UNIVERSITY OF CALIFORNIA
SANTA CRUZ

ON PEER LOSS: ITS THEORY AND THE APPLICATIONS TO THE PROBLEM OF LEARNING WITH NOISY LABELS

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE AND ENGINEERING

by

Xingyu Li

June 2020

The Thesis of Xingyu Li
is approved:

Professor Yang Liu, Chair

Professor David Helmbold

Professor James Davis

Quentin Williams
Acting Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

On peer loss: its theory and the applications to the problem of learning with noisy labels

by

Xingyu Li

Peer loss [1] is a new family of loss functions proposed to deal with the problem of learning with noisy labels. It claims to handle a wide range of label noise in binary classification tasks without explicitly estimating the noise rates. Numerical experiments demonstrate the effectiveness of peer loss. However, its extension to the multi-class classification remains unclear, and its working mechanism is not fully understood.

In this thesis, we study the theory of peer loss from three distinct perspectives. Follow the original method in [1], we first consider the multi-class extension of peer loss and investigate its noise tolerance properties. From this perspective, we see peer loss as a class of loss functions inspired by the truthful and proper scoring rules in the peer prediction literature. It turns out that this perspective is a static one and cannot provide a satisfactory explanation of how peer loss works in practical training. To gain an intuitive picture of the working mechanism, we further develop a divergence perspective towards peer loss, expressing it as the difference between two KL divergences. Thus, we recognize that peer loss has a built-in regularization effect, encouraging the model to make confident predictions. This regularization effect partially explains why peer loss works well under the label noise, as the existence of noise often blurs the data distribution and makes the resulting model prediction uncertain. Finally, we show that peer loss potentially suggests a new type of risk in decision theory, i.e., the correlation risk. This new

perspective helps us to understand better what the model learns when trained with peer loss. To complete the discussion of the correlation risk perspective, we develop a novel method to investigate the training dynamics of peer loss. This dynamical analysis justifies that with peer loss, the resulting model tends to grasp the positive correlations in the training datasets.

In addition to the theoretical analysis, we also carry out extensive numerical experiments. Those experiments on benchmark image datasets demonstrate the effectiveness of peer loss on the multi-class classification tasks under a wide range of label noise. Our experiments on the 2-dimension synthetic dataset reveal that the models trained with peer loss tend to produce hard decision boundaries. This phenomenon accords with our theoretical analysis that peer loss encourages the model to make confident predictions.

Acknowledgments

First of all, I want to thank my wife, Hui Zhang. I appreciate her support and patience during the two years when I am studying aboard.

I owe my special thanks to my supervisor, Professor Yang Liu. I'm lucky to have a chance to work with him at UC Santa Cruz. I admire his enthusiasm and taste in doing research. He is full of talent ideas and cares about his students. I wish to keep collaborating with him in my future academic career.

Many thanks to Zhaowei Zhu, Jiaheng Wei, and Jialu Wang. You are wonderful teammates and my best friends here. Every discussion is thought-provoking, and every party is shining in my memory.

Last, I want to express my gratitude to Professor David Helmbold and James Davis for being my thesis committees.

Table of Notations

$[K]$	abbreviation of the set $\{1, 2, \cdots, K\}$
$X, Y, \widetilde{Y}$	random variables stand for the feature, the clean label, and the noisy label
$x, y, \tilde{y}$	the values of the feature X, the clean label Y, and the noisy label $\widetilde{Y}$
$x_n, y_n, \tilde{y}_n$	the n-th sample of the feature, the clean label, and the noisy label
Ω_X	the domain of the feature X
$\mathcal{S}_{K-1}$	$K-1$ dimensional simplex
$h_y(x)$	the y-th component of the model prediction at x
$cl(x)$	the prediction of the classifier cl at x
$\ell(\cdot)$	a differentiable loss function
$\mathcal{D}$	clean data distribution
$\mathcal{D}_X, \mathcal{D}_Y$	marginal distributions of the feature and label of the clean data
$\widetilde{\mathcal{D}}$	noisy data distribution
$\widetilde{\mathcal{D}}_X, \widetilde{\mathcal{D}}_Y$	marginal distributions of the feature and label of the noisy data
D	empirical clean data distribution
D_X, D_Y	empirical marginal distributions of the feature and label of the clean data
$\widetilde{D}$	empirical noisy data distribution
$\widetilde{D}_X, \widetilde{D}_Y$	empirical marginal distributions of the feature and label of the noisy data
$\mathbb{P}(\cdot)$	distribution of a random variable, e.g., $\mathbb{P}(X)$
$\boldsymbol{P}(\cdot), \boldsymbol{Q}(\cdot)$	probability function for a random variable, e.g., $\boldsymbol{P}(x)$
T	the transition matrix
e_i	noise rate of flipping into the wrong class i. It is used only in the uniform off-diagonal noise model

Chapter 1

Introduction

The pioneering work of Yang and Guo [1] proposed a new class of loss functions called *peer loss* functions, which claims to handle label noise without explicitly estimating the noise rates. Peer loss is distinct from other methods dealing with label noise through viewing denoising as an elicitation problem. It embeds the statistical information about the correlation between features and labels in the sampling process. We will see that peer loss suggests an extension to the population risk, and it is possible to introduce a new framework in decision theory.

This thesis is primarily devoted to the investigation of the theory behind peer loss. We also include numerical experiment results to support the theoretical analysis and show its effectiveness in practical applications. Our discussions are confined to the classification problem in the supervised learning paradigm.

