博士論文

Reliable Machine Learning from Limited Data and Supervision: A Risk Modification Approach

（限られたデータと教師からの高信頼機械学習：リスク修正アプローチ）

石田 隆

Contents

Contents i

Abstract v

List of Figures viii

List of Tables xii

1 Introduction 1
- 1.1 Machine Learning . . . 1
  - 1.1.1 What is Machine Learning? . . . 1
  - 1.1.2 Learning Paradigms . . . 3
  - 1.1.3 Reliable Machine Learning . . . 5
- 1.2 Learning from Weak Supervision . . . 6
  - 1.2.1 Binary Classification from Weak Supervision . . . 6
  - 1.2.2 Multi-Class Classification from Weak Supervision . . . 8
- 1.3 Learning with Limited Data . . . 9
- 1.4 Contribution: A Risk Modification Approach . . . 14
  - 1.4.1 Risk Rewriting . . . 15
  - 1.4.2 Risk Correction . . . 15
- 1.5 Proposed Methods . . . 16
  - 1.5.1 Learning from Positive-Confidence Data . . . 16
  - 1.5.2 Learning from Complementary Labels . . . 17
  - 1.5.3 Flooding: A Novel Regularizer to Avoid Overfitting . . . 18
- 1.6 Organization . . . 19
- 1.7 Publications Related to This Dissertation . . . 19

2 Background and Preliminaries 21
- 2.1 Data and Distribution . . . 21

2.2 Binary Classification . . . 22
2.2.1 Classification Risk . . . 22
2.2.2 Bayes Risk and Bayes Error . . . 23
2.2.3 Models . . . 24
2.2.4 Binary Loss Functions . . . 26
2.2.5 Regularization . . . 27
2.3 Multi-Class Classification . . . 27
2.3.1 Classification Risk . . . 27
2.3.2 Multi-Class Loss Functions . . . 28

3 Learning from Positive-Confidence Data 31
3.1 Introduction . . . 31
3.2 Problem Formulation . . . 34
3.3 Pconf Classification . . . 34
3.3.1 Empirical Risk Minimization (ERM) Framework . . . 34
3.3.2 Theoretical Analysis . . . 37
3.3.3 Implementation . . . 40
3.4 Experiments . . . 40
3.4.1 Synthetic Experiments with Linear Models . . . 40
3.4.2 Benchmark Experiments with Neural Network Models . . . 43
3.5 Conclusion . . . 47
3.5.1 Summary . . . 47
3.5.2 Recent Advances . . . 47

4 Learning from Complementary Labels 49
4.1 Introduction . . . 50
4.2 Complementary-Label Learning with Symmetric Losses . . . 51
4.2.1 Formulation . . . 51
4.2.2 Theoretical Analysis . . . 55
4.2.3 Incorporation of Ordinary Labels . . . 57
4.2.4 Experiments . . . 57
4.3 Complementary-label Learning with Arbitrary Losses . . . 62
4.3.1 Formulation . . . 63
4.3.2 Necessity of Risk Correction . . . 67
4.3.3 Non-Negative Risk Estimator . . . 68
4.3.4 Approximate Non-Negative Risk Estimator . . . 69
4.3.5 Experiments . . . 71
4.4 Conclusion . . . 72

5 Flooding: A Novel Regularizer to Avoid Overfitting **75**
5.1 Introduction . . . 75
5.2 Flooding: How to Avoid Zero Training Loss . . . 80
5.2.1 Preliminaries . . . 80
5.2.2 Algorithm . . . 81
5.2.3 Implementation . . . 82
5.3 Does Flooding Generalize Better? . . . 82
5.3.1 Synthetic Datasets . . . 83
5.3.2 Benchmark Experiments . . . 87
5.4 *Why* Does Flooding Generalize Better? . . . 89
5.4.1 Memorization . . . 89
5.4.2 Performance and Gradients . . . 91
5.4.3 Flatness . . . 91
5.4.4 Theoretical Insight . . . 94
5.5 Conclusion . . . 94

6 Conclusions and Future Work **95**
6.1 Conclusions . . . 95
6.2 Short Term Future Work . . . 96
6.2.1 Future Work for Positive-Confidence Learning . . . 97
6.2.2 Future Work for Complementary-Label Learning . . . 97
6.2.3 Future Work for Flooding . . . 98
6.3 Future Directions . . . 99
6.3.1 Looking for Yet Another Type of Supervision . . . 99
6.3.2 Shifting Our Focus to Lowering the Total Cost . . . 99

Appendices **101**
A Appendices for Chapter 3 . . . 101
A.1 CNN Architecture . . . 101
A.2 AutoEncoder Architecture . . . 101
B Appendices for Chapter 4 . . . 102
B.1 Proof of Lemma 4.3 . . . 102
B.2 Proof of Lemma 4.4 . . . 104
B.3 Proof of Lemma 4.5 . . . 105
B.4 Proof of Theorem 4.6 . . . 105
B.5 Benchmark Datasets . . . 105
C Appendices for Chapter 5 . . . 106
C.1 Benchmark Datasets . . . 106

C.2 Proof of Theorem 5.1 107

Acknowledgments **109**

Bibliography **110**

Abstract

We are surrounded by an immense number of automated and intelligent systems empowered by machine learning in our daily lives. We are starting to take machine learning applications such as object detection, handwriting recognition, speech recognition, and text generation for granted, which were not easily available just a decade ago.

Although machine learning seems to have been successfully applied in the real world, building a practical machine learning system is still extremely difficult. For example, collecting a correct class label can be difficult. In this case, we may instead need to learn from *weak supervision*, where the supervision of the data can be given in a weaker or alternative form than the usual supervision. Another example is learning from *limited data*. This may cause overfitting issues, especially when models with a large capacity is used. These examples are common real-world scenarios due to high data collecting and data labeling costs, and various practical constraints. In this dissertation, we focus on making machine learning more *reliable* under weak supervision or limited data. We summarize our contributions as follows.

In Chapter 3, we introduce our novel problem setting of learning a binary classifier from only positive data, without any negative data or unlabeled data. As an example, this task is important in purchase prediction. We can easily collect customer data from our own company (positive data), but not from rival companies (negative data). However, even in this challenging scenario, we still want to perform binary classification. Our finding is that if one can equip positive data with confidence (positive-confidence), one can successfully learn a binary classifier with the optimal convergence rate to solve this problem, which we call *positive-confidence classification*. The key technique is to reformulate the classification risk into a formulation that only requires the positive-confidence data, even though naively computing the classification risk requires both positive and negative data. This leads to a simple empirical risk minimization framework that is model-, loss-, and optimization-independent. We also show the consistency and an estimation error bound for positive-confidence classification, and experimental results showing the effectiveness of our method.

In Chapter 4, we discuss a new type of weak supervision called *complementary labels*, which are helpful for multi-class classification. A complementary label specifies a class that a pattern does *not* belong to. Collecting complementary labels would be less laborious than collecting ordinary labels, since annotators/labelers do not have to carefully choose the correct class from a long list of candidate classes. However, complementary labels are less informative than ordinary labels and thus a suitable approach is needed to better learn from them. For this challenging problem, we show that an unbiased estimator to the classification risk can be obtained only from complementarily labeled data, if a loss function satisfies a particular symmetric condition. We then derive estimation error bounds for the proposed method and prove that the learned classifier achieves the optimal convergence rate. We also show that learning from complementary labels can be easily combined with learning from ordinary labels, i.e., ordinary supervised learning, resulting in even better generalization performance. We further extend the unbiased risk estimator to arbitrary losses and models, and improve it by a non-negative correction and a gradient ascent trick. We show experimental results demonstrating the usefulness of our approach.

In Chapter 5, we introduce a novel regularizer that can be used to avoid overfitting. Overparameterized deep networks have the capacity to make the empirical risk go to zero, resulting in an overconfident model with degraded test performance. While previous regularization methods indirectly cope with this problem, we propose a direct solution called *flooding* that intentionally prevents further reduction of the empirical risk when it reaches a reasonably small value, which we call the flood level. Our approach makes the flooded empirical risk float around the flood level by performing mini-batched gradient descent as usual but gradient *ascent* if the empirical risk is below the flood level. This can be implemented with one line of code and is compatible with any stochastic optimizer and other regularizers. With flooding, we will have a "random walk" with the same non-zero empirical risk, and we expect the model to go into an area with a flat loss landscape that leads to better generalization. We experimentally show that flooding improves the generalization performance and as a byproduct, induces a double descent curve of the test loss.

In Chapter 6, we summarize our contributions in this dissertation and conclude. We discuss possible extensions for positive-confidence classification, complementary-label classification, and flooding. Finally, we discuss future directions for reliable machine learning.

In summary, this dissertation is devoted to making machine learning more reliable under limited data and supervision. A common approach in all of our methods is *risk modification*. We start by observing the classification risk as the final target which we want to minimize by training our classifier. Then, we take a modification

step by either rewriting or correcting the risk. Risk rewriting is used in positive-confidence learning and complementary-label learning. Since we do not have access to fully labeled data, we rewrite the classification risk in an alternative formulation that utilizes weakly supervised data. This leads to a theoretically grounded algorithm with an unbiased estimator of the classification risk and an estimation error bound of the learned classifier achieving the optimal convergence rate. Another technique is risk correction. In our regularization method, we employ this risk correction technique, by putting a lower-bound on the empirical risk. We also use a risk correction technique in complementary-label learning to avoid severe overfitting. A notable advantage of the risk modification approach is that it usually produces a framework that can be applied to a variety of domains, optimizers, and models. To conclude, this dissertation demonstrates that risk rewriting and risk correction can be a powerful approach in building practical and useful algorithms for learning from limited data and supervision.

List of Figures

1.1 Relationship between previous works (du Plessis et al., 2014, 2015; Sakai et al., 2017; Kiryo et al., 2017; Lu et al., 2020) and this dissertation. Blue is risk rewriting and red is risk correction. Arrows (→) indicate how ideas in one paper were utilized in, extended in, or inspired another paper (or chapter). The years in the right-hand side show the years of publication in Ishida et al. (2017, 2018, 2019, 2020). 15

3.1 Comparisons of positive-negative (PN) classification, soft-label classification, one-class classification, positive-unlabeled (PU) classification, and positive-confidence (Pconf) classification. 32

3.2 Illustrations based on a single trial of the four setups used in experiments with various Gaussian distributions. The red and green lines are decision boundaries obtained by Pconf and Weighted classification, respectively, where only positive data with confidence are used (no negative data). The black boundary is obtained by O-SVM, which uses only hard-labeled positive data. The blue boundary is obtained by the fully-supervised method using data from both classes. Histograms of confidence of positive data are shown below. 42

4.1 The mean accuracy and standard deviation for five trials. We performed the experiments for varying proportions of ordinary and complementary labels. 63

4.2 The left and middle graphs shows the total risk (4.61) (in black color) and the risk decomposed into each *ordinary* class term (4.62) (in other colors) for training data with linear and MLP models, respectively. The right graph shows the corresponding test accuracy for both models. 67

4.3 Experimental results for various datasets and models. Dark colors show the mean accuracy of 5 trials and light colors show standard deviation. 73

5.1 (a) shows 3 different concepts related to overfitting. [A] shows the generalization gap increases, while the training and test losses decrease. [B] also shows the increasing gap, but the test loss starts to rise. [C] shows the training loss becoming (near-)zero. We avoid [C] by *flooding* the bottom area, visualized in (b), which forces the training loss to stay around a constant. This leads to a decreasing test loss once again. We confirm these claims in experiments with CIFAR-10 shown in (c)–(d). 76

5.2 A visual explanation of how flooding repeats gradient decent and gradient ascent. b is the flooding level and J is the original learning objective. Gradient descent happens when $J > b$ but gradient ascent happens when $J < b$. 77

5.3 Comparison of flooding with different mini-batch sizes. We report the mean accuracy and standard deviation over four trials with different mini-batch sizes (1, 10, and 50) and datasets (Two Gaussians, Spiral, and Sinusoid). 86

5.4 Comparison of different implementations for flooding. We report the mean accuracy and standard deviation of five trials with different datasets (Two Gaussians, Spiral, and Sinusoid). We compare three methods: Baseline (without flooding), flooding with (5.14) (sq), and flooding with (5.6) (abs). 87

5.5 Learning curves of the test loss showing that adding flooding leads to lower test loss. The black dotted line shows the baseline without flooding. The colored lines show the learning curves with flooding for different flooding levels. We show the learning curves for $b \in \{0.01, 0.02, \ldots, 0.10\}$. KMNIST is Kuzushiji-MNIST, C10 is CIFAR-10, and C100 is CIFAR-100. MLP, BN, DA, and LRD stand for multi-layer perceptron, batch normalization, data augmentation and learning rate decay, respectively. 88

5.6 Vertical axis is the training accuracy and the horizontal axis is the flood level. Marks are placed on the flood level that was chosen based on validation accuracy. 90

5.7 Relationship between test loss and amplitude of gradient (with training or test loss). Each point ("∘" or "+") in the figures corresponds to a single model at a certain epoch. We remove the first 10 epochs and plot the rest. "∘" is used for the method with flooding and "+" is used for the method without flooding. The large black "∘" and "+" show the epochs with early stopping. The color becomes lighter (purple → yellow) as the training proceeds. 92
5.8 One-dimensional visualization of flatness. We visualize the training/test loss with respect to perturbation. We depict the results for 3 models: the model when the empirical risk with respect to training data is below the flooding level for the first time during training (dotted blue), the model at the end of training with flooding (solid blue), and the model at the end of training without flooding (solid red). 93

List of Tables

1.1 Conceptual comparisons of regularizers. ✓/× stands for yes/no. "Target training loss" indicates the ability to specify the amount of training loss we want to have at the end of training. "Domain independent" means it can be used for various domains, e.g., vision and language. "Task independent" means it can be used for various tasks, e.g., classification and regression. "Model independent" means it can be used for various models, e.g., linear, kernel, and neural network models. $\boldsymbol{x}$ is the input and y is the class label. 10

3.1 Comparison of the proposed Pconf classification with other methods, with varying degrees of overlap between the positive and negative distributions. We report the mean and standard deviation of the classification accuracy over 20 trials. We show the best and equivalent methods based on the 5% t-test in bold, excluding the fully-supervised method and O-SVM whose settings are different from the others. . . . 43

3.2 Mean and standard deviation of the classification accuracy with noisy positive confidence. The experimental setup is the same as Table 3.1, except that positive confidence scores for positive data are noisy. Std. is the standard deviation of Gaussian noise. 44

3.3 Mean and standard deviation of the classification accuracy over 20 trials for the Fashion-MNIST dataset with fully-connected three hidden-layer neural networks. Pconf classification was compared with the baseline Weighted classification method, Auto-Encoder method and fully-supervised method, with *T-shirt* as the positive class and different choices for the negative class. The best and equivalent methods are shown in bold based on the 5% t-test, excluding the Auto-Encoder method and fully-supervised method. 45

3.4 Mean and standard deviation of the classification accuracy over 20 trials for the CIFAR-10 dataset with convolutional neural networks. Pconf classification was compared with the baseline Weighted classification method, Auto-Encoder method and fully-supervised method, with *airplane* as the positive class and different choices for the negative class. The best and equivalent methods are shown in bold based on the 5% t-test, excluding the Auto-Encoder method and fully-supervised method. 46

4.1 Mean and standard deviation of classification accuracy over five trials in percentage, when the number of classes is changed. "PC" is (4.18), "OVA" is (4.17), "sigmoid" is (4.32), and "ramp" is (4.33). Best and equivalent methods (with 5% t-test) are bold. 58

4.2 Mean and standard deviation of classification accuracy over 20 trials in percentage. "PC/S" is the proposed method for pairwise comparison formulation with sigmoid loss, "PL" is partial label with squared hinge loss, "ML" is multi-label, and "OL" is classification from ordinary labels. Best and equivalent methods (with 5% t-test excluding "OL") are bold. # train denotes the total number of training and validation samples in each class. # test denotes the number of test samples in each class. 60

4.3 Means and standard deviations of classification accuracy over 10 trials in percentage. "OL" is the ordinary label method, "CL" is the complementary label method, and "OL & CL" is a combination method that uses both ordinarily and complementarily labeled data. Best and equivalent methods are highlighted in boldface. "Class" denotes the class labels used for the experiment and "Dim" denotes the dimensionality d of patterns to be classified. # train denotes the number of ordinarily/complementarily labeled data for training and validation in each class. # test denotes the number of test data in each class. 61

5.1 Experimental results for the synthetic data. The average and standard deviation of the accuracy of each method over 10 trials. Sub-table (A) shows the results without early stopping. Sub-table (B) shows the results with early stopping. The **boldface** denotes the best and comparable method in terms of the average accuracy according to the t-test at the significance level 1%. The average and standard deviation of the chosen flood level is also shown. 85

5.2 Benchmark datasets. Reporting accuracy for all combinations of early stopping and flooding. We compare "w/o flood" and "w/ flood" and the better one is shown in **boldface**. The best setup for each dataset is shown with **underline**. "–" means that flood level of zero was optimal. "LRD" stands for learning rate decay and "DA" stands for data augmentation. C10 is CIFAR-10 and C100 is CIFAR-100. 90

1 Summary statistics of benchmark datasets. Fashion is Fashion-MNIST and Kuzushiji is Kuzushiji-MNIST. 106

Chapter 1

Introduction

In this chapter, we explain the motivation of this dissertation by beginning with a background of machine learning. We then explain the difficulty of deploying machine learning systems in the real world, and motivate the necessity of reliable machine learning research that can cope with many of the issues. Within reliable machine learning, we explain learning from weak supervision and learning from limited data in detail. Finally, we explain the contributions and organization of this dissertation.

1.1 Machine Learning

In this section, we explain the background of machine learning, benefits of the machine learning approach, learning paradigms, and the necessity of reliable machine learning in the real world.

1.1.1 What is Machine Learning?

We are surrounded by an immense number of *automated* and *intelligent* machines in our daily lives, which did not exist just one or two decades ago. We can immediately list up a dozen of these by observing our smartphones in our pockets. The camera function can detect positions of human faces when taking pictures of friends and family. A large number of photos are tagged and categorized based on the person that is in a photograph. Instead of finger tapping on the tiny QWERTY keyboards to write text, we can swipe through the letters (Goodfellow, 2020), or even better, we can simply "talk" to the smartphone, and the speech will be translated into text in real-time. We also have the luxury of handwriting with our fingers or with stylus-pens, and the smartphone will automatically recognize the characters. Mailing apps can provide us with phrases to use for replying, video-sharing apps provide recommendation for

the next video to watch, media apps provide machine-generated news, and health apps provide an estimated number of steps taken per day. The battery lifespan is optimized to last longer, based on our daily charging habits. Countless automated and intelligent software is running on our smartphones and on our apps.

One of the fundamental technologies behind all of these examples is *machine learning* (Bishop, 2006; Mohri et al., 2012; Shalev-Shwartz and Ben-David, 2014; Sugiyama, 2015). Machine learning is the process of automatically extracting important patterns, trends, or knowledge from data or experience with a computer, without the need of a human trying to encode domain specific knowledge into rules through computer programs. For example, assume we have a dataset of many spam emails and non-spam emails. A naive approach might be to make a list of known spammers based on email addresses, domains, or IP addresses included in the emails labeled as spam (Harker, 1997). Then it would be simple to write a program with an *if-then* statement to detect future spam emails. This will not work so well since spammers continue to make new entities. A slightly better approach would be to write a computer program that calculates a spam score, by counting the number of keywords that experts on spam have agreed to have a high chance of being included in a spam email.

This might work to some extent, but many challenges will remain. What if spam can be better characterized by combinations of keywords instead of single keywords? What if the order of keywords that appear is important? This will require a much more complex computer program and require more time with spam experts to get a list of combinations or orders of keywords.

On the other hand, a supervised learning approach, one of the most successful areas within machine learning that aims to predict the output based on input data, can be a good alternative (Pantel and Lin, 1998). The computer would try to learn the mapping between the email contents (and other available features such as sender information) and email's output label of spam and non-spam, in an automatic way with a predictive power on previously unseen email examples. The same system can be used for different languages as long as we prepare a set of spam and non-spam emails for the target language. If spammers start sending out new types of spams, we can swap the email collection to more recent examples and use the same machine learning system again to learn the new patterns.

As we can see in the spam application, a machine learning approach can be helpful to automate tasks that require extensive human supervision and complex rule-based computer programs (Shalev-Shwartz and Ben-David, 2014). A key point of this machine learning approach is that it shifts the burden of writing complex programs to the process of collecting a labeled dataset, where the human domain

knowledge is indirectly used in preparing spam/non-spam labels. This approach can be tremendously powerful for tasks that are beyond human capabilities (Shalev-Shwartz and Ben-David, 2014). Difficult tasks can be too complex to break down into rule-based programs, but it may be surprisingly easy to prepare a dataset with human-supervised labels instead. Even in the relatively simple spam detection case, Pantel and Lin (1998) discussed how experienced computational linguists were not able to provide good keyword combinations. On the other hand, most people can identify a spam email when they see one, even if they are neither a linguist, a security engineer, a police officer, nor a researcher on criminology.

In the last decade, this usefulness of machine learning has been empowered by an increasing amount of data, cheaper storage, better computing power, efficient human labeling with crowdsourcing platforms (Howe, 2008), and better deep neural networks (Goodfellow et al., 2016). Machine learning is used to further tackle even more challenging tasks, and has been improving its performance, sometimes achieving a human or superhuman level, in image classification (Russakovsky et al., 2015), natural language processing (Devlin et al., 2019), speech recognition (Hinton et al., 2012a), and games (Silver et al., 2016). This success has led to the adoption of machine learning in various science areas such as bioinformatics (Olson et al., 2018), robotics (Kober et al., 2013), and neuroscience (Vu et al., 2018), and has been transforming industries such as agriculture (Liakos et al., 2018), transportation (Bojarski et al., 2016), healthcare (Litjens et al., 2017), and finance (Heaton et al., 2017), just to name a few.

1.1.2 Learning Paradigms

Traditionally, it is common to classify machine learning problems and algorithms into *supervised learning* and *unsupervised learning* (Murphy, 2012).

Supervised Learning

In supervised learning, the learner is first given training data. Each data has its corresponding target output. We have already seen spam detection as an example of supervised learning (Pantel and Lin, 1998), where the data is a set of emails and the output is spam/non-spam. Another famous example is the hand-written digit image recognition. We are given pixel data in the form of matrices, which are the patterns, and each image has an output value which represents the digits: $\{0, \ldots, 9\}$. Our goal is to learn a classifier from such a training dataset, and then correctly classify test digit images (or test emails) into each digit class (or spam/non-spam) with the classifier (Bishop, 2006). These examples with discrete output values are

called *classification*. Spam detection is called *binary classification* because it has two output labels, spam and non-spam, while digit image recognition is called *multi-class classification* because it has more than two output labels. If the output is a continuous value, for example the temperature or the housing price, the problem would be called *regression*. If we want to predict the order of data, such as the order of websites for search engine results, the problem is called *ranking* (Mohri et al., 2012).

Tasks related to predicting, detecting, filtering, classifying, scoring, and ordering can often be formulated as a supervised learning problem, and hence it has many real-world applications.

Unsupervised Learning

In unsupervised learning, the training dataset consists of only input data, and we are not given any output values. Our goal is to find interesting patterns or discover some kind of knowledge from only the input (Murphy, 2012).

An example is *clustering*, where our aim is to discover groups of similar input from the data. A second example is *density estimation*, where our aim is to estimate the probability density from the input data. A third example is *dimension reduction*, where the aim is to reduce the dimension of the input data. This process is often used as a pre-processing step for supervised learning. Dimension reduction is also used for *visualization*, where the dimension is reduced to 2 or 3, so that we can visually understand the input data (Maaten and Hinton, 2008).

Since output information is not required in unsupervised learning, we do not need to additionally label our collected data. However, if our goal is supervised learning, e.g., classification, unsupervised learning is not suitable.

In between Supervised and Unsupervised Learning

Although supervised and unsupervised learning are often introduced as the main machine learning paradigms, there are other paradigms that do not necessarily fit into these traditional classes.

Weakly supervised learning (Zhou, 2018) is located in the middle of supervised and unsupervised learning. Unlike ordinary supervised learning, the output value or label is not fully given and thus imperfect. Usually we still aim for the same goal as supervised learning, which is to learn an input-output relationship and predict the true output of test samples, regardless of the disadvantage of the weak supervision in the training dataset. Depending on how the supervision is imperfect, there are many directions of research. We will discuss weakly supervised learning in depth in Section 1.2.

Reinforcement learning (Sutton and Barto, 2018) can also be regarded as a research field in between supervised and unsupervised learning. In reinforcement learning, an agent must learn a mapping from a state to an action by trying to maximize a reward. The agent does not receive explicit supervision for which action to take, but receives a reward signal instead. Interactions between the agent and the environment and the trade-off between exploration-exploitation are important aspects that do not appear in supervised and unsupervised learning.

Self-supervised learning (Jing and Tian, 2020) tries to extract or invent supervision from the given unlabeled data. Take a language model as an example. The language modelling task is to predict the next word given the past sequence of words (Goldberg, 2017). Similarly, in next sentence prediction, which was used to train Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), two sentences are given, and the task is to predict whether the second sentence comes after the first sentence in the original text or not. In computer vision, one example is predicting the relationship between a rotated image and the original image (Gidaris et al., 2018). Another is predicting the relative position of multiple patches within a single image (Doersch et al., 2015). In these tasks, a human annotator does not have to explicitly give labels, but instead, the task is designed so that the labels can be attached automatically from the unlabeled data. Self-supervised learning tasks are useful in learning representations and can be helpful for downstream tasks including supervised learning tasks.

1.1.3 Reliable Machine Learning

Although machine learning seems to be successful in a wide variety of tasks, many of the successful examples deal with easy and ideal situations. In order for machine learning to be truly useful for various problems and tasks, we need to satisfy a number of factors, and many real-world challenges have been hindering machine learning to be adopted in various areas or have occasionally caused problems for existing applications. Machine learning systems might be used in an environment that may include outliers or that may differ largely from how the training data has been collected (Sugiyama and Kawanabe, 2012; Ganin and Lempitsky, 2015; du Plessis and Sugiyama, 2014b). There may be malicious entities that try to fool the machine learning system, which can become a critical issue for internet security or autonomous driving (Szegedy et al., 2014; Goodfellow et al., 2015). The supervision in the training dataset may include incorrect labels, which is often the case when they are collected through crowdsourcing platforms (Natarajan et al., 2013; Patrini et al., 2017; Han et al., 2018; Ghosh et al., 2017). Business constraints constantly change,

with new regulations and more demand for privacy, leading to more difficulty for machine learning systems (Dwork, 2008). Under these challenges, machine learning systems can either fail with severe performance deterioration or machine learning cannot be applied anymore due to lack of data resources and supervision. Thus it is important for machine learning systems to be *reliable* in these various undependable environments.

In the next sections, we will discuss two scenarios where *reliability* becomes important, which will be the focus of this dissertation: learning from weak supervision and learning from limited data.

1.2 Learning from Weak Supervision

When using machine learning in the real world, it is important that it is possible to collect training data in a cheap and easy way. Collecting data has become extremely cheap in some applications. For example in web services, users usually upload data such as photos or videos themselves. Even if users do not explicitly upload data, they can still generate data by clicking, reading, watching, communicating, and many other online activities. In most cases, the collected data is in the form of an *unlabeled* data. Even if a user uploads a picture of a dog, a human labeler would need to annotate this as a dog in order for this to become *labeled* data. Although we have crowdsourcing platforms that allow us to label data in an efficient and cheap way, the labeling cost can still become high when we want to label a larger amount of data, and we may not be able to rely on crowdsourcing when class labels require special knowledge of certain domains, e.g., labeling of medical images. This is one of the bottlenecks of supervised learning, and various machine learning frameworks have been developed in order to learn from an alternative type of supervision, called *limited* or *weak supervision*. In Section 1.2.1, we will review the literature of weak supervision in the binary classification setup. In Section 1.2.2, we will review the literature of weak supervision in the multi-class classification setup.

1.2.1 Binary Classification from Weak Supervision

One of the early examples of weak supervision is *semi-supervised learning* (Chapelle et al., 2006). Semi-supervised learning deals with the problem when we have some labeled examples along with a large set of unlabeled examples. Early works on semi-supervised learning such as Chapelle et al. (2003) and Belkin et al. (2006) assumed that nearby samples have the same labels or input data can be placed on a low-dimensional manifold. These ideas based on distributional assumptions have been

further refined to work under deep networks, and have shown to perform well recently (Lee, 2013; Laine and Aila, 2017; Sajjadi et al., 2016; Miyato et al., 2018). However, in problems when these assumptions do not hold, the performance can become worse. Sakai et al. (2017) proposed a semi-supervised learning method for binary classification based on empirical risk minimization that does not rely on distributional assumptions. The empirical risk will be defined formally in the next chapter. It can be informally regarded as a performance measure calculated with respect to the finite training data sampled from the underlying distribution and is non-negative. It is better for a smaller value.

Another line of work is *positive-unlabeled (PU) learning* (Elkan and Noto, 2008). In PU learning, the training dataset includes only positive and unlabeled data, and assumes no access to negative data. This can be useful when negative data is hard to collect or expensive to label. For example, when we want to predict the relevance of digital advertisements to an user in websites, it would be tempting to use the "clicked" cases as positive (relevant) and "non-clicked" cases as negative (not relevant). However, "non-clicked" cases includes cases where the user was potentially interested in the advertisement but did not have time to click on it (positives), in addition to cases where the user was truly not interested in the advertisement (negatives). In this situation, it would be more natural to assume the "non-clicked" data as an unlabeled dataset which includes both positive and negative samples. In du Plessis et al. (2014, 2015), they used a rewriting trick, to transform the original classification risk with positive and negative distributions to a risk with positive and unlabeled distributions. In Kiryo et al. (2017), they used a risk correction trick, in order to prevent the empirical risk from going to negative. This was effective in preventing overfitting.

In *similar-unlabeled learning* (Bao et al., 2018), the training dataset will have pairs including two samples, labeled as "similar", meaning that the label is the same among the two samples. In addition to that, unlabeled data will be available for training. This can be useful when we cannot obtain explicit class labels, but may be easier to obtain similarity pairs. An extension was proposed in Shimada et al. (2019) to incorporate dissimilar pairs. A risk rewriting trick was used to utilize similar/dissimilar pairs and unlabeled data.

In *unlabeled-unlabeled learning* (du Plessis et al., 2013; Lu et al., 2019, 2020), the goal is to learn a binary classifier (or identify the classification boundary) from two unlabeled sets of data. The two data is assumed to have different class proportions but has the same class conditional distributions. In Lu et al. (2019), a risk rewriting trick was used to learn from these unlabeled data.

In all of the weakly supervised learning problems introduced above, it is assumed

that unlabeled data is easy to prepare. In some real-world applications, however, even unlabeled data may be difficult to obtain. Can we design new binary classification algorithms based on the assumption that even unlabeled data is not available?

1.2.2 Multi-Class Classification from Weak Supervision

Most *semi-supervised learning* methods introduced in Section 1.2.1 were already proposed for the multi-class classification scenario.

Multi-positive and unlabeled learning (Xu et al., 2017) is an extension of the binary PU learning to the multi-class setup. Consider multi-class classification problem with K classes. We have labeled samples from $K-1$ classes, and unlabeled samples from all K classes.

In *partial-label learning* (Cour et al., 2011), we have a few candidate classes for each one of the samples. We do not know which one of the candidates is the true class of that sample, but we know the true class is included in the candidates. Our goal is to predict the true label of test samples. Consider a case where several people are shown in a photo, and this is tagged with their names (Cour et al., 2011). If we extract the faces in this photo, we can use the tags that were attached to this photo as candidate names and use partial-label learning to learn a classifier.

Zero-shot learning (Akata et al., 2013) considers a multi-class classification problem with $K+J$ classes. We have training data for only seen classes from $\{1,\dots,K\}$, while test data will be coming from unseen classes from $\{K+1,\dots,K+J\}$. In *generalized zero-shot learning* (Xian et al., 2017), the test data will include all classes from $\{1,\dots,K,K+1,\dots,K+J\}$. We are also given *attributes* or *descriptions* for each of the seen and unseen classes, which characterize the properties or give semantic information of each class. Zero-shot learning methods try to transfer knowledge from the seen class to the unseen class through these additional information.

These advances on multi-class classification from weak supervision has been explored extensively and were shown to work well in experiments. However, labeling is still a bottleneck. Even if unlabeled data can be additionally used, we still need to collect precise labels or candidate labels. In zero-shot learning and multi-PU learning, we still require labels for the multi-positives or seen classes. This can become especially costly for a multi-class problem with many classes, since annotators will need to identify the correct class out of a huge number of candidates. Can we design another weakly supervised problem setting that lowers this annotation cost?

1.3 Learning with Limited Data

The standard procedure of machine learning would be to first learn the model with a training dataset, and then use or deploy that learned model in the real world. Ideally, we would want to *generalize* to previously unseen examples, and correctly predict the output of those examples. This is called *generalization*, and is one of the central topics in machine learning. A challenge in designing algorithms that generalize well is to avoid *overfitting*. Overfitting can easily happen with finite data, and is generally explained as the phenomenon where the machine learning model has a high performance on training data (fitting too much on training data), but performs worse on test data which was not used for training. In order to prevent overfitting from happening, various methods known as *regularization*, also known as *regularizers*, have been proposed.

The name "regularization" dates back to at least Tikhonov regularization for the ill-posed linear least-squares problem (Tikhonov, 1943; Tikhonov and Arsenin, 1977). One example is to modify $\boldsymbol{X}^\top \boldsymbol{X}$ (where $\boldsymbol{X}$ is the design matrix) to become "regular" by adding a term to the objective function. There are many other regularization methods that are based on the principle of adding an additional term to the learning objective, and we will call these methods as "narrow regularizers".

More recently, "regularization" has further evolved to a more general meaning including various methods that alleviate overfitting but do not necessarily have a step to regularize a singular matrix or add a regularization term to the objective function. For example, Goodfellow et al. (2016) defines regularization as "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error." We refer to this type of regularization as "general regularizers". In the following, we will review both narrow and general regularizers. See Table 1.1 for a summary of the regularization methods in chronological order.

Narrow Regularizers

- ℓ_2 *regularization* is a generalization of Tikhonov regularization (Tikhonov, 1943) and can be applied to non-linear models. It is also called *ridge regression*. These methods implicitly assume that the optimal model parameters are close to zero.

- ℓ_1 *regularization* (Tibshirani, 1996) adds a term which is a sum of absolute values of the parameters. This is based on the sparsity assumption that the optimal model has only a few non-zero parameters. Many interesting extensions have been proposed, including the *elastic net* (Zou and Hastie, 2005), *group lasso* (Yuan and Lin, 2006) and *fused lasso* (Tibshirani et al., 2005).

Table 1.1: Conceptual comparisons of regularizers. ✓/× stands for yes/no. "Target training loss" indicates the ability to specify the amount of training loss we want to have at the end of training. "Domain independent" means it can be used for various domains, e.g., vision and language. "Task independent" means it can be used for various tasks, e.g., classification and regression. "Model independent" means it can be used for various models, e.g., linear, kernel, and neural network models. $\boldsymbol{x}$ is the input and y is the class label.

