eta J^T J \nabla_{\mathbf{w}} l \tag{C.6}$$
$$= -\eta \left\{ \mathbf{e}\mathbf{e}^T + r^2 (I - \mathbf{e}\mathbf{e}^T) \right\} \nabla_{\mathbf{w}} l. \tag{C.7}$$

Therefore, the steepest descent direction with the spherical coordinate system does not perform the automatic turning of the given learning rate shown in Chapter 4.3. It performs only the scale invariance shown in Chapter 4.4.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[2] E. Agliari, A. Barra, A. Galluzzi, F. Guerra, and F. Moauro. Multitasking associative networks. *Physical review letters*, 109(26):268101, 2012.

[3] S. Akaho and K. Takabatake. Information geometry of contrastive divergence. In *Proceedings of International Conference on Information Theory and Statistical Learning*, 2008.

[4] S.-i. Amari. Neural theory of association and concept-formation. *Biological Cybernetics*, 26(3):175–185, 1977.

[5] S.-i. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[6] S.-i. Amari. Natural gradient learning and its dynamics in singular regions. In *Information Geometry and Its Applications*, pages 279–314. Springer, 2016.

[7] S.-i. Amari, T.-P. Chen, and A. Cichocki. Stability analysis of learning algorithms for blind source separation. *Neural Networks*, 10(8):1345–1351, 1997.

[8] S.-i. Amari, H. Park, and T. Ozeki. Singularities affect dynamics of learning in neuromanifolds. *Neural computation*, 18(5):1007–1065, 2006.

[9] M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. *arXiv preprint arXiv:1511.06464*, 2015.

[10] V. Badrinarayanan, B. Mishra, and R. Cipolla. Symmetry-invariant optimization in deep networks. *arXiv preprint arXiv:1511.01754*, 2015.

[11] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[12] P. Baldi, P. Sadowski, and D. Whiteson. Enhanced higgs boson to τ+ τ- search with deep learning. *Physical review letters*, 114(11):111801, 2015.

[13] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[14] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–828, 2013.

[15] Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.

[16] Y. Bengio and O. Delalleau. Shallow vs. deep sum-product networks. In *Proceedings of Advances in Neural Information Processing*, 2011.

[17] Y. Bengio, D.-H. Lee, J. Bornschein, and Z. Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[18] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In *Proceedings of International Conference on Machine Learning*, 2013.

[19] Y. Bengio, E. Thibodeau-Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *Proceedings of International Conference on Machine Learning*, 2014.

[20] M. Bianchini and F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

[21] W. N. Butos. *The Social Science of Hayek's The Sensory Order*, volume 13. Emerald Group Publishing, 2010.

[22] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2005.

[23] T. Chen and S.-i. Amari. Unified stabilization approach to principal and minor components extraction algorithms. *Neural Networks*, 14(10):1377–1387, 2001.

[24] T. Chen, Y. Hua, and W.-Y. Yan. Global convergence of Oja's subspace algorithm for principal component extraction. *IEEE Transactions on Neural Networks*, 9(1):58–67, 1998.

[25] K. Cho, A. Ilin, and T. Raiko. Improved learning of gaussian-bernoulli restricted boltzmann machines. In *Proceedings of International Conference on Artificial Neural Networks*, 2011.

[26] F. Cousseau, T. Ozeki, and S.-i. Amari. Dynamics of learning in multilayer perceptrons near singularities. *Neural Networks, IEEE Transactions on*, 19(8):1313–1328, 2008.

[27] G. Dahl, A. Mohamed, and G. E. Hinton. Phone recognition with the mean-covariance restricted Boltzmann machine. In *Proceedings of Advances in Neural Information Processing Systems*, 2010.

[28] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of Advances in Neural Information Processing Systems*, 2014.

[29] G. Desjardins, R. Pascanu, A. Courville, and Y. Bengio. Metric-free natural gradient for joint-training of boltzmann machines. *arXiv preprint arXiv:1301.3545*, 2013.