We first study the properties of the optimal classifier learned using peer loss. This is a static analysis, as it does not take into account the training process. We shall view peer loss from three perspectives: (1) as a class of loss functions inspired by the truthful proper scoring rules. (2) as divergence that measures the difference between the model predictions and the training data distribution. (3) as a new type of risk, i.e., the correlation risk, targeting a different learning objective

compared to the population risk. In the first perspective, we concern the extension of peer loss to the multi-class classification cases, following the method in the original work [1]. In the divergence perspective, we prove that peer loss possesses an equivalent form as the difference between two Kullback–Leibler (KL) divergences. This abstract yet intuitive form helps to clarify the working mechanism behind peer loss. Finally, the correlation risk perspective is the most ambitious one. We demonstrate that the formulation of peer loss can be reinterpreted as a new type of risk, which we call the correlation risk. This risk puts more emphasis on the correlation in the data rather than purely focuses on prediction accuracy. Hence, it would be preferable than the population risk when properties, such as fairness, matter.

Static analysis alone is not the complete story since the training process is an indispensable component of any machine learning algorithm. However, due to the high complexity of neural network models, there exists no universal/efficient method to analyze the training dynamics. In our dynamical analysis, we exploit the methods from evolutionary game theory [2]. Instead of explicitly investigating the evolution of the network function, we turn to study the stable steady states in the searching space, which is composed of all possible network functions. In this way, we manage to show that with the peer loss, the neural network model indeed tends to grasp the positive correlations in the dataset. This dynamical version of the classification-calibration argument complements our discussion about the correlation risk in the static analysis.

Peer loss is initially proposed as a method to deal with the problem of learning with noisy labels. Presently, this remains to be the major place of applying peer loss. We place a brief review on the learning with noisy labels problem at the end of this chapter to avoid disturbing the smoothness of theoretical analysis in later

content. The rest chapters are organized as follows: In chapter 2 and chapter 3, we present the static and dynamical analysis, respectively. Experiment results are summarized in chapter 4. At last, conclusion and future works are discussed in chapter 5. Detailed proofs of theorems and propositions and additional experiment results can be found in the appendix.

1.1 Learning with noisy labels

1.1.1 Motivation

Nowadays, supervised learning dominates in the majority of applications of machine learning algorithms. Its great success crucially relies on the availability of large scale quality datasets. This is an intrinsic consequence of its statistical nature. Nonetheless, data collection inevitably suffer from noise, due to its requirements of scalability, decentralization, and the imperfect process of human annotating. One salient example is the build of ImageNet [3]. The existence of label noise degrades the resulting machine learning model, as the noise would weaken the feature-label connection and introduce fake correlations. Furthermore, with the vast amount of parameters, present neural network models can easily overfit the noisy data [4], which harms the model's generalization performance.

1.1.2 Description of the label noise

Generally speaking, with the label noise, the original clean data distribution $\mathbb{P}(X, Y)$ has been transformed into a noisy distribution $\mathbb{P}(X, \widetilde{Y})$. By the chain rule, one always has

$$\boldsymbol{P}(x, \tilde{y} = i) = \sum_{j \in [K]} \boldsymbol{P}(\tilde{y} = i | x, y = j) \boldsymbol{P}(x, y = j).$$

It is convenient to introduce a transition matrix $T_{i,j}(x) := \boldsymbol{P}(\tilde{y} = i|x, y = j)$, which summarizes the label noise. We will refer to the off-diagonal terms of $T(x)$ as error rates in later content. Seeking a solution concerning the general label noise is too hard. Thus we will compromise and consider the feature-independent noise instead. Namely, we only deal with feature-independent transition matrix $T_{ij} = \boldsymbol{P}(\tilde{y} = i|y = j)$. We are now ready to formally state the problem we propose to solve using peer loss.

1.1.3 Problem statement: classification with noisy labels

Consider a multi-class classification problem, where the random variable X denotes a high-dimensional feature, and the random variable Y indicates the label for a particular instance of X. Let $\Omega_X \subseteq \mathbb{R}^d$ and $\mathcal{Y} \in \{1, 2, \cdots, K\}$ be the domains, from which X and Y take values, respectively. Here, K is the number of classes. For simplicity, we adopt $[K] := \{1, 2, \cdots, K\}$ in the later content. We further denote the joint distribution of X and Y as $\mathcal{D}$. The training samples (x_n, y_n) are drawn from $\mathcal{D}$ in an i.i.d. fashion and composes the training dataset $D := \{(x_n, y_n)\}_{n\in[N]}$. The classification task targets at learning a classifier $cl : \Omega_X \to \mathcal{Y}$ that maps an instance feature x to its label y accurately.

In practice, instead of directly searching for an optimal classifier r^*_D, one minimizes the empirical risk with respect to some loss function ℓ as:

$$h^*_{D,\ell} = \arg\min_{h\in\mathcal{H}} \frac{1}{N} \sum_{n\in[N]} \ell(h(x_n), y_n), \tag{1.1}$$

where $h : \Omega_X \to \mathcal{S}_{K-1}$ is the network function specified by the underlying neural network model. $\mathcal{H}$ is the collection of all possible h. $\mathcal{S}_{K-1}$ refers to a $K-1$ dimension simplex. The y-th component $h_y(x)$ is interpreted as the predicted

probability of x being labeled as y. The corresponding classifer is normally generated by $cl(x) := \arg\max_{y\in[K]} h_y(x)$. Commonly, for classification task the loss function ℓ is chosen as the Cross-Entropy (CE) loss, i.e. $\ell(h(x), y) = -\ln(h_y(x))$.