Regularization methods in chronological order	target training loss	domain independent	task independent	model independent	Main assumption
ℓ_2 regularization (Tikhonov, 1943)	×	✓	✓	✓	Optimal model params are close to 0
Weight decay (Hanson and Pratt, 1988)	×	✓	✓	✓	Optimal model params are close to 0
Early stopping (Morgan and Bourlard, 1990)	×	✓	✓	✓	Overfitting occurs in later epochs
Double backprop (Drucker and LeCun, 1992)	×	✓	✓	×	Small change of $\boldsymbol{x}$ do not change output
ℓ_1 regularization (Tibshirani, 1996)	×	✓	✓	✓	Optimal model has to be sparse
Image data augmentation (Simard et al., 2003)	×	×	✓	✓	Input is invariant to the translations
Multi-task regularization (Evgeniou and Pontil, 2004)	×	✓	×	✓	Different tasks have similar params
Manifold regularization (Belkin et al., 2006)	×	✓	×	✓	$\boldsymbol{x}$ is on manifold & output is smooth
Dropout (Hinton et al., 2012b)	×	✓	✓	×	Existence of complex co-adaptations
Knowledge distillation (Hinton et al., 2014)	×	✓	×	✓	Teacher gives "dark knowledge"
Adversarial training (Goodfellow et al., 2015)	×	×	×	✓	Existence of local sensitivity
Batch normalization (Ioffe and Szegedy, 2015)	×	✓	✓	×	Existence of internal covariate shift
Label smoothing (Szegedy et al., 2016)	×	✓	×	✓	True posterior is not a one-hot vector
Confidence penalty (Pereyra et al., 2017)	×	✓	×	✓	True posterior's entropy is not small
Focal loss (Lin et al., 2017)	×	✓	×	✓	Overconfidence with cross-entropy loss
Mixup (Zhang et al., 2018)	×	×	×	✓	Linear relationship between $\boldsymbol{x}$ and y
Flooding (Chapter 5) (Ishida et al., 2020)	✓	✓	✓	✓	Learning until zero loss is harmful

- *Double back-propagation* (Drucker and LeCun, 1992) adds a gradient-based term that penalizes large Jacobian. It explicitly pushes the input gradients to zero, making the minimum broader. Recently, similar ideas have been explored (Sokolić et al., 2017) and has been used for *contractive autoencoders* (Rifai et al., 2011).

- The *confidence penalty* (Pereyra et al., 2017) adds a confidence penalty term that penalizes low entropy of the output distribution of the classifier. The motivation is similar to label smoothing, which we will discuss shortly.

- *Manifold regularization* (Belkin et al., 2006) was proposed for the setting of semi-supervised learning. It is based on the assumptions that inputs are on a manifold and outputs are smooth. It adds an additional term that forces close samples to have similar outputs.

- *Multi-task regularization* (Evgeniou and Pontil, 2004) is another problem-specific method. It was proposed for the setting of multi-task learning and adds an additional term that forces different tasks to have similar parameters.

- *Knowledge distillation* (Hinton et al., 2014) is a teacher-student framework, where the knowledge of a large neural network (teacher) is distilled to a smaller model (student). This is helpful in situations where we want to deal with edge computing or when we have constraints for storage or inference time. Hinton et al. (2014) proposed a method that first trains the teacher on the training data, and then adds a regularization term that forces the student to have softmax probabilities that are similar to the teacher's softmax probabilities. This can be regarded as a special case of label smoothing, where the softmax probabilities are specified by the teacher.

- *Adversarial training* (Szegedy et al., 2014) was originally aiming to increase adversarial robustness in the test stage with adversarial examples, but Goodfellow et al. (2015) found that it was also helpful in generalizing better to the original test dataset. They proposed to add a penalty term that trains adversarial samples near original samples to have the same output as the original.

General Regularizers

- *Dropout* (Hinton et al., 2012b; Srivastava et al., 2014) is a regularization method for training neural networks. During training, each unit is "dropped" according to a drop probability and the remaining weights are trained by backpropagation. In Srivastava et al. (2014), they explained how this can be helpful to avoid

co-adaptation between units. Dropout has been widely used in many neural networks (Devlin et al., 2019), and many interesting extensions have been proposed since then, including DropConnect (Wan et al., 2013), Spatial Dropout (Tompson et al., 2015), Cutout (DeVries and Taylor, 2017), and DropBlock (Ghiasi et al., 2018).

- *Batch normalization* (Ioffe and Szegedy, 2015), also known as *batch norm*, is a helpful regularizer that normalizes the mean and standard deviation of units in each layer based on a mini-batch. Ioffe and Szegedy (2015) assumed the existence of internal covariate shift and explained that batch norm helps to fix the shift. They also showed that batch normalization can sometimes eliminate the need for dropout in experiments.

- *Early stopping* (Morgan and Bourlard, 1990) is a simple model selection technique that chooses the model after the best performing epoch based on performance on validation data. Note that validation data is only used for hyperparameter selection, and is usually kept separated from training and test datasets. A common phenomenon in experiments with large neural networks is that the performance based on the validation dataset becomes better in the early stage of training, but it starts to become worse in the latter stage of training. The idea is to keep the model with the best performance based on validation data so far. When training is finished, instead of using the model at the last point, we use the model that we kept with the best performance. In practice, we can stop training after the validation performance becomes worse for a few epochs in a row (Goodfellow et al., 2016).

- *Label smoothing* (Szegedy et al., 2016) is a popular technique that modifies the one-hot label of the training data with a smoothed version with $1-\epsilon$ instead of 1 and $\frac{\epsilon}{K-1}$ instead of 0, where K is the number of classes, and ϵ typically takes a small positive constant value such as 0.1 or 0.2. This modification will prevent the neural network from giving overconfident softmax probabilities. Instead of using the same constant for all training samples, Li et al. (2020) proposed a method where the smoothness is dependent on the data.

- The *focal loss* (Lin et al., 2017) is a modification of the softmax cross-entropy loss which was orginally proposed for object detection. Mukhoti et al. (2020) showed that the focal loss has implicit regularization effects by making the gradient norms for confident samples to be lower. Charoenphakdee et al. (2020) showed the focal loss is classification-calibrated (Bartlett et al., 2006) but is not strictly proper.

- *Data augmentation* (Shorten and Khoshgoftaar, 2019) is a popular technique for increasing training data artificially. Simard et al. (2003) proposed simple image augmentation techniques such as rotating and skewing. Since then, various ways of augmentation (Shorten and Khoshgoftaar, 2019) have been explored, especially in the image domain. Recently, sophisticated automation strategies (Cubuk et al., 2019; Lim et al., 2019; Hataya et al., 2020) have been proposed in order to go beyond the labor-intensive procedures of designing augmentation manually.

- *Mixup* (Zhang et al., 2018) is a method that trains with convex combinations of pairs of input data and their labels. This regularization will enforce linear behavior in-between training samples. It has been applied in Verma et al. (2019), Berthelot et al. (2019), and Kolesnikov et al. (2020) and is becoming an essential regularization tool for developing new systems.

So far, we have seen various regularization methods that add an additional penalty term to the learning objective, modify the model architecture, modify the output labels, or increase artificial data in order to alleviate overfitting. Although these regularization methods—both the *narrow* and *general* ones—already work well in practice and have become the de facto standard tools (Bishop, 2006; Goodfellow et al., 2016), most of them are domain-, task-, loss-, or model-dependent, or have additional assumptions on the optimal model parameters.

- *Domain-specific methods*: Data augmentation and methods such as mixup (Zhang et al., 2018) are mostly designed for the vision domain. They may require some efforts when applying to other domains (Guo et al., 2019; Thulasidasan et al., 2019).

- *Task-specific methods*: Label smoothing (Szegedy et al., 2016), confidence penalty (Pereyra et al., 2017), and knowledge distillation (Hinton et al., 2014) are used for problems with class labels and harder to use with regression or ranking. Manifold regularization (Belkin et al., 2006) and multi-task regularization (Evgeniou and Pontil, 2004) are proposed for certain tasks.

- *Loss-specific methods*: The focal loss (Lin et al., 2017) is a modification of a specific loss function. Label smoothing (Szegedy et al., 2016), the confidence penalty (Pereyra et al., 2017), and knowledge distillation (Hinton et al., 2014) assume that trained models can be used as class-posterior probability estimators, and use loss functions such as the softmax cross-entropy loss function.

- *Model-specific methods*: Batch normalization (Ioffe and Szegedy, 2015) and dropout (Srivastava et al., 2014) are designed for neural network architectures with hidden layers.

- *Additional assumptions*: ℓ_2 and ℓ_1 regularization assume that the optimal model parameters are close to zero or is sparse. Although dropout (Srivastava et al., 2014), early stopping (Morgan and Bourlard, 1990), label smoothing (Szegedy et al., 2016), and also weight decay (which shrinks the parameter weights during gradient updates (Hanson and Pratt, 1988)) do not add a regularization term to the learning objective explicitly, it is known that these four methods are equivalent or similar to ℓ_2 regularization under certain conditions (Bishop, 1995; Goodfellow et al., 2016; Wager et al., 2013; Lukasik et al., 2020; Loshchilov and Hutter, 2019). This implies that they may have similar assumptions on the optimal model parameters.

Modern machine learning tasks are applied to complex problems where the optimal model parameters are not necessarily close to zero or are not sparse, and it would be ideal if we can properly add regularization effects to the optimization stage without such assumptions. Is it possible to propose a regularization method that has an alternative assumption? Is it also possible to have a domain-, task-, loss-, and model-*independent* method?

1.4 Contribution: A Risk Modification Approach

In this dissertation, we will propose positive-confidence learning, complementary-label learning, and flooding. Positive-confidence learning tries to learn a binary classifier when we have access to only positive data with confidence. Complementary-label learning tries to learn a multi-class classifier when the class label in the training dataset specifies the class that the sample does *not* belong to. Flooding is a regularizer that aims to avoid overfitting.

One important concept that underlies all of our proposed methods is "risk modification". We start by observing the classification risk as the final target which we want to minimize by training our classifier. Then, we take a *modification* step by either *rewriting* or *correcting* the risk. In this section, we discuss benefits of taking this approach and our main contributions of this dissertation from the viewpoint of risk modification. We visualize the relationship between previous works in Fig. 1.1.

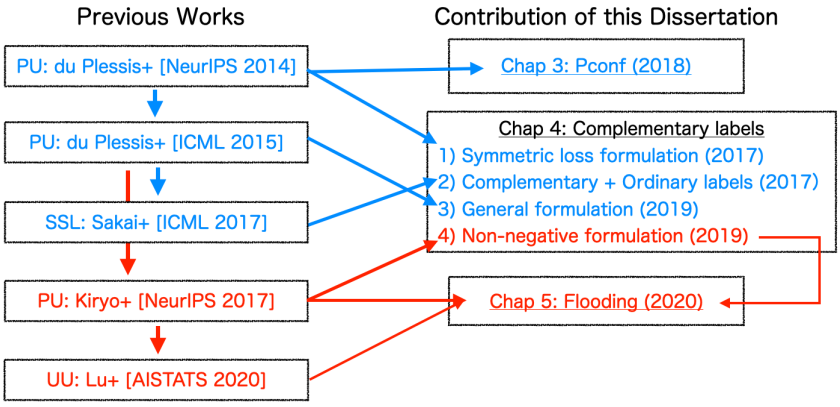


Figure 1.1: Relationship between previous works (du Plessis et al., 2014, 2015; Sakai et al., 2017; Kiryo et al., 2017; Lu et al., 2020) and this dissertation. Blue is risk rewriting and red is risk correction. Arrows (→) indicate how ideas in one paper were utilized in, extended in, or inspired another paper (or chapter). The years in the right-hand side show the years of publication in Ishida et al. (2017, 2018, 2019, 2020).

1.4.1 Risk Rewriting

In positive-confidence learning and complementary-label learning, we do not have access to the data sampled from a clean joint distribution of input data and output labels. We *rewrite* the classification risk in an alternative formulation that utilizes weakly supervised data and prove that it is equivalent to the original classification risk. This leads to a theoretically grounded algorithm with an unbiased estimator of the classification risk and an estimation error bound of the learned classifier achieving the optimal convergence rate. This usually produces a framework that can potentially be applied to a variety of domains and various choices of optimizers and models. A similar risk rewriting trick was used in previous works such as Natarajan et al. (2013), du Plessis et al. (2014), and Sakai et al. (2017), to propose better algorithms for *existing* problem settings. On the other hand, we propose *novel* problem settings that can be useful in handling different types of supervision.

1.4.2 Risk Correction

Although the risk rewriting approach does not prohibit us from using any type of models, using a large neural network model can lead to severe overfitting. This was especially a serious issue for positive-unlabeled (PU) learning (Elkan and Noto, 2008; du Plessis et al., 2014). The good news of the risk rewriting approach is that we

can identify the issue easily. In Kiryo et al. (2017), a negative empirical risk was observed after training for a while in PU learning, which cannot happen with the empirical risk in the original formulation. Kiryo et al. (2017) proposed a way to force the empirical risk to become non-negative, and showed that it was effective. We found that a similar issue occurs in complementary-label learning, and we propose correcting the risk to become non-negative.

Flooding can also be regarded as a risk correction method. In ordinary learning, the training loss will never go below zero, but we go a step further by forcing the training loss to not go below a *positive* flood level, under the assumption that learning until zero training loss is harmful for generalization.

Many regularization methods also try to avoid learning too much, but with an indirect approach, e.g., adding penalty terms, limiting the number of training epochs, decaying the learning rate, and modifying the output labels to be smooth. Since they are indirect, they cannot aim for a specific training loss value that they want to have at the end of training. With overparametrized neural networks, the training loss can often still go to zero even if these regularizers are used during training. On the other hand, flooding is based on a risk correction approach, which allows us to directly aim for a specific value of the training loss we want to keep till the end of training.

1.5 Proposed Methods

In this section, we will describe and summarize the contributions for the three proposed methods: positive-confidence learning (Section 1.5.1), complementary-label learning (Section 1.5.2), and flooding (Section 1.5.3).

1.5.1 Learning from Positive-Confidence Data

Can we learn a binary classifier from only positive data, without any negative data or unlabeled data? As an example, this task can be important in purchase prediction. We can easily collect customer data from our own company (positive data), but not from rival companies (negative data). We still want to perform binary classification in this challenging scenario.

We show that if one can equip positive data with confidence (positive-confidence), one can successfully learn a binary classifier with the optimal convergence rate to solve this problem, which we call *positive-confidence classification*. In the example of purchase prediction, often times, our customers are asked to answer questionnaires/-surveys on how strong their buying intention was over rival products. This may be transformed into a probability between 0 and 1 by pre-processing, and then it can

be used as positive-confidence, which is all we need for *positive-confidence learning*.

This is related to one-class classification which is aimed at "describing" the positive class by clustering-related methods, but one-class classification does not have the ability to tune hyper-parameters and their aim is not on "discriminating" positive and negative classes.

Positive-confidence classification is also related to PU classification (Elkan and Noto, 2008; du Plessis et al., 2014), which uses hard-labeled positive data and additional unlabeled data for constructing a binary classifier (see Section 1.2.1). A practical advantage of our positive-confidence classification method over typical PU classification methods is that, in addition to not requiring unlabeled data, our method does not involve estimation of the class-prior probability (du Plessis and Sugiyama, 2014a), which is required in standard PU classification methods but is known to be highly challenging in practice (Blanchard et al., 2010; Scott and Blanchard, 2009). This is enabled by the additional confidence information which indirectly includes the information of the class prior probability, bridging the class conditionals and class posteriors.

A key technique in our proposed method was to reformulate the classification risk into a formulation that only requires the positive-confidence data, even though naively computing the classification risk requires both positive and negative data. This leads to a simple empirical risk minimization framework that is model-, loss-, and optimization-independent. We also showed the consistency and an estimation error bound for *positive-confidence classification*.

In synthetic experiments, we investigate the behavior of the proposed method, against various data distributions and when positive-confidence can be noisy. We also show the effectiveness of our approach in benchmark datasets.

This problem and algorithm will be introduced in Chapter 3.

1.5.2 Learning from Complementary Labels

As we discussed in Section 1.2.2, collecting labeled data is costly and thus a critical bottleneck in real-world multi-class classification tasks. To mitigate this problem, we proposed a novel setting, namely *learning from complementary labels* for multi-class classification. A complementary label specifies a class that a pattern does not belong to. Collecting complementary labels would be less laborious than collecting ordinary labels, since annotators/labelers do not have to carefully choose the correct class from a long list of candidate classes. However, complementary labels are less informative than ordinary labels and thus a suitable approach is needed to better learn from them.

We show that an unbiased estimator to the multi-class classification risk can be obtained only from complementarily labeled data, if a loss function satisfies a particular symmetric condition. We derive estimation error bounds for the proposed method and prove that the optimal parametric convergence rate is achieved. We propose an extension for complementary-label learning with an unbiased estimator of the classification risk, for arbitrary losses and models. We further improved the risk estimator by a non-negative correction and a gradient ascent trick.

We further show that learning from complementary labels can be easily combined with learning from ordinary labels (i.e., ordinary supervised learning), providing a highly practical implementation of the proposed method.

Note that even though unbiased estimators were actively explored for the binary classification problem (Sakai et al., 2017; du Plessis et al., 2014, 2015; Natarajan et al., 2013), it was not so common in multi-class classification until a few years ago. In noisy-label learning, the work by Natarajan et al. (2013) was extended to the multi-class case in Patrini et al. (2017). In PU learning, du Plessis et al. (2014) was extended to the multi-class case in Xu et al. (2017).

This problem and algorithms of complementary labels will be introduced in Chapter 4.

1.5.3 Flooding: A Novel Regularizer to Avoid Overfitting

Overparameterized deep networks have the capacity to memorize training data with zero training error. Even after memorization, the training loss continues to approach zero, making the model overconfident and the test performance degraded. Since existing regularizers do not directly aim to avoid zero training loss, they often fail to maintain a moderate level of the the training loss, ending up with a too small or too large loss.

To cope with this problem, we propose a direct solution called *flooding* that intentionally prevents further reduction of the training loss when it reaches a reasonably small value, which we call the flood level. We simply modify the empirical risk as

$$\tilde{R}(\boldsymbol{g}) = |\hat{R}(\boldsymbol{g}) - b| + b, \tag{1.1}$$

where $\hat{R}(\boldsymbol{g})$ is the empirical risk, $\boldsymbol{g}$ is the classifier, and b is the flood level. Our approach makes the flooded empirical risk float around the flood level by performing mini-batched gradient descent as usual but gradient ascent if the empirical risk is below the flood level. This can be implemented with one line of code, and is compatible with any stochastic optimizer and other regularizers. With flooding, the model will continue to "random walk" with the same non-zero empirical risk, and we expect

it to drift into an area with a flat loss landscape that leads to better generalization. We experimentally showed that flooding improves performance and as a byproduct, induces a double descent curve of the test loss with respect to the training epochs.

This regularization method will be introduced in Chapter 5.

1.6 Organization

So far, we have discussed the background of machine learning, reliable machine learning, and contributions of this dissertation. The rest of this dissertation is organized as follows.

In Chapter 2, we will review the basic problem setting of supervised learning, especially binary and multi-class classification and introduce our notations. This will be the background for reading the rest of the dissertation.

Chapter 3 and Chapter 4 deal with learning from weak supervision. In Chapter 3, we introduce binary classification problem with positive-confidence data and propose an algorithm for the problem setting. This is based on our publication in Ishida et al. (2018).

In Chapter 4, we introduce multi-class classification with complementary labels and propose an algorithm for this problem setting. This is based on our publications in Ishida et al. (2017) and Ishida et al. (2019).

In Chapter 5, we shift our focus to the topic of learning from limited data. We introduce our novel regularization method that aims to avoid overfitting which we call flooding. This is based on our publication in Ishida et al. (2020).

Finally, we conclude and discuss future directions in Chapter 6.

1.7 Publications Related to This Dissertation

1. T. Ishida, G. Niu, W. Hu, M. Sugiyama. Learning from Complementary Labels. In *Advances in Neural Information Processing Systems 30 (NeurIPS2017)*.

2. T. Ishida, G. Niu, and M. Sugiyama. Binary Classification From Positive-Confidence Data. In *Advances in Neural Information Processing Systems 31 (NeurIPS2018)*.

3. T. Ishida, G. Niu, A. K. Menon, and M. Sugiyama. Complementary-Label Learning for Arbitrary Losses and Models. In *Proceedings of Thirty-sixth International Conference on Machine Learning (ICML2019)*.

4. T. Ishida, I. Yamane, T. Sakai, G. Niu, M. Sugiyama. Do We Need Zero Training Loss After Achieving Zero Training Error? In *Proceedings of Thirty-seventh International Conference on Machine Learning (ICML2020)*.

Chapter 2

Background and Preliminaries

In this chapter, we review the background and preliminaries for supervised learning, with an emphasis on classification. We have already introduced the main concepts and examples of supervised learning in the previous chapter, but now we will deal with technical backgrounds that will be used from the next chapter.

2.1 Data and Distribution

Supervised learning is the task of learning a mapping between input and output. Suppose $\mathcal{X}$ is the input space, and $\mathcal{Y}$ is the output space. $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ is a d-dimensional input, also known as a pattern, a sample, or an example. $y \in \mathcal{Y}$ is the supervision. In classification, the output is discrete with $\mathcal{Y} = \{1, 2, \ldots, K\}$, where K is the number of classes. The classification problem can be categorized into binary classification when there are two classes ($K = 2$) and multi-class classification when there are more classes ($K \geq 3$). Note that in binary classification, it is conventional to use $\mathcal{Y} = \{-1, +1\}$ or $\mathcal{Y} = \{0, 1\}$ instead of $\mathcal{Y} = \{1, 2\}$. This simplifies the formulation in some cases, and we will adopt $\mathcal{Y} = \{-1, +1\}$ for binary classification. In multi-class classification, we sometimes use a one-hot vector, $\boldsymbol{e}_y$, which takes 1 for the yth element and 0 for others.

We regard $\boldsymbol{x}$ and y as random variables and assume the underlying distribution has a joint probability density $p(\boldsymbol{x}, y)$. This is usually unknown, but our training samples are independently and identically distributed (i.i.d.) from $p(\boldsymbol{x}, y)$:

$$\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}, y), \tag{2.1}$$

where n is the number of training samples.

We will also have a (smaller) validation dataset and a test dataset that are sam-

pled i.i.d. from $p(\boldsymbol{x}, y)$, that will be used for model/hyper-parameter selection and performance evaluation, respectively.

2.2 Binary Classification

In this section, we consider binary classification with $\mathcal{Y} = \{-1, +1\}$.

2.2.1 Classification Risk

If we are given an input $\boldsymbol{x}$ and a score function $g\colon \mathbb{R}^d \to \mathbb{R}$, we can give a class prediction by $\hat{y} = \mathrm{sign}(g(\boldsymbol{x}))$. $\mathrm{sign}(\cdot)$ is 1 if the sign of the argument is non-negative and -1 otherwise. Since the score function g is used to classify $\boldsymbol{x}$ into the positive or negative class, it is also called a *classifier*. The *classification risk* is defined as

$$R(g) = \mathbb{E}_{p(\boldsymbol{x},y)}[\ell(g(\boldsymbol{x}), y)], \tag{2.2}$$

where $\mathbb{E}_{p(\boldsymbol{x},y)}$ is the expectation over $(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)$ and ℓ is a loss function $\ell\colon \mathbb{R} \times \{-1, +1\} \to \mathbb{R}_+$. Note that $\mathbb{R}_+ = \{x \in \mathbb{R} | x \geq 0\}$. There are some cases where a loss function that can become negative is considered, e.g., van Rooyen et al. (2015), but in this dissertation, we only consider non-negative loss functions.

With a *margin*, which is defined as $z = yg(\boldsymbol{x})$, we can also have a margin loss function with a single argument: $\ell\colon \mathbb{R} \to \mathbb{R}_+$. For simplicity, we use the same notation of ℓ. Using a margin is helpful because when y and $g(\boldsymbol{x})$ has the same (or a different) sign, this means that the classifier's output is correct (or incorrect). With the margin loss, the classification risk is defined as

$$R(g) = \mathbb{E}_{p(\boldsymbol{x},y)}[\ell(yg(\boldsymbol{x}))]. \tag{2.3}$$

It is sometimes helpful to rewrite the classification risk in the following ways:

$$\begin{aligned} R(g) &= \mathbb{E}_{p(\boldsymbol{x})}[p(y = +1|\boldsymbol{x}) \cdot \ell(g(\boldsymbol{x})) + p(y = -1|\boldsymbol{x}) \cdot \ell(-g(\boldsymbol{x}))] \\ &= \pi_+ \mathbb{E}_{p(\boldsymbol{x}|y=+1)}[\ell(g(\boldsymbol{x}))] + \pi_- \mathbb{E}_{p(\boldsymbol{x}|y=-1)}[\ell(-g(\boldsymbol{x}))], \end{aligned} \tag{2.4}$$

where $\pi_+ = p(y = +1)$ and $\pi_- = p(y = -1)$. One of the most important loss functions is the *zero-one loss*, which is defined as

$$\ell_{01}(z) := \begin{cases} 0 & \text{if} \quad z > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2.5}$$

This means that we have a loss of 0 when the margin is positive, but loss of 1 otherwise. This corresponds to having a penalty only when the classifier gives a wrong prediction. The *classification error* is the classification risk for the zero-one loss:

$$R_{01}(g) := \mathbb{E}_{p(\boldsymbol{x},y)}[\ell_{01}(g(\boldsymbol{x}), y)]. \tag{2.6}$$

The goal of binary classification is to learn a score function g that minimizes the classification error $R_{01}(g)$. In optimization, we consider the minimization of the risk with an almost surely differentiable *surrogate loss function* $\ell(z)$ $(\neq \ell_{01}(z))$ to make the problem more tractable. A surrogate loss function typically takes a small value for a large z. Furthermore, since $p(\boldsymbol{x}, y)$ is usually unknown and there is no way to exactly evaluate $R(g)$, we minimize its empirical version calculated from the training data instead:

$$\widehat{R}(g) := \frac{1}{n}\sum_{i=1}^{n} \ell(g(\boldsymbol{x}_i), y_i). \tag{2.7}$$

We call $\widehat{R}$ the *empirical risk*. Since we are minimizing the empirical risk to learn g with respect to our classifier set $\mathcal{G}$, this approach is called *empirical risk minimization (ERM)* (Vapnik, 1995). When we use the term "training/test loss", this will mean the empirical risk with respect to the surrogate loss function ℓ over the training/test data, respectively. We refer to the "training/test error" as the empirical risk with respect to ℓ_{01} over the training/test data, respectively (which is equal to one minus accuracy) (Zhang, 2004b).

2.2.2 Bayes Risk and Bayes Error

What is the best achievable classification risk? We can use the concept of the *Bayes risk*, which is defined as

$$R^* := \inf_h R(h), \tag{2.8}$$

where the infimum is taken over all measurable functions $h\colon \mathbb{R}^d \to \mathbb{R}$. The Bayes risk is often referred to as the *Bayes error* if the zero-one loss is used:

$$R^*_{01} = \inf_h R_{01}(h). \tag{2.9}$$

A classifier that can achieve the Bayes error R^*_{01} is called the *Bayes classifier* or *Bayes optimal classifier*. An alternative way to define the Bayes error is by looking at the conditional probabilities (Mohri et al., 2012). The *conditional Bayes error* (or

noise) for any $\boldsymbol{x} \in \mathcal{X}$ is

$$r(\boldsymbol{x}) = \min[p(y = +1|\boldsymbol{x}), p(y = -1|\boldsymbol{x})]. \tag{2.10}$$

This can be regarded as the average error of the Bayes classifier for $\boldsymbol{x}$. Then, the Bayes error can be regarded as the average of the conditional Bayes error:

$$R^*_{01} = \mathbb{E}_{p(\boldsymbol{x})}[r(\boldsymbol{x})]. \tag{2.11}$$

We have $R^*_{01} = 0$ only in the deterministic case where $r(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathcal{X}$. In classification, this is called a *class-separable problem*. In practice, it is more common to encounter a stochastic case where $R^*_{01} > 0$.

2.2.3 Models

The term *model* is used interchangeably with a classifier, but a model is more likely to be used when we are interested in the design or the architecture of the classifier.

Linear-in-Parameter Model

We first demonstrate a *linear-in-parameter model* for the binary classifier $g(\boldsymbol{x})$. We can consider the binary classifier of the form

$$g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + \beta, \tag{2.12}$$

where $^\top$ denotes the transpose, $\boldsymbol{w} \in \mathbb{R}^b$ is the weight parameter, $\beta \in \mathbb{R}$ is a bias parameter, and $\boldsymbol{\phi}(\boldsymbol{x})\colon \mathbb{R}^d \to \mathbb{R}^b$ is the basis functions. If we redefine $\boldsymbol{w}$ and $\boldsymbol{\phi}(\boldsymbol{x})$ as

$$\boldsymbol{w} := \begin{pmatrix} \boldsymbol{w} \\ \beta \end{pmatrix}, \quad \phi(\boldsymbol{x}) := \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}) \\ 1 \end{pmatrix}, \tag{2.13}$$

it can be rewritten in a simpler form as

$$g'(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}). \tag{2.14}$$

One example of the basis function that is often used in practice is the Gaussian kernel model. If $\{\boldsymbol{x}_i\}_{i=1}^n$ is the set of n samples, Gaussian kernel model is defined as

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_1\|^2}{2\sigma^2}\right) \\ \vdots \\ \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_n\|^2}{2\sigma^2}\right) \end{pmatrix}, \tag{2.15}$$

where $\sigma > 0$ is a bandwidth parameter and $\|\cdot\|$ is the Euclidean norm. Note that $b = n$ in this case.

Linear-in-Input Model

A *linear-in-input model* is a special case of the linear-in-parameter model, and can be derived by using $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$ as the basis function. Then we have

$$g(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + \beta, \tag{2.16}$$

with $\boldsymbol{w} \in \mathbb{R}^d$.

Neural Network Model

Recently, *neural network models* are becoming popular in various areas, e.g., computer vision and natural language processing. One of the simplest neural network is the *feedforward neural network* with a single hidden layer. If the hidden layer has M units, each unit $m \in \{1, \ldots, M\}$ in the hidden layer is a linear combination of the inputs:

$$a_m^{(1)} = \sum_{i=1}^{d} w_{mi}^{(1)} x_i + b_m^{(1)}, \tag{2.17}$$

where $\boldsymbol{w}_m^{(1)} \in \mathbb{R}^d$ and $b_m^{(1)} \in \mathbb{R}$ are the weight parameters and bias parameter for unit m between the input and hidden layer. $w_{mi}^{(1)}$ is the i-th element of $\boldsymbol{w}_m^{(1)}$. Each unit is followed by a non-linear activation function $h^{(1)}(\cdot)$:

$$c_m^{(1)} = h^{(1)}(a_m). \tag{2.18}$$

For example, we may use a *rectified linear unit (ReLU)* (Nair and Hinton, 2010):

$$h^{(1)}(a_m) = \max(0, a_m). \tag{2.19}$$

These outputs go through another transformation with a linear combination and we end up with a single output unit:

$$a^{(2)} = \sum_{m=1}^{M} w_m^{(2)} c_m^{(1)} + b^{(2)}, \tag{2.20}$$

where $\boldsymbol{w}^{(2)} \in \mathbb{R}^M$ and $b^{(2)} \in \mathbb{R}$ are the weight parameters and bias parameter for the output unit. $w_m^{(2)}$ is the m-th element of $\boldsymbol{w}^{(2)}$. Finally, we have another non-linear

activation function $h^{(2)}(\cdot)$:

$$c^{(2)} = h^{(2)}(a^{(2)}). \tag{2.21}$$

In binary classification, the logistic sigmoid function may be used:

$$h^{(2)}(a^{(2)}) = \frac{1}{1 + \exp(-a^{(2)})}. \tag{2.22}$$

This can easily be extended to multiple hidden layers and multiple output units for multi-class classification (Bishop, 2006; Goodfellow et al., 2016).

Designing a good neural network architecture has been one of the central topics in machine learning in the past few decades. Many specialized architectures have been developed depending on the type of data, e.g., convolutional neural networks for grid-structured data, recurrent neural networks for sequential data, and graph neural networks for graph data, and innovative building blocks have been developed, e.g., dropout (Srivastava et al., 2014), batch normalization (Ioffe and Szegedy, 2015), residual connections (He et al., 2016), attention (Bahdanau et al., 2015), and self-attention (Vaswani et al., 2017).

2.2.4 Binary Loss Functions

We have discussed that in practice, we can swap the zero-one loss function $\ell(z)$ with a surrogate loss function $\ell(z)$ which typically takes a small value for a large z. Some popular surrogate loss functions are the following.

- *Logistic loss*: $\ell(z) = \log(1 + \exp(-z))$.
- *Squared loss*: $\ell(z) = (1 - z)^2$.
- *Hinge loss*: $\ell(z) = \max(0, 1 - z)$.
- *Squared-hinge loss*: $\ell(z) = \big(\max(0, 1 - z)\big)^2$.
- *Exponential loss*: $\ell(z) = \exp(-z)$.
- *Ramp loss*: $\frac{1}{2}\min\big(2, \max(0, 1 - z)\big)$.
- *Sigmoid loss*: $1/\big(1 + \exp(z)\big)$.

Different loss functions are used for different purposes. For example, the logistic loss is used when we want to train a probabilistic classifier (Sugiyama, 2015) and the hinge loss is used for support vector machines (Vapnik, 1995).

The logistic, squared, hinge, squared-hinge, and exponential losses are convex while the ramp and sigmoid losses are non-convex. Using a linear-in-parameter model

with a convex loss function can lead to convex optimization problems, and can be solved efficiently. When we use a squared loss with a linear-in-parameter model, we can derive an analytical solution. We may use gradient descent methods for differentiable loss functions (the logistic, squared, exponential, squared-hinge, and sigmoid loss), or sub-gradient descent methods for sub-differentiable loss functions (the hinge and ramp loss).

2.2.5 Regularization

If we simply minimize the empirical risk, we may suffer from overfitting. The learned classifier may achieve a small error on training data, but may have a higher error on unseen test data. In practice, we often add regularizers to avoid overfitting. A simple way would be to add an additional regularization term $\Omega(g)$ and minimize the total of the two terms instead: $\min_{g \in \mathcal{G}} \widehat{R}(g) + \lambda \Omega(g)$. λ is a positive hyper-parameter that controls the strength of the regularization term. For example, in ℓ_2 regularization, we add the term $\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$ when model parameters are $\boldsymbol{w}$. See Section 1.3 for various regularizers.

2.3 Multi-Class Classification

In this section, we consider multi-class classification with $\mathcal{Y} = \{1, \ldots, K\}$ with $K \geq 3$. We review the multi-class classification risk and multi-class loss functions. Note that the discussions on the Bayes risk/error, classification model, and regularization can be extended to the multi-class case in a straightforward way.

2.3.1 Classification Risk

The classification risk is defined as

$$R(\boldsymbol{g}) = \mathbb{E}_{p(\boldsymbol{x},y)}[\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), y)], \tag{2.23}$$

where a multi-class loss function is $\mathcal{L}\colon \mathbb{R}^K \times [K] \to \mathbb{R}_+$ and $\boldsymbol{g}\colon \mathbb{R}^d \to \mathbb{R}^K$ is a vector-valued score function. Typically, our classifier $f\colon \mathbb{R}^d \to [K]$ is constructed as

$$f(\boldsymbol{x}) := \operatorname*{arg\,max}_{k \in [K]} g_k(\boldsymbol{x}), \tag{2.24}$$

where $g_z(\cdot)$ is the k-th element of $\boldsymbol{g}(\cdot)$. In case of a tie, arg max returns the smallest argument. This is aligned with popular implementations, e.g., NumPy (Harris et al., 2020).

The zero-one loss is

$$\mathcal{L}_{01}(\boldsymbol{v}, k') := \begin{cases} 0 & \text{if} \quad \arg\max_{k\in\{1,\ldots,K\}} v_k = k', \\ 1 & \text{otherwise,} \end{cases} \tag{2.25}$$

where $\boldsymbol{v} := (v_1, \ldots, v_K)^\top \in \mathbb{R}^K$. The classification error is the classification risk for $\mathcal{L} = \mathcal{L}_{01}$:

$$R_{01}(\boldsymbol{g}) := \mathbb{E}_{p(\boldsymbol{x},y)}[\mathcal{L}_{01}(\boldsymbol{g}(\boldsymbol{x}), y)]. \tag{2.26}$$

Similarly to binary classification, the goal of multi-class classification is to learn a score function $\boldsymbol{g}$ that minimizes the classification error $R_{01}(\boldsymbol{g})$. Again, we consider minimizing the empirical risk, an empirical version calculated from the training data:

$$\widehat{R}(\boldsymbol{g}) := \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}_i), y_i), \tag{2.27}$$

where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ are i.i.d. samples from $p(\boldsymbol{x}, y)$.