[30] G. Desjardins, K. Simonyan, R. Pascanu, et al. Natural neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2015.

[31] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[32] S. Fiori. Quasi-geodesic neural learning algorithms over the orthogonal group: A tutorial. *Journal of Machine Learning Research*, 6(May):743–781, 2005.

[33] K. Fukumizu. A regularity condition of the information matrix of a multilayer perceptron network. *Neural networks*, 9(5):871–879, 1996.

[34] K. Fukumizu. Effect of batch learning in multilayer neural network. In *Proceedings of International Conference on Neural Information Processing*, 1998.

[35] K. Fukumizu and S.-i. Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000.

[36] M. Gabrié, E. W. Tramel, and F. Krzakala. Training restricted boltzmann machine via the thouless-anderson-palmer free energy. In *Proceedings of Advances in Neural Information Processing Systems*, 2015.

[37] R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points-online stochastic gradient for tensor decomposition. In *Proceedings of Conference on Learning Theory*, pages 797–842, 2015.

[38] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2010.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[40] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. *Proceedings of International Conference on Machine Learning*, 2013.

[41] R. Grosse and R. Salakhudinov. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. *Journal of Machine Learning Research*, 37:2304–2313, 2015.

[42] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[43] G. E. Hinton. A practical guide to training restricted Boltzmann machines. Technical report, 2010-003, University of Toronto, 2010.

[44] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[45] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[46] A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

[47] A. Hyvärinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on Neural Networks*, 18(5):1529–1531, 2007.

[48] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2001.

[49] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[50] B. Jiang, T.-Y. Wu, and W. H. Wong. Convergence of contrastive divergence with annealed learning rate in exponential family. *arXiv preprint arXiv:1605.06220*, 2016.

[51] R. Karakida, M. Okada, and S.-i. Amari. Analyzing feature extraction by contrastive divergence learning in RBMs. In *Deep Learning and Representation Learning Workshop: NIPS 2014*, 2014.

[52] R. Karakida, M. Okada, and S.-i. Amari. Adaptive natural gradient learning algorithms for unnormalized statistical models. In *Proceedings of International Conference on Artificial Neural Networks*, 2016.

[53] R. Karakida, M. Okada, and S.-i. Amari. Dynamical analysis of contrastive divergence learning: Restricted boltzmann machines with gaussian visible units. *Neural Networks*, 79:78–87, 2016.

[54] R. Karakida, M. Okada, and S.-i. Amari. Maximum likelihood learning of rbms with gaussian visible units on the stiefel manifold. In *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2016.

[55] K. Kawaguchi. Deep learning without poor local minima. *Proceedings of Advances in Neural Information Processing Systems*, 2016.

[56] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[57] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[58] O. Krause, A. Fischer, T. Glasmachers, and C. Igel. Approximation properties of dbns with binary hidden units and real-valued visible units. In *Proceedings of International Conference on Machine Learning*, 2013.

[59] A. Krizhevsky, G. E. Hinton, et al. Factored 3-way restricted boltzmann machines for modeling natural images. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2010.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*.

[61] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of International Conference on Machine Learning*, 2008.

[62] N. Le Roux and Y. Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, 20(6):1631–1649, 2008.

[63] N. Le Roux and Y. Bengio. Deep belief networks are compact universal approximators. *Neural computation*, 22(8):2192–2207, 2010.

[64] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[65] H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area V2. In *Proceedings of Advances in Neural Information Processing Systems*, 2008.

[66] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of International Conference on Machine Learning*, 2013.

[67] J. Martens. Deep learning via hessian-free optimization. In *Proceedings of International Conference on Machine Learning*, 2010.

[68] J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of International Conference on Machine Learning*, 2015.

[69] J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of International Conference on Machine Learning*, 2011.

[70] J. Martens and I. Sutskever. Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012.