This empirical risk minimization process is closely related to the search of the Bayes optimal classifier, which is defined as

$$cl^*_{Bayes}(x) := \arg\max_{i\in[K]} \boldsymbol{P}(x, y = i),$$

where $\boldsymbol{P}(x, y)$ refers to the joint probability of the data distribution. Clearly, when the model perfectly learns the data distribution, i.e. $h_y(x) = \boldsymbol{P}(y|x)$, the resulting classifier will be just the Bayes optimal classifier.

In the presence of label noise, we would face the joint distribution between the feature X and the noisy label $\widetilde{Y}$, which is denoted as $\widetilde{\mathcal{D}}$. Correspondingly, we will have a noisy training dataset $\widetilde{D} := \{(x_n, \tilde{y}_n)\}_{n\in[N]}$. Now, minimizing the empirical risk leads to

$$h^*_{\widetilde{D},\ell} = \arg\min_{h\in\mathcal{H}} \frac{1}{N} \sum_{n\in[N]} \ell(h(x_n), \tilde{y}_n).$$

It is well known [4] that the opitmal classifier $h^*_{\widetilde{D},\ell}$ can easily overfit the noisy dataset $\widetilde{D}$ and, thus, degrade the resulting model.

Any label noise tolerant method tries to make $h^*_{\widetilde{D},\ell}$ closer to $h^*_{D,\ell}$. There exist many different types of methods to deal with label noise, as reviewed in section 1.2. In the present thesis, we propose to address the label noise issue through designing a loss function (i.e., the peer loss) such that the classifier trained on noisy datasets approximates the performance of the classifier trained over clean datasets. Under idea situation, we expect $h^*_{\widetilde{D},\ell}$ and $h^*_{D,\ell}$ generate the same classifier.

1.2 Related works

1.2.1 Learning with the estimate of noise rates

A good number of works have been devoted to studying how to make deep neural networks robust to label noise. This literature started with the random classification noise model, where observed labels are flipped independently with a certain probability [5, 6, 7, 8, 9, 10, 11]. More recent efforts focused on learning with asymmetric noisy data (or also referred as *class-conditional* random classification noise (CCN)) [12, 9, 10, 11, 13, 14].

Under the supervised learning paradigm, one popular choice is to use the surrogate loss [9, 10, 11, 13, 14], which use the transition matrix to define unbiased estimates of the true losses. A common difficulty in the surrogate loss methods is the explicit estimation of the transition matrix. Often an additional clean dataset is required to provide a reference [15, 14, 16]. Some methods exploit the transition matrix in different ways, including to correct the loss or network outputs [17] and re-weight the importance of training samples [16]. We also note that [18] recently proposes an information-theoretic loss, which is robust to both symmetric and asymmetric noise rates.

1.2.2 Learning without the estimate of noise rates

In order to avoid explicit estimation of the label noise, recent efforts try to solve the problem of learning with noisy labels by modeling it as a semi-supervised learning problem [19, 20, 21]. The high-level idea is to filter out the noisy samples during training and treat them as unlabeled data, on which semi-supervised learning technique will be applied. The above procedure requires a sufficient effort on hyperparameter tuning (e.g., deciding on whether a sample is noisy or not). Fur-

ther, the performance improvement is shown to be mainly due to semi-supervised learning in the latter stages.

Chapter 2

Static Analysis

In this chapter, we study the theory and intuition behind peer loss from three perspectives, emphasizing its working mechanism and the properties of the resulting optimal classifier. The detailed architecture of the neural network model is ignored, with only assuming its expressivity is high enough to include the optimal solution. This assumption is justified as, in practice, the neural network models are typically over-parametrized [22] comparing to the scale of the underlying task.

2.1 Peer loss as a loss fucntion inspired by the truthful and proper scoring rule

The design of peer loss [1] is inspired by the truthful and proper scoring rules [23, 24] in the peer prediction literature, which aim at eliciting truthful reports from agents without knowing the ground truth information. The original work [1] focuses on the binary classification tasks and proves that peer loss is label-noise tolerant as long as the noisy data remains informative. This section shows that the original arguments of peer loss can be readily extended to the multi-class

classification situations. The difficulty lies in finding the proper constraints on the type of label noise rather than on extending the original formulation of peer loss.

All missing proofs in this section can be found in the appendix.

2.1.1 The peer loss for binary classification

The orignal peer loss [1] is motivated by the Correlated Agreement (CA) mechanism [24] from peer prediction literature, which concerns a Δ matrix,

$$\Delta_{i,j} := \boldsymbol{P}(r = i, r' = j) - \boldsymbol{P}(r = i)\boldsymbol{P}(r' = j),$$

where r and r' are two sources of reports, and i, j refer to the possible outcomes. For example, if there are K classes then i, j can take values from 1 to K. This Δ matrix captures the correlation between the different sources of reports.

To build a truthful and proper scoring function from CA, Liu and Guo [1] introduce an auxiliary matrix $M := \text{sgn}(\Delta)$, where the element-wise operation $\text{sgn}(x) = 1$ for $x > 0$ and $\text{sgn}(x) = 0$ otherwise. Then the desired scoring function is defined as

$$S(r_n, r'_n) := M(r_n, r'_n) - M(r_{n_1}, r'_{n_2}),$$

where the subscripts n, n_1 and n_2 refer to three independent tasks for which the two resources r and r' report their outcomes. For each task n, we randomly choose other two tasks, n_1 and n_2, to calculate the score.