2.3.2 Multi-Class Loss Functions

In order to minimize the classification risk, we need to define the multi-class loss function $\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), y)$. Here we review four multi-class loss functions: *one-versus-all*, *pairwise-comparison*, *multi-class SVM*, and *softmax cross-entropy*. For the first three, we will utilize binary surrogate loss functions $\ell(z) : \mathbb{R} \to \mathbb{R}^+$ (that incurs a large loss for small z) to design multi-class loss functions.

One-Versus-All Loss

We first introduce the *one-versus-all* (OVA) (Zhang, 2004a) loss function which is defined as

$$\mathcal{L}_{\text{OVA}'}(\boldsymbol{g}(\boldsymbol{x}), y) = \ell\big(g_y(\boldsymbol{x})\big) + \sum_{y'\neq y} \ell\big(-g_{y'}(\boldsymbol{x})\big). \tag{2.28}$$

For a pattern $\boldsymbol{x}$ that belongs to class y, the one-versus-all loss encourages binary classifier $g_y(\boldsymbol{x})$ to take a large value, and all other binary classifiers $g_{y'}(\boldsymbol{x})$ to take small values.

It is also useful to normalize the "rest" loss by $K-1$,

$$\mathcal{L}_{\text{OVA}}(\boldsymbol{g}(\boldsymbol{x}), y) = \ell\big(g_y(\boldsymbol{x})\big) + \frac{1}{K-1}\sum_{y'\neq y} \ell\big(-g_{y'}(\boldsymbol{x})\big), \tag{2.29}$$

which will be used in Chapter 4.

Pairwise-Comparison Loss

The second example is the *pairwise-comparison* (PC) (Vapnik, 1998) loss function which is defined as

$$\mathcal{L}_{\mathrm{PC}}\big(\boldsymbol{g}(\boldsymbol{x}), y\big) = \sum_{y' \neq y} \ell\big(g_y(\boldsymbol{x}) - g_{y'}(\boldsymbol{x})\big). \tag{2.30}$$

For a pattern $\boldsymbol{x}$ that belongs to class y, the pairwise-comparison loss encourages binary classifier $g_y(\boldsymbol{x})$ to take a larger value than other binary classifiers $g_{y'}(\boldsymbol{x})$. In other words, pairwise-comparison loss only pays attention to the sign of the difference of classifiers, while the one-versus-all loss pays attention to the sign of each classifiers.

Multi-Class SVM Loss

The third example is the *multi-class SVM* (Crammer and Singer, 2001) loss function which is defined as

$$\mathcal{L}_{\mathrm{MS}}(\boldsymbol{g}(\boldsymbol{x}), y) = \ell\big(g_y(\boldsymbol{x}) - \max_{y' \neq y} g_{y'}(\boldsymbol{x})\big). \tag{2.31}$$

For a pattern $\boldsymbol{x}$ that belongs to class y, the multi-class SVM loss encourages binary classifier $g_y(\boldsymbol{x})$ to take a larger value than the largest value of all other binary classifiers $g_{y'}(\boldsymbol{x})$. It can be regarded as a modification of the PC loss.

Softmax Cross-Entropy Loss

The final example is the *softmax cross-entropy* (CE) (Murphy, 2012) loss function which is defined as

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{g}(\boldsymbol{x}), y) := -\log \frac{\exp(g_y(\boldsymbol{x}))}{\sum_{k \in [K]} \exp(g_k(\boldsymbol{x}))}. \tag{2.32}$$

It can be regarded as a multi-class extension of the *logistic loss*. A modification of the softmax cross-entropy loss called the *focal loss* was proposed in Lin et al. (2017), to put more focus on the harder examples:

$$\mathcal{L}_{\mathrm{F}}(\boldsymbol{g}(\boldsymbol{x}), y) := -\left(1 - \frac{\exp(g_y(\boldsymbol{x}))}{\sum_{k \in [K]} \exp(g_k(\boldsymbol{x}))}\right)^{\gamma} \log \frac{\exp(g_y(\boldsymbol{x}))}{\sum_{k \in [K]} \exp(g_k(\boldsymbol{x}))}, \tag{2.33}$$

where $\gamma > 0$ is a hyper-parameter.

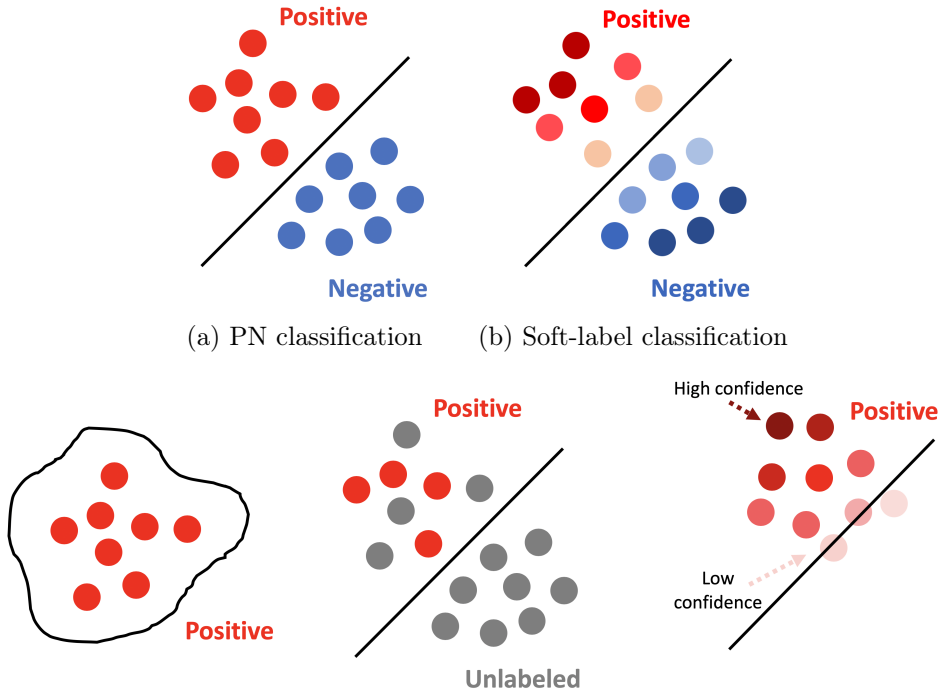
Chapter 3

Learning from Positive-Confidence Data

Can we learn a binary classifier from only positive data, without any negative data or unlabeled data? In this chapter, we show that if one can equip positive data with confidence (positive-confidence), one can successfully learn a binary classifier, which we name *positive-confidence (Pconf) classification*. Our work is related to one-class classification which is aimed at "describing" the positive class by clustering-related methods, but one-class classification does not have the ability to tune hyper-parameters and their aim is not on "discriminating" positive and negative classes. For the Pconf classification problem, we provide a simple empirical risk minimization framework that is model-, loss-, and optimization-independent. We theoretically establish the consistency and an estimation error bound, and demonstrate the usefulness of the proposed method with synthetic and benchmark experiments.

3.1 Introduction

In this chapter, we deal with the binary classification problem from positive data equipped with confidence. At a glance, being restricted from collecting negative or even unlabeled data but having access to confidence information may seem peculiar. However, such a *positive-confidence (Pconf) classification* scenario is conceivable in various real-world problems. For example, in purchase prediction, we can easily collect customer data from our own company (positive data), but not from rival companies (negative data). Often times, our customers are asked to answer questionnaires/surveys on how strong their buying intention was over rival products. This may be transformed into a probability between 0 and 1 by pre-processing, and



(a) PN classification (b) Soft-label classification

(c) One-class classification (d) PU classification (e) Pconf classification

Figure 3.1: Comparisons of positive-negative (PN) classification, soft-label classification, one-class classification, positive-unlabeled (PU) classification, and positive-confidence (Pconf) classification.

then it can be used as positive-confidence, which is all we need for Pconf classification as we will see in this chapter.

Another example is a common task for app developers, where they need to predict whether app users will continue using the app or unsubscribe in the future. The critical issue is that depending on the privacy/opt-out policy or data regulation, they need to fully discard the unsubscribed user's data. Hence, developers will not have access to users who quit using their services, but they can associate a positive-confidence score with each remaining user by, e.g., how actively they use the app.

In these potential applications, as long as positive-confidence data can be collected, Pconf classification allows us to obtain a classifier that discriminates between positive and negative data.

Pconf classification is related to one-class classification, which is aimed at "describing" the positive class typically from hard-labeled positive data without confidence. To the best of our knowledge, previous one-class methods are motivated geometrically (Tax and Duin, 2004; Schölkopf and Smola, 2001), by information the-

ory (Sugiyama et al., 2014), or by density estimation (Breunig et al., 2000). However, due to the descriptive nature of all previous methods, there is no systematic way to tune hyper-parameters to "classify" positive and negative data. In the conceptual example in Figure 3.1, one-class methods do not have any knowledge of the negative distribution, such that the negative distribution is in the lower right of the positive distribution. Therefore, even if we have an infinite number of training data, one-class methods will still require regularization to have a tight boundary in all directions, wherever the positive posterior becomes low. Note that even if we knew that the negative distribution lies in the lower right of the positive distribution, it is still impossible to find the decision boundary, because we still need to know the degree of overlap between the two distributions and the class prior. One-class methods are designed for and work well for anomaly detection, but have critical limitations if the problem of interest is "classification".

On the other hand, Pconf classification is aimed at constructing a discriminative classifier and thus hyper-parameters can be objectively chosen to discriminate between positive and negative data. We will later see that Pconf classification relies on the key ingredient recurring throughout this dissertation, which is the empirical risk minimization (ERM; Vapnik (1995)) and this makes it suitable for binary classification.

Pconf classification is also related to positive-unlabeled (PU) classification (Elkan and Noto, 2008; du Plessis et al., 2014, 2015), which uses hard-labeled positive data and additional unlabeled data for constructing a binary classifier. A practical advantage of our Pconf classification method over typical PU classification methods is that our method does not involve estimation of the *class-prior probability*, which is required in standard PU classification methods (du Plessis et al., 2014, 2015; Kiryo et al., 2017), but is known to be challenging in practice (Scott and Blanchard, 2009; Blanchard et al., 2010; Menon et al., 2015; du Plessis et al., 2017). This is enabled by the additional confidence information which indirectly includes the information of the class prior probability, bridging class conditionals and class posteriors.

Finally, Pconf classification has connections with soft-label learning (Nguyen et al., 2011). In soft-label learning, in addition to the class labels, we have an auxiliary information that corresponds to how strong the labeler feels about his or her labeling decision. Soft-label learning has access to data from all classes, while Pconf classification only has access to positive data.

In this chapter, we propose a simple ERM framework for Pconf classification and theoretically establish the consistency and an estimation error bound. We then provide an example of implementation to Pconf classification by using linear-in-parameter models (such as Gaussian kernel models), which can be implemented eas-

ily and can be computationally efficient. Finally, we experimentally demonstrate the practical usefulness of the proposed method for training linear-in-parameter models and deep neural networks.

3.2 Problem Formulation

In this section, we formulate our Pconf classification problem. Suppose that a pair of d-dimensional pattern $\boldsymbol{x} \in \mathbb{R}^d$ and its class label $y \in \{+1, -1\}$ follow an unknown probability distribution with density $p(\boldsymbol{x}, y)$. Our goal is to train a binary classifier $g(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ so that the classification risk $R(g)$ is minimized:

$$R(g) = \mathbb{E}_{p(\boldsymbol{x},y)}[\ell(yg(\boldsymbol{x}))], \tag{3.1}$$

where $\mathbb{E}_{p(\boldsymbol{x},y)}$ denotes the expectation over $p(\boldsymbol{x}, y)$, and $\ell(z)$ is a loss function. When margin z is small, $\ell(z)$ typically takes a large value. Since $p(\boldsymbol{x}, y)$ is unknown, the ordinary ERM approach (Vapnik, 1995) replaces the expectation with the average over training data drawn independently from $p(\boldsymbol{x}, y)$.

However, in the Pconf classification scenario, we are only given positive data equipped with *confidence* $\mathcal{X} := \{(\boldsymbol{x}_i, r_i)\}_{i=1}^n$, where $\boldsymbol{x}_i$ is a positive pattern drawn independently from $p(\boldsymbol{x}|y = +1)$ and r_i is the positive confidence given by $r_i = p(y = +1|\boldsymbol{x}_i)$. Note that this equality does not have to strictly hold as later shown in Section 3.4. Since we have no access to negative data in the Pconf classification scenario, we cannot directly employ the standard ERM approach. In the next section, we show how the classification risk can be estimated only from Pconf data.

3.3 Pconf Classification

In this section, we propose an ERM framework for Pconf classification and derive an estimation error bound for the proposed method. Finally we give examples of practical implementations.

3.3.1 Empirical Risk Minimization (ERM) Framework

Let $\pi_+ = p(y = +1)$ and $r(\boldsymbol{x}) = p(y = +1|\boldsymbol{x})$, and let $\mathbb{E}_+$ denote the expectation over $p(\boldsymbol{x}|y = +1)$. Then the following theorem holds, which forms the basis of our approach:

Theorem 3.1. *The classification risk (3.1) can be expressed as*

$$R(g) = \pi_+ \mathbb{E}_+ \left[\ell\big(g(\boldsymbol{x})\big) + \frac{1 - r(\boldsymbol{x})}{r(\boldsymbol{x})} \ell\big(-g(\boldsymbol{x})\big) \right], \tag{3.2}$$

if we have $p(y = +1|\boldsymbol{x}) \neq 0$ for all $\boldsymbol{x}$ sampled from $p(\boldsymbol{x})$.

Proof. The classification risk (3.1) can be expressed and decomposed as

$$\begin{aligned} R(g) &= \sum_{y=\pm 1} \int \ell\big(yg(\boldsymbol{x})\big) p(\boldsymbol{x}|y) p(y) \mathrm{d}\boldsymbol{x} \\ &= \int \ell\big(g(\boldsymbol{x})\big) p(\boldsymbol{x}|y=+1) p(y=+1) \mathrm{d}\boldsymbol{x} + \int \ell\big(-g(\boldsymbol{x})\big) p(\boldsymbol{x}|y=-1) p(y=-1) \mathrm{d}\boldsymbol{x} \\ &= \pi_+ \mathbb{E}_+[\ell(g(\boldsymbol{x}))] + \pi_- \mathbb{E}_-[\ell(-g(\boldsymbol{x}))], \end{aligned} \tag{3.3}$$

where $\pi_- = p(y = -1)$ and $\mathbb{E}_-$ denotes the expectation over $p(\boldsymbol{x}|y = -1)$. Since

$$\begin{aligned} \pi_+ p(\boldsymbol{x}|y=+1) + \pi_- p(\boldsymbol{x}|y=-1) &= p(\boldsymbol{x}, y=+1) + p(\boldsymbol{x}, y=-1) \\ &= p(\boldsymbol{x}) \\ &= \frac{p(\boldsymbol{x}, y=+1)}{p(y=+1|\boldsymbol{x})} \\ &= \frac{\pi_+ p(\boldsymbol{x}|y=+1)}{r(\boldsymbol{x})}, \end{aligned} \tag{3.4}$$

where the third equality requires the assumption of $p(y = +1|\boldsymbol{x}) \neq 0$ stated in Theorem 3.1. we have

$$\pi_- p(\boldsymbol{x}|y=-1) = \pi_+ p(\boldsymbol{x}|y=+1) \left(\frac{1 - r(\boldsymbol{x})}{r(\boldsymbol{x})} \right). \tag{3.5}$$

Then the second term in (3.3) can be expressed as

$$\begin{aligned} \pi_- \mathbb{E}_-[\ell(-g(\boldsymbol{x}))] &= \int \pi_- p(\boldsymbol{x}|y=-1) \ell(-g(\boldsymbol{x})) \mathrm{d}\boldsymbol{x} \\ &= \int \pi_+ p(\boldsymbol{x}|y=+1) \left(\frac{1 - r(\boldsymbol{x})}{r(\boldsymbol{x})} \right) \ell(-g(\boldsymbol{x})) \mathrm{d}\boldsymbol{x} \\ &= \pi_+ \mathbb{E}_+ \left[\frac{1 - r(\boldsymbol{x})}{r(\boldsymbol{x})} \ell(-g(\boldsymbol{x})) \right], \end{aligned} \tag{3.6}$$

which concludes the proof. □

Equation (3.2) does not include the expectation over negative data, but only includes the expectation over positive data and their confidence values. Furthermore,

when (3.2) is minimized with respect to g, unknown π_+ is a proportional constant and thus can be safely ignored. Conceptually, the assumption of $p(y=+1|\boldsymbol{x}) \neq 0$ is implying that the support of the negative distribution is the same or is included in the support of the positive distribution.

Based on this, we propose the following ERM framework for Pconf classification:

$$\min_g \sum_{i=1}^{n} \Big[\ell\big(g(\boldsymbol{x}_i)\big) + \frac{1-r_i}{r_i} \ell\big(-g(\boldsymbol{x}_i)\big) \Big]. \tag{3.7}$$

It might be tempting to consider a similar empirical formulation as follows:

$$\min_g \sum_{i=1}^{n} \Big[r_i \ell\big(g(\boldsymbol{x}_i)\big) + (1-r_i) \ell\big(-g(\boldsymbol{x}_i)\big) \Big]. \tag{3.8}$$

Equation (3.8) means that we weigh the positive loss with positive-confidence r_i and the negative loss with negative-confidence $1-r_i$. This is quite natural and may look straightforward at a glance. However, if we simply consider the population version of the objective function of (3.8), we have

$$\begin{aligned}
&\mathbb{E}_+ \Big[r(\boldsymbol{x}) \ell\big(g(\boldsymbol{x})\big) + \big(1-r(\boldsymbol{x})\big) \ell\big(-g(\boldsymbol{x})\big) \Big] \\
&= \mathbb{E}_+ \Big[p(y=+1|\boldsymbol{x}) \ell\big(g(\boldsymbol{x})\big) + p(y=-1|\boldsymbol{x}) \ell\big(-g(\boldsymbol{x})\big) \Big] \\
&= \mathbb{E}_+ \Big[\sum_{y \in \{\pm 1\}} p(y|\boldsymbol{x}) \ell\big(y g(\boldsymbol{x})\big) \Big] \\
&= \mathbb{E}_+ \Big[\mathbb{E}_{p(y|\boldsymbol{x})} \big[\ell\big(y g(\boldsymbol{x})\big) \big] \Big],
\end{aligned} \tag{3.9}$$

which is *not* equivalent to the classification risk $R(g)$ defined by (3.1). If the outer expectation was over $p(\boldsymbol{x})$ instead of $p(\boldsymbol{x}|y=+1)$ in (3.9), then it would be equal to (3.1). This implies that if we had a different problem setting of having positive confidence equipped for $\boldsymbol{x}$ sampled from $p(\boldsymbol{x})$, this would be trivially solved by a naive weighting idea.

From this viewpoint, (3.7) can be regarded as an application of *importance sampling* (Fishman, 1996; Sugiyama and Kawanabe, 2012) to (3.8) to cope with the distribution difference between $p(\boldsymbol{x})$ and $p(\boldsymbol{x}|y=+1)$, but with the advantage of *not* requiring training data from the test distribution $p(\boldsymbol{x})$.

In summary, our ERM formulation of (3.7) is different from naive confidence-weighted classification of (3.8). We further show in Section 3.3.2 that the minimizer of (3.7) converges to the true risk minimizer, while the minimizer of (3.8) converges to a different quantity and hence learning based on (3.8) is inconsistent.

3.3.2 Theoretical Analysis

Here we derive an estimation error bound for the proposed method. To begin with, let $\mathcal{G}$ be our function class for ERM. Assume there exists $C_g > 0$ such that $\sup_{g\in\mathcal{G}} \|g\|_\infty \leq C_g$ as well as $C_\ell > 0$ such that $\sup_{|z|\leq C_g} \ell(z) \leq C_\ell$. The existence of C_ℓ may be guaranteed for all reasonable ℓ given a reasonable $\mathcal{G}$ in the sense that C_g exists. As usual (Mohri et al., 2012), assume $\ell(z)$ is Lipschitz continuous for all $|z| \leq C_g$ with a (not necessarily optimal) Lipschitz constant L_ℓ.

Denote by $\widehat{R}(g)$ the objective function of (3.7) times π_+, which is unbiased in estimating $R(g)$ in (3.1) according to Theorem 3.1. Subsequently, let $g^* = \arg\min_{g\in\mathcal{G}} R(g)$ be the true risk minimizer, and $\hat{g} = \arg\min_{g\in\mathcal{G}} \widehat{R}(g)$ be the empirical risk minimizer, respectively. The estimation error is defined as $R(\hat{g}) - R(g^*)$, and we are going to bound it from above.

In Theorem 3.1, $(1 - r(\boldsymbol{x}))/r(\boldsymbol{x})$ is playing a role inside the expectation, for the fact that

$$r(\boldsymbol{x}) = p(y = +1 \mid \boldsymbol{x}) > 0 \text{ for } \boldsymbol{x} \sim p(\boldsymbol{x} \mid y = +1). \tag{3.10}$$

In order to derive any error bound based on statistical learning theory, we should ensure that $r(\boldsymbol{x})$ could never be too close to zero. To this end, assume there is $C_r > 0$ such that $r(\boldsymbol{x}) \geq C_r$ almost surely. We may trim $r(\boldsymbol{x})$ and then analyze the bounded but biased version of $\widehat{R}(g)$ alternatively. For simplicity, only the unbiased version is involved after assuming C_r exists.

Lemma 3.2. *For any $\delta > 0$, the following uniform deviation bound holds with probability at least $1 - \delta$ (over repeated sampling of data for evaluating $\widehat{R}(g)$):*

$$\sup_{g\in\mathcal{G}} |\widehat{R}(g) - R(g)| \leq 2\pi_+ \left(L_\ell + \frac{L_\ell}{C_r}\right) \mathfrak{R}_n(\mathcal{G}) + \pi_+ \left(C_\ell + \frac{C_\ell}{C_r}\right) \sqrt{\frac{\ln(2/\delta)}{2n}}, \tag{3.11}$$

where $\mathfrak{R}_n(\mathcal{G})$ is the Rademacher complexity of $\mathcal{G}$ for $\mathcal{X}$ of size n drawn from $p(\boldsymbol{x} \mid y = +1)$.[1]

Proof. By assumption, it holds almost surely that

$$\frac{1 - r(\boldsymbol{x})}{r(\boldsymbol{x})} \leq \frac{1}{C_r}; \tag{3.12}$$

due to the existence of C_ℓ, the change of $\widehat{R}(g)$ will be no more than $(C_\ell + C_\ell/C_r)/n$ if some $\boldsymbol{x}_i$ is replaced with $\boldsymbol{x}_i'$.

[1] $\mathfrak{R}_n(\mathcal{G}) = \mathbb{E}_{\mathcal{X}}\mathbb{E}_{\sigma_1,\ldots,\sigma_n}[\sup_{g\in\mathcal{G}} \frac{1}{n} \sum_{\boldsymbol{x}_i\in\mathcal{X}} \sigma_i g(\boldsymbol{x}_i)]$ where $\sigma_1, \ldots, \sigma_n$ are n Rademacher variables following Mohri et al. (2012).

Consider a single direction of the uniform deviation: $\sup_{g\in\mathcal{G}} \widehat{R}(g) - R(g)$. Note that the change of $\sup_{g\in\mathcal{G}} \widehat{R}(g) - R(g)$ shares the same upper bound with the change of $\widehat{R}(g)$, and *McDiarmid's inequality* (McDiarmid, 1989) implies that

$$\begin{aligned}\Pr\left\{\sup_{g\in\mathcal{G}} \widehat{R}(g) - R(g) - \mathbb{E}_{\mathcal{X}}\left[\sup_{g\in\mathcal{G}} \widehat{R}(g) - R(g)\right] \geq \epsilon\right\} \\ \leq \exp\left(-\frac{2\epsilon^2 n}{(C_\ell + C_\ell/C_r)^2}\right),\end{aligned} \tag{3.13}$$

or equivalently, with probability at least $1-\delta/2$,

$$\sup_{g\in\mathcal{G}} \widehat{R}(g) - R(g) \leq \mathbb{E}_{\mathcal{X}}\left[\sup_{g\in\mathcal{G}} \widehat{R}(g) - R(g)\right] + \left(C_\ell + \frac{C_\ell}{C_r}\right)\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{3.14}$$

Since $\widehat{R}(g)$ is unbiased, it is routine to show that (Mohri et al., 2012)

$$\begin{aligned}\mathbb{E}_{\mathcal{X}}\left[\sup_{g\in\mathcal{G}} \widehat{R}(g) - R(g)\right] &\leq 2\mathfrak{R}_n\left(\left(1 + \frac{1-r}{r}\right)\circ\ell\circ\mathcal{G}\right) \\ &\leq 2\left(1 + \frac{1}{C_r}\right)\mathfrak{R}_n(\ell\circ\mathcal{G}) \\ &\leq 2\left(L_\ell + \frac{L_\ell}{C_r}\right)\mathfrak{R}_n(\mathcal{G}),\end{aligned} \tag{3.15}$$

which proves this direction.

The other direction $\sup_{g\in\mathcal{G}} R(g) - \widehat{R}(g)$ can be proven similarly. □

Lemma 3.2 guarantees that with high probability $\widehat{R}(g)$ concentrates around $R(g)$ for all $g \in \mathcal{G}$, and the degree of such concentration is controlled by $\mathfrak{R}_n(\mathcal{G})$. Based on this lemma, we are able to establish an estimation error bound, as follows:

Theorem 3.3. *For any $\delta > 0$, with probability at least $1-\delta$ (over repeated sampling of data for training $\hat{g}$), we have*

$$R(\hat{g}) - R(g^*) \leq 4\pi_+\left(L_\ell + \frac{L_\ell}{C_r}\right)\mathfrak{R}_n(\mathcal{G}) + 2\pi_+\left(C_\ell + \frac{C_\ell}{C_r}\right)\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{3.16}$$

Proof. Based on Lemma 3.2, the estimation error bound (3.16) is proven through

$$\begin{aligned}R(\hat{g}) - R(g^*) &= \left(\widehat{R}(\hat{g}) - \widehat{R}(g^*)\right) + \left(R(\hat{g}) - \widehat{R}(\hat{g})\right) + \left(\widehat{R}(g^*) - R(g^*)\right) \\ &\leq 0 + 2\sup_{g\in\mathcal{G}}\left|\widehat{R}(g) - R(g)\right| \\ &\leq 4\pi_+\left(L_\ell + \frac{L_\ell}{C_r}\right)\mathfrak{R}_n(\mathcal{G}) + 2\pi_+\left(C_\ell + \frac{C_\ell}{C_r}\right)\sqrt{\frac{\ln(2/\delta)}{2n}},\end{aligned} \tag{3.17}$$

where $\widehat{R}(\hat{g}) \leq \widehat{R}(g^*)$ by the definition of $\widehat{R}$. □

Theorem 3.3 guarantees learning with (3.7) is consistent (Ledoux and Talagrand, 1991): $n \to \infty$ always means $R(\hat{g}) \to R(g^*)$. Consider linear-in-parameter models defined by

$$\mathcal{G} = \{g(\boldsymbol{x}) = \langle w, \phi(\boldsymbol{x})\rangle_{\mathcal{H}} \mid \|w\|_{\mathcal{H}} \leq C_w, \|\phi(\boldsymbol{x})\|_{\mathcal{H}} \leq C_\phi\}, \tag{3.18}$$

where $\mathcal{H}$ is a Hilbert space, $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ is the inner product in $\mathcal{H}$, $w \in \mathcal{H}$ is the normal, $\phi : \mathbb{R}^d \to \mathcal{H}$ is a feature map, and $C_w > 0$ and $C_\phi > 0$ are constants (Schölkopf and Smola, 2001). It is known that $\mathfrak{R}_n(\mathcal{G}) \leq C_w C_\phi/\sqrt{n}$ (Mohri et al., 2012) and thus $R(\hat{g}) \to R(g^*)$ in $\mathcal{O}_p(1/\sqrt{n})$, where $\mathcal{O}_p$ denotes the order in probability. This order is already the optimal parametric rate and cannot be improved without additional strong assumptions on $p(\boldsymbol{x}, y)$, ℓ and $\mathcal{G}$ jointly (Mendelson, 2008). Additionally, if ℓ is strictly convex we have $\hat{g} \to g^*$, and if the aforementioned $\mathcal{G}$ is used $\hat{g} \to g^*$ in $\mathcal{O}_p(1/\sqrt{n})$ (Boyd and Vandenberghe, 2004).

At first glance, learning with (3.8) is numerically more stable; however, it is generally inconsistent, especially when g is linear in parameters and ℓ is strictly convex. Denote by $\widehat{R}'(g)$ the objective function of (3.8) times π_+, which is unbiased to

$$R'(g) = \pi_+\mathbb{E}_+\mathbb{E}_{p(y|\boldsymbol{x})}[\ell(yg(\boldsymbol{x}))] \tag{3.19}$$

rather than $R(g)$. By the same technique for proving (3.11) and (3.16), it is not difficult to show that with probability at least $1-\delta$,

$$\sup_{g\in\mathcal{G}} |\widehat{R}'(g) - R'(g)| \leq 4\pi_+ L_\ell \mathfrak{R}_n(\mathcal{G}) + 2\pi_+ C_\ell \sqrt{\frac{\ln(2/\delta)}{2n}}, \tag{3.20}$$

and hence

$$R'(\hat{g}') - R'(g'^*) \leq 8\pi_+ L_\ell \mathfrak{R}_n(\mathcal{G}) + 4\pi_+ C_\ell \sqrt{\frac{\ln(2/\delta)}{2n}}, \tag{3.21}$$

where

$$g'^* = \arg\min_{g\in\mathcal{G}} R'(g) \quad \text{and} \quad \hat{g}' = \arg\min_{g\in\mathcal{G}} \widehat{R}'(g).$$

As a result, when the strict convexity of $R'(g)$ and $\widehat{R}'(g)$ is also met, we have $\hat{g}' \to g'^*$. This demonstrates the inconsistency of learning with (3.8), since $R'(g) \neq R(g)$ which leads to $g'^* \neq g^*$ given any reasonable $\mathcal{G}$.

3.3.3 Implementation

Finally we give examples of implementations. As a classifier g, let us consider a linear-in-parameter model $g(\boldsymbol{x}) = \boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{x})$, where $^\top$ denotes the transpose, $\boldsymbol{\phi}(\boldsymbol{x})$ is a vector of basis functions, and $\boldsymbol{\alpha}$ is a parameter vector. Then from (3.7), the ℓ_2-regularized ERM is formulated as

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \left[\ell\big(\boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\big) + \frac{1-r_i}{r_i} \ell\big(-\boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)\big) \right] + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \boldsymbol{R} \boldsymbol{\alpha}, \tag{3.22}$$

where λ is a non-negative constant and $\boldsymbol{R}$ is a positive semi-definite matrix. In practice, we can use any loss functions such as squared loss $\ell_{\mathrm{S}}(z) = (z-1)^2$, hinge loss $\ell_{\mathrm{H}}(z) = \max(0, 1-z)$, and ramp loss $\ell_{\mathrm{R}}(z) = \min(1, \max(0, 1-z))$. In the experiments in Section 3.4, we use the logistic loss $\ell_L(z) = \log(1+e^{-z})$, which yields,

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \left[\log\big(1+e^{-\boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)}\big) + \frac{1-r_i}{r_i} \log\big(1+e^{\boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{x}_i)}\big) \right] + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \boldsymbol{R} \boldsymbol{\alpha}. \tag{3.23}$$

The above objective function is continuous and differentiable, and therefore optimization can be efficiently performed, for example, by quasi-Newton (Nocedal and Wright, 2006) or stochastic gradient methods (Shalev-Shwartz and Ben-David, 2014).

3.4 Experiments

In this section, we numerically illustrate the behavior of the proposed method on synthetic datasets for linear models. We further demonstrate the usefulness of the proposed method on benchmark datasets for deep neural networks that are highly nonlinear models. The implementation is based on PyTorch (Paszke et al., 2019) and Sklearn (Pedregosa et al., 2011)[2].

3.4.1 Synthetic Experiments with Linear Models

Setup

We used two-dimensional Gaussian distributions with means $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ and covariance matrices $\boldsymbol{\Sigma}_+$ and $\boldsymbol{\Sigma}_-$, for $p(\boldsymbol{x}|y=+1)$ and $p(\boldsymbol{x}|y=-1)$, respectively. For these parameters, we tried various combinations visually shown in Figure 3.2. The specific parameters used for each setup are:

[2]Our code is available online: http://github.com/takashiishida/pconf.

- Setup A: $\boldsymbol{\mu}_+ = [0,0]^\top$, $\boldsymbol{\mu}_- = [-2,5]^\top$, $\boldsymbol{\Sigma}_+ = \begin{bmatrix} 7 & -6 \\ -6 & 7 \end{bmatrix}$, $\boldsymbol{\Sigma}_- = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$.
- Setup B: $\boldsymbol{\mu}_+ = [0,0]^\top, \boldsymbol{\mu}_- = [0,4]^\top, \boldsymbol{\Sigma}_+ = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$, $\boldsymbol{\Sigma}_- = \begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix}$.
- Setup C: $\boldsymbol{\mu}_+ = [0,0]^\top, \boldsymbol{\mu}_- = [0,8]^\top, \boldsymbol{\Sigma}_+ = \begin{bmatrix} 7 & -6 \\ -6 & 7 \end{bmatrix}$, $\boldsymbol{\Sigma}_- = \begin{bmatrix} 7 & 6 \\ 6 & 7 \end{bmatrix}$.
- Setup D: $\boldsymbol{\mu}_+ = [0,0]^\top, \boldsymbol{\mu}_- = [0,4]^\top, \boldsymbol{\Sigma}_+ = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$, $\boldsymbol{\Sigma}_- = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

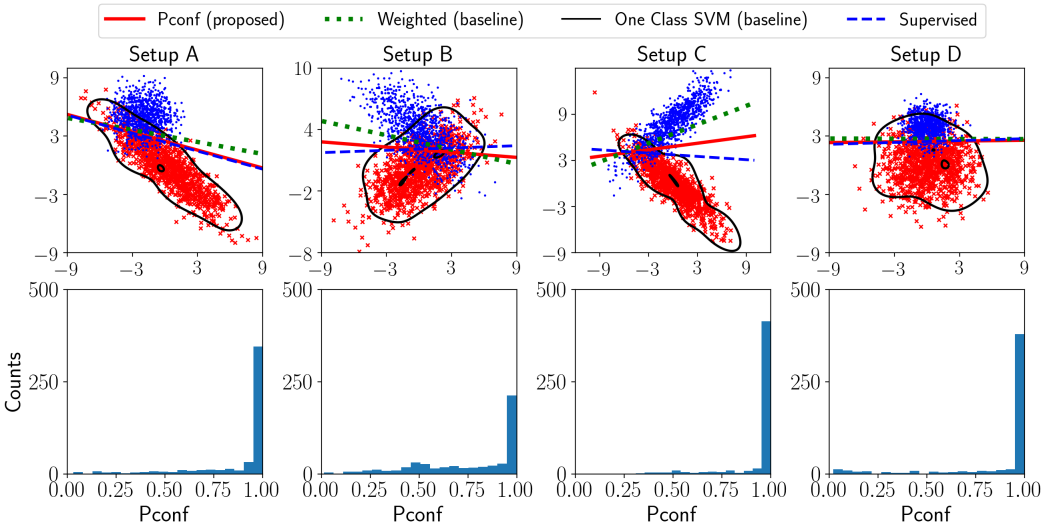
In the case of using two Gaussian distributions, $p(y = +1|\boldsymbol{x}) > 0$ is satisfied for any $\boldsymbol{x}$ sampled from $p(\boldsymbol{x})$, which is a necessary condition for applying Theorem 3.1. 500 positive data and 500 negative data were generated independently from each distribution for training.[3] Similarly, 1,000 positive and 1,000 negative data were generated for testing. We compared our proposed method (3.7) with the weighted classification method (3.8), a regression based method (predict the confidence value itself and post-process output to a binary signal by comparing it to 0.5), one-class support vector machine (O-SVM, Schölkopf et al. (2001)) with the Gaussian kernel, and a fully-supervised method based on the empirical version of (3.1). Note that the proposed method, weighted method, and regression based method only use Pconf data, O-SVM only uses (hard-labeled) positive data, and the fully-supervised method uses both positive and negative data.