[71] G. Montavon, K.-R. Müller, and M. Cuturi. Wasserstein training of boltzmann machines. *arXiv preprint arXiv:1507.01972*, 2015.

[72] G. Montufar and N. Ay. Refinements of universal approximation results for deep belief networks and restricted boltzmann machines. *Neural Computation*, 23(5):1306–1319, 2011.

[73] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2014.

[74] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of International Conference on Machine Learning*, 2010.

[75] S. Nakajima, M. Sugiyama, S. D. Babacan, and R. Tomioka. Global analytic solution of fully-observed variational bayesian matrix factorization. *Journal of Machine Learning Research*, 14(Jan):1–37, 2013.

[76] B. Neyshabur, R. R. Salakhutdinov, and N. Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2015.

[77] Y. Nishimori and S. Akaho. Learning algorithms utilizing quasi-geodesic flows on the stiefel manifold. *Neurocomputing*, 67:106–135, 2005.

[78] E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(01):61–68, 1989.

[79] E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935, 1992.

[80] E. Oja. The nonlinear PCA learning rule in independent component analysis. *Neurocomputing*, 17(1):25–45, 1997.

[81] Y. Ollivier. Riemannian metrics for neural networks I: feedforward networks. *Information and Inference*, 4(2):108–153, 2015.

[82] H. Park, S.-i. Amari, and K. Fukumizu. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, 2000.

[83] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

[84] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. Exponential expressivity in deep neural networks through transient chaos. *arXiv preprint arXiv:1606.05340*, 2016.

[85] T. Raiko, H. Valpola, and Y. LeCun. Deep learning made easier by linear transformations in perceptrons. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2012.

[86] M. Ranzato, V. Mnih, J. M. Susskind, and G. E. Hinton. Modeling natural images using gated mrfs. *IEEE transactions on pattern analysis and machine intelligence*, 35(9):2206–2222, 2013.

[87] M. Rattray, D. Saad, and S.-i. Amari. Natural gradient descent for on-line learning. *Physical review letters*, 81(24):5461, 1998.

[88] N. L. Roux, P.-A. Manzagol, and Y. Bengio. Topmoumoute online natural gradient algorithm. In *Proceedings of Advances in Neural Information Processing Systems*, 2008.

[89] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[90] R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2009.

[91] T. Salimans and D. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Proceedings of Advances in Neural Information Processing Systems*, 2016.

[92] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of International Conference on Learning Representations*, 2014.

[93] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *Proceedings of International Conference on Machine Learning*, 2013.

[94] S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

[95] F. Seide, G. Li, X. Chen, and D. Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[96] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[97] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, pages 194–281. The MIT Press, 1986.

[98] J. Sohl-Dickstein, P. B. Battaglino, and M. R. DeWeese. New method for parameter estimation in probabilistic models: minimum probability flow. *Physical review letters*, 107(22):220601, 2011.

[99] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[100] I. Sutskever and T. Tieleman. On the convergence properties of contrastive divergence. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2010.

[101] T. Tieleman and G. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of Annual International Conference on Machine Learning*, 2009.

[102] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B*, 61(3):611–622, 1999.

[103] R. Velu and G. C. Reinsel. *Multivariate reduced-rank regression: theory and applications*, volume 136. Springer Science & Business Media, 2013.

[104] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of International Conference on Machine Learning*, 2008.

[105] A. Vinnikov and S. Shalev-Shwartz. K-means recovers ica filters when independent components are sparse. In *Proceedings of International Conference on Machine Learning*, 2014.

[106] N. Wang, J. Melchior, and L. Wiskott. Gaussian-binary restricted Boltzmann machines on modeling natural image statistics. arXiv:1401.5900, 2014.

[107] H. Wei, J. Zhang, F. Cousseau, T. Ozeki, and S.-i. Amari. Dynamics of learning near singularities in layered networks. *Neural computation*, 20(3):813–843, 2008.

[108] M. Welling, S. Osindero, and G. E. Hinton. Learning sparse topographic representations with products of student-t distributions. In *Proceedings of Advances in neural information processing systems*, 2002.