At this point, Liu and Guo [1] make a key analogy, viewing reports r and r' as the *Bayes optimal* classifier's prediction $\hat{y}^* = cl^*(x)$ and the label $\tilde{y}$ from the noisy dataset (the clean dataset can be viewed as a special case of a noisy dataset with zero noise rates), respectively. Further, different samples in the dataset are

recognized as the different tasks in the peer prediction situation. In this way, a generic peer loss is defined as

$$\ell_g(cl(x_n), \tilde{y}_n) := (1 - M(cl(x_n), \tilde{y}_n)) - (1 - M(cl(x_{n_1}), \tilde{y}_{n_2})),$$

which corresponds to the negative scoring function. Note that the loss function is defined for a general classifier $cl(x)$, while M is determined by the Δ matrix between the Bayes optimal classifier $cl^*(x)$ and the noisy label. The first term evaluates the classifier's prediction on x_n using corresponding label $\tilde{y}_n$, and the second term, a.k.a. the peer term, punishes the over-agreement between the classifier's prediction and the noisy label.

For the binary classification tasks with label noise, the freedom of the transition matrix is reduced to 2 and can be summarised by two noise rates: $e_+ = \boldsymbol{P}(\tilde{y} = -1|y = +1)$ and $e_- = \boldsymbol{P}(\tilde{y} = +1|y = -1)$. When $e_+ + e_- < 1$, we say the noisy data is informative. In other word, the true signal in the noisy dataset tends to dominate the nosiy signal. For example, let $e_+ = 0.1$ and $e_- = 0.8$ and assume $\boldsymbol{P}(y = +1) = \boldsymbol{P}(y = -1)$ in the original clean dataset. We find $\boldsymbol{P}(\tilde{y} = +1) = 0.9\boldsymbol{P}(y = +1) + 0.8\boldsymbol{P}(y = -1)$, where the first term in the r.h.s. corresponds to the true signal, and the second term corresponds to the noisy signal. Clearly, the true signal clearly dominates the noisy one. It has been proved (see [1], lemma 2) that if the noisy data is informative, then $M = I_{2\times 2}$, i.e., the identity matrix. With this simplification, the generic peer loss reduces to the 0-1 peer loss:

$$\mathbb{1}_{peer}(cl(x_n), \tilde{y}_n) = \mathbb{1}(cl(x_n), \tilde{y}_n) - \mathbb{1}(cl(x_{n_1}), \tilde{y}_{n_2}),$$

where n, n_1 and n_2 are the indices of three independent random samples from the

training data. We call samples n_1 and n_2 the peer samples of sample n. Since the 0-1 loss $\mathbb{1}(\cdot,\cdot)$ is not differentiable, one can replace it by any other differentiable (convex) functions $\ell(\cdot)$, and get

$$\ell_{peer}(h(x_n), \tilde{y}_n) = \ell(h(x_n), \tilde{y}_n) - \ell(h(x_{n_1}), \tilde{y}_{n_2}).$$

Thus we arrived at the standard form of peer loss as proposed in [1]. Note that argument of ℓ_{peer} is the model output $h(x)$ rather than the classifier $cl(x)$.

2.1.2 The peer loss for multiclass classification

Now, we consider the extension of peer loss to multi-class classification tasks. Based on the above arguments, it is easy to see that the procedure of deriving the generic peer loss is irrelevant to the number of classes. Thus, the key of the extension is to find a new informative condition for the multi-class cases.

Following the method in [1], let's consider the Δ matrix between the Bayes optimal classifier and the noisy labels in a K-class classification task, i.e. $\Delta_{ij} = \boldsymbol{P}(cl^*(x) = i, \tilde{y} = j) - \boldsymbol{P}(cl^*(x) = i)\boldsymbol{P}(\tilde{y} = j)$. We assume the Bayes optimal classifier is independent to the noisy label conditioned on the ture label. Since the Bayes optimal classifier is not perfect, we can treat its predictions as noisy labels as well and thus define the corresponding noise transition matrix $T^*_{ij} = \boldsymbol{P}(cl^*(x) = i|Y = j)$. Then the Δ matrix can be rewritten as

$$\begin{aligned}\Delta_{ij} = &\sum_{k}^{K} T^*_{ik} \cdot T_{jk} \cdot \boldsymbol{P}(y = k) \\ &- \sum_{l}^{K} T^*_{il} \cdot \boldsymbol{P}(y = l) \sum_{m}^{K} T_{jm} \boldsymbol{P}(y = m).\end{aligned}$$

Above summations can be viewed as expectations over the distribution of clean

labels. For example, $\sum_m^K T_{jm} \cdot \boldsymbol{P}(y = m)$ is the average noise rate of flipping into class j. Thus

$$\Delta_{ij} = \mathbb{E}_Y[T^*_{i,Y} \cdot T_{j,Y}] - \mathbb{E}_Y[T^*_{i,Y}]\mathbb{E}_Y[T_{j,Y}].$$

Note in $T_{i,Y}$, we treat the second index of the subscript as a random variable Y. In this way, we may interpret Δ as the covariance matrix between the noise rates of the Bayes optimal classifier and the noisy labels. Clearly, if cl^* and the noisy label are positively correlated in the case of flipping into the same class, i.e., $\Delta_{ii} > 0$, and negatively correlated in the case of flipping into different classes, i.e., $\Delta_{ij} < 0, i \neq j$, we will have M being the identity matrix $I_{K\times K}$. Next, we present the following two label noise settings that satisfy the above condition.