In the proposed, weighted, fully-supervised methods, linear-in-input model $g(\boldsymbol{x}) = \boldsymbol{\alpha}^\top \boldsymbol{x} + b$ and the logistic loss were commonly used and *vanilla gradient descent* with $5,000$ epochs (full-batch size) and learning rate 0.001 was used for optimization. For the regression-based method, we used the squared loss and analytical solution (Hastie et al., 2009). For the purpose of clear comparison of the risk, we did not use regularization in this toy experiment. An exception was O-SVM, where the user is required to subjectively pre-specify regularization parameter ν and Gaussian bandwidth γ. We set them at $\nu = 0.05$ and $\gamma = 0.1$.[4]

[3] Negative training data are used only in the fully-supervised method that is tested for performance comparison.

[4] If we naively use default parameters in Sklearn (Pedregosa et al., 2011) instead, which is the usual case in the real world without negative data for validation, the classification accuracy of O-SVM is worse for all setups except D in Table 3.1, which demonstrates the difficulty of using O-SVM.

Figure 3.2: Illustrations based on a single trial of the four setups used in experiments with various Gaussian distributions. The red and green lines are decision boundaries obtained by Pconf and Weighted classification, respectively, where only positive data with confidence are used (no negative data). The black boundary is obtained by O-SVM, which uses only hard-labeled positive data. The blue boundary is obtained by the fully-supervised method using data from both classes. Histograms of confidence of positive data are shown below.



Analysis with true positive-confidence

Our first experiments were conducted when true positive-confidence was known. The positive-confidence $r(\boldsymbol{x})$ was analytically computed from the two Gaussian densities and given to each positive data. The results in Table 3.1 show that the proposed Pconf method is significantly better than the baselines in all cases. In most cases, the proposed Pconf method has similar accuracy compared with the fully supervised case, excluding Setup C where there is a few percent loss. Note that the naive weighted method is consistent if the model is correctly specified, but becomes inconsistent if misspecified (Sugiyama and Kawanabe, 2012).

Analysis with noisy positive-confidence

In the above toy experiments, we assumed that true positive confidence $r(\boldsymbol{x}) = p(y = +1|\boldsymbol{x})$ is exactly accessible, but this can be unrealistic in practice. To investigate the influence of noise in positive-confidence, we conducted experiments with noisy positive-confidence.

As noisy positive confidence, we added zero-mean Gaussian noise with standard

Table 3.1: Comparison of the proposed Pconf classification with other methods, with varying degrees of overlap between the positive and negative distributions. We report the mean and standard deviation of the classification accuracy over 20 trials. We show the best and equivalent methods based on the 5% t-test in bold, excluding the fully-supervised method and O-SVM whose settings are different from the others.

Setup	Pconf	Weighted	Regression	O-SVM	Supervised
A	$\mathbf{89.7 \pm 0.6}$	88.7 ± 1.2	68.4 ± 6.5	76.0 ± 3.5	89.8 ± 0.7
B	$\mathbf{81.2 \pm 1.1}$	78.1 ± 1.8	73.2 ± 3.2	71.3 ± 2.3	81.4 ± 1.0
C	$\mathbf{90.2 \pm 9.1}$	82.7 ± 13.1	50.5 ± 1.7	90.8 ± 1.2	93.6 ± 0.5
D	$\mathbf{91.5 \pm 0.5}$	90.8 ± 0.7	64.6 ± 5.3	57.1 ± 4.8	91.4 ± 0.5

deviation chosen from $\{0.01, 0.05, 0.1, 0.2\}$. As the standard deviation gets larger, more noise will be incorporated into positive-confidence. When the modified positive-confidence was over 1 or below 0.01, we clipped it to 1 or rounded up to 0.01 respectively.

The results are shown in Table 3.2. As expected, the performance starts to deteriorate as the confidence becomes more noisy (i.e., as the standard deviation of Gaussian noise is larger), but the proposed method still works reasonably well in almost all cases.

3.4.2 Benchmark Experiments with Neural Network Models

Here, we use more realistic benchmark datasets and more flexible neural network models for experiments.

Fashion-MNIST

The *Fashion-MNIST dataset*[5] consists of 70,000 examples where each sample is a 28×28 gray-scale image (input dimension is 784), associated with a label from 10 fashion item classes. We standardized the data to have zero mean and unit variance.

First, we chose "T-shirt/top" as the positive class, and another item for the negative class. The binary dataset was then divided into four sub-datasets: a training set, a validation set, a test set, and a dataset for learning a probabilistic classifier to estimate positive-confidence. Note that we ask labelers for positive-confidence values in real-world Pconf classification, but we obtained positive-confidence values through a probabilistic classifier here.

[5] https://github.com/zalandoresearch/fashion-mnist

Table 3.2: Mean and standard deviation of the classification accuracy with noisy positive confidence. The experimental setup is the same as Table 3.1, except that positive confidence scores for positive data are noisy. Std. is the standard deviation of Gaussian noise.

Setup A			Setup B		
Std.	Pconf	Weighted	Std.	Pconf	Weighted
0.01	**89.8 ± 0.6**	88.8 ± 0.9	0.01	**81.2 ± 0.9**	78.2 ± 1.4
0.05	**89.7 ± 0.6**	88.3 ± 1.1	0.05	**80.7 ± 2.3**	78.1 ± 1.4
0.10	**89.2 ± 0.7**	87.6 ± 1.4	0.10	**80.8 ± 1.2**	77.8 ± 1.5
0.20	**85.9 ± 2.5**	**85.8 ± 2.5**	0.20	**77.8 ± 1.4**	**77.2 ± 1.9**

Setup C			Setup D		
Std.	Pconf	Weighted	Std.	Pconf	Weighted
0.01	**92.4 ± 1.7**	84.0 ± 8.2	0.01	**91.6 ± 0.5**	90.6 ± 0.9
0.05	**92.2 ± 3.3**	78.5 ± 11.3	0.05	**91.5 ± 0.5**	89.9 ± 1.2
0.10	**90.8 ± 9.5**	72.6 ± 12.9	0.10	**90.8 ± 0.7**	88.7 ± 1.8
0.20	**88.0 ± 9.5**	65.5 ± 13.1	0.20	**87.7 ± 0.8**	85.5 ± 3.7

We used logistic regression with the same network architecture as a probabilistic classifier to generate confidence.[6] However, instead of weight decay, we used *dropout* (Srivastava et al., 2014) with rate 50% after each fully-connected layer, and early-stopping with 20 epochs, since softmax output of flexible neural networks tends to be extremely close to 0 or 1 (Goodfellow et al., 2016), which is not suitable as a representation of confidence. Furthermore, we rounded up positive confidence less than 1% to 1% to stabilize the optimization process.

We compared Pconf classification (3.7) with weighted classification (3.8) and fully-supervised classification based on the empirical version of (3.1). We used the logistic loss for these methods. We also compared our method with Auto-Encoder (Hinton and Salakhutdinov, 2006) as a one-class classification method.

Except Auto-Encoder, we used a fully-connected neural network of three hidden layers (d-100-100-100-1) with *rectified linear units (ReLU)* (Nair and Hinton, 2010) as the activation functions, and weight decay candidates were chosen from $\{10^{-7}, 10^{-4}, 10^{-1}\}$. *Adam* (Kingma and Ba, 2015) was again used for optimization with 200 epochs and mini-batch size 100.

To select hyper-parameters with validation data, we used the zero-one loss ver-

[6] Both positive and negative data are used to train the probabilistic classifier to estimate confidence, and this data is separated from any other process of experiments.

sions of (3.7) and (3.8) for Pconf classification and weighted classification, respectively, since no negative data were available in the validation process and thus we could not directly use the classification accuracy. On the other hand, the classification accuracy was directly used for hyper-parameter tuning of the fully-supervised method, which is extremely advantageous. We reported the test accuracy of the model with the best validation score out of all epochs.

Auto-Encoder was trained with (hard-labeled) positive data, and we classified test data into positive class if the mean squared error (MSE) is below a threshold of 70% quantile, and into negative class otherwise. Since we have no negative data for validating hyper-parameters, we sort the MSEs of training positive data in ascending order. We set the weight decay to 10^{-4}. The architecture is d-100-100-100-100 for encoding and the reversed version for decoding, with *ReLU* after hidden layers and *Tanh* after the final layer.

Table 3.3: Mean and standard deviation of the classification accuracy over 20 trials for the Fashion-MNIST dataset with fully-connected three hidden-layer neural networks. Pconf classification was compared with the baseline Weighted classification method, Auto-Encoder method and fully-supervised method, with *T-shirt* as the positive class and different choices for the negative class. The best and equivalent methods are shown in bold based on the 5% t-test, excluding the Auto-Encoder method and fully-supervised method.

P / N	Pconf	Weighted	Auto-Encoder	Supervised
T-shirt / trouser	**92.14 ± 4.06**	85.30 ± 9.07	71.06 ± 1.00	98.98 ± 0.16
T-shirt / pullover	**96.00 ± 0.29**	**96.08 ± 1.05**	70.27 ± 1.22	96.17 ± 0.34
T-shirt / dress	**91.52 ± 1.14**	89.31 ± 1.08	53.82 ± 0.93	96.56 ± 0.34
T-shirt / coat	**98.12 ± 0.33**	**98.13 ± 1.12**	68.74 ± 0.98	98.44 ± 0.13
T-shirt / sandal	**99.55 ± 0.22**	87.83 ± 18.79	82.02 ± 0.49	99.93 ± 0.09
T-shirt / shirt	**83.70 ± 0.46**	**83.60 ± 0.65**	57.76 ± 0.55	85.57 ± 0.69
T-shirt / sneaker	**89.86 ± 13.32**	58.26 ± 14.27	83.70 ± 0.26	100.00 ± 0.00
T-shirt / bag	**97.56 ± 0.99**	95.34 ± 1.00	82.79 ± 0.70	99.02 ± 0.29
T-shirt / ankle boot	**98.84 ± 1.43**	88.87 ± 7.86	85.07 ± 0.37	99.76 ± 0.07

Table 3.4: Mean and standard deviation of the classification accuracy over 20 trials for the CIFAR-10 dataset with convolutional neural networks. Pconf classification was compared with the baseline Weighted classification method, Auto-Encoder method and fully-supervised method, with *airplane* as the positive class and different choices for the negative class. The best and equivalent methods are shown in bold based on the 5% t-test, excluding the Auto-Encoder method and fully-supervised method.

P / N	Pconf	Weighted	Auto-Encoder	Supervised
airplane / auto	**82.68 ± 1.89**	76.21 ± 2.43	75.13 ± 0.42	93.96 ± 0.58
airplane / bird	**82.23 ± 1.21**	80.66 ± 1.60	54.83 ± 0.39	87.76 ± 4.97
airplane / cat	85.18 ± 1.35	**89.60 ± 0.92**	61.03 ± 0.59	92.90 ± 0.58
airplane / deer	**87.68 ± 1.36**	**87.24 ± 1.58**	55.60 ± 0.53	93.35 ± 0.77
airplane / dog	**89.91 ± 0.85**	**89.08 ± 1.95**	62.64 ± 0.63	94.61 ± 0.45
airplane / frog	**90.80 ± 0.98**	81.84 ± 3.92	62.52 ± 0.68	95.95 ± 0.40
airplane / horse	**89.82 ± 1.07**	85.10 ± 2.61	67.55 ± 0.73	95.65 ± 0.37
airplane / ship	**69.71 ± 2.37**	**70.68 ± 1.45**	52.09 ± 0.42	81.45 ± 8.87
airplane / truck	81.76 ± 2.09	**86.74 ± 0.85**	73.74 ± 0.38	92.10 ± 0.82

CIFAR-10

The *CIFAR-10 dataset* [7] consists of 10 classes, with 5,000 images in each class. Each image is given in a $32 \times 32 \times 3$ format. We chose "airplane" as the positive class and one of the other classes as the negative class in order to construct a dataset for binary classification. We used the neural network architecture specified in Appendix A.1.

For the probabilistic classifier, the same architecture as that for Fashion-MNIST was used except *dropout* with rate 50% was added after the first two fully-connected layers. For Auto-Encoder, the MSE threshold was set to 80% quantile, and we used the architecture specified in Appendix A.2. Other details such as the loss function and weight-decay follow the same setup as the Fashion-MNIST experiments.

Results

The results in Table 3.3 and Table 3.4 show that in most cases, Pconf classification either outperforms or is comparable to the weighted classification baseline, outperforms Auto-Encoder, and is even comparable to the fully-supervised method in some cases.

[7] https://www.cs.toronto.edu/~kriz/cifar.html

3.5 Conclusion

In this section, we conclude by summarizing this chapter and discussing recent advances on positive-confidence learning.

3.5.1 Summary

We proposed a novel problem setting and algorithm for binary classification from positive data equipped with confidence. Our key contribution was to show that an unbiased estimator of the classification risk can be obtained for positive-confidence data, without negative data or even unlabeled data. This was achieved by reformulating the classification risk based on both positive and negative data, to an equivalent expression that only requires positive-confidence data. Theoretically, we established an estimation error bound, and experimentally demonstrated the usefulness of our algorithm.

3.5.2 Recent Advances

In this subsection, we will introduce recent advances on positive-confidence learning.

Skewed confidence In practice, the confidence given by the labeler may be skewed due to bias arising in the labeling process. In Section 3.4.1, we demonstrated how our method can be robust to some extent when adding a specific type of noise to the confidence. If we know our confidence is skewed, can we take that into account to aim for better generalization? Recently, a method to adjust the skewed confidence and learn a better classifier was proposed (Shinoda et al., 2020). An exponential model was used for the skewed confidence, and the knowledge of misclassification rate of positive samples was assumed. Then, the model parameter was chosen so that the squared error between the empirical classification error of the positive samples and the underlying misclassification rate of the positive class is minimized. In synthetic and benchmark experiments, the proposed method in Shinoda et al. (2020) was significantly better than the original method we discussed in this chapter, when the confidence was skewed. Furthermore, an experiment with a dataset of driver drowsiness prediction with real-world confidence annotation was shown.

Negative confidence There has been several extensions to utilize confidence of negative data or unlabeled data. Yabe and Zempo (2020) proposed an extension that utilize negative-confidence for negative samples, which is helpful when we only have a limited number of negative training samples. In Wang et al. (2020), they proposed

an binary classification problem with positive and unlabeled samples, but assumed that the unlabeled samples are equipped with negative confidence.

Chapter 4

Learning from Complementary Labels

Collecting labeled data is costly and thus a critical bottleneck in real-world multi-class classification tasks. In this chapter, we propose a novel setting, namely *learning from complementary labels* for multi-class classification to mitigate this problem. A complementary label specifies a class that a pattern does not belong to. Collecting complementary labels would be less laborious than collecting ordinary labels, since annotators/labelers do not have to carefully choose the correct class from a long list of candidate classes. However, complementary labels are less informative than ordinary labels and thus a suitable approach is needed to better learn from them.

We show that an unbiased estimator to the multi-class classification risk can be obtained only from complementarily labeled data, if a loss function satisfies a particular symmetric condition. We derive estimation error bounds for the proposed method and prove that the optimal convergence rate is achieved. We further show that learning from complementary labels can be easily combined with learning from ordinary labels (i.e., ordinary supervised learning), providing a highly practical implementation of the proposed method.

We propose an extension for complementary-label learning with an unbiased estimator of the classification risk, for arbitrary losses. We further improved the risk estimator by a non-negative correction and gradient ascent trick. We demonstrate the usefulness of complementary-label learning in experiments with benchmark datasets.

4.1 Introduction

In ordinary supervised classification problems, each training pattern is equipped with a label which specifies the class the pattern belongs to. Although supervised classifier training is effective, the cost of labeling training patterns is often expensive.

We consider a novel weakly supervised classification scenario: instead of ordinary class labels, only a *complementary label* which specifies a class the pattern does *not* belong to is available. If the number of classes is large, choosing a correct class label from many candidate classes is laborious, while choosing one of the incorrect class labels would be much easier and thus less costly. Classification with complementary labels is essentially equivalent to classification with ordinary labels in the binary classification setup, because complementary label 1 (i.e., not class 1) immediately means ordinary label 2. On the other hand, complementary labels are less informative than ordinary labels in K-class problems for $K > 2$, because complementary label 1 only means either of the ordinary labels $2, 3, \ldots, K$.

The complementary classification problem may be solved by the method of learning from *partial labels* (Cour et al., 2011), where multiple candidate classes are provided to each training pattern — complementary label $\bar{y}$ can be regarded as an extreme case of a partial label given to all $K - 1$ classes other than class $\bar{y}$. Even though the proposed method of Cour et al. (2011) shows statistical consistency, it does not give an unbiased estimator of the classification risk. Furthermore, it has different assumptions, e.g., dominance relation, compared to our assumption in our problem. Another possibility to solve the complementary classification problem is to consider a multi-label setup (Read et al., 2011), where each pattern can belong to multiple classes — complementary label $\bar{y}$ is translated into a negative label for class $\bar{y}$ and positive labels for the other $K - 1$ classes.

Our contribution is to give a direct risk minimization framework for the complementary classification problem. We then show that the classification risk can be rewritten with complementary labels with certain symmetric conditions for the loss functions. Theoretically, we establish the estimation error bounds for the proposed method, showing that learning from complementary labels is also consistent; the order of these bounds is the optimal rate. We further show that learning from complementary labels can be easily combined with learning from ordinary labels (i.e., ordinary supervised learning), providing a highly practical implementation of the proposed method. This will be discussed in Section 4.2.

We propose an extension for complementary-label learning with an unbiased estimator of the classification risk, for arbitrary losses and models. We further improved the risk estimator by a non-negative correction and gradient ascent trick. This will

be discussed in Section 4.3.

4.2 Complementary-Label Learning with Symmetric Losses

In this section, we introduce our complementary-label learning framework with symmetric losses. We show the behavior of the proposed method theoretically and empirically. Finally, we will show that we can learn from both ordinary labels and complementary labels.

4.2.1 Formulation

Suppose that d-dimensional pattern $\boldsymbol{x} \in \mathbb{R}^d$ and its class label $y \in \{1, \ldots, K\}$ are sampled independently from an unknown probability distribution with density $p(\boldsymbol{x}, y)$. Recall that the goal of ordinary multi-class classification is to learn a score function $\boldsymbol{g}(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^K$ that minimizes the classification risk with multi-class loss $\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), y\big)$:

$$R(\boldsymbol{g}) = \mathbb{E}_{p(\boldsymbol{x},y)}\big[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), y\big)\big], \tag{4.1}$$

where $\mathbb{E}$ denotes the expectation. The prediction for $\boldsymbol{x}$ with the score function $\boldsymbol{g}$ is

$$\hat{y} = \operatorname*{arg\,max}_{y \in \{1,\ldots,K\}} g_y(\boldsymbol{x}), \tag{4.2}$$

where $g_y(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ is a binary classifier for class y versus the rest. Then, together with a binary loss $\ell(z) : \mathbb{R} \to \mathbb{R}$ that incurs a large loss for large z, the *one-versus-all* (OVA) loss[1] or the *pairwise-comparison* (PC) loss defined as follows are used as the multi-class loss (Zhang, 2004a):

$$\mathcal{L}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}), y) = \ell\big(g_y(\boldsymbol{x})\big) + \frac{1}{K-1}\sum_{y' \neq y} \ell\big(-g_{y'}(\boldsymbol{x})\big), \tag{4.3}$$

$$\mathcal{L}_{\mathrm{PC}}\big(\boldsymbol{g}(\boldsymbol{x}), y\big) = \sum_{y' \neq y} \ell\big(g_y(\boldsymbol{x}) - g_{y'}(\boldsymbol{x})\big). \tag{4.4}$$

Finally, the expectation over unknown $p(\boldsymbol{x}, y)$ in Eq.(4.1) is approximated by the empirical average over independent and identically distributed training samples to obtain a practical classification formulation.

[1] We normalize the "rest" loss by $K - 1$ to be consistent with the discussion in the following sections.

However, our situation do not follow this standard setup. We consider the situation where, instead of ordinary class label y, we are given only *complementary label* $\bar{y}$ which specifies a class pattern $\boldsymbol{x}$ does *not* belong to. Our goal is to still learn a classifier that minimizes the classification risk (4.1), but only from complementarily labeled training samples $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$. We assume that $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$ are drawn independently from an unknown probability distribution with density

$$\bar{p}(\boldsymbol{x}, \bar{y}) = \frac{1}{K-1} \sum_{y \neq \bar{y}} p(\boldsymbol{x}, y). \tag{4.5}$$

The coefficient $1/(K-1)$ is for the normalization purpose: it is natural to assume $\bar{p}(\boldsymbol{x}, \bar{y}) = (1/Z) \sum_{y \neq \bar{y}} p(\boldsymbol{x}, y)$ since all $p(\boldsymbol{x}, y)$ for $y \neq \bar{y}$ contribute to $\bar{p}(\boldsymbol{x}, \bar{y})$; in order to ensure that $\bar{p}(\boldsymbol{x}, \bar{y})$ is a valid joint density such that $\mathbb{E}_{\bar{p}(\boldsymbol{x}, \bar{y})}[1] = 1$, we take $Z = K - 1$.

Let us consider a *complementary* loss $\bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y})$ for a complementary sample $(\boldsymbol{x}, \bar{y})$. Then we have the following theorem:

Theorem 4.1. *The classification risk (4.1) can be expressed as*

$$R(\boldsymbol{g}) = (K-1)\mathbb{E}_{\bar{p}(\boldsymbol{x}, \bar{y})}\big[\bar{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}), \bar{y}\big)\big] - M_1 + M_2, \tag{4.6}$$

if there exist constants $M_1, M_2 \geq 0$ such that the complementary loss satisfies for all $\boldsymbol{x}$ and y

$$\sum_{\bar{y}=1}^{K} \bar{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}), \bar{y}\big) = M_1 \quad \textit{and} \quad \bar{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}), y\big) + \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), y\big) = M_2. \tag{4.7}$$

Proof. According to (4.5),

$$(K-1)\mathbb{E}_{\bar{p}(\boldsymbol{x}, \bar{y})}[\bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y})] = (K-1) \int \sum_{\bar{y}=1}^{K} \bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) \bar{p}(\boldsymbol{x}, \bar{y}) \mathrm{d}\boldsymbol{x} \tag{4.8}$$

$$= (K-1) \int \sum_{\bar{y}=1}^{K} \bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) \left(\frac{1}{K-1} \sum_{y \neq \bar{y}} p(\boldsymbol{x}, y) \right) \mathrm{d}\boldsymbol{x} \tag{4.9}$$

$$= \int \sum_{y=1}^{K} \sum_{\bar{y} \neq y} \bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) p(\boldsymbol{x}, y) \mathrm{d}\boldsymbol{x} \tag{4.10}$$

$$= \mathbb{E}_{p(\boldsymbol{x},y)} \left[\sum_{\bar{y} \neq y} \bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) \right] \tag{4.11}$$

$$= M_1 - \mathbb{E}_{p(\boldsymbol{x},y)}[\bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), y)], \tag{4.12}$$

where the fifth equality is due to the first equation in (4.7). Subsequently,

$$(K-1)\mathbb{E}_{\bar{p}(\boldsymbol{x},\bar{y})}[\bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y})] - \mathbb{E}_{p(\boldsymbol{x},y)}[\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), y)] \tag{4.13}$$

$$= M_1 - \mathbb{E}_{p(\boldsymbol{x},y)}[\bar{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), y) + \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), y)] \tag{4.14}$$

$$= M_1 - M_2, \tag{4.15}$$

where the second equality is due to the second equation in (4.7). □

With the expression (4.6), the classification risk (4.1) can be naively approximated in an unbiased fashion by the sample average as

$$\widehat{R}(\boldsymbol{g}) = \frac{K-1}{n} \sum_{i=1}^{n} \bar{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}_i), \bar{y}_i\big) - M_1 + M_2. \tag{4.16}$$

Let us define the complementary losses for the OVA loss $\mathcal{L}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}), y)$ and the PC loss $\mathcal{L}_{\mathrm{PC}}\big(\boldsymbol{g}(\boldsymbol{x}), y\big)$ as

$$\bar{\mathcal{L}}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) = \frac{1}{K-1} \sum_{y \neq \bar{y}} \ell\big(g_y(\boldsymbol{x})\big) + \ell\big(-g_{\bar{y}}(\boldsymbol{x})\big), \tag{4.17}$$

$$\bar{\mathcal{L}}_{\mathrm{PC}}\big(\boldsymbol{g}(\boldsymbol{x}), \bar{y}\big) = \sum_{y \neq \bar{y}} \ell\big(g_y(\boldsymbol{x}) - g_{\bar{y}}(\boldsymbol{x})\big). \tag{4.18}$$

Then we have the following theorem:

Theorem 4.2. *If binary loss $\ell(z)$ satisfies*

$$\ell(z) + \ell(-z) = 1, \tag{4.19}$$

then $\bar{\mathcal{L}}_{\mathrm{OVA}}$ satisfies conditions (4.7) with $M_1 = K$ and $M_2 = 2$, and $\bar{\mathcal{L}}_{\mathrm{PC}}$ satisfies conditions (4.7) with $M_1 = K(K-1)/2$ and $M_2 = K-1$.

Proof. From Eq.(4.19), we have

$$\sum_{\bar{y}=1}^{K} \bar{\mathcal{L}}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) = \frac{1}{K-1} \sum_{\bar{y}=1}^{K} \sum_{y \neq \bar{y}} \ell\big(g_y(\boldsymbol{x})\big) + \sum_{\bar{y}=1}^{K} \ell\big(-g_{\bar{y}}(\boldsymbol{x})\big) \tag{4.20}$$

$$= \sum_{\bar{y}=1}^{K} \big(\ell\big(g_{\bar{y}}(\boldsymbol{x})\big) + \ell\big(-g_{\bar{y}}(\boldsymbol{x})\big)\big) \tag{4.21}$$

$$= K, \tag{4.22}$$

$$\mathcal{L}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}), y) + \bar{\mathcal{L}}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}), y) = \ell\big(g_y(\boldsymbol{x})\big) + \frac{1}{K-1}\sum_{\bar{y}\neq y} \ell\big(-g_{\bar{y}}(\boldsymbol{x})\big) \tag{4.23}$$

$$+ \frac{1}{K-1}\sum_{y'\neq y} \ell\big(g_{y'}(\boldsymbol{x})\big) + \ell\big(-g_y(\boldsymbol{x})\big) \tag{4.24}$$

$$= 2, \tag{4.25}$$

$$\sum_{\bar{y}=1}^{K} \bar{\mathcal{L}}_{\mathrm{PC}}\big(\boldsymbol{g}(\boldsymbol{x}), \bar{y}\big) = \sum_{\bar{y}=1}^{K}\sum_{y\neq\bar{y}} \ell\big(g_y(\boldsymbol{x}) - g_{\bar{y}}(\boldsymbol{x})\big) \tag{4.26}$$

$$= \sum_{\bar{y}=1}^{K-1}\sum_{y=\bar{y}+1}^{K} \Big(\ell\big(g_y(\boldsymbol{x}) - g_{\bar{y}}(\boldsymbol{x})\big) + \ell\big(g_{\bar{y}}(\boldsymbol{x}) - g_y(\boldsymbol{x})\big)\Big) \tag{4.27}$$

$$= \frac{K(K-1)}{2}, \tag{4.28}$$

$$\mathcal{L}_{\mathrm{PC}}(\boldsymbol{g}(\boldsymbol{x}), y) + \bar{\mathcal{L}}_{\mathrm{PC}}(\boldsymbol{g}(\boldsymbol{x}), y) = \sum_{y'\neq y} \ell\big(g_y(\boldsymbol{x}) - g_{y'}(\boldsymbol{x})\big) + \sum_{y'\neq y} \ell\big(g_{y'}(\boldsymbol{x}) - g_y(\boldsymbol{x})\big) \tag{4.29}$$

$$= K - 1. \tag{4.30}$$

□

For example, the following binary losses satisfy the symmetric condition (4.19):

$$\text{Zero-one loss: } \ell_{01}(z) = \begin{cases} 0 & \text{if } z > 0, \\ 1 & \text{if } z \leq 0, \end{cases} \tag{4.31}$$

$$\text{Sigmoid loss: } \ell_{\mathrm{S}}(z) = \frac{1}{1+e^{z}}, \tag{4.32}$$

$$\text{Ramp loss: } \ell_{\mathrm{R}}(z) = \frac{1}{2}\max\Big(0, \min\big(2, 1-z\big)\Big). \tag{4.33}$$

Note that these losses are non-convex (du Plessis et al., 2014). In practice, the sigmoid loss or ramp loss may be used for training a classifier, while the zero-one loss

may be used for tuning hyper-parameters.

4.2.2 Theoretical Analysis

In the following, we establish the estimation error bounds for the proposed method.

Let $\mathcal{G} = \{g(\boldsymbol{x})\}$ be a function class for empirical risk minimization and $\sigma_1, \ldots, \sigma_n$ be n Rademacher variables. Then the Rademacher complexity of $\mathcal{G}$ for $\mathcal{X}$ of size n drawn from $p(\boldsymbol{x})$ is defined as follows (Mohri et al., 2012):

$$\mathfrak{R}_n(\mathcal{G}) = \mathbb{E}_{\mathcal{X}}\mathbb{E}_{\sigma_1,\ldots,\sigma_n}\left[\sup_{g\in\mathcal{G}}\frac{1}{n}\sum_{\boldsymbol{x}_i\in\mathcal{X}}\sigma_i g(\boldsymbol{x}_i)\right]; \tag{4.34}$$

define the Rademacher complexity of $\mathcal{G}$ for $\bar{\mathcal{X}}$ of size n drawn from $\bar{p}(\boldsymbol{x})$ as

$$\bar{\mathfrak{R}}_n(\mathcal{G}) = \mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma_1,\ldots,\sigma_n}\left[\sup_{g\in\mathcal{G}}\frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}}\sigma_i g(\boldsymbol{x}_i)\right]. \tag{4.35}$$

Note that $\bar{p}(\boldsymbol{x}) = p(\boldsymbol{x})$ and thus $\bar{\mathfrak{R}}_n(\mathcal{G}) = \mathfrak{R}_n(\mathcal{G})$, which enables us to express the obtained theoretical results using the standard Rademacher complexity $\mathfrak{R}_n(\mathcal{G})$.

To begin with, let $\tilde{\ell}(z) = \ell(z) - \ell(0)$ be the shifted loss such that $\tilde{\ell}(0) = 0$, and $\widetilde{\mathcal{L}}_{\mathrm{OVA}}$ and $\widetilde{\mathcal{L}}_{\mathrm{PC}}$ be losses defined following (4.17) and (4.18) but with $\tilde{\ell}$ instead of ℓ; let L_ℓ be any (not necessarily the best) Lipschitz constant of ℓ. Define the corresponding function classes as follows:

$$\mathcal{H}_{\mathrm{OVA}} = \{(\boldsymbol{x}, \bar{y}) \mapsto \widetilde{\mathcal{L}}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) \mid g_1, \ldots, g_K \in \mathcal{G}\}, \tag{4.36}$$

$$\mathcal{H}_{\mathrm{PC}} = \{(\boldsymbol{x}, \bar{y}) \mapsto \widetilde{\mathcal{L}}_{\mathrm{PC}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) \mid g_1, \ldots, g_K \in \mathcal{G}\}. \tag{4.37}$$

Then we can obtain the following lemmas.

Lemma 4.3. *Let $\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{OVA}})$ be the Rademacher complexity of $\mathcal{H}_{\mathrm{OVA}}$ for $\mathcal{S}$ of size n drawn from $\bar{p}(x, \bar{y})$ defined as*

$$\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{OVA}}) = \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma_1,\ldots,\sigma_n}\left[\sup_{h\in\mathcal{H}_{\mathrm{OVA}}}\frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}}\sigma_i h(\boldsymbol{x}_i, \bar{y}_i)\right]. \tag{4.38}$$

Then, $\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{OVA}}) \leq \frac{K(K+1)}{K-1}L_\ell\mathfrak{R}_n(\mathcal{G})$.

A proof is given in Section B.1.

Lemma 4.4. *Let $\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{PC}})$ be the Rademacher complexity of $\mathcal{H}_{\mathrm{PC}}$ defined similarly to $\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{OVA}})$. Then, $\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{PC}}) \leq 2K(K-1)L_\ell \mathfrak{R}_n(\mathcal{G})$.*

A proof is given in Section B.2. Based on Lemmas 4.3 and 4.4, we can derive the uniform deviation bounds of $\widehat{R}(\boldsymbol{g})$.

Lemma 4.5. *For any $\delta > 0$, with probability at least $1-\delta$,*

$$\sup_{g_1,\ldots,g_K \in \mathcal{G}} \left| \widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}) \right| \leq 2K(K-1)L_\ell \mathfrak{R}_n(\mathcal{G}) + (K-1)\sqrt{\frac{2\ln(2/\delta)}{n}}, \tag{4.39}$$

where $\widehat{R}(\boldsymbol{g})$ is w.r.t. $\bar{\mathcal{L}}_{\mathrm{OVA}}$, and

$$\sup_{g_1,\ldots,g_K \in \mathcal{G}} \left| \widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}) \right| \leq 4K(K-1)^2 L_\ell \mathfrak{R}_n(\mathcal{G}) + (K-1)^2\sqrt{\frac{\ln(2/\delta)}{2n}}, \tag{4.40}$$

where $\widehat{R}(\boldsymbol{g})$ is w.r.t. $\bar{\mathcal{L}}_{\mathrm{PC}}$.

A proof is given in Section B.3. Let $(g_1^*, \ldots, g_K^*)$ be the true risk minimizer and $(\hat{g}_1, \ldots, \hat{g}_K)$ be the empirical risk minimizer, i.e.,

$$(g_1^*, \ldots, g_K^*) = \operatorname*{arg\,min}_{g_1,\ldots,g_K \in \mathcal{G}} R(\boldsymbol{g}) \quad \text{and} \quad (\hat{g}_1, \ldots, \hat{g}_K) = \operatorname*{arg\,min}_{g_1,\ldots,g_K \in \mathcal{G}} \widehat{R}(\boldsymbol{g}). \tag{4.41}$$

Finally, based on Lemma 4.5, we can establish the estimation error bounds.

Theorem 4.6. *For any $\delta > 0$, with probability at least $1-\delta$,*

$$R(\hat{\boldsymbol{g}}) - R(\boldsymbol{g}^*) \leq 4K(K+1)L_\ell \mathfrak{R}_n(\mathcal{G}) + (K-1)\sqrt{\frac{8\ln(2/\delta)}{n}}, \tag{4.42}$$

if $(\hat{g}_1, \ldots, \hat{g}_K)$ is trained by minimizing $\widehat{R}(\boldsymbol{g})$ is w.r.t. $\bar{\mathcal{L}}_{\mathrm{OVA}}$, and

$$R(\hat{\boldsymbol{g}}) - R(\boldsymbol{g}^*) \leq 8K(K-1)^2 L_\ell \mathfrak{R}_n(\mathcal{G}) + (K-1)^2\sqrt{\frac{2\ln(2/\delta)}{n}}, \tag{4.43}$$

if $(\hat{g}_1, \ldots, \hat{g}_K)$ is trained by minimizing $\widehat{R}(\boldsymbol{g})$ is w.r.t. $\bar{\mathcal{L}}_{\mathrm{PC}}$.

A proof is given in Section B.4. Theorem 4.6 guarantees learning from complementary labels is also consistent: as $n \to \infty$, $R(\hat{\boldsymbol{g}}) \to R(\boldsymbol{g}^*)$. Consider linear-in-parameter models defined by

$$\mathcal{G} = \{g(\boldsymbol{x}) = \langle w, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}} \mid \|w\|_{\mathcal{H}} \leq C_w, \|\phi(\boldsymbol{x})\|_{\mathcal{H}} \leq C_\phi\}, \tag{4.44}$$

where here $\mathcal{H}$ is a Hilbert space with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, $w \in \mathcal{H}$ is a normal, $\phi : \mathbb{R}^d \to \mathcal{H}$ is a feature map, and $C_w > 0$ and $C_\phi > 0$ are constants (Schölkopf and Smola, 2001). It is known that $\mathfrak{R}_n(\mathcal{G}) \leq C_w C_\phi / \sqrt{n}$ (Mohri et al., 2012) and thus $R(\hat{\boldsymbol{g}}) \to R(\boldsymbol{g}^*)$ in $\mathcal{O}_p(1/\sqrt{n})$ if this $\mathcal{G}$ is used, where $\mathcal{O}_p$ denotes the order in probability. This order is already the optimal parametric rate and cannot be improved without additional strong assumptions on $\bar{p}(\boldsymbol{x}, \bar{y})$, ℓ and $\mathcal{G}$ jointly.