[109] M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Proceedings of Advances in neural information processing systems*, 2004.

[110] C. Williams and F. Agakov. An analysis of contrastive divergence learning in Gaussian Boltzmann machines. Technical report, EDI-INF-RR-0120, The University of Edinburgh, 2002.

[111] R. Williams. *Feature discovery through error-correction learning*. ICS Report 8501, University of California, 1985.

[112] T.-Y. Wu, B. Jiang, Y. Jin, and W. H. Wong. Convergence of contrastive divergence algorithm in exponential family. *arXiv preprint arXiv:1603.05729*, 2016.

[113] W.-Y. Yan, U. Helmke, and J. B. Moore. Global analysis of Oja's flow for neural networks. *IEEE Transactions on Neural Networks*, 5(5):674–683, 1994.

[114] Y. Yoshida, R. Karakida, M. Okada, and S.-i. Amari. Statistical mechanical analysis of online learning with weight normalization in single layer perceptron. *under review*, 2016.

[115] A. Yuille. The convergence of contrastive divergences. In *Proceedings of Advances in Neural Information Processing Systems*, 2005.

List of Publications

Refereed Papers

[1] **Ryo Karakida**, Yasuhiko Igarashi, Kenji Nagata and Masato Okada, Inter-Layer Correlation in a Feed-Forward Network with Intra-Layer Common Noise, Journal of the Physical Society of Japan, 82, 064007, 2013.

[2] **Ryo Karakida**, Masato Okada, Shun-ichi Amari, Dynamical analysis of contrastive divergence learning: Restricted Boltzmann machines with Gaussian visible units, Neural Networks, 79, 78-87, 2016.

[3] Yoshihiro Nagano, **Ryo Karakida**, Norifumi Watanabe, Atsushi Aoyama, Masato Okada, Input Response of Neural Network Model with Lognormally Distributed Synaptic Weights, Journal of the Physical Society of Japan, 85, 074001, 2016.

[4] Yuki Yoshida, **Ryo Karakida**, Masato Okada, Shun-ichi Amari, Statistical Mechanical Analysis of Online Learning with Weight Normalization in Single Layer Perceptron, Journal of the Physical Society of Japan, accepted, 2017.

Refereed Proceedings of International Conferences and Workshops

[1] **Ryo Karakida**, Masato Okada, Shun-ichi Amari, Analysing Feature Extraction by Contrastive Divergence Learning in RBMs Deep Learning And Representation Learning Workshop: NIPS2014, Montreal, Dec. 2014.

[2] **Ryo Karakida**, Masato Okada, Shun-ichi Amari, Maximum likelihood learning of RBMs with Gaussian visible units on the Stiefel manifold European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Bruges, April 2016.

[3] **Ryo Karakida**, Masato Okada, Shun-ichi Amari, Adaptive Natural Gradient Learning Based on Riemannian Metric of Score Matching 4th International Conference on Learning Representations - Workshop Track (ICLR 2016), San Juan, May 2016.

[4] **Ryo Karakida**, Masato Okada, Shun-ichi Amari, Adaptive natural gradient learning algorithms for unnormalized statistical models 25th International Conference on Artificial Neural Networks (ICANN 2016), Barcelona, Sep. 2016.

List of Awards

[1] **Ryo Karakida**, IEEE CISJ Young Researcher Award, Technical Committee on Neurocomputing, Tokyo, Japan, March 2015.

[2] **Ryo Karakida**, JNNS Young Presenter Award, 24st Annual Conference of the Japanese Neural Network Society (JNNS2014), Hokkaido, Japan, August 2014.

[3] **Ryo Karakida**, Best Student Presentation Award, Information-based Induction Sciences (IBIS2015), Tsukuba, Japan, Nov. 2015.

[4] **Ryo Karakida**, Best Student Paper Award, 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN2016), Bruges, Belgium, April 2016.