Sparse noise model:

This setting corresponds to the disjoint pairwise noises, in which we try to capture the settings where particular pairs of classes are easily mistaken between each other. For instance, an image of a plane may look like a bird, while it is unlikely to mistake a plane with a truck. One can specify this type of noise by a pair of classes (i, j) and a pair of transition matrix elements (T_{ij}, T_{ji}). Formally, suppose we have chosen C pairs, $(i_c, j_c)_{c\in[C]}$. Generally, the transition matrix is of the form (without losing generality, assume $i_c < j_c$, note there will be C pairs):

$$T = \begin{pmatrix} \ddots & & & & \\ & 1 - T_{i_c j_c} & \dots & T_{j_c i_c} & \\ & \vdots & \ddots & \vdots & \\ & T_{i_c j_c} & \dots & 1 - T_{j_c i_c} & \\ & & & & \ddots \end{pmatrix}.$$

The off-diagonal elements $T_{i,j} \neq 0$ only if $i = i_c, j = j_c$ or $i = j_c, j = i_c$.

Lemma 1. *Let there be C pairs in the K classes. If $T_{i_cj_c} + T_{j_ci_c} < 1$ and $T^*_{i_cj_c} + T^*_{j_ci_c} < 1$ for all $c \in [C]$, then $M = I_{K\times K}$.*

Uniform off-diagonal noise model:

More generally, we consider the case in which any clean labels can be flipped into every other possible class. We assume that the noise rates of flipping into a given class from other classes to be the same. Mathematically, it reads $T_{ij} = T_{ik}, \forall i \neq j \neq k$. Note that this is different from saying the noise is uniform. Different classes have different chances of having a wrong label (i.e., $T_{ii} \neq T_{jj}, i \neq j$.) Let $e_i = T_{ij}, \forall i \neq j$, the transition matrix T takes the following form

$$T = \begin{pmatrix} T_{11} & e_1 & e_1 & \dots & e_1 \\ e_2 & T_{22} & e_2 & \dots & e_2 \\ e_3 & e_3 & T_{33} & \dots & e_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_K & e_K & e_K & \dots & T_{KK} \end{pmatrix},$$

where the diagonal terms can be computed by the conservation of probability, i.e. $T_{ii} = 1 - \sum_{j\neq i} e_j$. Compared to the sparse noise setting, this is a much more chaotic noise setting with high average noise rates. We believe this noise model is more suitable for testing the robustness of our method.

If we further assume $e^*_i = T^*_{ij} = T^*_{ik}, \forall i \neq j \neq k$ and $T^*_{ii} = 1 - \sum_{j\neq i} e^*_j$ for the Bayes optimal classifier. Then, we have

Lemma 2. *If $\sum_{j\in[K]} e^*_j < 1$ and $\sum_{k\in[K]} e_k < 1$, then $M = I_{K\times K}$.*

We admit that **none** of the above-mentioned conditions are as concise and clean as in the binary classification case. However, pending the above conditions

holds, we can write down the 0-1 peer loss for multi-class situations

$$\mathbb{1}_{peer}(cl(x_n), \tilde{y}_n) = \mathbb{1}(cl(x_n), \tilde{y}_n) - \mathbb{1}(cl(x_{n_1}), \tilde{y}_{n_2}),$$

and, similarly, the peer loss in its standard form

$$\ell_{peer}(h(x_n), \tilde{y}_n) = \ell(h(x_n), \tilde{y}_n) - \ell(h(x_{n_1}), \tilde{y}_{n_2}). \tag{2.1}$$

Not surprisingly, this is in the same form as for the binary classification case. All the difficulties hide in finding the right informative conditions of the label noise so that the above formulation is valid.

2.1.3 Noise tolerance properties of peer loss

Peer loss is proved to possess excellent noise tolerance properties in the binary classification tasks. We further elaborate on its noise tolerance property in the multi-class classification case. For an arbitrary noise transition matrix, the analysis will be intractable. Thus, we will focus on the uniform off-diagonal noise model. We show under uniform off-diagonal noise, ℓ_{peer} is robust to label noise for an arbitrary hypothesis class:

Theorem 1. *The expectation of peer loss is invariant to label noise up to an affine transformation:*

$$\mathbb{E}_{\widetilde{\mathcal{D}}}[\ell_{peer}(h(X), \widetilde{Y})] = \left(1 - \sum_{j \in [K]} e_j\right) \mathbb{E}_{\mathcal{D}}[\ell_{peer}(h(X), Y)]. \tag{2.2}$$

This above theorem states that peer loss is invariant subject to a broad setting of multi-class label noise. Therefore optimizing peer loss over noisy data is equivalent to optimizing it over clean data. With the above, we can further prove

that when the classes are balanced, optimizing the 0-1 peer loss induces the Bayes optimal classifier.