4.2.3 Incorporation of Ordinary Labels

In many practical situations, we may also have ordinarily labeled data in addition to complementarily labeled data. For example, in crowdsourcing (Howe, 2008), we may choose one of the classes randomly by following the uniform distribution, with probability $\frac{1}{K-1}$ for each class, and ask crowdworkers whether a pattern belongs to the chosen class or not. Then the pattern is treated as ordinarily labeled if the answer is yes; otherwise, the pattern is regarded as complementarily labeled.

In such cases, we want to leverage both kinds of labeled data to obtain more accurate classifiers. To this end, motivated by Sakai et al. (2017), let us consider a convex combination of the classification risks derived from ordinarily labeled data and complementarily labeled data:

$$R(\boldsymbol{g}) = \alpha \mathbb{E}_{p(\boldsymbol{x},y)}[\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), y)] + (1-\alpha)\Big[(K-1)\mathbb{E}_{\bar{p}(\boldsymbol{x},\bar{y})}[\overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y})] - M_1 + M_2\Big], \tag{4.45}$$

where $\alpha \in [0, 1]$ is a hyper-parameter that interpolates between the two risks. The combined risk (4.45) can be naively approximated by the sample averages as

$$\widehat{R}(\boldsymbol{g}) = \frac{\alpha}{m}\sum_{j=1}^{m} \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}_j), y_j) + \frac{(1-\alpha)(K-1)}{n}\sum_{i=1}^{n} \overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}_i), \bar{y}_i), \tag{4.46}$$

where $\{(\boldsymbol{x}_j, y_j)\}_{j=1}^m$ are ordinarily labeled data and $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^n$ are complementarily labeled data. Our risk estimator (4.46) can utilize both kinds of labeled data to obtain better classifiers[2]. We will experimentally demonstrate the usefulness of this combination method in Section 4.2.4.

4.2.4 Experiments

We experimentally evaluate the performance of the proposed method.

[2]Note that when pattern $\boldsymbol{x}$ has already been equipped with ordinary label y, giving complementary label $\bar{y}$ does not bring us any additional information (unless the ordinary label is noisy).

Table 4.1: Mean and standard deviation of classification accuracy over five trials in percentage, when the number of classes is changed. "PC" is (4.18), "OVA" is (4.17), "sigmoid" is (4.32), and "ramp" is (4.33). Best and equivalent methods (with 5% t-test) are bold.

Method	3 cls	4 cls	5 cls	6 cls	7 cls	8 cls	9 cls	10 cls
OVA Sigmoid	**96.2 (0.4)**	**91.5 (1.2)**	**90.2 (0.9)**	**79.2 (4.2)**	**75.2 (2.7)**	68.4 (2.5)	**61.9 (5.2)**	52.4 (5.4)
OVA Ramp	**95.9 (0.5)**	**90.5 (1.1)**	**89.9 (1.0)**	**77.0 (5.3)**	**73.2 (1.5)**	62.9 (5.0)	54.1 (5.8)	48.9 (3.3)
PC Sigmoid	**95.9 (1.1)**	**90.8 (0.6)**	**89.3 (1.3)**	**80.8 (2.0)**	**76.9 (3.2)**	**72.2 (2.2)**	**66.4 (3.9)**	**60.4 (0.6)**
PC Ramp	**95.9 (1.1)**	**90.6 (1.2)**	88.2 (1.2)	**80.1 (2.4)**	**74.7 (3.2)**	**70.0 (2.6)**	**62.3 (4.5)**	55.1 (3.5)

Comparison between proposed methods

Here we first compare the performance between four variations of the proposed approach: The two formulations, OVA (4.17) and PC (4.18), each with the sigmoid loss (4.32) and ramp loss (4.33). We used the MNIST hand-written digit dataset (with all patterns standardized to have zero mean and unit variance), with different number of classes: 3 classes (digits "1" to "3") to 10 classes (digits "1" to "9" and "0"). From each class, we selected 500 samples for training and 500 samples for testing, and generated complementary labels by randomly selecting one of the complementary classes. From the training dataset, we left out 25% for validation for hyper-parameter tuning based on the zero-one loss objective version of (4.17) or (4.18).

For all the methods, we used a linear-in-input model $g_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \boldsymbol{x} + b_k$ as the binary classifier, where $^\top$ denotes the transpose, $\boldsymbol{w}_k \in \mathbb{R}^d$ is the weight parameters, and $b_k \in \mathbb{R}$ is a bias parameter in class $k \in \{1, \ldots, K\}$. We added an ℓ_2-regularization term, with hyper-parameter candidates $\lambda \in \{10^{-4}, 10^{-3}, \ldots, 10^4\}$. Adam (Kingma and Ba, 2015) was used for optimization with 5,000 iterations.

We reported the mean and standard deviation of the classification accuracy over five trials in Table 4.1. From the results, we can see that PC tends to outperform OVA. A possible explanation for this is that the PC formulation is a more direct approach for classification (Vapnik, 1998) — it takes the sign of the difference of the classifiers, instead of the sign of each classifier as in OVA. When we compare the sigmoid loss and the ramp loss in PC, the sigmoid loss tends to outperform the ramp loss and hence we use only PC with sigmoid loss for the following experiments.

Benchmark Experiments

Next, we compare our proposed method, PC with the sigmoid loss, with two baseline methods. The first baseline is one of the state-of-the-art partial label (PL) methods (Cour et al., 2011) with the squared hinge loss $\ell(z) = (\max(0, 1-z))^2$. The second baseline method is a multi-label (ML) formulation, where complementary label $\bar{y}$ is translated into a negative label for class $\bar{y}$ and positive labels for the other $K-1$ classes. This formulation yields the loss $\mathcal{L}_{\mathrm{ML}}(\boldsymbol{g}(\boldsymbol{x}), \bar{y}) = \sum_{y \neq \bar{y}} \ell\big(g_y(\boldsymbol{x})\big) + \ell\big(-g_{\bar{y}}(\boldsymbol{x})\big)$, where we used the squared loss $\ell(z) = (z-1)^2$ as the binary loss.

Another interesting comparison is between learning from complementary labels and ordinary labels. For the ordinary-label (OL) method, we used the unnormalized version of (4.3) with the squared loss. For better comparison of the two settings, we gave only $\frac{1}{K-1}$ times as many samples to OL since one ordinary label can be regarded as $K-1$ complementary labels. We used a one-hidden-layer neural network (d-3-1) with rectified linear units (ReLU) (Nair and Hinton, 2010) as a activation function, and weight decay candidates were chosen from $\{10^{-7}, 10^{-4}, 10^{-1}\}$.

We evaluated the classification performance with the following benchmark datasets: WAVEFORM1 ($d = 21$), WAVEFORM2 ($d = 40$), SATIMAGE ($d = 36$), SHUTTLE ($d = 9$), SEGMENTATION ($d = 19$), PENDIGITS ($d = 16$), MNIST ($d = 784$), DRIVE ($d = 48$), LETTER ($d = 16$), VOWEL ($d = 12$), and USPS ($d = 256$). MNIST and USPS can be downloaded from the website of the late Sam Roweis[3], and all other datasets can be downloaded from *UCI machine learning repository*.[4] For datasets with 10 or more classes, we chose five classes with an equal number of samples. In Table 4.2, the specification of the datasets as well as the mean and standard deviation of the classification accuracy over 20 trials is reported. From the results, we can see that the proposed method is either competitive or outperforms the baseline methods in many of the datasets. It is also interesting that the proposed classification method from complementary labels can be competitive with classification from ordinary labels in many of the datasets.

Combination of Ordinary and Complementary Labels

Finally, we demonstrate the usefulness of combining ordinarily and complementarily labeled data. We used (4.46), with hyperparameter α fixed at $1/2$ for simplicity. We divided our training dataset by $1 : (K-1)$ ratio, where one subset was labeled

[3] See http://cs.nyu.edu/~roweis/data.html.
[4] See http://archive.ics.uci.edu/ml/.

Table 4.2: Mean and standard deviation of classification accuracy over 20 trials in percentage. "PC/S" is the proposed method for pairwise comparison formulation with sigmoid loss, "PL" is partial label with squared hinge loss, "ML" is multi-label, and "OL" is classification from ordinary labels. Best and equivalent methods (with 5% t-test excluding "OL") are bold. # train denotes the total number of training and validation samples in each class. # test denotes the number of test samples in each class.

Dataset	Class	# train	# test	PC/S	PL	ML	OL
WAVEFORM1	$1 \sim 3$	1230	406	**85.7(0.9)**	84.1(1.5)	84.7(1.6)	85.8(0.9)
WAVEFORM2	$1 \sim 3$	1221	400	**84.4(1.3)**	**83.1(2.7)**	81.8(2.3)	86.7(1.8)
SATIMAGE	$1 \sim 7$	415	211	**67.2(7.0)**	54.8(6.8)	51.6(6.0)	67.9(4.2)
SHUTTLE	$1, 4, 5$	2458	809	94.9(9.7)	**97.5(0.7)**	90.4(11.8)	97.5(0.8)
SEGMENTATION	$1 \sim 7$	29	299	**36.1(6.8)**	31.7(5.8)	26.6(5.4)	58.6(4.5)
PENDIGITS	$1 \sim 5$	719	336	**79.4(9.5)**	73.2(6.4)	**75.9(7.7)**	78.8(2.9)
	$6 \sim 10$	719	335	**77.7(3.8)**	65.5(6.4)	72.0(8.6)	74.7(4.6)
	even #	719	335	**74.0(7.3)**	58.5(9.9)	65.7(6.3)	74.8(5.5)
	odd #	719	336	**88.5(5.9)**	74.6(4.4)	79.1(6.1)	84.0(8.8)
MNIST	$1 \sim 5$	5842	980	**88.4(4.2)**	71.5(7.4)	56.6(12.4)	77.9(0.4)
	$6 \sim 10$	5421	892	**83.4(2.6)**	67.4(8.1)	50.5(13.7)	77.0(4.5)
	even #	5421	892	**85.3(2.2)**	70.4(6.7)	61.7(11.1)	76.7(1.4)
	odd #	5842	958	**85.0(3.7)**	67.3(8.6)	57.3(13.0)	76.5(0.7)
DRIVE	$1 \sim 5$	3931	1280	**87.6(5.9)**	72.7(7.0)	64.2(12.6)	79.3(5.1)
	$6 \sim 10$	3958	1318	**84.9(5.7)**	73.1(5.8)	69.7(9.3)	81.6(2.9)
	even #	3932	1295	**82.4(5.6)**	72.9(6.6)	63.2(12.8)	83.5(5.3)
	odd #	3931	1310	**76.9(8.0)**	60.0(6.9)	51.6(9.3)	65.4(3.3)
LETTER	$1 \sim 5$	565	171	**79.6(5.5)**	67.6(6.0)	71.0(9.3)	82.2(4.3)
	$6 \sim 10$	550	178	**73.2(6.3)**	63.9(6.1)	61.2(10.6)	75.9(5.6)
	$11 \sim 15$	556	177	**73.3(5.9)**	66.6(3.4)	59.0(10.1)	75.4(5.0)
	$16 \sim 20$	550	184	**71.5(5.9)**	64.9(5.2)	63.5(7.0)	73.9(5.3)
	$21 \sim 25$	585	167	**76.2(6.0)**	68.3(8.1)	63.1(11.2)	77.1(5.1)
VOWEL	$1 \sim 5$	48	42	**35.6(9.0)**	**37.0(9.3)**	31.5(6.7)	54.9(6.7)
	$6 \sim 10$	48	42	**32.6(7.5)**	**34.1(7.7)**	**30.0(9.8)**	53.0(4.4)
	even #	48	42	**36.6(9.0)**	**39.9(10.5)**	**33.3(7.8)**	62.1(5.6)
	odd #	48	42	**28.2(9.0)**	**28.8(7.2)**	23.2(4.8)	54.0(5.5)
USPS	$1 \sim 5$	652	166	**70.1(5.2)**	62.8(7.2)	45.8(5.9)	76.2(2.3)
	$6 \sim 10$	542	147	**64.3(4.7)**	**61.4(5.9)**	41.7(5.3)	76.9(5.1)
	even #	556	147	**70.6(5.4)**	63.7(7.2)	48.4(5.3)	75.7(2.7)
	odd #	542	166	**63.1(4.3)**	57.8(6.8)	37.8(5.7)	73.6(3.4)

Table 4.3: Means and standard deviations of classification accuracy over 10 trials in percentage. "OL" is the ordinary label method, "CL" is the complementary label method, and "OL & CL" is a combination method that uses both ordinarily and complementarily labeled data. Best and equivalent methods are highlighted in boldface. "Class" denotes the class labels used for the experiment and "Dim" denotes the dimensionality d of patterns to be classified. # train denotes the number of ordinarily/complementarily labeled data for training and validation in each class. # test denotes the number of test data in each class.

Dataset	Class	Dim	# train	# test	OL ($\alpha = 1$)	CL ($\alpha = 0$)	OL & CL ($\alpha = \frac{1}{2}$)
WAVEFORM1	$1 \sim 3$	21	413/826	408	85.3(0.8)	86.0(0.4)	**86.9**(**0.5**)
WAVEFORM2	$1 \sim 3$	40	411/821	411	82.7(1.3)	82.0(1.7)	**84.7**(**0.6**)
SATIMAGE	$1 \sim 7$	36	69/346	211	74.9(4.9)	70.1(5.6)	**81.2**(**1.1**)
PENDIGITS	$1 \sim 5$	16	144/575	336	**91.3**(**2.1**)	84.7(3.2)	**93.1**(**2.0**)
	$6 \sim 10$		144/575	335	**86.3**(**3.5**)	78.3(6.2)	**87.8**(**2.8**)
	even #		144/575	336	94.3(1.7)	91.0(4.3)	**95.8**(**0.6**)
	odd #		144/575	335	**85.6**(**2.0**)	75.9(3.1)	**86.9**(**1.1**)
	$1 \sim 10$		72/647	335	61.7(4.3)	41.1(5.7)	**66.9**(**2.0**)
DRIVE	$1 \sim 5$	48	780/3121	1305	92.1(2.6)	89.0(2.1)	**94.2**(**1.0**)
	$6 \sim 10$		795/3180	1290	**87.0**(**3.0**)	86.5(3.1)	**89.5**(**2.1**)
	even #		657/3284	1314	**91.4**(**2.9**)	81.8(4.6)	**91.8**(**3.3**)
	odd #		790/3161	1255	91.1(1.5)	86.7(2.9)	**93.4**(**0.5**)
	$1 \sim 10$		397/3570	1292	**75.2**(**2.8**)	40.5(7.2)	**77.6**(**2.2**)
LETTER	$1 \sim 5$	16	113/452	171	85.2(1.3)	77.2(6.1)	**89.5**(**1.6**)
	$6 \sim 10$		110/440	178	81.0(1.7)	77.6(3.7)	**84.6**(**1.0**)
	$11 \sim 15$		111/445	177	81.1(2.7)	76.0(3.2)	**87.3**(**1.6**)
	$16 \sim 20$		110/440	184	81.3(1.8)	77.9(3.1)	**84.7**(**2.0**)
	$21 \sim 25$		117/468	167	86.8(2.7)	81.2(3.4)	**91.1**(**1.0**)
	$1 \sim 25$		22/528	167	11.9(1.7)	6.5(1.7)	**31.0**(**1.7**)
USPS	$1 \sim 5$	256	130/522	166	83.8(1.7)	76.5(5.3)	**89.5**(**1.3**)
	$6 \sim 10$		108/434	147	79.2(2.1)	67.6(4.3)	**85.5**(**2.4**)
	even #		108/434	166	79.6(2.7)	67.4(4.4)	**84.8**(**1.4**)
	odd #		111/445	147	82.7(1.9)	72.9(6.2)	**87.3**(**2.2**)
	$1 \sim 10$		54/488	147	43.7(2.6)	28.5(3.6)	**59.3**(**2.2**)

ordinarily while the other was labeled complementarily[5]. From the training dataset, we left out 25% of the data for validating hyperparameters based on the zero-one loss version of (4.46). Other details such as standardization, the model and optimization, and weight-decay candidates follow the previous experiments.

We compared three methods: the ordinary label (OL) method corresponding to $\alpha = 1$, the complementary label (CL) method corresponding to $\alpha = 0$, and the combination (OL & CL) method with $\alpha = 1/2$. The PC and sigmoid losses were commonly used for all methods.

We reported the means and standard deviations of the classification accuracy over 10 trials in Table 4.3. From the results, we can see that OL & CL tends to outperform OL and CL, demonstrating the usefulnesses of combining ordinarily and complementarily labeled data.

We also performed experiments that fix the total number of training data, but changed the proportion of ordinary and complementary labels. With the MNIST dataset, we tried varying proportions of ordinary and complementary labels from $\{0.0, 0.1, \ldots, 1.0\}$, where a smaller proportion means less complementary labels. The learning rate and weight decay were both fixed at $1e{-}4$ and we trained for 100 epochs. We used the pairwise comparison multi-class loss function with the binary sigmoid loss function, for both ordinary labels and complementary labels. We combined the two empirical risk shown in Eq.(4.45). We set α to be equivalent to the proportion of ordinary labels in the training dataset.

The mean accuracy and standard deviation for five trials are shown in Fig. 4.1. We can observe that when an ordinary label is given for all training data, it performs the best. As we transform more and more of them into a complementary label, the accuracy gradually decreases, which is intuitive since complementary labels are less informative.

4.3 Complementary-label Learning with Arbitrary Losses

So far, we have shown complementary-label learning but required strong restrictions on the loss functions, allowing only one-versus-all and pairwise comparison multi-class loss functions (Zhang, 2004a), with certain non-convex binary losses. This is a severe limitation since the softmax cross-entropy loss, which cannot be expressed by the two losses above, is the most popular loss in deep learning nowadays.

Recently, Yu et al. (2018) proposed a different formulation for complementary labels by employing the forward loss correction technique (Patrini et al., 2017) to

[5]We used $K - 1$ times more complementarily labeled data than ordinarily labeled data since a single ordinary label corresponds to $(K - 1)$ complementary labels.

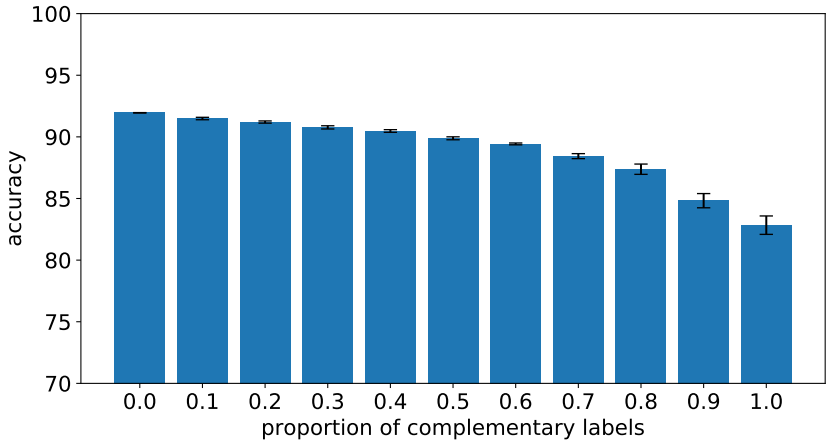


Figure 4.1: The mean accuracy and standard deviation for five trials. We performed the experiments for varying proportions of ordinary and complementary labels.

adjust the learning objective, but limiting the loss function to softmax cross-entropy loss. Their proposed risk estimator is not necessarily *unbiased* but the minimizer is theoretically guaranteed to be *consistent* with the minimizer of the risk for ordinary labels (under an implicit assumption on the model for convergence analysis). They also extended the problem setting to where complementary labels are chosen in an uneven (biased) way.

In this section, we first derive an unbiased risk estimator with a general loss function, making *any* loss functions available for use: not only the softmax cross-entropy loss function but other convex/non-convex loss functions can also be applied. We also do not have implicit assumptions on the classifier, allowing both linear and non-linear models.

Finally, our proposed unbiased risk estimator has an issue that the classification risk can attain negative values after learning, leading to overfitting. We further propose a non-negative correction to the original unbiased risk estimator to improve our estimator. The modified objective is no longer guaranteed to be an unbiased risk estimator, but the unbiased risk estimator can still be used for validation procedures for this modified learning objective. We experimentally show that our proposed method is comparable to or better than other methods.

4.3.1 Formulation

First, we introduce a few notations. We use $\eta_k(\boldsymbol{x}) = p(y = k|\boldsymbol{x})$ for the class-conditional probability for class k, $\boldsymbol{\eta}(\boldsymbol{x}) = [\eta_1(\boldsymbol{x}), \eta_2(\boldsymbol{x}), \ldots, \eta_K(\boldsymbol{x})]^\top$ for a vector of

class-conditional probabilities for each classes, and

$$\boldsymbol{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x})) = [\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), 1), \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), 2), \ldots, \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), K)]^\top \tag{4.47}$$

for a vector of losses for each classes. We define the class prior for class k as $\pi_k = p(y = k)$. We define similarly for the complementary label distribution, with $\overline{\eta}_k(\boldsymbol{x}) = \overline{p}(\overline{y} = k|\boldsymbol{x})$, $\overline{\boldsymbol{\eta}}(\boldsymbol{x}) = [\overline{\eta}_1(\boldsymbol{x}), \overline{\eta}_2(\boldsymbol{x}), \ldots, \overline{\eta}_K(\boldsymbol{x})]^\top$, and $\overline{\pi}_k = \overline{p}(\overline{y} = k)$. Two useful equivalent expressions of classification risk (4.1) used later are

$$R(\boldsymbol{g}) = \mathbb{E}_X[\boldsymbol{\eta}(x)^\top \boldsymbol{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}))] = \sum_{k=1}^{K} \pi_k \mathbb{E}_{p(\boldsymbol{x}|y=k)}\Big[\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), k)\Big], \tag{4.48}$$

Since the marginal distribution is equivalent for ordinary distribution and complementary distribution, we can rewrite our assumption in (4.5) as

$$\overline{\boldsymbol{\eta}}(\boldsymbol{x}) = \boldsymbol{T}\boldsymbol{\eta}(\boldsymbol{x}) \tag{4.49}$$

where $\boldsymbol{T} \in \mathbb{R}^{K \times K}$ is a matrix that takes 0 on diagonals and $\frac{1}{K-1}$ on non-diagonals.

Next, we describe our general unbiased risk formulation. We give the following theorem, which allows unbiased estimation of the classification risk from complementarily labeled patterns:

Theorem 4.7. *For any ordinary distribution $p(\boldsymbol{x}, y)$ and complementary distribution $\overline{p}(\boldsymbol{x}, \overline{y})$ related by (4.49) with decision function $\boldsymbol{g}$, and loss $\boldsymbol{\mathcal{L}}$, we have*

$$R(\boldsymbol{g}) = \mathbb{E}_{(\boldsymbol{x},\overline{y})\sim\overline{p}(\boldsymbol{x},\overline{y})}[\overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}), \overline{y})], \tag{4.50}$$

for the complementary loss

$$\overline{\boldsymbol{\mathcal{L}}}\big(\boldsymbol{g}(\boldsymbol{x})\big) := \Big(-(K-1)\boldsymbol{I}_K + \mathbf{1}\mathbf{1}^\top \Big) \cdot \boldsymbol{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x})\big), \tag{4.51}$$

or equivalently,

$$\overline{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}), k\big) = -(K-1) \cdot \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), k\big) + \sum_{j=1}^{K} \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), j\big), \tag{4.52}$$

where $\boldsymbol{I}_K$ is a $K \times K$ identity matrix and $\mathbf{1}$ is a K-dimensional column vector with 1 in each element.

Proof. First of all,

$$\begin{aligned} p(\boldsymbol{x}, \overline{y}) &= \frac{1}{K-1} \sum_{y \neq \overline{y}} p(\boldsymbol{x}, y) \\ &= \frac{1}{K-1} \Big(\sum_{y=1}^{K} p(\boldsymbol{x}, y) - p(\boldsymbol{x}, \overline{y}) \Big) \\ &= \frac{1}{K-1} \big(p(\boldsymbol{x}) - p(\boldsymbol{x}, \overline{y}) \big). \end{aligned} \tag{4.53}$$

The first equality holds since the marginal distribution is equivalent for $p(\boldsymbol{x}, y)$ and $\overline{p}(\boldsymbol{x}, y)$ and we assume (4.49). Consequently,

$$\begin{aligned} \overline{p}(\overline{y}|\boldsymbol{x}) &= \frac{\overline{p}(\boldsymbol{x}, \overline{y})}{p(\boldsymbol{x})} \\ &= \frac{1}{K-1} \cdot \Big(1 - \frac{p(x, \overline{y})}{p(\boldsymbol{x})} \Big) \\ &= \frac{1}{K-1} \cdot \big(1 - p(\overline{y}|\boldsymbol{x}) \big) \\ &= -\frac{1}{K-1} p(\overline{y}|\boldsymbol{x}) + \frac{1}{K-1}. \end{aligned} \tag{4.54}$$

More simply, we have $\boldsymbol{\eta}(\boldsymbol{x}) = -(K-1)\overline{\boldsymbol{\eta}}(\boldsymbol{x}) + \mathbf{1}$. Finally, we transform the classification risk,

$$\begin{aligned} R(\boldsymbol{g}) &= \mathbb{E}_{p(\boldsymbol{x},y)}[\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), y)] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}[\boldsymbol{\eta}^\top \boldsymbol{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}\big[\big(-(K-1)\overline{\boldsymbol{\eta}}^\top + \mathbf{1}^\top \big) \boldsymbol{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x})\big) \big] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}\big[-(K-1)\overline{\boldsymbol{\eta}}^\top \boldsymbol{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x})\big) + \mathbf{1}^\top \boldsymbol{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x})\big) \big] \\ &= \mathbb{E}_{(\boldsymbol{x},\overline{Y}) \sim \overline{\mathcal{D}}}\big[-(K-1) \cdot \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), \overline{y}\big) \big] + \mathbf{1}^\top \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}\big[\boldsymbol{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x})\big) \big] \\ &= \sum_{k=1}^{K} \overline{\pi}_k \cdot \mathbb{E}_{\boldsymbol{x} \sim \overline{p}(\boldsymbol{x}|k)} \Big[-(K-1) \cdot \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), k\big) + \mathbf{1}^\top \boldsymbol{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x})\big) \Big] \\ &= \overline{R}(\boldsymbol{g}) \end{aligned} \tag{4.55}$$

for the complementary loss, $\overline{\mathcal{L}}(k, \boldsymbol{g}) := -(K-1)\mathcal{L}(\boldsymbol{g}, k) + \mathbf{1}^\top \boldsymbol{\mathcal{L}}(\boldsymbol{g})$, which concludes the proof. □

It is worth noting that, in the above derivation, there are no constraints on the loss function and classifier. Thus, we can use any loss (convex/non-convex) and any model (linear/non-linear) for complementary learning.

Next, we show the relationship between our proposed framework and previous complementary-label learning in Section 4.2.

Corollary 4.8. *If one-versus-all loss (4.17) or pairwise comparison loss (4.18) is used with binary loss function that satisfy $\ell(z)+\ell(-z)=1$, the classification risk can be written as*

$$R(\boldsymbol{g}) = (K-1)\mathbb{E}_{\overline{p}(\boldsymbol{x},\overline{y})}\left[\overline{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}),\overline{y}\big)\right] - M_1 + M_2, \tag{4.56}$$

where M_1 and M_2 are non-negative constants that satisfy

$$\sum_{\overline{y}=1}^{K} \overline{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}),\overline{y}\big) = M_1 \tag{4.57}$$

for all $\boldsymbol{x}$ and

$$\overline{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}),\overline{y}\big) + \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}),\overline{y}\big) = M_2 \tag{4.58}$$

for all $\boldsymbol{x}$ and $\overline{y}$.

Proof.

$$\begin{aligned}
\overline{R}(\boldsymbol{g}) &= \mathbb{E}_{\overline{p}(\boldsymbol{x},\overline{y})}[\overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}),\overline{y})] \\
&= \mathbb{E}_{\overline{p}(\boldsymbol{x},\overline{y})}[-(K-1)\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}),\overline{y}) + \sum_{j=1}^{K}\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}),j)] \\
&= \mathbb{E}_{\overline{p}(\boldsymbol{x},\overline{y})}\big[-(K-1)[M_2 - \overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}),\overline{y})] + M_1\big] \\
&= (K-1)\mathbb{E}_{\overline{p}(\boldsymbol{x},\overline{y})}[\overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}),\overline{y})] + M_1 - (K-1)M_2 \\
&= (K-1)\mathbb{E}_{\overline{p}(\boldsymbol{x},\overline{y})}[\overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}),\overline{y})] - M_1 + M_2
\end{aligned} \tag{4.59}$$

The second equality holds because we use (4.52). The third equality holds because we are using losses that satisfy $\sum_j \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}),j) = M_1$ for all x and $\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}),\overline{y}) + \overline{\mathcal{L}}(\boldsymbol{g}(\boldsymbol{x}),\overline{y}) = M_2$ for all x and $\overline{y}$. The 4th equality rearranges terms. The 5th equality holds because $M_1 - (K-1)M_2 = -M_1 + M_2$ for $\overline{\mathcal{L}}_{\mathrm{OVA}}$ and $\overline{\mathcal{L}}_{\mathrm{PC}}$. This can be easily shown by using $M_1 = K$ and $M_2 = 2$ for $\overline{\mathcal{L}}_{\mathrm{OVA}}$, and $M_1 = K(K-1)/2$ and $M_2 = K-1$ for $\overline{\mathcal{L}}_{\mathrm{PC}}$. □

Since this is equivalent to the first two theorems in Section 4.2, this is a generalization of the unbiased complementary-label learning framework with symmetric conditions.

The key idea of the proof in Theorem 4.7 is to not rely on the condition that $\sum_{k=1}^{K}\overline{\mathcal{L}}\big(\boldsymbol{g}(\boldsymbol{x}),k\big)$ is a constant for all $\boldsymbol{x}$, used previously, which is inspired by the

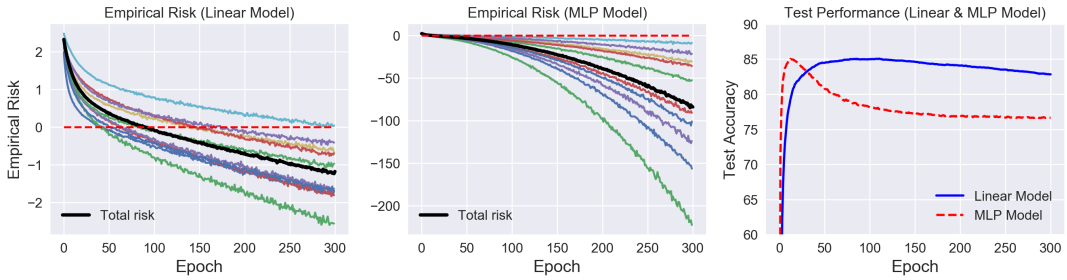


Figure 4.2: The left and middle graphs shows the total risk (4.61) (in black color) and the risk decomposed into each *ordinary* class term (4.62) (in other colors) for training data with linear and MLP models, respectively. The right graph shows the corresponding test accuracy for both models.

property of binary 0-1 loss ℓ_{01}, where $\ell_{01}(z)$ is 1 if $z < 0$ and 0 otherwise. Such a technique was also used when designing unbiased risk estimators for learning from positive and unlabeled data in a binary classification setup (du Plessis et al., 2014), but was later shown to be unnecessary (du Plessis et al., 2015). Note that Theorem 4.7 can be regarded as a special case of a framework proposed for learning from weak labels (Cid-Sueiro et al., 2014).

By using (4.52), the classification risk can be written as

$$R(\boldsymbol{g}) = \sum_{k=1}^{K} \overline{\pi}_k \mathbb{E}_{\overline{p}(\boldsymbol{x}|y=k)} \Big[-(K-1) \cdot \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), k\big) + \sum_{j=1}^{K} \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), j\big) \Big]. \tag{4.60}$$

Here, we rearrange our complementarily labeled dataset as $\{\mathcal{X}_k\}_{k=1}^{K}$, where $\mathcal{X}_k$ denotes the samples complementarily labeled as class k. Then, this expression of the classification risk can be approximated by,

$$\widehat{R}(\boldsymbol{g}) = \sum_{k=1}^{K} \frac{\widehat{\overline{\pi}}_k}{|\mathcal{X}_k|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_k} \Big[-(K-1) \cdot \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}_i), k\big) + \sum_{j=1}^{K} \mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}_i), j\big) \Big]. \tag{4.61}$$

4.3.2 Necessity of Risk Correction

The original expression of the classification risk (4.1) includes an expectation over non-negative loss $\mathcal{L} : [K] \times \mathbb{R}^K \to \mathbb{R}_+$, so the risk and its empirical approximator are both lower-bounded by zero. On the other hand, the expression (4.60) derived above contains a negative element. Although (4.60) is still non-negative by definition, due to the negative term, its empirical estimator can go negative, leading to overfitting.

We elaborate on this issue with an illustrative numerical example. In the left

graph of Figure 4.2, we show an example of training a linear model trained on the handwritten digits dataset MNIST[6], with complementary labels generated to satisfy (4.49). We used *Adam* (Kingma and Ba, 2015) for optimization with learning rate $5\mathrm{e}-5$, mini-batch size of 100, and weight decay of $1\mathrm{e}-4$ with 300 epochs. The empirical classification risk (4.61) is shown in black. We can see that the empirical classification risk continues decreasing and can go below zero at around 100 epochs. The test accuracy on the right graph hits the peak also at around epoch 100 and then the accuracy gradually deteriorates.

This issue stands out even more significantly when we use a flexible model. The middle graph shows the empirical classification risk for a multilayer perceptron (MLP) with one hidden layer (500 units), where *ReLU* (Nair and Hinton, 2010) was used as the activation function. The optimization setup was the same as the case of the linear model above. We can see the empirical risk decreasing much more quickly and going negative. Correspondingly, as the right graph shows, the test accuracy drops significantly after the empirical risk goes negative.

In fact, a similar issue is already implicit in the formulation with symmetric losses: According to Corollary 4.8, the unbiased risk estimator includes subtraction of a positive constant term which increases with respect to the number of classes. This means that the learning objective of symmetric losses has a (negative) lower bound.

4.3.3 Non-Negative Risk Estimator

As we saw in Section 4.3.2, our risk estimator can suffer from overfitting due to the negative issue. Here, we propose a correction to the risk estimator to overcome this problem.