Theorem 2. *When the true label Y has equal prior $\boldsymbol{P}(Y = k) = 1/K, \forall k \in [K]$, we have*

$$cl^*_{Bayes} = \arg\min_{cl \in \mathcal{F}} \ \mathbb{E}_{\widetilde{\mathcal{D}}}\Big[\mathbb{1}_{peer}(cl(X), \tilde{Y})\Big].$$

Remark 1. *Unlike for the 0-1 peer loss, we do not have any clear result that connects the optimal classifier obtained by optimizing the general peer loss ℓ_{peer} and the Bayes optimal classifier. It turns out we need to adopt a new viewpoint towards this problem, which will be elaborated in detail in section 2.3.3.*

2.1.4 The α peer loss and a brief summary of the experiments on benchmark image datasets

Now, we consider the practical implementation of peer loss to the learning with noisy labels problem. In this way, we want to investigate the practical effectiveness of peer loss under different types of label noise, not limiting to the sparse noise and uniform off-diagonal noise mentioned in the previous subsections.

The experiments suggest that for peer loss to work in the multi-class classification case, we need to introduce the α peer loss, which is defined as follows.

$$\ell_{p-\alpha}(h(x_n), \tilde{y}_n) = \ell(h(x_n), \tilde{y}_n) - \alpha \cdot \ell(h(x_{n_1}), \tilde{y}_{n_2}).$$

Recall that n, n_1 and n_2 refer to three independent random samples. n_1 and n_2 are called the peer samples of sample n. The only difference with the definition (2.1) is that we introduce a hyper-parameter α to tune the contribution of the second term (a.k.a. the peer term). In our experiments, the α increases as the training process progress. Initially, the α is set to 0 and gradually increases to a relatively

large value, say 5 or 10. We will stop increasing the α and fix its final value throughout the remaining training process. We find such a way of tuning α works in most situations of our experiments. To understand this behavior is one of the main targets of the later sections in this chapter.

Besides, we would like to provide a brief summary of the experiment results here. On the one hand, this thesis is mainly devoted to the theory of peer loss. We do not want to disturb the theoretical analysis. On the other hand, these experiments reveal interesting behaviors of peer loss that need to be explained in our theoretical framework. For the details about the experiments, the readers can refer to chapter 4. Now, we list the major observations:

1. As mentioned, we find it necessary to introduce a hyper-parameter α to control the peer term's contribution. In addition, we usually find an increasing α that starts from 0 will make the model work well.

2. Even though the formulation and noise tolerance properties of peer loss in the multi-class case rely on restricted assumptions about the type of label noise, we find the α peer loss works well in general situations, such the Clothing1M dataset.

3. The experiments on 2-dimension synthetic datasets show that the model trained with peer loss tend to have a hard decision boundary, i.e., the model is confident about their predictions.

In later content, we will explore extensions of the present theory of peer loss and provide explanations for those observations.

2.2 Peer loss as the difference between K-L divergences

In the previous section, we discussed the multi-class extension of peer loss and proved relevant label noise tolerance properties. The drawback of the previous formulation is two-fold: First, the noise tolerance relies on the complete cancellation of label noise, which is limited to some particular noise types and cannot be justified easily in real-life practice. Second, our numerical experiments suggest that one should adopt a hyper-parameter α to tune the peer term's contribution to make peer loss work in multi-class classification. Thus, the previous theory cannot completely explain the success of peer loss in practice, and there should be another way to explain the working mechanism of peer loss better.

In this section, we focus on developing an intuitive formulation that can help us better understand why and how peer loss works in practical learning with noisy label problems. It is worth noting that we only consider the expectation over the data distribution $\mathcal{D}$ rather than the empirical distribution D.

2.2.1 Deriving the divergence form

Let's start from the a special version of peer loss

$$\ell_{peer}\left(h\left(x_n\right), y_n\right) = \ell_{CE}\left(h\left(x_n\right), y_n\right) - \ell_{CE}\left(h\left(x_{n_1}\right), y_{n_2}\right),$$

where we focus on the CE loss ℓ_{CE} and $h(\cdot)$ is the output of the neural network. The pair (x_n, y_n) refer to the normal training samples, while (x_{n_1}, y_{n_2}) are the peer samples.

Taking expectation of ℓ_{peer} over the training data distribution $\mathcal{D} \sim \mathbb{P}(X, Y)$,

one finds

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[\ell_{peer}(h(X),Y)\right] &= \mathbb{E}_{\mathcal{D}}\left[\ell_{CE}(h(X),Y)\right] - \mathbb{E}_{\mathcal{D}_Y}\left[\mathbb{E}_{\mathcal{D}_X}\left[\ell_{CE}\left(h\left(X\right),Y\right)\right]\right] \\ &= -\int_{\Omega_X} dx \sum_{y\in[K]} \boldsymbol{P}(x,y)\log \boldsymbol{Q}(y|x) + \\ &\qquad \int_{\Omega_X} dx \sum_{y\in[K]} \boldsymbol{P}(x)\boldsymbol{P}(y)\log \boldsymbol{Q}(y|x),\end{aligned} \tag{2.3}$$

where $\mathcal{D}_X$ and $\mathcal{D}_Y$ are the marginal distribution for the features and labels, respectively. Ω_X denote the domain of feature x, and $[K] = \{1, 2, \cdots, K\}$ refers to the set of possible classes. In the last line, we plug in the explicit form of CE loss and write the expectation as a sum-integration over the domain of feature x and label y. The conditional probability $\boldsymbol{Q}(y|x)$ stands for the prediction of the underlying neural network model, while $\boldsymbol{P}(x,y)$ and $\boldsymbol{P}(x)\boldsymbol{P}(y)$ refer to the probabilities of the joint and marginal-product of the training distribution. We call the first sum-integration as *CE term* and the second as *peer term*.

The key observation is that $\boldsymbol{P}(x,y)$, $\boldsymbol{P}(x)$, and $\boldsymbol{P}(y)$ do not depend on the model parameters and, thus, will not contribute to the gradient. In this sense, we say such terms are "constant". Thus, one can add an arbitrary function of those "constant" terms to the loss function without affecting the resulting optimal classifier. For the CE term, we have

$$\begin{aligned}\text{CE term} \rightarrow & -\int_{\Omega_X} dx \sum_{y\in[K]} \left[\boldsymbol{P}(x,y)\log \boldsymbol{Q}(y|x)\boldsymbol{P}(x) - \boldsymbol{P}(x,y)\log \boldsymbol{P}(x,y)\right] \\ = & -\int_{\Omega_X} dx \sum_{y\in[K]} \boldsymbol{P}(x,y)\log \frac{\boldsymbol{Q}(x,y)}{\boldsymbol{P}(x,y)} \\ = & D_{KL}(\mathbb{Q}(X,Y)\|\mathbb{P}(X,Y)).\end{aligned}$$

In the above, we used $\boldsymbol{Q}(x,y) = \boldsymbol{Q}(y|x)\boldsymbol{P}(x)$ as in classification task the model prediction does not affect the feature distribution. So that optimizing the CE

term (i.e., the CE loss) leads to minimizing the K-L divergence between the joint distributions of the training data and the model prediction. The global minimum of this divergence lies at $\mathbb{Q}(X,Y) = \mathbb{P}(X,Y)$, i.e., when the model prediction completely fits the data distribution. Thus, we see intuitively the CE loss tends to overfit to the noisy data.

Similarly, for the peer term

$$\begin{aligned}\text{peer term} \to & \int_{\Omega_X} dx \sum_{y\in[K]} [\boldsymbol{P}(x)\boldsymbol{P}(y)\log \boldsymbol{Q}(y|x)\boldsymbol{P}(x) - \boldsymbol{P}(x)\boldsymbol{P}(y)\log \boldsymbol{P}(x)\boldsymbol{P}(y)] \\ = & \int_{\Omega_X} dx \sum_{y\in[K]} \boldsymbol{P}(x)\boldsymbol{P}(y)\log \frac{\boldsymbol{Q}(x,y)}{\boldsymbol{P}(x)\boldsymbol{P}(y)} \\ = & - D_{KL}(\mathbb{Q}(X,Y)\|\mathbb{P}(X)\times\mathbb{P}(Y)).\end{aligned}$$

Because of the minus sign, minimizing the peer term pushes $\mathbb{Q}(X,Y)$ away from the marginal-product $\mathbb{P}(X)\times\mathbb{P}(Y)$. Also, we note that there is no global minimum in this situation. Now the expectation of the peer loss can be rewritten as

$$\mathbb{E}_{\mathcal{D}}\left[\ell_{peer}(f(X),Y)\right] = D_{KL}(\mathbb{Q}(X,Y)\|\mathbb{P}(X,Y)) - D_{KL}(\mathbb{Q}(X,Y)\|\mathbb{P}(X)\times\mathbb{P}(Y)). \tag{2.4}$$

2.2.2 Intuition: Two anchors, correlation regularizer

Based on equation (2.4), there are two driving forces in the peer loss.

- In optimizing the CE term, the model prediction $\mathbb{Q}(X,Y)$ is getting closer to the data distribution $\mathbb{P}(X,Y)$. Certainly, this would cause overfit. Especially in the presence of noise, the model can overfit to the noisy signal in the training set.

- In the peer term, the K-L divergence measures the "distance"[1] between the marginal-product $\mathbb{P}(X) \times \mathbb{P}(Y)$ and the model prediction $\mathbb{Q}(X, Y)$. Since the distribution $\mathbb{P}(X) \times \mathbb{P}(Y)$ is the least correlated one, minimizing the peer term encourages the model prediction to encode more correlation between X and Y.

Intuitively, we see $\mathbb{P}(X, Y)$ and $\mathbb{P}(X) \times \mathbb{P}(Y)$ as two anchors that provide a reference for our optimization. Anchor $\mathbb{P}(X, Y)$ is an attractor and anchor $\mathbb{P}(X) \times \mathbb{P}(Y)$ is a repellent.

As the effects of the peer term are not definitive, i.e., it just encourages a higher correlation between X and Y in the model prediction without specifying a concrete target, we will call this term a *correlation regularizer*. We naturally expect to introduce a hyper-parameter to tune the importance of the correlation regularizer, thus resulting in the α peer loss. Putting in the divergence form, the α peer loss reads

$$\mathbb{E}_{\mathcal{D}}\left[\ell_{peer}(f(X), Y)\right] = D_{KL}(\mathbb{Q}(X, Y) \| \mathbb{P}(X, Y)) - \alpha \cdot D_{KL}(\mathbb{Q}(X, Y) \| \mathbb{P}(X) \times \mathbb{P}(Y)),$$

where α is the hyper-parameter that controls the contribution of the regularizer during the training.

In the next subsection, we will show that the above intuition is consistent with all the properties of peer loss and experimental observations.

2.2.3 Interpretation of properties of peer loss

Label noise tolerance: The first and foremost thing is to clarify why peer loss helps to reduce the effect of label noise. Here we assume that

[1] Just a metaphor, as K-L divergence is not a distance measure.

Adding noise will degrade the correlation between the features and the labels.

In other words, the noise should be "random" and make the data less informative[2]. By making this assumption, we exclude a subset of adversarial noise, which introduces a fake correlation that can be stronger than the original true signal. Dealing with such kinds of adversarial noise requires quite different techniques.