Each term in the risk with ordinary labels (right-hand side of (4.48)), which corresponds to each class, is non-negative. We can reformulate (4.60) in order to show the counterpart for each non-negative term in the right-hand side of (4.48) for complementarily labeled data as

$$R(\boldsymbol{g};\ell)=\sum_{k=1}^{K}\Big[-(K-1)\overline{\pi}_k\cdot\mathbb{E}_{\overline{p}(\boldsymbol{x}|\overline{y}=k)}\big[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}),k\big)\big]+\sum_{j=1}^{K}\overline{\pi}_j\cdot\mathbb{E}_{\overline{p}(\boldsymbol{x}|\overline{y}=j)}\big[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}),k\big)\big]\Big]. \tag{4.62}$$

These counterparts (4.62) were originally non-negative when ordinary labels were used. In the left and middle graphs of Figure 4.2, we plot the decomposed risks

[6] See http://yann.lecun.com/exdb/mnist/.

with respect to each *ordinary* class (4.62) (shown in different colors). We can see that the decomposed risks for all classes become negative eventually. Based on this observation, our basic idea for correction is to enforce non-negativity for each ordinary class, with the expression based on complementary labels. More specifically, we propose a non-negative version by

$$\sum_{k=1}^{K} \max\left\{0, \left[-(K-1)\overline{\pi}_k \cdot \mathbb{E}_{\overline{p}(\boldsymbol{x}|y=k)}\left[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), k\big)\right] + \sum_{j=1}^{K} \overline{\pi}_j \cdot \mathbb{E}_{\overline{p}(\boldsymbol{x}|y=j)}\left[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}), k\big)\right]\right]\right\}. \tag{4.63}$$

(4.63) is equivalent to (4.62), since $\max\{0, a\} = a$ if a is non-negative. By using the datasets used for (4.61), this non-negative risk can be naïvely approximated by the sample average as

$$\sum_{k=1}^{K} \max\left\{0, \left[-(K-1) \cdot \frac{\overline{\pi}_k}{|\mathcal{X}_k|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_k} \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}_i), k) + \sum_{j=1}^{K} \frac{\overline{\pi}_j}{|\mathcal{X}_j|} \sum_{\boldsymbol{x}_{i'} \in \mathcal{X}_j} \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}_{i'}), k)\right]\right\}. \tag{4.64}$$

The empirical version of (4.62) may suffer from a negative objective, but (4.64) is non-negative (even though their population versions are equivalent.)

Enforcing the reformulated risk to become non-negative was previously explored in Kiryo et al. (2017), in the context of binary classification from positive and unlabeled data. The positive class risk is already bounded below by zero in their case (because they have true positive labels), so there was a max operator only on the negative class risk. We follow their footsteps, but since our setting is a multi-class scenario and also differs by not having *any* true labels, we put a max operator on each of the K classes.

4.3.4 Approximate Non-Negative Risk Estimator

Implementation with Max Operator

We now illustrate how to design a practical implementation under stochastic optimization for our non-negative risk estimator. An unfortunate issue is that the minimization of (4.64) is not point-wise due to the max-operator, thus cannot be used directly for stochastic optimization methods with mini-batch. However, an upper

Algorithm 1 Complementary-label learning with gradient ascent

Input: complementarily labeled training data $\{\mathcal{X}_k\}_{k=1}^K$, where $\mathcal{X}_k$ denotes the samples complementarily labeled as class k;
Output: model parameter θ for $g(x;\theta)$

Let $\mathcal{A}$ be an external SGD-like stochastic optimization algorithm such as Kingma and Ba (2015)
while no stopping criterion has been met:
  Shuffle $\{\mathcal{X}_j\}_j^K$ into B mini-batches;
  for $b = 1$ **to** B:
    Denote $\{\mathcal{X}_j^b\}$ as the b-th mini-batch for complementary class j
    Denote $r_k^b(\theta) = -(K-1)\overline{\pi}_k \cdot \widehat{\mathbb{E}}_{\overline{p}(\boldsymbol{x}|\overline{y}=k)}[\mathcal{L}(\boldsymbol{g},k);\mathcal{X}_k^b] + \sum_{j=1}^K \overline{\pi}_j \cdot \widehat{\mathbb{E}}_{\overline{p}(\boldsymbol{x}|y=j)}[\mathcal{L}(\boldsymbol{g},k);\mathcal{X}_j^b]$
    if $\min_k[r_1^b(\theta),\ldots,r_k^b(\theta),\ldots,r_K^b(\theta)] > -\beta$:
      Denote $L^b(\theta) = \sum_{k=1}^K r_k^b(\theta)$
      Set gradient $\nabla_\theta L^b(\theta)$;
      Update θ by $\mathcal{A}$ with its current step size η;
    else:
      Denote $\widetilde{L}^b(\theta) = \sum_{k=1}^K \min\{-\beta, r_k^b(\theta)\}$
      Set gradient $-\nabla_\theta \widetilde{L}^b(\theta)$;
      Update θ by $\mathcal{A}$ with a discounted step size $\gamma\eta$;

bound of the risk can be minimized in parallel by using mini-batch as the following,

$$\frac{1}{B}\sum_{b=1}^{N}\sum_{k=1}^{K}\max\Big\{0,-(K-1)\overline{\pi}_k\cdot\widehat{\mathbb{E}}_{\overline{P}_k}\big[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}),k\big);\mathcal{X}_k^b\big]+\sum_{j=1}^{K}\overline{\pi}_j\cdot\widehat{\mathbb{E}}_{\overline{P}_j}\big[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}),k\big);\mathcal{X}_j^b\big]\Big\}, \tag{4.65}$$

where $\widehat{\mathbb{E}}$ is the empirical version of the expectation and B is the number of mini-batches.

Implementation with Gradient Ascent

If the objective is negative for a certain mini-batch, the previous implementation based on the max operator will prevent the objective to further *decrease*. However, if the objective is already negative, that mini-batch has already started to overfit. The max operator cannot contribute to decrease the degree of overfitting. From this perspective, there is still room to improve the overfitting issue, and it would be preferable to *increase* itself to make this mini-batch less overfitted.

Our idea is the following. We denote the risk that corresponds to the kth ordinary

class for the bth mini-batch as

$$r_k^b(\theta) = -(K-1)\overline{\pi}_k \cdot \widehat{\mathbb{E}}_{\overline{p}(\boldsymbol{x}|\overline{y}=k)}\big[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}),k\big);\mathcal{X}_k^b\big] + \sum_{j=1}^{K} \overline{\pi}_j \cdot \widehat{\mathbb{E}}_{\overline{p}(\boldsymbol{x}|\overline{y}=j)}\big[\mathcal{L}\big(\boldsymbol{g}(\boldsymbol{x}),k\big);\mathcal{X}_j^b\big], \tag{4.66}$$

and the total risk as

$$L^b(\theta) = \sum_{k=1}^{K} r_k^b(\theta). \tag{4.67}$$

When $\min_k\{r_k^b(\theta)\}_{k=1}^K \geq -\beta$, we conduct gradient descent as usual with gradient $\nabla_\theta L^b(\theta)$. On the other hand, if $\min_k\{r_k^b(\theta)\}_{k=1}^K < -\beta$, we first squash the class-decomposed risks over $-\beta$ to $-\beta$ with a min operator, and then sum the results:

$$\widetilde{L}^b(\theta) = \sum_{k=1}^{K} \min\{-\beta, r_k^b(\theta)\}. \tag{4.68}$$

Next we set the gradient in the opposite direction with $-\nabla_\theta \widetilde{L}^b(\theta)$. Conceptually, we are going *up* the gradient $\nabla_\theta \widetilde{L}^b(\theta)$ for *only* the class-decomposed risks below $-\beta$, to avoid the class-decomposed risks that are already large to further increase. Note that β is a hyper-parameter that controls the tolerance of negativity. $\beta = 0$ would mean there is zero tolerance, but in practice we can also have $-\beta \neq 0$ for a threshold that allows some negative ($-\beta < 0$) or positive ($-\beta > 0$) amount. The procedure is shown in detail in Algorithm 1.

4.3.5 Experiments

We compare the 3 methods that we have proposed in Section 4.3, which are *Free* (Unbiased risk estimator that is loss assumption free, based on Eq. (4.61)), *Max Operator* (based on Eq. (4.65)), and *Gradient Ascent* (based on Alg. 1). For *Gradient Ascent*, we used $\beta = 0$ and $\gamma = 1$ for simplicity. Mini-batch size was set to 256. We also compare with two baseline methods: Pairwise comparison (PC) with ramp loss (from Section 4.2) and *Forward* correction from Yu et al. (2018). For training, we used only complementarily labeled data, which was generated so that the assumption of (4.49) is satisfied. This is straightforward when the dataset has a uniform (ordinarily-labeled) class prior, because it reduces to just choosing a class randomly other than the true class.

In Appendix B.5, we explain the details of the datasets used in the experiments: MNIST, Fashion-MNIST, Kuzushiji-MNIST, and CIFAR-10. The implementation is

based on PyTorch (Paszke et al., 2019) and our demo code is available online[7].

Comparison of All Epochs During Training

Setup For MNIST, Fashion-MNIST, and Kuzushiji-MNIST, a linear-in-input model with a bias term and a MLP model ($d - 500 - 1$) was trained with softmax cross-entropy loss function (except *PC*) for 300 epochs. Weight decay of $1e-4$ for weight parameters and learning rate of $5e-5$ for Adam (Kingma and Ba, 2015) was used.

For CIFAR-10, DenseNet (Huang et al., 2017) and ResNet-34 (He et al., 2016) were used with weight decay of $5e-4$ and initial learning rate of $1e-2$. For optimization, stochastic gradient descent was used with the momentum set to 0.9. Learning rate was halved every 30 epochs.

Results We show the accuracy for all 300 epochs on test data to demonstrate how the issues discussed in Section 4.3.2 appear and how different implementations in Section 4.3.4 are effective. In Figure 4.3, we show the mean and standard deviation of test accuracy for 4 trials on test data evaluated with ordinary labels.

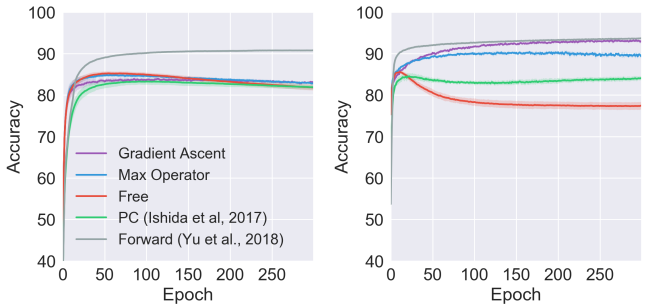
First we compare our 3 proposed methods with each other. For linear models in MNIST, Fashion-MNIST, and Kuzushiji-MNIST, all proposed methods work similarly. However in the case of using a more flexible MLP model or using DenseNet/ResNet in CIFAR-10, we can see that *Free* is the worst, *Max Operator* is better and *Gradient Ascent* is the best out of the proposed three methods for most of the epochs (*Free* < *Max Operator* < *Gradient Ascent*). These results are consistent with the discussions of overfitting in Section 4.3.2 and the motivations for different implementations in Section 4.3.4.

Next, we compare with baseline methods. For linear models, the forward method seem to work well. However for deep models (MLP, DenseNet, and ResNet), the superiority stands out for *Gradient Ascent* for many datasets, but in some cases, the forward method seems to be a good choice.

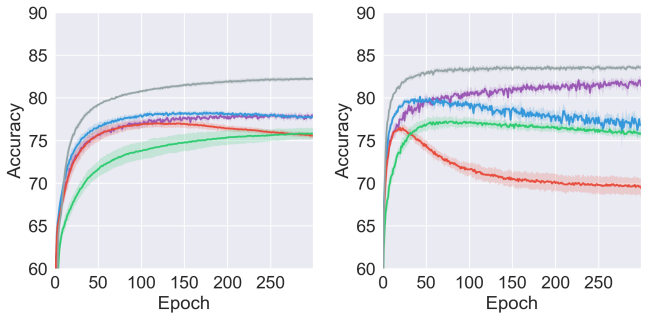
4.4 Conclusion

We proposed a novel problem setting and algorithm for multi-class classification from complementary labels. We first showed a formulation with symmetric losses. We also showed how our formulation can be combined with ordinary-label learning. We established an estimation error bound, and experimentally demonstrated the usefulness of our algorithm. Then, we showed a general formulation that can be

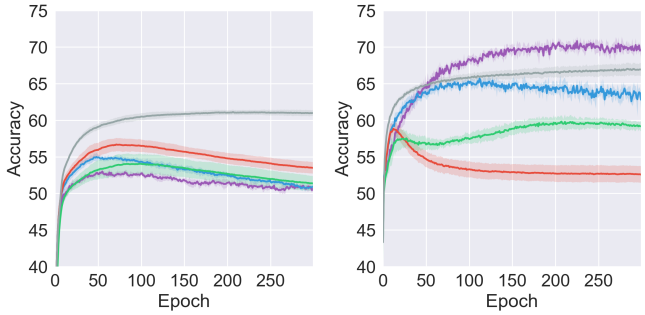
[7] https://github.com/takashiishida/comp



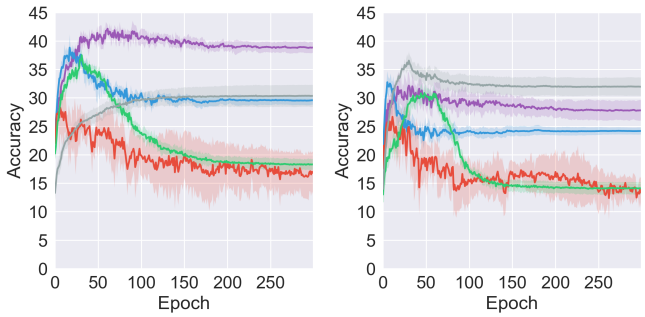
(a) MNIST, linear (b) MNIST, MLP



(c) Fashion MNIST, linear (d) Fashion MNIST, MLP



(e) Kuzushi MNIST, linear (f) Kuzushi MNIST, MLP



(g) CIFAR-10, DenseNet (h) CIFAR-10, ResNet

Figure 4.3: Experimental results for various datasets and models. Dark colors show the mean accuracy of 5 trials and light colors show standard deviation.

used with arbitrary losses. We further improved our risk estimator by a non-negative correction and a gradient ascent trick.

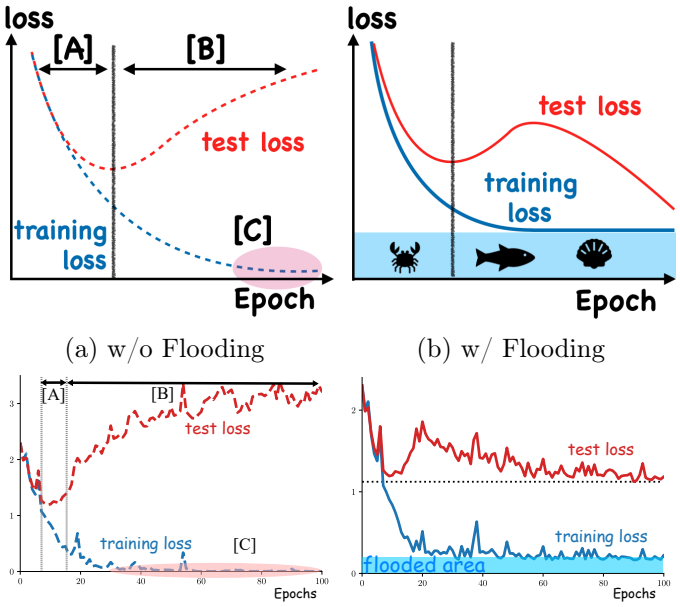
Chapter 5

Flooding: A Novel Regularizer to Avoid Overfitting

Overparameterized deep networks have the capacity to memorize training data with zero training error. Even after memorization, the training loss continues to approach zero, making the model overconfident and the test performance degraded. While existing regularizers indirectly try to cope with this issue we propose a direct solution called *flooding* that intentionally prevents further reduction of the training loss when it reaches a reasonably small value, which we call the *flood level*. Our approach makes the loss float around the flood level by doing mini-batched gradient descent as usual but gradient *ascent* if the training loss is below the flood level. This can be implemented with one line of code and is compatible with any stochastic optimizer and other regularizers. With flooding, the model will continue to "random walk" with the same non-zero training loss, and we expect it to go into an area with a flat loss landscape that leads to better generalization. We experimentally show that flooding improves performance and, as a byproduct, induces a double descent curve of the test loss.

5.1 Introduction

As we have discussed in Section 1.3, "overfitting" is one of the biggest interests and concerns in the machine learning community (Ng, 1997; Caruana et al., 2000; Belkin et al., 2018; Roelofs et al., 2019; Werpachowski et al., 2019). One way of identifying overfitting is to see whether the generalization gap, the test minus the training loss, is increasing or not (Goodfellow et al., 2016). We can further decompose the situation of the generalization gap increasing into two stages: The first stage is when both the



(a) w/o Flooding (b) w/ Flooding

(c) C10 w/o Flooding (d) C10 w/ Flooding

Figure 5.1: (a) shows 3 different concepts related to overfitting. [A] shows the generalization gap increases, while the training and test losses decrease. [B] also shows the increasing gap, but the test loss starts to rise. [C] shows the training loss becoming (near-)zero. We avoid [C] by *flooding* the bottom area, visualized in (b), which forces the training loss to stay around a constant. This leads to a decreasing test loss once again. We confirm these claims in experiments with CIFAR-10 shown in (c)–(d).

training and test losses are decreasing, but the training loss is decreasing faster than the test loss ([A] in Fig. 5.1a.) The next stage is when the training loss is decreasing, but the test loss is increasing.

Within stage [B], after learning for even more epochs, the training loss will continue to decrease and may become (near-)zero. This is shown as [C] in Fig. 5.1a. If we continue training even after the model has memorized (Zhang et al., 2017; Arpit et al., 2017; Belkin et al., 2018) the training data completely with zero error, the training loss can easily become (near-)zero especially with overparametrized models. Recent works on overparametrization and double descent curves (Belkin et al., 2019; Nakkiran et al., 2020) have shown that learning until zero training error is meaningful to achieve a lower generalization error. However, whether zero training *loss* is necessary after achieving zero training *error* remains an open issue.

In this chapter, we propose a method to make the training loss *float* around a small constant value, in order to prevent the training loss from approaching zero. This is analogous to *flooding* the bottom area with water, and we refer to the constant

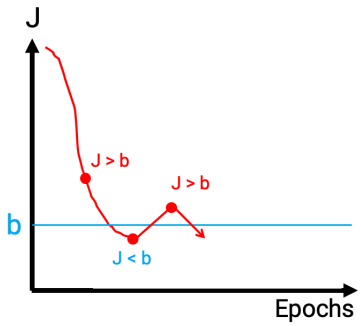


Figure 5.2: A visual explanation of how flooding repeats gradient decent and gradient ascent. b is the flooding level and J is the original learning objective. Gradient descent happens when $J > b$ but gradient ascent happens when $J < b$.

value as the *flood level*. Note that even if we add flooding, we can still memorize the training data. Our proposal only forces the training *loss* to become positive, which does not necessarily mean the training *error* will become positive, as long as the flood level is not too large. The idea of flooding is shown in Fig. 5.1b, and we show learning curves before and after flooding with a benchmark dataset in Fig. 5.1c and Fig. 5.1d. [1]

Algorithm and implementation Our algorithm of flooding is surprisingly simple. If the original learning objective is J, the proposed modified learning objective $\widetilde{J}$ with flooding is

$$\widetilde{J}(\boldsymbol{\theta}) = |J(\boldsymbol{\theta}) - b| + b, \tag{5.1}$$

where $b > 0$ is the flood level specified by the user, and $\boldsymbol{\theta}$ is the model parameter.[2]

The gradient of $\tilde{J}$ w.r.t. $\boldsymbol{\theta}$ will point in the same direction as that of $J(\boldsymbol{\theta})$ when $J(\boldsymbol{\theta}) > b$ but in the opposite direction when $J(\boldsymbol{\theta}) < b$. This means that when the learning objective is above the flood level, there is a "gravity" effect with gradient *descent*, but when the learning objective is below the flood level, there is a "buoyancy" effect with gradient *ascent* (see Fig. 5.2). In practice, this will be performed with a mini-batch and will be compatible with any stochastic optimizers. It can also be used along with other regularization methods.

During flooding, the training loss will repeat going below and above the flood level. The model will continue to "random walk" with the same non-zero training loss, and we expect it to drift into an area with a flat loss landscape that leads to

[1] Note that Figure 5.1c shows the learning curves for the first 80 epochs for CIFAR-10 and ResNet-18 (He et al., 2016) without flooding. Figure 5.1d shows the learning curves with flooding, when the flooding level is 0.18.

[2] Adding back b will not affect the gradient but will ensure $\widetilde{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta})$ when $J(\boldsymbol{\theta}) > b$.

better generalization (Hochreiter and Schmidhuber, 1997; Chaudhari et al., 2017; Keskar et al., 2017; Li et al., 2018). In experiments, we show that during this period of random walk, there is an increase in flatness of the loss function (See Section 5.4.3).

This modification can be incorporated into existing machine learning code easily: Add one line of code for Eq. (5.1) after evaluating the original objective function $J(\boldsymbol{\theta})$. A minimal working example with a mini-batch in PyTorch (Paszke et al., 2019) is demonstrated below to show the additional one line of code:

```
outputs = model(inputs)
loss = criterion(outputs, labels)
flood = (loss-b).abs()+b  # This is it!
optimizer.zero_grad()
flood.backward()
optimizer.step()
```

It may be hard to set the flood level without expert knowledge on the domain or task. We can circumvent this situation easily by treating the flood level as a hyper-parameter. We may exhaustively evaluate the accuracy for the predefined hyper-parameter candidates with a validation dataset, which can be performed in parallel.

Previous regularization methods Many previous regularization methods (see Section 1.3) also aim at avoiding *training too much* in various ways including restricting the parameter norm to become small by decaying the parameter weights (Hanson and Pratt, 1988), raising the difficulty of training by dropping activations of neural networks (Srivastava et al., 2014), avoiding the model to output a hard label by smoothing the training labels (Szegedy et al., 2016), and simply stopping training at an earlier phase (Morgan and Bourlard, 1990). These methods can be considered as indirect ways to control the training loss, by also introducing additional assumptions such as the optimal model parameters are close to zero. Although making the regularization effect stronger would make it harder for the training loss to approach zero, it is still hard to maintain the right level of training loss till the end of training. In fact, for overparametrized deep networks, applying small regularization would not stop the training loss becoming (near-)zero, making it even harder to choose a hyper-parameter that corresponds to a specific level of loss.

Flooding, on the other hand, is a direct solution to the issue that the training loss becomes (near-)zero. Flooding intentionally prevents further reduction of the training loss when it reaches a reasonably small value, and the flood level corresponds to the level of training loss that the user wants to keep.

Avoiding Over-Minimization of the Empirical Risk It is commonly observed that the empirical risk goes below zero, and it causes overfitting (Kiryo et al., 2017) in *weakly supervised learning* when an equivalent form of the risk expressed with the given weak supervision is alternatively used (Natarajan et al., 2013; Cid-Sueiro et al., 2014; du Plessis et al., 2014, 2015; Patrini et al., 2017; van Rooyen and Williamson, 2018). Kiryo et al. (2017) proposed a gradient ascent technique to keep the empirical risk non-negative. This idea has been generalized and applied to other weakly supervised settings (Han et al., 2020; Lu et al., 2020). We have also seen how a similar idea was useful in Section 4.3 for complementary-label learning.

Although flooding also places a lower bound on the empirical risk, the motivation is different: First, while Kiryo et al. (2017) and others aim to fix the negative empirical risk to become non-negative, our original empirical risk is already non-negative. Instead, we are aiming to sink the original empirical risk by modifying it with a *positive* lower bound. Second, the problem settings are different. Weakly supervised learning methods require certain loss corrections or sample corrections (Han et al., 2020) before the non-negative correction, but we work on the original empirical risk without any setting-specific modifications.

Early stopping (Morgan and Bourlard, 1990) may be a naive solution to this problem where the empirical risk becomes too small. However, performance of early stopping highly depends on the training dynamics and is sensitive to the randomness in the optimization method and mini-batch sampling. This suggests that early stopping at the optimal epoch in terms of a single training path does not necessarily perform well in another round of training. This makes it difficult to use hyper-parameter selection techniques such as cross-validation that requires re-training a model more than once. In our experiments, we will demonstrate how flooding performs even better than early stopping.

Double Descent Curves with Overparametrization Recently, there has been increasing attention on the phenomenon of "double descent," named by Belkin et al. (2019), to explain the two regimes of deep learning: The first one (underparametrized regime) occurs where the model complexity is small compared to the number of samples, and the test error as a function of model complexity decreases with low model complexity but starts to increase after the model complexity is large enough. This follows the classical view of machine learning that excessive complexity leads to poor generalization. The second one (overparametrized regime) occurs when an even larger model complexity is considered. Then increasing the complexity only decreases test error, which leads to a double descent shape. The phase of decreasing test error often occurs after the training error becomes zero. This follows the modern

view of machine learning that bigger models lead to better generalization.[3]

As far as we know, the discovery of double descent curves dates back to at least Krogh and Hertz (1991), where they theoretically showed the double descent phenomenon under a linear regression setup. Recent works (Belkin et al., 2019; Nakkiran et al., 2020) have shown empirically that a similar phenomenon can be observed with deep learning methods. Nakkiran et al. (2020) observed that the double descent curves for the *test error* can be shown not only as a function of model complexity, but also as a function of the epoch number.

To the best of our knowledge, the epoch-wise double descent curve was not observed for the *test loss* before but was observed in our experiments after using flooding with only about 100 epochs. Investigating the connection between epoch-wise double descent curves for the test loss and previous double descent curves (Krogh and Hertz, 1991; Belkin et al., 2019; Nakkiran et al., 2020) is out of the scope of this chapter but is an important future direction.

Organization This chapter is organized as the following. In Section 5.2, we explain our proposed method. In Section 5.3, we show experiments with synthetic and benchmark datasets. In Section 5.4, we investigate four properties of flooding. In Section 5.5, we conclude.

5.2 Flooding: How to Avoid Zero Training Loss

In this section, we propose our regularization method, *flooding*. Note that this section and the following sections only consider multi-class classification for simplicity.

5.2.1 Preliminaries

Consider input variable $\boldsymbol{x} \in \mathbb{R}^d$ and output variable $y \in [K] := \{1, \ldots, K\}$, where K is the number of classes. They follow an unknown joint probability distribution with density $p(\boldsymbol{x}, y)$. We denote the score function by $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^K$. For any test data point $\boldsymbol{x}_0$, our prediction of the output label will be given by $\widehat{y}_0 := \arg\max_{z \in [K]} g_z(\boldsymbol{x}_0)$, where $g_z(\cdot)$ is the z-th element of $\boldsymbol{g}(\cdot)$, and in case of a tie, arg max returns the smallest argument. Let $\mathcal{L} : \mathbb{R}^K \times [K] \to \mathbb{R}$ denote a loss function. $\mathcal{L}$ can be the *zero-one loss*,

$$\mathcal{L}_{01}(\boldsymbol{v}, z') := \begin{cases} 0 & \text{if} \quad \arg\max_{z \in \{1,\ldots,K\}} v_z = z', \\ 1 & \text{otherwise}, \end{cases} \tag{5.2}$$

[3] https://www.eff.org/ai/metrics

where $\boldsymbol{v} := (v_1, \ldots, v_K)^\top \in \mathbb{R}^K$, or a surrogate loss such as the softmax cross-entropy loss,

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{v}, z') := -\log \frac{\exp(v_{z'})}{\sum_{z \in [K]} \exp(v_z)}. \tag{5.3}$$

For a surrogate loss $\mathcal{L}$, we denote the *classification risk* by

$$R(\boldsymbol{g}) := \mathbb{E}_{p(\boldsymbol{x},y)}[\mathcal{L}(\boldsymbol{g}(\boldsymbol{x}), y)] \tag{5.4}$$

where $\mathbb{E}_{p(\boldsymbol{x},y)}[\cdot]$ is the expectation over $(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)$. We use $R_{01}(\boldsymbol{g})$ to denote Eq. (5.4) when $\mathcal{L} = \mathcal{L}_{01}$ and call it the *classification error*.

The goal of multi-class classification is to learn $\boldsymbol{g}$ that minimizes the classification error $R_{01}(\boldsymbol{g})$. In optimization, we consider the minimization of the risk with a almost surely differentiable surrogate loss $R(\boldsymbol{g})$ instead to make the problem more tractable. Furthermore, since $p(\boldsymbol{x}, y)$ is usually unknown and there is no way to exactly evaluate $R(\boldsymbol{g})$, we minimize its empirical version calculated from the training data instead:

$$\widehat{R}(\boldsymbol{g}) := \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{g}(\boldsymbol{x}_i), y_i), \tag{5.5}$$

where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ are i.i.d. sampled from $p(\boldsymbol{x}, y)$. We call $\widehat{R}$ the *empirical risk*.

We would like to clarify some of the undefined terms used so far in this chapter. The "train/test loss" is the empirical risk with respect to the surrogate loss function $\mathcal{L}$ over the training/test data, respectively. We refer to the "training/test error" as the empirical risk with respect to $\mathcal{L}_{01}$ over the training/test data, respectively (which is equal to one minus accuracy) (Zhang, 2004b). [4]

5.2.2 Algorithm

With flexible models, $\widehat{R}(\boldsymbol{g})$ w.r.t. a surrogate loss can easily become small if not zero, as we mentioned in Section 5.1; see [C] in Fig. 5.1a. We propose a method that "floods the bottom area and sinks the original empirical risk" as in Fig. 5.1b so that the empirical risk cannot go below the flood level. More technically, if we denote the flood level as b, our proposed training objective with flooding is a simple fix.

[4] Also see Guo et al. (2017) for a discussion of the empirical differences of loss and error with neural networks.

Definition 1. *The* flooded empirical risk *is defined as*[5]

$$\widetilde{R}(\boldsymbol{g}) = |\widehat{R}(\boldsymbol{g}) - b| + b. \tag{5.6}$$

Note that when $b = 0$, then $\widetilde{R}(\boldsymbol{g}) = \widehat{R}(\boldsymbol{g})$. The gradient of $\widetilde{R}(\boldsymbol{g})$ w.r.t. model parameters will point to the same direction as that of $\widehat{R}(\boldsymbol{g})$ when $\widehat{R}(\boldsymbol{g}) > b$ but in the opposite direction when $\widehat{R}(\boldsymbol{g}) < b$. This means that when the learning objective is above the flood level, we perform gradient *descent* as usual (gravity zone), but when the learning objective is below the flood level, we perform gradient *ascent* instead (buoyancy zone).

The issue is that in general, we seldom know the optimal flood level in advance. This issue can be mitigated by searching for the optimal flood level b^* with a hyper-parameter optimization technique. In practice, we can search for the optimal flood level by performing the exhaustive search in parallel.

5.2.3 Implementation

For large scale problems, we can employ mini-batched stochastic optimization for efficient computation. Suppose that we have M disjoint mini-batch splits. We denote the empirical risk (5.5) with respect to the m-th mini-batch by $\widehat{R}_m(\boldsymbol{g})$ for $m \in \{1, \ldots, M\}$. Our mini-batched optimization performs gradient descent updates in the direction of the gradient of $|\widehat{R}_{\mathrm{m}}(\boldsymbol{g}) - b| + b$. By the convexity of the absolute value function and Jensen's inequality, we have

$$\widetilde{R}(\boldsymbol{g}) \leq \frac{1}{M} \sum_{m=1}^{M} \left(|\widehat{R}_{\mathrm{m}}(\boldsymbol{g}) - b| + b \right). \tag{5.7}$$

This indicates that mini-batched optimization will simply minimize an upper bound of the full-batch case with $\widetilde{R}(\boldsymbol{g})$.

5.3 Does Flooding Generalize Better?

In this section, we show experimental results to demonstrate that adding flooding leads to better generalization. The implementation in this section and the next is based on PyTorch (Paszke et al., 2019) and demo code is available.[6] Experiments were carried out with NVIDIA GeForce GTX 1080 Ti, NVIDIA Quadro RTX 5000

[5] Strictly speaking, Eq. (5.1) is different from Eq. (5.6), since Eq. (5.1) can be a mini-batch version of Eq. (5.6).

[6] https://github.com/takashiishida/flooding

and Intel Xeon Gold 6142.

5.3.1 Synthetic Datasets

The aim of synthetic experiments is to study the behavior of flooding with a controlled setup.

Data We use three types of synthetic data: Two Gaussians, Sinusoid (Nakkiran et al., 2019), and Spiral (Sugiyama, 2015). Below we explain how these data were generated.

Two Gaussians Data: We used two 10-dimensional Gaussian distributions (one for each class) with covariance matrix identity and means $\mu_{\mathrm{P}} = [0, 0, \ldots, 0]^\top$ and $\mu_{\mathrm{N}} = [m, m, \ldots, m]^\top$, where $m = 1.0$. The training, validation, and test sample sizes are 100, 100, and 20000, respectively.

Sinusoid Data: The sinusoid data (Nakkiran et al., 2019) are generated as follows. We first draw input data points from the 2-dimensional standard Gaussian distribution, i.e., $\boldsymbol{x} \sim N(\mathbf{0}_2, \boldsymbol{I}_2)$, where $\mathbf{0}_2$ is the two-dimensional zero vector, and $\boldsymbol{I}_2$ 2×2 identity matrix. Then put class labels based on

$$y = \mathrm{sign}(\boldsymbol{x}^\top \boldsymbol{w} + \sin(\boldsymbol{x}^\top \boldsymbol{w'})), \tag{5.8}$$

where $\boldsymbol{w}$ and $\boldsymbol{w'}$ are any two 2-dimesional vectors such that $\boldsymbol{w} \perp \boldsymbol{w'}$. The training, validation, and test sample sizes are 100, 100, and 20000, respectively.

Spiral Data: The spiral data (Sugiyama, 2015) is two-dimensional synthetic data. Let

$$\theta_1^+ := 0, \theta_2^+, \ldots, \theta_{n^+}^+ := 4\pi \tag{5.9}$$

be equally spaced n^+ points in the interval $[0, 4\pi]$, and

$$\theta_1^- := 0, \theta_2^-, \ldots, \theta_{n^-}^- := 4\pi \tag{5.10}$$

be equally spaced n^- points in the interval $[0, 4\pi]$. Let positive and negative input data points be

$$\boldsymbol{x}_i^+ := \theta_i[\cos(\theta_i), \sin(\theta_i)]^\top + \tau\boldsymbol{\nu}_i^+, \tag{5.11}$$

$$\boldsymbol{x}_i^- := (\theta + \pi)[\cos(\theta), \sin(\theta)]^\top + \tau\boldsymbol{\nu}_i^- \tag{5.12}$$

for $i = 1, \ldots, n$, where τ controls the magnitude of the noise, $\boldsymbol{\nu}_i^+$ and $\boldsymbol{\nu}_i^-$ are i.i.d. distributed according to the two-dimensional standard normal distribution. Then,

we make data for classification by

$$\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} := \{(\boldsymbol{x}_i^+, +1)\}_{i^+=1}^{n^+} \cup \{(\boldsymbol{x}_i^-, -1)\}_{i^-=1}^{n^-}, \tag{5.13}$$

where $n := n^+ + n^-$. The training, validation, and test sample sizes are 100, 100, and 10000 per class respectively.

Settings

We perform three experiments with synthetic datasets. For the first experiment, we aim to investigate whether flooding contributes to better generalization. We use a five-hidden-layer feedforward neural network with 500 units in each hidden layer with the ReLU activation function (Nair and Hinton, 2010). We train the network for 500 epochs with the logistic loss and the Adam (Kingma and Ba, 2015) optimizer with 100 mini-batch size and learning rate of 0.001. The flood level is chosen from $b \in \{0, 0.01, 0.02, \ldots, 0.50\}$. We tried adding label noise to our dataset, by flipping 1% (Low), 5% (Middle), or 10% (High) of the labels randomly. This label noise is added to both the training and the test dataset, so it corresponds to varying the Bayes risk, i.e., the difficulty of classification. Note that the training with $b = 0$ is identical to the baseline method without flooding. We report the test accuracy of the flood level with the best validation accuracy. We first conduct experiments without early stopping, which means that the last epoch was chosen for all flood levels.

The second experiment aims to investigate the effect of mini-batch size. In this experiment, we use the same model except that we have one hidden layer, the flood level is chosen from $b \in \{0, 0.01, 0.02, \ldots, 0.35\}$, we train for 450 epochs, and we fix the noise rate to 5%. The mini-batch size is chosen from 1, 10, and 50. Early-stopping was not used.

The last experiment compares a different implementation of flooding. Instead of using the absolute operator in Eq. (5.6), we try the following version[7]:

$$\widetilde{R}_{\mathrm{sq}}(\boldsymbol{g}) = (\widehat{R}(\boldsymbol{g}) - b)^2 + b. \tag{5.14}$$

The mini-batch size is 100, and other settings are same as the second experiment.

Results

The average and standard deviation of the accuracy of each method over 10 trials are summarized in Table 5.1.

[7] We also tried $\widetilde{R}_{\max}(\boldsymbol{g}) = \max(\widehat{R}(\boldsymbol{g}), b) + b$ but did not seem to have any improvement over the baseline (w/o flooding) in preliminary experiments.

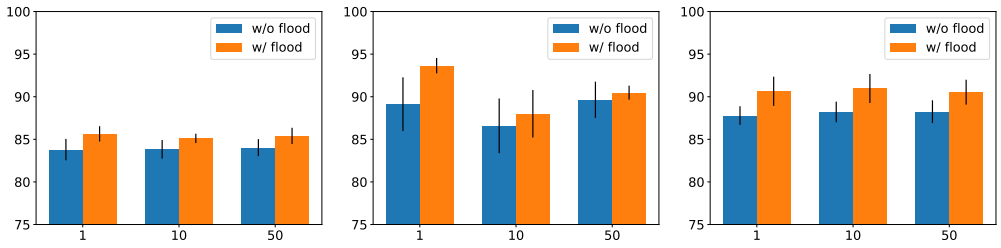
Table 5.1: Experimental results for the synthetic data. The average and standard deviation of the accuracy of each method over 10 trials. Sub-table (A) shows the results without early stopping. Sub-table (B) shows the results with early stopping. The **boldface** denotes the best and comparable method in terms of the average accuracy according to the t-test at the significance level 1%. The average and standard deviation of the chosen flood level is also shown.

(A) Without Early Stopping

Data	Label Noise	Without Flooding	With Flooding	Chosen b
Two Gaussians	Low	90.52%(0.71)	**92.13% (0.48)**	0.17 (0.10)
	Middle	84.79% (1.23)	**88.03% (1.00)**	0.22 (0.09)
	High	78.44% (0.92)	**83.59% (0.92)**	0.32 (0.08)
Spiral	Low	97.72% (0.26)	**98.72% (0.20)**	0.03 (0.01)
	Middle	89.94% (0.59)	**93.90% (0.90)**	0.12 (0.03)
	High	82.38% (1.80)	**87.64% (1.60)**	0.24 (0.05)
Sinusoid	Low	**94.62% (0.89)**	**94.66% (1.35)**	0.05 (0.04)
	Middle	87.85% (1.02)	**90.19% (1.41)**	0.11 (0.07)
	High	78.47% (2.39)	**85.78% (1.34)**	0.25 (0.08)

(B) With Early Stopping

Without Flooding	With Flooding	Chosen b
90.41% (0.98)	**92.13% (0.52)**	0.16 (0.12)
85.85% (2.07)	**88.15% (0.72)**	0.23 (0.08)
81.09% (2.23)	**83.87% (0.65)**	0.32 (0.11)
97.72% (0.68)	**98.26% (0.50)**	0.02 (0.01)
91.37% (1.43)	**93.51% (0.84)**	0.10 (0.04)
84.48% (1.52)	**88.01% (0.82)**	0.22 (0.06)
94.52% (0.85)	**95.42% (1.13)**	0.03 (0.03)
90.56% (1.46)	**91.16% (1.38)**	0.13 (0.07)
83.88% (1.49)	**85.10% (1.34)**	0.22 (0.10)



(a) Two Gaussians (b) Spiral (c) Sinusoid

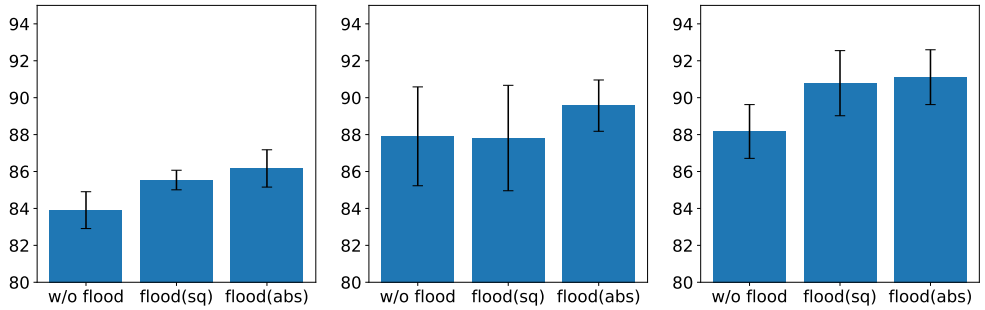
Figure 5.3: Comparison of flooding with different mini-batch sizes. We report the mean accuracy and standard deviation over four trials with different mini-batch sizes (1, 10, and 50) and datasets (Two Gaussians, Spiral, and Sinusoid).

From (A) in Table 5.1, we can see that the method with flooding often improves test accuracy over the baseline method without flooding. As we mentioned in the introduction, it can be harmful to keep training a model until the end without flooding. However, with flooding, the model at the final epoch has good prediction performance according to the results, which implies that flooding helps the late-stage training improve test accuracy.

We also conducted experiments with early stopping, meaning that we chose the model that recorded the best validation accuracy during training. The results are reported in sub-table (B) of Table 5.1. Compared with sub-table (A), we see that early stopping improves the baseline method without flooding in many cases. This indicates that training longer without flooding was harmful in our experiments. On the other hand, the accuracy for flooding combined with early stopping is often close to that with early stopping, meaning that training until the end with flooding tends to be already as good as doing so with early stopping. The table shows that flooding often improves or retains the test accuracy of the baseline method without flooding even after deploying early stopping. Flooding does not hurt performance but can be beneficial for methods used with early stopping.

It is worth noting that the average of the chosen flood level b is larger for a larger label noise. Since the increase of the label noise is expected to increase the Bayes risk, the results imply a positive correlation for the Bayes risk and the optimal flood level.

The experiments with different mini-batch sizes are shown in Figure 5.3. In Section 4.3.4, we discussed how a mini-batch case is an upper bound of the full-batch case with the flooded empirical risk $\tilde{R}(\boldsymbol{g})$. However, we can see that practically this is not an issue, since in Figure 5.3, we do not see a trend where smaller mini-batch size lead to worse performance. The experiments when the mini-batch size is



(a) Two Gaussians (b) Spiral (c) Sinusoid

Figure 5.4: Comparison of different implementations for flooding. We report the mean accuracy and standard deviation of five trials with different datasets (Two Gaussians, Spiral, and Sinusoid). We compare three methods: Baseline (without flooding), flooding with (5.14) (sq), and flooding with (5.6) (abs).

1 corresponds to the case when we add flooding in a sample-wise fashion. Except for Sinusoid, we did not observe a positive effect by adding flooding in a sample-wise fashion.

Finally, the results of comparing flooding with the squared implementation is shown in Figure 5.4. The squared implementation performs better than the baseline (w/o flooding) in Two Gaussians and Sinusoid, but performs the same in Spiral. The absolute implementation performs the best compared with the baseline (w/o flooding) and the squared implementation.[8] To conclude, we recommend using the absolute implementation rather than the squared implementation in practice. The experiments in the rest of this section will use the absolute implementation for flooding.

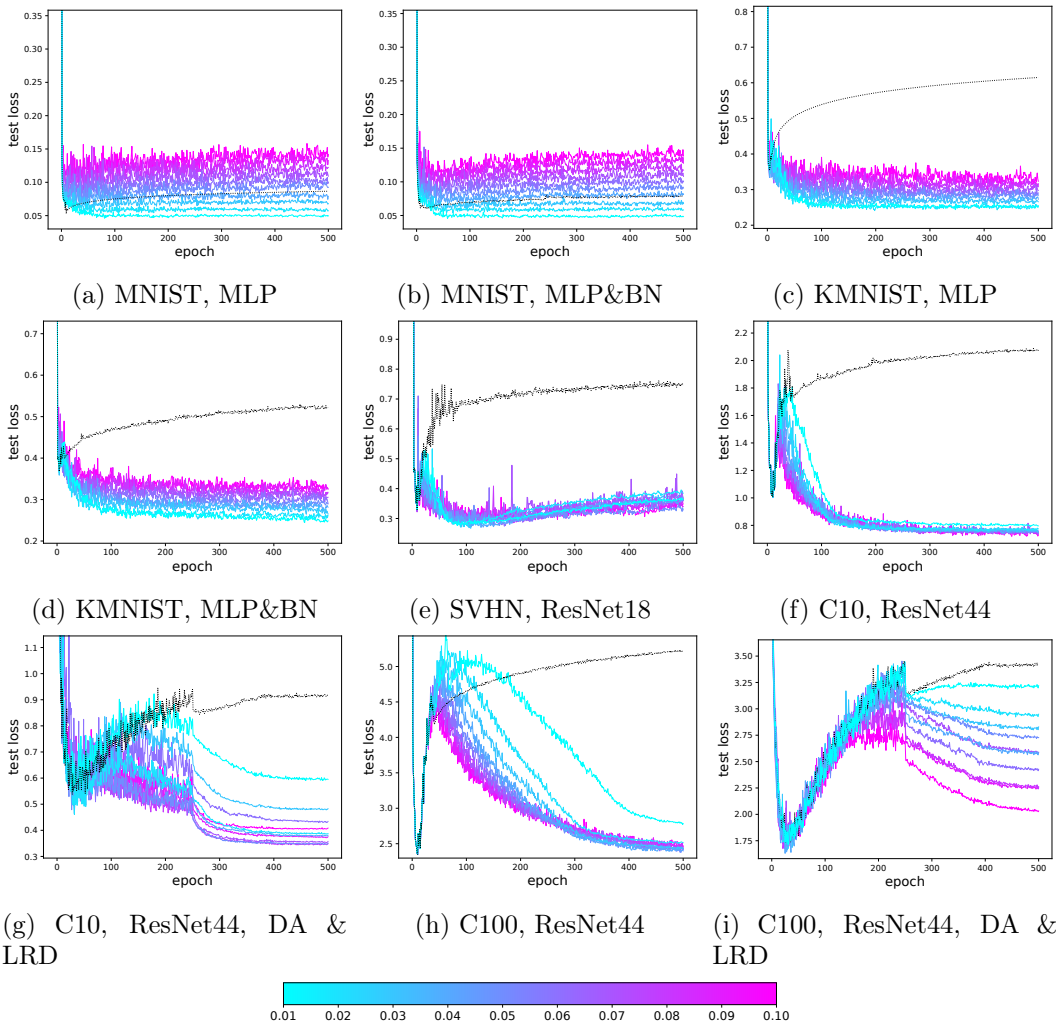
5.3.2 Benchmark Experiments

We next perform experiments with benchmark datasets. Not only do we compare with the baseline without flooding, we also compare or combine with other general regularization methods, which are early stopping, weight decay, batch normalization, data augmentation, and learning rate decay.

Settings

We use the following benchmark datasets: MNIST, Kuzushiji-MNIST, SVHN, CIFAR-10, and CIFAR-100. The details of the benchmark datasets can be found in Ap-

[8] We also compared with a max implementation with $\max(\widehat{R}(\boldsymbol{g}), b) + b$, but did not perform better than the baseline in our preliminary experiments.



(a) MNIST, MLP (b) MNIST, MLP&BN (c) KMNIST, MLP

(d) KMNIST, MLP&BN (e) SVHN, ResNet18 (f) C10, ResNet44

(g) C10, ResNet44, DA & LRD (h) C100, ResNet44 (i) C100, ResNet44, DA & LRD

Figure 5.5: Learning curves of the test loss showing that adding flooding leads to lower test loss. The black dotted line shows the baseline without flooding. The colored lines show the learning curves with flooding for different flooding levels. We show the learning curves for $b \in \{0.01, 0.02, \ldots, 0.10\}$. KMNIST is Kuzushiji-MNIST, C10 is CIFAR-10, and C100 is CIFAR-100. MLP, BN, DA, and LRD stand for multi-layer perceptron, batch normalization, data augmentation and learning rate decay, respectively.

pendix C.1. Stochastic gradient descent (Robbins and Monro, 1951) is used with learning rate of 0.1 and momentum of 0.9 for 500 epochs. For MNIST and Kuzushiji-MNIST, we use a two-hidden-layer feedforward neural network with 1000 units and the ReLU activation function (Nair and Hinton, 2010). We also compared adding batch normalization (Ioffe and Szegedy, 2015). For SVHN, we used ResNet-18 from He et al. (2016) with the implementation provided in Paszke et al. (2019). For MNIST, Kuzushiji-MNIST, and SVHN, we compared adding weight decay with rate 1×10^{-5}. For CIFAR-10 and CIFAR-100, we used ResNet-44 from He et al. (2016) with the implementation provided in Idelbayev (2020). For CIFAR-10 and CIFAR-100, we compared adding simple data augmentation (random crop and horizontal flip) and learning rate decay (multiply by 0.1 after 250 and 400 epochs). We split the original training dataset into training and validation data with with a proportion of 80 : 20 except for when we used data augmentation, we used 85 : 15. We perform the exhaustive hyper-parameter search for the flood level with candidates from $\{0.00, 0.01, \ldots, 0.10\}$. We used the original labels and did not add label noise. We deployed early stopping in the same way as in Section 5.3.1.

Results

We show the results in Table 5.2. For each dataset, the best performing setup and combination of regularizers always use flooding. Combining flooding with early stopping, weight decay, data augmentation, batch normalization, and/or learning rate decay usually has complementary effects. As a byproduct, we were able to produce a double descent curve for the test loss with a relatively few number of epochs, shown in Fig. 5.5.

5.4 *Why* Does Flooding Generalize Better?

In this section, we investigate four key properties of flooding to seek potential reasons for better generalization.

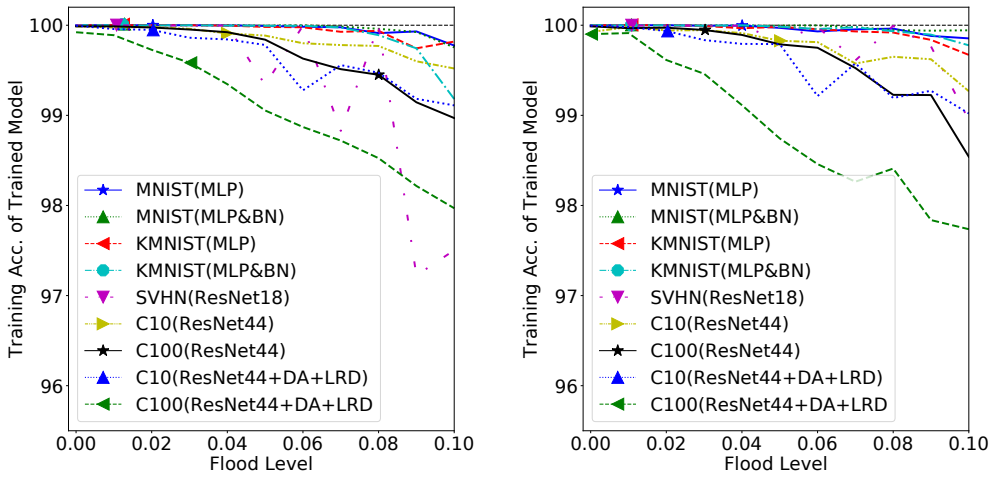
5.4.1 Memorization

Can we maintain memorization (Zhang et al., 2017; Arpit et al., 2017; Belkin et al., 2018) even after adding flooding? We investigate if the trained model has zero training error (100% accuracy) for the flood level that was chosen with validation data. The results are shown in Fig. 5.6.

All datasets and settings show downward curves, implying that the model will give up eventually on memorizing all training data as the flood level becomes higher. A

Table 5.2: Benchmark datasets. Reporting accuracy for all combinations of early stopping and flooding. We compare "w/o flood" and "w/ flood" and the better one is shown in **boldface**. The best setup for each dataset is shown with **underline**. "–" means that flood level of zero was optimal. "LRD" stands for learning rate decay and "DA" stands for data augmentation. C10 is CIFAR-10 and C100 is CIFAR-100.

		w/o early stopping		w/ early stopping	
Dataset	Model & Setup	w/o flood	w/ flood	w/o flood	w/ flood
MNIST	MLP	98.45%	**98.76%**	98.48%	**98.66%**
	MLP w/ weight decay	98.53%	**98.58%**	98.51%	**98.64%**
	MLP w/ batch norm	98.60%	**98.72%**	**98.66%**	98.65%
Kuzushiji	MLP	92.27%	**93.15%**	92.24%	**92.90%**
	MLP w/ weight decay	92.21%	**92.53%**	92.24%	**93.15%**
	MLP w/ batch norm	92.98%	**93.80%**	92.81%	**93.74%**
SVHN	ResNet18	92.38%	**92.78%**	92.41%	**92.79%**
	ResNet18 w/ weight decay	93.20%	–	92.99%	**93.42%**
C10	ResNet44	**75.38%**	75.31%	74.98%	**75.52%**
	ResNet44 w/ DA & LRD	88.05%	**89.61%**	88.06%	**89.48%**
C100	ResNet44	**46.00%**	45.83%	**46.87%**	46.73%
	ResNet44 w/ DA & LRD	63.38%	**63.70%**	63.24%	–



(a) w/o early stopping (b) w/ early stopping

Figure 5.6: Vertical axis is the training accuracy and the horizontal axis is the flood level. Marks are placed on the flood level that was chosen based on validation accuracy.

more interesting observation is the position of the optimal flood level (the one chosen by validation accuracy which is marked with $\star$, $\triangle$, $\triangleleft$, $\circ$, $\triangledown$ or $\triangleright$). We can observe that the marks are often plotted at zero error. These results are consistent with recent empirical works that imply zero training error leads to lower generalization error (Belkin et al., 2019; Nakkiran et al., 2020), but we further demonstrate that zero training loss may be harmful under zero training error.

5.4.2 Performance and Gradients

We visualize the relationship between test performance (loss or accuracy) and gradient amplitude of the training/test loss in Fig. 5.7, where the gradient amplitude is the ℓ_2 norm of the *filter-normalized gradient* of the loss. The filter-normalized gradient is the gradient appropriately scaled depending on the magnitude of the corresponding convolutional filter, similarly to Li et al. (2018).

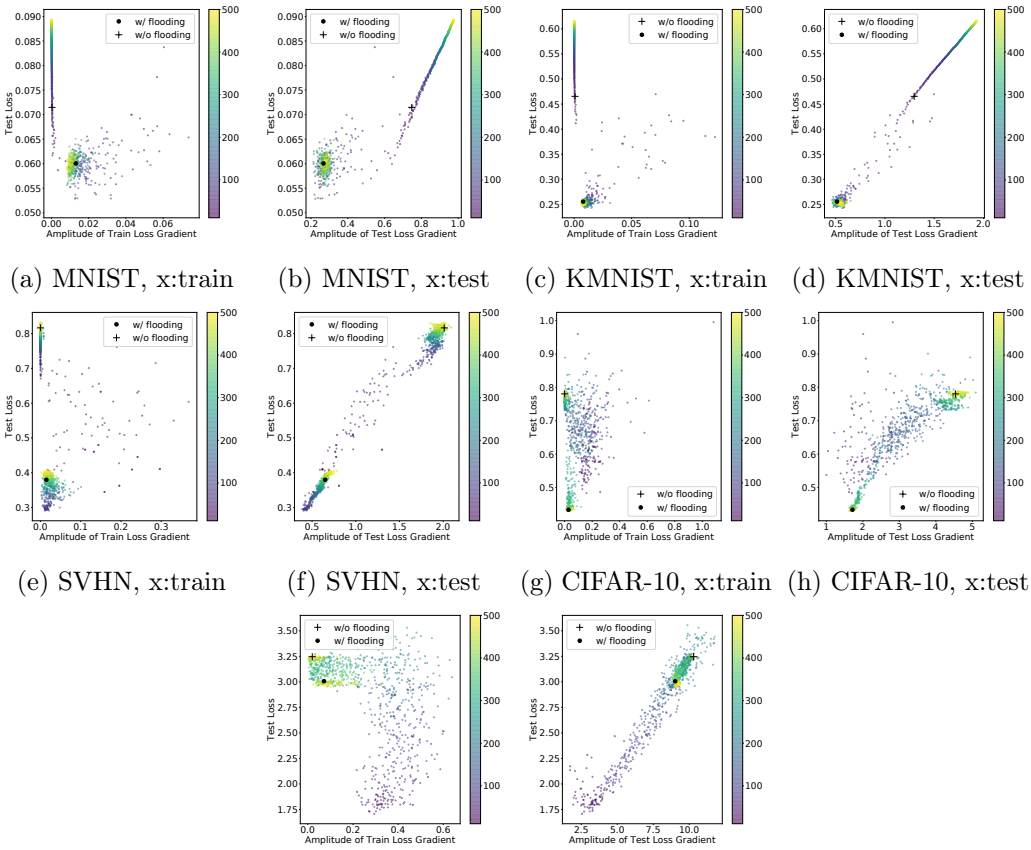
More specifically, for each filter of every convolutional layer, we multiply the corresponding elements of the gradient by the norm of the filter. Note that a fully connected layer is a special case of convolutional layer and is also subject to this scaling.

For the sub-figures with gradient amplitude of training loss on the horizontal axis, "$\circ$" marks (w/ flooding) are often plotted on the right of "$+$" marks (w/o flooding), which implies that flooding prevents the model from staying at a local minimum. For the sub-figures with gradient amplitude of test loss on the horizontal axis, the method with flooding ("$\circ$") improves performance while the gradient amplitude becomes smaller.

5.4.3 Flatness

We give a one-dimensional visualization of flatness (Li et al., 2018). We compare the flatness of the model right after the empirical risk with respect to a mini-batch becomes smaller than the flood level, $\widehat{R}_m(\boldsymbol{g}) < b$, for the first time (dotted blue) and the model after training (solid blue). We also compare them with the model trained by the baseline method without flooding, and training is finished (solid red).

In Fig. 5.8, the test loss becomes lower and flatter during the training with flooding. Note that the training loss, on the other hand, continues to float around the flood level until the end of training after it enters the flooding zone. We expect that the model makes a random walk and escapes regions with sharp loss landscapes during the period. This may be a possible reason for better generalization results with our proposed method.

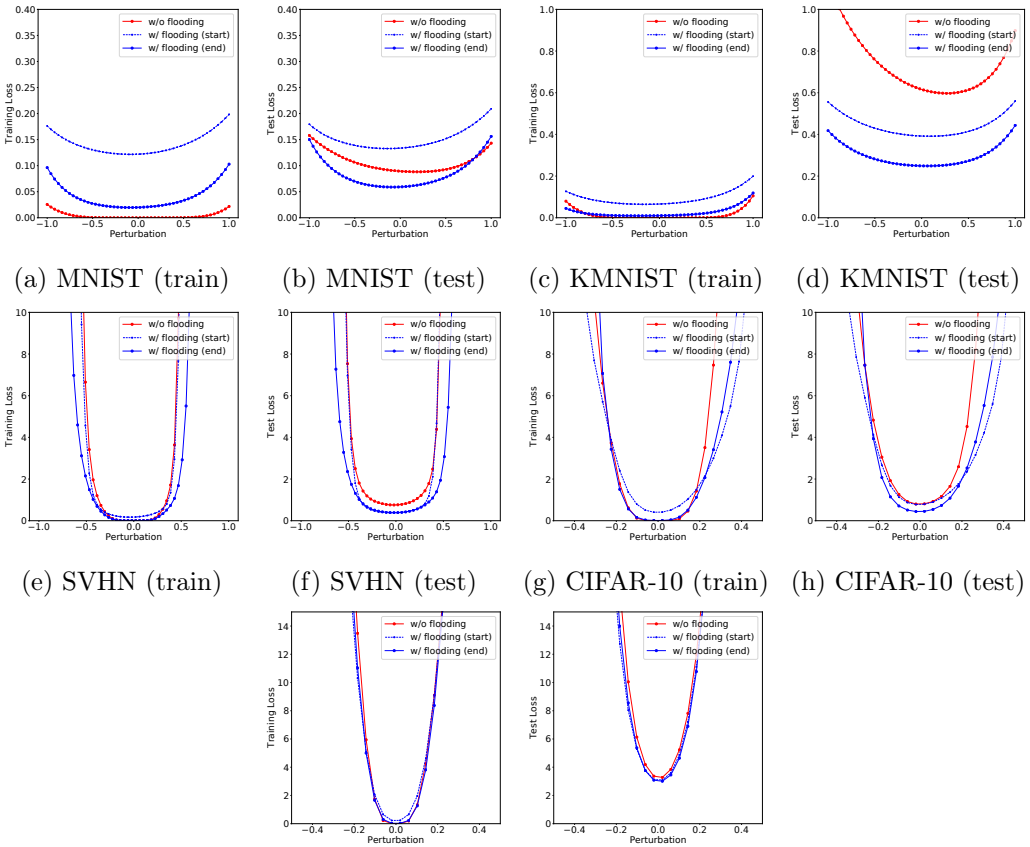


(a) MNIST, x:train (b) MNIST, x:test (c) KMNIST, x:train (d) KMNIST, x:test

(e) SVHN, x:train (f) SVHN, x:test (g) CIFAR-10, x:train (h) CIFAR-10, x:test

(i) CIFAR-100, x:train (j) CIFAR-100, x:test

Figure 5.7: Relationship between test loss and amplitude of gradient (with training or test loss). Each point ("∘" or "+") in the figures corresponds to a single model at a certain epoch. We remove the first 10 epochs and plot the rest. "∘" is used for the method with flooding and "+" is used for the method without flooding. The large black "∘" and "+" show the epochs with early stopping. The color becomes lighter (purple → yellow) as the training proceeds.



(a) MNIST (train) (b) MNIST (test) (c) KMNIST (train) (d) KMNIST (test)

(e) SVHN (train) (f) SVHN (test) (g) CIFAR-10 (train) (h) CIFAR-10 (test)

(i) CIFAR-100 (train) (j) CIFAR-100 (test)

Figure 5.8: One-dimensional visualization of flatness. We visualize the training/test loss with respect to perturbation. We depict the results for 3 models: the model when the empirical risk with respect to training data is below the flooding level for the first time during training (dotted blue), the model at the end of training with flooding (solid blue), and the model at the end of training without flooding (solid red).

5.4.4 Theoretical Insight

We show that the mean squared error (MSE) of the flooded risk estimator is smaller than that of the original risk estimator without flooding, with the condition that the flood level is between the original training loss and the test loss, in the following theorem.

Theorem 5.1. *Fix any measurable vector-valued function* $\boldsymbol{g}$*. If the flood level* b *satisfies* $b \leq R(\boldsymbol{g})$*, we have*

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) \geq \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})). \tag{5.15}$$

If b *further satisfies* $b < R(\boldsymbol{g})$ *and* $\Pr[\widehat{R}(\boldsymbol{g}) < b] > 0$*, the inequality will be strict:*

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) > \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})). \tag{5.16}$$

See Appendix C.2 for formal discussions and the proof.

5.5 Conclusion

We proposed a novel regularization method called *flooding* that keeps the training loss to stay around a small constant value, to avoid zero training loss. In our experiments, the optimal flood level often maintained memorization of training data, with zero error. With flooding, our experiments confirmed that the test accuracy improves for various synthetic and benchmark datasets, and we theoretically showed that the MSE will be reduced under certain conditions.

As a byproduct, we were able to produce a double descent curve for the test loss when flooding was used. An important future direction is to study the relationship between this and the double descent curves from previous works (Krogh and Hertz, 1991; Belkin et al., 2019; Nakkiran et al., 2020).

Chapter 6

Conclusions and Future Work

In this chapter, we conclude by summarizing our main contributions and discuss future directions.

6.1 Conclusions

In this dissertation, we presented reliable machine learning algorithms for weak supervision and limited data. We summarize our main contributions as follows.

- In Chapter 3, we introduced our novel problem setting of learning a binary classifier from only positive data, without any negative data or unlabeled data. We showed that if we can equip positive data with confidence, we can successfully learn a binary classifier with the optimal convergence rate. The key technique was to reformulate the classification risk into a formulation that only required the positive-confidence data, from a formulation that originally required both positive and negative data. This led to a simple empirical risk minimization framework that is model-, optimization-, and loss-independent. We showed the consistency and an estimation error bound. Our experimental results demonstrated the effectiveness of our method.

- In Chapter 4, we introduced another type of weak supervision called complementary labels, which is helpful for multi-class classification. A complementary label specifies a class that a pattern does not belong to. We showed that an unbiased estimator to the classification risk can be obtained only from complementarily labeled data, if a loss function satisfies a particular symmetric condition. We then derived estimation error bounds for the proposed method and proved that the optimal convergence rate is achieved. We also showed that learning from complementary labels can be easily combined with learning from

ordinary labels, i.e., ordinary supervised learning. We showed experimental results to confirm the usefulness of our approach. We further extend the unbiased risk estimator to arbitrary losses and models, and improve it by a non-negative correction and a gradient ascent trick. We showed experiments with benchmark datasets to demonstrate the usefulness of this extension.

- In Chapter 5, we introduced a novel regularizer that can be used to avoid overfitting. We proposed a regularizer called flooding that intentionally prevents further reduction of the training loss when it reaches a reasonably small value, which we call the flood level. Our approach makes the flooded empirical risk float around the flood level by performing mini-batched gradient descent as usual but gradient ascent if the empirical risk is below the flood level. With flooding, the model will continue performing a "random walk" with the same non-zero empirical risk, and we expect it to drift into an area with a flat loss landscape that leads to better generalization. We experimentally showed that flooding improves the generalization performance and as a byproduct, induces a double descent curve of the test loss.

A recurring theme throughout this dissertation was *risk modification*. We either took a *risk rewriting* approach or a *risk correction* approach. In Pconf classification and complementary-label classification, we rewrote the risk in order to utilize weakly supervised data. When we extended the complementary-label framework to be compatible with any loss functions, we encountered a situation where the empirical risk becomes negative. We corrected the risk to become non-negative, and showed that this was an effective way to avoid performance deterioration. We used a similar idea in flooding—we avoided minimizing the empirical risk too much, by directly correcting the risk to stay around the flood level.

Many machine learning algorithms have different settings, components, and assumptions, but one of the few factors in common is that they often have a target empirical risk, or more generally, an objective function, that they want to minimize (or maximize). The advantage of the risk modification approach is that instead of focusing on specific settings or components, we directly deal with the empirical risk. This leads to an algorithm that will usually become compatible with a wide variety of models, loss functions, and optimizers, and potentially can be used for various domains and tasks.

6.2 Short Term Future Work

We discuss future work of this dissertation.

6.2.1 Future Work for Positive-Confidence Learning

What if we have access to negative data in addition to positive-confidence data? In order to learn from positive-confidence and negative data, we can rewrite the classification risk as follows:

$$\begin{aligned}
R(g) &= \mathbb{E}_{p(\boldsymbol{x},y)}[\ell(yg(\boldsymbol{x})] \\
&= \pi_+\mathbb{E}_+[\ell(g(\boldsymbol{x}))] + \pi_-\mathbb{E}_-[\ell(-g(\boldsymbol{x}))] \\
&= \pi_+\mathbb{E}_+[\ell(g(\boldsymbol{x}))] + \lambda\pi_-\mathbb{E}_-[\ell(-g(\boldsymbol{x}))] + (1-\lambda)\pi_-\mathbb{E}_-[\ell(-g(\boldsymbol{x}))] \\
&= \pi_+\mathbb{E}_+[\ell(g(\boldsymbol{x}))] + \lambda\pi_-\mathbb{E}_-[\ell(-g(\boldsymbol{x}))] + (1-\lambda)\pi_+\mathbb{E}_+\left[\frac{1-r(\boldsymbol{x})}{r(\boldsymbol{x})}\ell(-g(\boldsymbol{x}))\right] \\
&= \pi_+\mathbb{E}_+\left[\ell(g(\boldsymbol{x})) + (1-\lambda)\frac{1-r(\boldsymbol{x})}{r(\boldsymbol{x})}\ell(-g(\boldsymbol{x}))\right] + \lambda\pi_-\mathbb{E}_-\left[\ell(-g(\boldsymbol{x}))\right], \qquad (6.1)
\end{aligned}$$

where $\lambda \in [0,1]$ is a hyper-parameter. $\lambda = 0$ corresponds to Pconf classification and $\lambda = 1$ corresponds to ordinary classification. We can learn a classifier by minimizing the empirical version of the above.

When we are collecting confidence for positive-confidence learning, we may encounter positive-confidence values of zero. Although by assumption this is impossible (since we are giving confidence to *positive* samples), there may be errors in data collection procedures and negative samples may be included in the training dataset. In this situation, the above formulation of positive-confidence and negative data can become handy.

6.2.2 Future Work for Complementary-Label Learning

In this dissertation, we focused on the risk modification approach. For future work, we can consider other approaches to further enhance the performance of complementary-label learning. Inspired by ideas from semi-supervised learning, we can consider adding a regularization term to encourage similar examples to have the same prediction. For example, we can think of propagating complementary labels to nearby samples: if a sample is *not* from class k, then a nearby sample also shouldn't be from class k.

It would also be interesting to explore the value of using complementary labels for (weakly-)supervised pretraining (Donahue et al., 2014; Mahajan et al., 2018) or in the fine-tuning stage after self-supervised pretraining (He et al., 2020; Chen et al., 2020).

6.2.3 Future Work for Flooding

In this subsection, we discuss future works for flooding.

Theoretical and Empirical Analysis for Flooding

In Chapter 5, we discussed four potential reasons that may explain the better generalization properties of flooding. But in order to have a better understanding of this regularizer, we will need to further investigate the properties of flooding theoretically. The phenomenon of the epoch-wise double descent curve that can be observed with flooding also needs investigation.

Recently, Zheng et al. (2020) compared various regularizers such as dropout (Srivastava et al., 2014), label smoothing (Szegedy et al., 2016), mixup (Zhang et al., 2018), adversarial training (Goodfellow et al., 2015), adversarial model perturbation (Zheng et al., 2020), and flooding (Chapter 5). The experimental results showed that flooding performs the best or second best with respect to confidence-calibration based on the expected calibration error (Guo et al., 2017) out of all methods. It would be interesting to further investigate this property from a theoretical point of view.

Class-Wise Flooding

We are interested in extending or improving the flooding algorithm. An example would be to employ different flood levels for different classes, instead of employing a single flood level for all samples. The motivation is that some classes may be more difficult than others, and the class-wise empirical risks should have different flood levels.

However, this increases the number of hyperparameters to K (the number of classes) from 1, and hyperparameter tuning would become impractical. It would be interesting to combine ideas from Bayesian optimization (Snoek et al., 2012) to cope with this issue.

Flooding in Other Tasks and Domains

In Chapter 5, we argued that flooding is task-independent and domain-independent. However, our experiments were based only on binary and multi-class classification for image datasets. It is important to investigate if it is *effective* in other tasks (e.g., regression, ranking, and clustering) and domains (e.g., natural language processing and tabular data).

6.3 Future Directions

In this section, we discuss potential future directions.

6.3.1 Looking for Yet Another Type of Supervision

In this dissertation, we explored if we can consider an alternative type of supervision, and investigated if we can propose algorithms to learn from such an alternative supervision. This exploration has led to positive-confidence learning (in Chapter 3) and complementary-label learning (in Chapter 4).

Recently, there have been other interesting proposals such as classification from multiclass label proportions or coarse-grained labels (Zhang et al., 2019), pairwise comparison binary classification which only has pairs of unlabeled data and a label showing which one is more likely to be positive (Feng et al., 2020), triplet comparison binary classification from triplet comparison data (Cui et al., 2020), and regression from pairwise comparison data (Xu et al., 2019).

An opposite direction is to explore *heavy supervision*. In some cases, labelers can provide rich information of the data they are observing. In fact, our idea of using confidence in Pconf classification can be regarded as making the supervision heavier for the positive samples while weakening supervision by removing negative samples. Another example is from Hancock et al. (2018), where they proposed a practical problem setting for classification where labelers provide a natural language explanation of why they chose certain class labels, and showed that utilizing this explanation when learning the classifier can boost the performance.

With ever changing environments and practical constraints, the demand for machine learning and methods to cope with various types of supervision will continue to increase in various areas in the industry and science. We believe research on novel types of supervision will remain to be an important topic.

6.3.2 Shifting Our Focus to Lowering the Total Cost

In this dissertation, we focused on learning from limited data and supervision with the aim of lowering the data collecting costs and labeling costs. In the long run, this is rather an intermediate goal, and our ultimate goal is to lower the total cost of starting a machine learning project and building a machine learning system. Collecting and labeling data is one of the early steps of a machine learning project, and starting a machine learning project will become much easier when the cost of data collection and labeling are low. However, the total cost may still remain high, since it is sometimes time-consuming and expensive to design, train, and evaluate machine

learning models. It will become important to shift our focus to lowering the total cost. *Hyperparameter optimization* (Yu and Zhu, 2020), *neural architecture search* (Elsken et al., 2019), and *Bayes error estimation* (Berisha et al., 2015; Sekeh et al., 2020; Noshad et al., 2019) are promising directions to lower the total cost.

Appendices

A Appendices for Chapter 3

In this section, we explain the neural network architectures that were used in Section 4.2.4.

A.1 CNN Architecture

- Convolution (3 in- /18 out-channels, kernel size 5).
- Max-pooling (kernel size 2, stride 2).
- Convolution (18 in- /48 out-channels, kernel size 5).
- Max-pooling (kernel size 2, stride 2).
- Fully-connected (800 units) with ReLU.
- Fully-connected (400 units) with ReLU.
- Fully-connected (1 unit).

A.2 AutoEncoder Architecture

- Convolution (3 in- /18 out-channels, kernel size 5, stride 1) with ReLU.
- Max-pooling (kernel size 2, stride 2).
- Convolutional layer (18 in- /48 out-channels, kernel size 5, stride 1) with ReLU.
- Max-pooling (kernel size 2, stride 2).
- Deconvolution (48 in- /18 out-channels, kernel size 5, stride 2) with ReLU.
- Deconvolution (18 in- /5 out-channels, kernel size 5, stride 2).
- Deconvolution (5 in- /3 out-channels, kernel size 4, stride 1) with Tanh.

B Appendices for Chapter 4

B.1 Proof of Lemma 4.3

Proof. By definition, $h(\boldsymbol{x}_i, \bar{y}_i) = \widetilde{\mathcal{L}}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}_i), \bar{y}_i)$ so that

$$\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{OVA}}) = \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}} \sigma_i\left(\frac{1}{K-1}\sum_{y\neq\bar{y}_i}\tilde{\ell}(g_y(\boldsymbol{x}_i)) + \tilde{\ell}(-g_{\bar{y}_i}(\boldsymbol{x}_i))\right)\right]. \tag{2}$$

After rewriting $\widetilde{\mathcal{L}}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}_i), \bar{y}_i)$, we can know that

$$\widetilde{\mathcal{L}}_{\mathrm{OVA}}(\boldsymbol{g}(\boldsymbol{x}_i), \bar{y}_i) = \frac{1}{K-1}\sum_{y}\tilde{\ell}(g_y(\boldsymbol{x}_i)) + \frac{K}{K-1}\tilde{\ell}(-g_{\bar{y}_i}(\boldsymbol{x}_i)), \tag{3}$$

and subsequently,

$$\begin{aligned}\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{OVA}}) \leq{}& \frac{1}{K-1}\mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}} \sigma_i\sum_{y}\tilde{\ell}(g_y(\boldsymbol{x}_i))\right] \\ &+ \frac{K}{K-1}\mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}} \sigma_i\tilde{\ell}(-g_{\bar{y}_i}(\boldsymbol{x}_i))\right]\end{aligned} \tag{4}$$

due to the sub-additivity of the supremum.

The first term is independent of $\bar{y}_i$ and thus

$$\begin{aligned}&\mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}} \sigma_i\sum_{y}\tilde{\ell}(g_y(\boldsymbol{x}_i))\right] \\ &= \mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i\sum_{y}\tilde{\ell}(g_y(\boldsymbol{x}_i))\right] \\ &\leq \sum_{y}\mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i\tilde{\ell}(g_y(\boldsymbol{x}_i))\right] \\ &= \sum_{y}\mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_y\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i\tilde{\ell}(g_y(\boldsymbol{x}_i))\right] \\ &= K\bar{\mathfrak{R}}_n(\tilde{\ell}\circ\mathcal{G}),\end{aligned} \tag{5}$$

which means the first term can be bounded by $K/(K-1)\cdot\bar{\mathfrak{R}}_n(\tilde{\ell}\circ\mathcal{G})$. The second

term is more involved. Let $I(\cdot)$ be the indicator function and $\alpha_i = 2I(y = \bar{y}_i) - 1$, then

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}\frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}}\sigma_i\tilde{\ell}(-g_{\bar{y}_i}(\boldsymbol{x}_i))\right] \\
&= \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}\frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}}\sigma_i\sum_{y}\tilde{\ell}(-g_y(\boldsymbol{x}_i))I(y=\bar{y}_i)\right] \\
&= \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}\frac{1}{2n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}}\sigma_i\sum_{y}\tilde{\ell}(-g_y(\boldsymbol{x}_i))(\alpha_i+1)\right] \\
&\leq \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}\frac{1}{2n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}}\alpha_i\sigma_i\sum_{y}\tilde{\ell}(-g_y(\boldsymbol{x}_i))\right] \\
&\quad + \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}\frac{1}{2n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}}\sigma_i\sum_{y}\tilde{\ell}(-g_y(\boldsymbol{x}_i))\right] \\
&= \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}\frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}}\sigma_i\sum_{y}\tilde{\ell}(-g_y(\boldsymbol{x}_i))\right], \qquad (6)
\end{aligned}$$

where we used that $\alpha_i\sigma_i$ has exactly the same distribution as σ_i. This can be similarly bounded by $\bar{\mathfrak{R}}_n(\tilde{\ell}\circ\mathcal{G})$ and the second term can be bounded by $K^2/(K-1)\cdot\bar{\mathfrak{R}}_n(\tilde{\ell}\circ\mathcal{G})$.

As a result,

$$\begin{aligned}
\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{OVA}}) &\leq \left(\frac{K}{K-1}+\frac{K^2}{K-1}\right)\bar{\mathfrak{R}}_n(\tilde{\ell}\circ\mathcal{G}) \\
&= \frac{K(K+1)}{K-1}\bar{\mathfrak{R}}_n(\tilde{\ell}\circ\mathcal{G}) \leq \frac{K(K+1)}{K-1}L_{\ell}\bar{\mathfrak{R}}_n(\mathcal{G}) \\
&= \frac{K(K+1)}{K-1}L_{\ell}\mathfrak{R}_n(\mathcal{G}), \qquad (7)
\end{aligned}$$

according to Talagrand's contraction lemma (Ledoux and Talagrand, 1991).[1] □

[1] We thank Katsura and Uchida (2020) for pointing out a miscalculation in our earlier version of this lemma.

B.2 Proof of Lemma 4.4

Proof. By definition,

$$\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{PC}}) = \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}} \sigma_i \left(\sum_{y'\neq\bar{y}_i} \tilde{\ell}(g_{y'}(\boldsymbol{x}_i) - g_{\bar{y}_i}(\boldsymbol{x}_i))\right)\right]. \tag{8}$$

Using the proof technique for handling the second term in the proof of Lemma 4.3, we have

$$\begin{aligned}
\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{PC}}) &\leq \mathbb{E}_{\mathcal{S}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{(\boldsymbol{x}_i,\bar{y}_i)\in\mathcal{S}} \sigma_i \sum_{y}\left(\sum_{y'\neq y} \tilde{\ell}(g_{y'}(\boldsymbol{x}_i) - g_{y}(\boldsymbol{x}_i))\right)\right] \\
&= \mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i \sum_{y}\left(\sum_{y'\neq y} \tilde{\ell}(g_{y'}(\boldsymbol{x}_i) - g_{y}(\boldsymbol{x}_i))\right)\right] \\
&\leq \sum_{y}\sum_{y'\neq y} \mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_y,g_{y'}\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i \tilde{\ell}(g_{y'}(\boldsymbol{x}_i) - g_{y}(\boldsymbol{x}_i))\right],
\end{aligned} \tag{9}$$

due to the sub-additivity of the supremum.

Let $\mathcal{G}_{y,y'} = \{\boldsymbol{x} \mapsto g_{y'}(\boldsymbol{x}) - g_y(\boldsymbol{x}) \mid g_y, g_{y'} \in \mathcal{G}\}$, then by Talagrand's contraction lemma (Ledoux and Talagrand, 1991),

$$\begin{aligned}
&\mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_y,g_{y'}\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i \tilde{\ell}(g_{y'}(\boldsymbol{x}_i) - g_{y}(\boldsymbol{x}_i))\right] \\
&= \bar{\mathfrak{R}}_n(\tilde{\ell}\circ\mathcal{G}_{y,y'}) \\
&\leq L_\ell \bar{\mathfrak{R}}_n(\mathcal{G}_{y,y'}) \\
&= L_\ell \mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_y,g_{y'}\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i (g_{y'}(\boldsymbol{x}_i) - g_{y}(\boldsymbol{x}_i))\right] \\
&\leq L_\ell \mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_y\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i g_y(\boldsymbol{x}_i)\right] + L_\ell \mathbb{E}_{\bar{\mathcal{X}}}\mathbb{E}_{\sigma}\left[\sup_{g_{y'}\in\mathcal{G}} \frac{1}{n}\sum_{\boldsymbol{x}_i\in\bar{\mathcal{X}}} \sigma_i g_{y'}(\boldsymbol{x}_i)\right] \\
&= 2L_\ell \bar{\mathfrak{R}}_n(\mathcal{G}) \\
&= 2L_\ell \mathfrak{R}_n(\mathcal{G}).
\end{aligned} \tag{10}$$

This proves that $\bar{\mathfrak{R}}_n(\mathcal{H}_{\mathrm{PC}}) \leq 2K(K-1)L_\ell \mathfrak{R}_n(\mathcal{G})$. □

B.3 Proof of Lemma 4.5

Proof. We prove the case of $\bar{\mathcal{L}}_{\text{OVA}}$; the other case is similar. We consider a single direction $\sup_{g_1,\ldots,g_K\in\mathcal{G}}(\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))$ with probability at least $1-\delta/2$; the other direction is similar too.

Given the symmetric condition (4.19), it must hold that $\|\bar{\mathcal{L}}_{\text{OVA}}\|_\infty = 2$ when $g_1,\ldots,g_K$ can be any measurable functions. Let a single $(\boldsymbol{x}_i, \bar{y}_i)$ be replaced with $(\boldsymbol{x}_i', \bar{y}_i')$, then the change of $\sup_{g_1,\ldots,g_K\in\mathcal{G}}(\widehat{R}(\boldsymbol{g})-R(\boldsymbol{g}))$ is no greater than $2(K-1)/n$. Apply McDiarmid's inequality (McDiarmid, 1989) to the single-direction uniform deviation $\sup_{g_1,\ldots,g_K\in\mathcal{G}}(\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))$ to get that with probability at least $1-\delta/2$,

$$\sup_{g_1,\ldots,g_K\in\mathcal{G}}(\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g})) \leq \mathbb{E}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}(\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))\right] + (K-1)\sqrt{\frac{2\ln(2/\delta)}{n}}. \quad (11)$$

Since $R(\boldsymbol{g}) = \mathbb{E}[\widehat{R}(\boldsymbol{g})]$, it is a routine work to show by symmetrization that (Mohri et al., 2012)

$$\mathbb{E}\left[\sup_{g_1,\ldots,g_K\in\mathcal{G}}(\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))\right] \leq 2(K-1)\bar{\mathfrak{R}}_n(\mathcal{H}_{\text{OVA}}) \leq 2K(K+1)L_\ell\mathfrak{R}_n(\mathcal{G}), \quad (12)$$

where the last line is due to Lemma 4.3. □

B.4 Proof of Theorem 4.6

Proof. Based on Lemma 4.5, the estimation error bounds can be proven through

$$\begin{aligned} R(\hat{\boldsymbol{g}}) - R(\boldsymbol{g}^*) &= \left(\widehat{R}(\hat{\boldsymbol{g}}) - \widehat{R}(\boldsymbol{g}^*)\right) + \left(R(\hat{\boldsymbol{g}}) - \widehat{R}(\hat{\boldsymbol{g}})\right) + \left(\widehat{R}(\boldsymbol{g}^*) - R(\boldsymbol{g}^*)\right) \\ &\leq 0 + 2\sup_{g_1,\ldots,g_K\in\mathcal{G}}\left|\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g})\right|, \end{aligned} \quad (13)$$

where we used that $\widehat{R}(\hat{\boldsymbol{g}}) \leq \widehat{R}(\boldsymbol{g}^*)$ by the definition of $\hat{\boldsymbol{g}}$. □

B.5 Benchmark Datasets

In the experiments in Section 4.3.5, we use 4 benchmark datasets explained below. The summary statistics of the four datasets are given in Table 1.

- MNIST[2] (Lecun et al., 1998) is a 10 class dataset of handwritten digits: $1, 2\ldots, 9$ and 0. Each sample is a 28×28 grayscale image.

[2] http://yann.lecun.com/exdb/mnist/

Table 1: Summary statistics of benchmark datasets. Fashion is Fashion-MNIST and Kuzushiji is Kuzushiji-MNIST.

Name	# Train	# Test	# Dim	# Classes	Model
MNIST	60k	10k	784	10	Linear, MLP
Fashion	60k	10k	784	10	Linear, MLP
Kuzushiji	60k	10k	784	10	Linear, MLP
CIFAR-10	50k	10k	2,048	10	DenseNet, Resnet

- Fashion-MNIST[3] (Xiao et al., 2017) is a 10 class dataset of fashion items: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. Each sample is a 28×28 grayscale image.
- Kuzushiji-MNIST[4] (Clanuwat et al., 2018) is a 10 class dataset of cursive Japanese ("Kuzushiji") characters. Each sample is a 28×28 grayscale image.
- CIFAR-10[5] is a 10 class dataset of various objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each sample is a colored image in $32 \times 32 \times 3$ RGB format. It is a subset of the 80 million tiny images dataset (Torralba et al., 2008).

C Appendices for Chapter 5

C.1 Benchmark Datasets

In the experiments in Section 5.3.2, we use five image benchmark datasets explained below.

- MNIST[6] (Lecun et al., 1998) is a 10 class dataset of handwritten digits: $1, 2 \ldots, 9$ and 0. Each sample is a 28×28 grayscale image. The number of training and test samples are 60,000 and 10,000, respectively.
- Kuzushiji-MNIST[7] (Clanuwat et al., 2018) is a 10 class dataset of cursive Japanese ("Kuzushiji") characters. Each sample is a 28×28 grayscale image. The number of training and test samples are 60,000 and 10,000, respectively.

[3] https://github.com/zalandoresearch/fashion-mnist
[4] https://github.com/rois-codh/kmnist
[5] https://www.cs.toronto.edu/~kriz/cifar.html
[6] http://yann.lecun.com/exdb/mnist/
[7] https://github.com/rois-codh/kmnist

- SVHN[8] (Netzer et al., 2011) is a 10 class dataset of house numbers from Google Street View images, in $32 \times 32 \times 3$ RGB format. 73257 digits are for training and 26032 digits are for testing.
- CIFAR-10[9] is a 10 class dataset of various objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each sample is a colored image in $32 \times 32 \times 3$ RGB format. It is a subset of the 80 million tiny images dataset (Torralba et al., 2008). There are 6,000 images per class, where 5,000 are for training and 1,000 are for test.
- CIFAR-100[10] is a 100 class dataset of various objects. Each class has 600 samples, where 500 samples are for training and 100 samples are for test. This is also a subset of the 80 million tiny images dataset (Torralba et al., 2008).

C.2 Proof of Theorem 5.1

Proof. If the flood level is b, then the proposed flooding estimator is

$$\widetilde{R}(\boldsymbol{g}) = |\widehat{R}(\boldsymbol{g}) - b| + b. \tag{14}$$

Since the absolute operator can be expressed with a max operator with $\max(a, b) = \frac{a+b+|a-b|}{2}$, the proposed estimator can be re-expressed as

$$\widetilde{R}(\boldsymbol{g}) = 2\max(\widehat{R}(\boldsymbol{g}), b) - \widehat{R}(\boldsymbol{g}) = A - \widehat{R}(\boldsymbol{g}). \tag{15}$$

For convenience, we used $A = 2\max(\widehat{R}(\boldsymbol{g}), b)$. From the definition of MSE,

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) = \mathbb{E}[(\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))^2], \tag{16}$$

and

$$\begin{aligned}\mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) &= \mathbb{E}[(\widetilde{R}(\boldsymbol{g}) - R(\boldsymbol{g}))^2] \\ &= \mathbb{E}[(A - \widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))^2)] \\ &= \mathbb{E}[A^2] - 2\mathbb{E}[A(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))] + \mathbb{E}[(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))^2].\end{aligned} \tag{17}$$

We are interested in the sign of

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) - \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) = \mathbb{E}[-4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - A^2 + 2A(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))]. \tag{18}$$

[8] http://ufldl.stanford.edu/housenumbers/
[9] https://www.cs.toronto.edu/~kriz/cifar.html
[10] https://www.cs.toronto.edu/~kriz/cifar.html

Define the inside of the expectation as $B = -4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - A^2 + 2A(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))$. B can be divided into two cases, depending on the outcome of the max operator:

$$\begin{aligned} B &= \begin{cases} -4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - 4\widehat{R}(\boldsymbol{g})^2 + 4\widehat{R}(\boldsymbol{g})(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g})) & \text{if} \quad \widehat{R}(\boldsymbol{g}) \geq b \\ -4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - 4b^2 + 4b(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g})) & \text{if} \quad \widehat{R}(\boldsymbol{g}) < b \end{cases} \\ &= \begin{cases} 0 & \text{if} \quad \widehat{R}(\boldsymbol{g}) \geq b \\ -4(b - \widehat{R}(\boldsymbol{g}))(b - R(\boldsymbol{g})) & \text{if} \quad \widehat{R}(\boldsymbol{g}) < b. \end{cases} \end{aligned} \tag{19}$$

In the latter case, B becomes positive when $\widehat{R}(\boldsymbol{g}) < b < R(\boldsymbol{g})$.

Therefore, if $b \leq R(\boldsymbol{g})$, we have

$$\begin{aligned} \mathrm{MSE}(\widehat{R}(\boldsymbol{g})) - \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) =& \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) \geq b] \Pr[\widehat{R}(\boldsymbol{g}) \geq b] \\ &+ \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) < b] \Pr[\widehat{R}(\boldsymbol{g}) < b] \\ =& \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) < b] \Pr[\widehat{R}(\boldsymbol{g}) < b] \\ \geq& 0. \end{aligned} \tag{20}$$

Furthermore, if $b < R(\boldsymbol{g})$ and $\Pr[\widehat{R}(\boldsymbol{g}) < b] > 0$, we have

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) - \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) = \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) < b] \Pr[\widehat{R}(\boldsymbol{g}) < b] > 0. \tag{21}$$

□

Acknowledgments

First and foremost, I would like to express my deepest gratitude to Prof. Masashi Sugiyama. This thesis could not have been completed without his tremendous support, guidance, and supervision. His encouragement kept me going for so many years, and I learned a lot throughout the journey.

I would like to express my deep gratitude to Prof. Issei Sato, Prof. Junya Honda, and Prof. Naoto Yokoya for the advice, discussions, and providing a nice research environment for students in the lab. I would like to thank Prof. Hiroshi Kori, Prof. Taiji Suzuki, and Prof. Hideki Nakayama for the valuable comments and suggestions for improving this dissertation.

I was fortunate to have the chance to work with Dr. Gang Niu. He has mentored me for many years and I learned a lot about writing a scientific paper from the numerous inspiring discussions. I am very grateful to Dr. Ikko Yamane and Dr. Tomoya Sakai. I had countless research discussions with them, and that formed the basis of how I tackle machine learning problems. I was also fortunate to have the chance to collaborate with Weihua Hu, Dr. Aditya Krishna Menon, and Prof. Chang Xu. I was often inspired by their expertise and learned a lot from this experience.

I had the joy of working or discussing with many incredible researchers, professors, and students. Thank you to Nontawat Charoenphakdee, Nan Lu, Kento Nozawa, Dr. Futoshi Futami, Soma Yokoi, Han Bao, Takeshi Teshima, Jeonghyun Song, Dr. Yoshihiro Nagano, Dr. Voot Tangkaratt, Prof. Bo Han, David Ha, Dr. Jie Luo, Prof. Takayuki Osa, Dr. Marthinus Christoffel du Plessis, Prof. Hiroaki Sasaki, and Dr. Kiyoshi Irie. I would like to thank all members of Sugiyama-Yokoya Lab. Thank you to Yuko Kawashima and Etsuko Yoshida for supporting me with lab-related activities.

I am grateful to the many people and coworkers who have supported me during my years at Sumitomo Mitsui Asset Management and Sumitomo Mitsui DS Asset Management. In particular, I am grateful to Kunio Yokoyama, Makoto Nagao, Takashi Kume, Dr. Jason Bennett, Dr. Naoya Kawadai, Prof. Anna Liu, and Takafumi Nohara.

I would like to thank the Google PhD Fellowship Program, Japan Society for the Promotion of Science (JSPS) Fellowship Program, Toyota-Dwango AI Scholarship Program, and JST ACT-X. The support from these fellowships, scholarships, and grants were helpful during my studies.

Last but not least, I would like to thank my family. A huge thank you to my wife. She gave me a lot of encouragement and support. Thank you to my parents for raising me, and thank you to my younger brother for the encouragement throughout the many years.

Bibliography

Akata, Z., Perronnin, F., Harchaoui, Z., and Schmid, C. (2013). Label-embedding for attribute-based classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. (2017). A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 233–242.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *2nd International Conference on Learning Representations*.

Bao, H., Niu, G., and Sugiyama, M. (2018). Classification from pairwise similarity and unlabeled data. In *Proceedings of the 35th International Conference on Machine Learning*, pages 452–461.

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *PNAS*, 116:15849–15854.

Belkin, M., Hsu, D. J., and Mitra, P. (2018). Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate. In *Advances in Neural Information Processing Systems 31*, pages 2300–2311.

Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434.

Berisha, V., Wisler, A., Hero, A. O., and Spanias, A. (2015). Empirically estimable classification bounds based on a nonparametric divergence measure. *IEEE Transactions on Signal Processing*, 64(3):580–591.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. (2019). MixMatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems 32*, pages 5049–5059.

Bishop, C. M. (1995). Regularization and complexity control in feed-forward networks. In *5th International Conference on Artificial Neural Networks*, pages 141–148.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Blanchard, G., Lee, G., and Scott, C. (2010). Semi-supervised novelty detection. *Journal of Machine Learning Research*, 11:2973–3009.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104.

Caruana, R., Lawrence, S., and Giles, C. L. (2000). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13*, pages 402–408.

Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning*. MIT Press.

Chapelle, O., Weston, J., and Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 16*, pages 601–608.

Charoenphakdee, N., Vongkulbhisal, J., Chairatanakul, N., and Sugiyama, M. (2020). On focal loss for class-posterior probability estimation: A theoretical perspective. *arXiv preprint arXiv:2011.09172*.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. (2017). Entropy-SGD: Biasing gradient descent into wide valleys. In *5th International Conference on Learning Representations*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607.

Cid-Sueiro, J., García-García, D., and Santos-Rodríguez, R. (2014). Consistency of losses for learning from weak labels. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 197–210.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical Japanese literature. In *NeurIPS Workshop on Machine Learning for Creativity and Design*.

Cour, T., Sapp, B., and Taskar, B. (2011). Learning from partial labels. *Journal of Machine Learning Research*, 12:1501–1536.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 113–123.

Cui, Z., Charoenphakdee, N., Sato, I., and Sugiyama, M. (2020). Classification from triplet comparison data. *Neural Computation*, 32(3):659–681.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on Machine Learning*, pages 647–655.

Drucker, H. and LeCun, Y. (1992). Improving generalisation performance using double back-propagation. *IEEE Transactions on Neural Networks*, 3(6):991–997.

du Plessis, M. C., Niu, G., and Sugiyama, M. (2013). Clustering unclustered data: Unsupervised binary labeling of two datasets having different class balances. In *TAAI*.

du Plessis, M. C., Niu, G., and Sugiyama, M. (2014). Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems 27*, pages 703–711.

du Plessis, M. C., Niu, G., and Sugiyama, M. (2015). Convex formulation for learning from positive and unlabeled data. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1386–1394.

du Plessis, M. C., Niu, G., and Sugiyama, M. (2017). Class-prior estimation for learning from positive and unlabeled data. *Machine Learning*, 106(4):463–492.

du Plessis, M. C. and Sugiyama, M. (2014a). Class prior estimation from positive and unlabeled data. *IEICE Transactions on Information and Systems*, E97-D(5):1358–1362.

du Plessis, M. C. and Sugiyama, M. (2014b). Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50:110–119.

Dwork, C. (2008). Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer.

Elkan, C. and Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *ACM SIGKDD*.

Feng, L., Shu, S., Lu, N., Han, B., Xu, M., Niu, G., An, B., and Sugiyama, M. (2020). Pointwise binary classification with pairwise confidence comparisons. *arXiv preprint arXiv:2010.01875*.

Fishman, G. S. (1996). *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1180–1189.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. (2018). Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems 31*, pages 10727–10737.

Ghosh, A., Kumar, H., and Sastry, P. (2017). Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *Sixth International Conference on Learning Representations*.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers.

Goodfellow, I. (2020). Generative adversarial networks. URL: https://www.youtube.com/watch?v=9d4jmPmTWmc. Keynote speech for "GANs for Good–A Virtual Expert Panel by DeepLearning.AI".

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations*.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330.

Guo, H., Mao, Y., and Zhang, R. (2019). Augmenting data with mixup for sentence classification: An empirical study. In *arXiv:1905.08941*.

Han, B., Niu, G., Yu, X., Yao, Q., Xu, M., Tsang, I. W., and Sugiyama, M. (2020). Sigua: Forgetting may make learning with noisy labels more robust. In *Proceedings of the 37th International Conference on Machine Learning*.

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I. W., and Sugiyama., M. (2018). Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems 31*, pages 8527–8537.

Hancock, B., Bringmann, M., Varma, P., Liang, P., Wang, S., and Ré, C. (2018). Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, page 1884. NIH Public Access.

Hanson, S. J. and Pratt, L. Y. (1988). Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems 1*, pages 177–185.

Harker, R. (1997). Selectively rejecting spam using sendmail. In *LISA*.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *Nature*, 585(7825):357–362.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer.

Hataya, R., Zdenek, J., Yoshizoe, K., and Nakayama, H. (2020). Faster autoaugment: Learning augmentation strategies using backpropagation. In *16th European Conference on Computer Vision*.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Heaton, J. B., Polson, N. G., and Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv:1207.0580*.

Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. In *NeurIPS 2014 Deep Learning Workshop*.

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

Hochreiter, S. and Schmidhuber, J. (1997). Flat minima. *Neural Computation*, 9:1–42.

Howe, J. (2008). *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Currency.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Idelbayev, Y. (2020). Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. https//github.com/akamaster/pytorch_resnet_cifar10. Accessed: 2020-05-31.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456.

Ishida, T., Niu, G., Hu, W., and Sugiyama, M. (2017). Learning from complementary labels. In *Advances in Neural Information Processing Systems 30*, pages 5639–5649.

Ishida, T., Niu, G., Menon, A. K., and Sugiyama, M. (2019). Complementary-label learning for arbitrary losses and models. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2971–2980.

Ishida, T., Niu, G., and Sugiyama, M. (2018). Binary classification from positive-confidence data. In *Advances in Neural Information Processing Systems 31*, pages 5917–5928.

Ishida, T., Yamane, I., Sakai, T., Niu, G., and Sugiyama, M. (2020). Do we need zero training loss after achieving zero training error? In *Proceedings of the 37th International Conference on Machine Learning*.

Jing, L. and Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Katsura, Y. and Uchida, M. (2020). Bridging ordinary-label learning and complementary-label learning. In *Proceedings of The 12th Asian Conference on Machine Learning*, pages 161–176.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations*.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Kiryo, R., Niu, G., du Plessis, M. C., and Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems 30*, pages 1675–1685.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big transfer (BiT): General visual representation learning. In *arXiv:1912.11370v3*.

Krogh, A. and Hertz, J. A. (1991). A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems 4*, pages 950–957.

Laine, S. and Aila, T. (2017). Temporal ensembling for semi-supervised learning. In *Fifth International Conference on Learning Representations*.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Ledoux, M. and Talagrand, M. (1991). *Probability in Banach Spaces: Isoperimetry and Processes*. Springer.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems 31*, pages 6389–6399.

Li, W., Dasarathy, G., and Berisha, V. (2020). Regularization via structural label smoothing. In *The 23rd International Conference on Artificial Intelligence and Statistics*, pages 1453–1463.

Liakos, K. G., Busato, P., Moshou, D., Pearson, S., and Bochtis, D. (2018). Machine learning in agriculture: A review. *Sensors*, 18(8):2674.

Lim, S., Kim, I., Kim, T., Kim, C., and Kim, S. (2019). Fast autoaugment. In *Advances in Neural Information Processing Systems 32*, pages 6665–6675.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations*.

Lu, N., Niu, G., Menon, A. K., and Sugiyama, M. (2019). On the minimal supervision for training any binary classifier from only unlabeled data. In *7th International Conference on Learning Representations*.

Lu, N., Zhang, T., Niu, G., and Sugiyama, M. (2020). Mitigating overfitting in supervised classification from two unlabeled datasets: A consistent risk correction approach. In *The 23rd International Conference on Artificial Intelligence and Statistics*, pages 1115–1125.

Lukasik, M., Bhojanapalli, S., Menon, A. K., and Kumar, S. (2020). Does label smoothing mitigate label noise? In *Proceedings of the 37th International Conference on Machine Learning*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and Van Der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision*, pages 181–196.

McDiarmid, C. (1989). On the method of bounded differences. In Siemons, J., editor, *Surveys in Combinatorics*, pages 148–188. Cambridge University Press.

Mendelson, S. (2008). Lower bounds for the empirical minimization algorithm. *IEEE Transactions on Information Theory*, 54(8):3797–3803.

Menon, A. K., van Rooyen, B., Ong, C. S., and Williamson, R. C. (2015). Learning from corrupted binary labels via class-probability estimation. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 125–134.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. MIT Press.

Morgan, N. and Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in Neural Information Processing Systems 2*, pages 630–637.

Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P. H., and Dokania, P. K. (2020). Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems 33*.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. Cambridge, MA.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2020). Deep double descent: Where bigger models and more data hurt. In *8th International Conference on Learning Representations*.

Nakkiran, P., Kaplun, G., Kalimeris, D., Yang, T., Edelman, B. L., Zhang, F., and Barak, B. (2019). SGD on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems 32*.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2013). Learning with noisy labels. In *Advances in Neural Information Processing Systems 26*, pages 1196–1204.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Ng, A. Y. (1997). Preventing "overfitting" of cross-validation data. In *Proceedings of the 14th International Conference on Machine Learning*.

Nguyen, Q., Valizadegan, H., and Hauskrecht, M. (2011). Learning classification with auxiliary probabilistic information. In *2011 IEEE 11th International Conference on Data Mining*, pages 477–486.

Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer.

Noshad, M., Xu, L., and Hero, A. (2019). Learning to benchmark: Determining best achievable misclassification error from training data. *arXiv preprint arXiv:1909.07192*.

Olson, R. S., La Cava, W., Mustahsan, Z., Varik, A., and Moore, J. H. (2018). Data-driven advice for applying machine learning to bioinformatics problems. *Pacific Symposium on Biocomputing*, pages 192–203.

Pantel, P. and Lin, D. (1998). Spamcop: A spam classification & organization program. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 12.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8026–8037.

Patrini, G., Rozza, A., Menon, A. K., Nock, R., and Qu, L. (2017). Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. In *arXiv:1701.06548*.

Read, J., Pfahringer, B., and Holmes, G. (2011). Classifier chains for mulit-label classification. *Machine Learning*, 85:333–359.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning*, pages 833—-840.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., and Schmidt, L. (2019). A meta-analysis of overfitting in machine learning. In *Advances in Neural Information Processing Systems 32*, pages 9179–9189.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems 29*, pages 1163–1171.

Sakai, T., du Plessis, M. C., Niu, G., and Sugiyama, M. (2017). Semi-supervised classification based on classification from positive and unlabeled data. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2998–3006.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471.

Schölkopf, B. and Smola, A. (2001). *Learning with Kernels*. MIT Press.

Scott, C. and Blanchard, G. (2009). Novelty detection: Unlabeled data definitely help. In *The 12th International Conference on Artificial Intelligence and Statistics*, pages 464–471.

Sekeh, S. Y., Oselio, B. L., and Hero, A. O. (2020). Learning to bound the multi-class bayes error. *IEEE Transactions on Signal Processing*, pages 3793 – 3807.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Shimada, T., Bao, H., Sato, I., and Sugiyama, M. (2019). Classification from pairwise similarities/dissimilarities and unlabeled data via empirical risk minimization. *arXiv preprint arXiv:1904.11717*.

Shinoda, K., Kaji, H., and Sugiyama, M. (2020). Binary classification from positive data with skewed confidence. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 3328–3334.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 958–963.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2951–2959.

Sokolić, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. (2017). Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280.

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

Sugiyama, M. (2015). *Introduction to statistical machine learning*. Morgan Kaufmann.

Sugiyama, M. and Kawanabe, M. (2012). *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. MIT Press.

Sugiyama, M., Niu, G., Yamada, M., Kimura, M., and Hachiya, H. (2014). Information-maximization clustering based on squared-loss mutual information. *Neural Computation*, 26(1):84–131.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, 2nd edition edition.

Szegedy, C., Vanhoucke, V., Ioffe, S., and Shlens, J. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *2nd International Conference on Learning Representations*.

Tax, D. M. J. and Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, 54(1):45–66.

Thulasidasan, S., Chennupati, G., Bilmes, J., Bhattacharya, T., and Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems 32*, pages 13888–13899.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58:267–288.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.

Tikhonov, A. N. (1943). On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39:195–198.

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of Ill Posed Problems*. Winston.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 30, pages 1958 – 1970.

van Rooyen, B., Menon, A., and Williamson, R. C. (2015). Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems 28*, pages 10–18.

van Rooyen, B. and Williamson, R. C. (2018). A theory of learning with corrupted labels. *Journal of Machine Learning Research*, 18:1–50.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley & Sons.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. (2019). Interpolation consistency training for semi-supervised learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3635–3641.

Vu, M.-A. T., Adalı, T., Ba, D., Buzsáki, G., Carlson, D., Heller, K., Liston, C., Rudin, C., Sohal, V. S., Widge, A. S., et al. (2018). A shared vision for machine learning in neuroscience. *Journal of Neuroscience*, 38(7):1601–1607.

Wager, S., Wang, S., and Liang, P. (2013). Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems 26*, pages 351–359.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1058–1066.

Wang, X., Ounis, I., and Macdonald, C. (2020). Negative confidence-aware weakly supervised binary classification for effective review helpfulness classification. In *29th ACM International Conference on Information and Knowledge Management*.

Werpachowski, R., György, A., and Szepesvári, C. (2019). Detecting overfitting via adversarial examples. In *Advances in Neural Information Processing Systems 32*, pages 7858–7868.

Xian, Y., Schiele, B., and Akata, Z. (2017). Zero-shot learning - the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, L., Honda, J., Niu, G., and Sugiyama, M. (2019). Uncoupled regression from pairwise comparison data. In *Advances in Neural Information Processing Systems 32*, pages 3992–4002.

Xu, Y., Xu, C., Xu, C., and Tao, D. (2017). Multi-positive and unlabeled learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3182–3188.

Yabe, T. and Zempo, K. (2020). Negative confidence recommendation system avoids local solution based on user's negative reactions. In *IEEE 9th Global Conference on Consumer Electronics*, pages 531–534.

Yu, T. and Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. *arXiv:2003.05689*.

Yu, X., Liu, T., Gong, M., and Tao, D. (2018). Learning with biased complementary labels. In *15th European Conference on Computer Vision*.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations*.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations*.

Zhang, T. (2004a). Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251.

Zhang, T. (2004b). Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85.

Zhang, Y., Charoenphakdee, N., and Sugiyama, M. (2019). Learning from indirect observations. *arXiv preprint arXiv:1910.04394*.

Zheng, Y., Zhang, R., and Mao, Y. (2020). Regularizing neural networks via adversarial model perturbation. *arXiv preprint arXiv:2010.04925*.

Zhou, Z.-H. (2018). A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.