Pending the above assumption holds, overfitting to the noisy data (a likely result when training using the CE loss solely) would lead to a model prediction with less correlation between X and Y. Adding the correlation regularizer will push the model closer to the "correct direction" (this is an ambiguous term, please refer to the α tuning part for more discussion).

By the above discussion, we see that so long as the noise decreases the correlation in the dataset, peer loss *could* help, not limited to particular label noise settings, which are necessary for a complete cancellation of the noise. Thus we can apply peer loss to much broader situations in real-life practice.

α tuning: Even though setting $\alpha = 1$ could get a complete cancellation of the uniform and sparse label noise in theory, we found in practical training that it generally performs better when adopting an increasing α.

This observation can be readily explained in our divergence picture. We note that the correlation regularizer is "blind", as it does involve any restriction on optimizing direction. If we start with $\alpha = 1$ from the beginning, the SGD update might lead us to anywhere provided $\mathbb{Q}(X, Y)$ encodes a high correlation between X and Y. To avoid such a situation, we need to make use of the anchor $\mathbb{P}(X, Y)$ to narrow down the searching area. More precisely, we start from $\alpha = 0$, letting the model grasp the basic pattern in the dataset. Then, α is increased to regularize

[2]This assumption may be justified to some extent base on the Data Processing Inequality (DPI).

the resulting model for suppressing the overfitting to the label noise.

Hard decision boundary: In the experiments on the two-dimension synthetic dataset, we found when applying peer loss with large α (of course, in an increasing manner), the decision boundary is much narrow than using CE loss or peer loss with small α. We call such a narrow decision boundary a hard decision boundary, which means the model is quite confident when making predictions.

This phenomenon can be understood well in our divergence picture as well. Roughly speaking, larger α encourage higher correlation between X and Y in model prediction, and the highest correlation corresponds to a deterministic function, i.e. $\boldsymbol{P}(y = i|x) \approx 1$ for some class i and 0 for the other classes. Hence, the resulting model would be more confident with its prediction, hence possess harder decision boundary. For example, let's consider a binary classification problem and assume balanced prior for label distribution, the peer term reads

$$\begin{aligned}\text{peer term} =& \int_{\Omega_X} dx \sum_{y\in[2]} \boldsymbol{P}(x)\boldsymbol{P}(y) \log \boldsymbol{Q}(y|x) \\ =& \frac{1}{2} \int_{\Omega_X} dx \sum_{y\in[2]} \boldsymbol{P}(x) \log \boldsymbol{Q}(y|x) \\ =& \frac{1}{2} \int dx\, \boldsymbol{P}(x) \left[\log \boldsymbol{Q}(y = 1|x) + \log(1 - \boldsymbol{Q}(y = 1|x))\right].\end{aligned}$$

Note the integrand is of form $\log t + \log(1 - t)$, $t \in (0, 1)$, which takes its minimum at $t \to 0$ or $t \to 1$. This implies $\boldsymbol{Q}(y = 1|x) \to 0$ or 1 are preferred solution, and such case, where relation between x and y become almost deterministic, represently the higheast level correlation between X and Y.

Remark 2. *The peer term could over-regularize the training, especially when working on a clean dataset. However, we argue that, at least for image classification tasks, the peer term will not harm. This is because, in practice, the*

distribution of images from different classes are usually well separated, i.e., it is rare that one can not distinguish a dog from a cat. Mathematically, one may define a well-separated dataset as the one with $\boldsymbol{P}(y = i|x) \approx 1$ *for a specific class* i *and* ≈ 0 *otherwise.*

Remark 3. *The divergence formulation also indicates possible extensions to peer loss. One option is to use a general* f*-divergence rather than the K-L divergence for the correlation regularizer, i.e.* $D_f(\mathbb{Q}(X,Y)\|\mathbb{P}(X) \times \mathbb{P}(Y))$*. This is one of our ongoing research projects. We found that even though the intuition does not change, the choice of* f*-divergence does matter. In other words, the correlation regularizer should contribute similarly to the CE term during the training.*

2.3 Peer loss as the correlation risk

From the divergence viewpoint, we can intuitively explain the working mechanism of peer loss in practical training. However, it is very hard to draw concrete conclusions on the properties of the resulting optimal classifier. This is certainly not satisfactory. Can we do better? It turns out that peer loss can be re-interpreted as the correlation risk, which differs from the population risk and does not target at learning the Bayes optimal.

In this section, we will show that the trace of the Correlated Agreement (CA) matrix Δ provides a measure of classification performance, which puts more emphasis on the minor classes than the accuracy measure does. We call this new measure the correlation measure. Correlation measure matches the correlation risk minimization just as accuracy measure matches the population risk minimization. Thus, it is possible to build a similar framework for correlation risk as to the one for Bayes risk.

Let us start from a brief review of the accuracy measure and its relation to the Bayes optimal classifier and the population risk.

2.3.1 The accuracy measure, Bayes optimal classifier, population risk and classification calibration

The confusion matrix $\mathbb{C}$ of a classifier is defined by $\mathbb{C}_{ij} := \boldsymbol{P}(\hat{y} = i, y = j)$. We use $\hat{y}$ to indicate the classifier's output, and y to indicate the label from the dataset. The accuracy of the classifier is the trace of the corresponding confusion matrix, i.e., $\mathrm{tr}(\mathbb{C})$. Furthermore,

$$\begin{aligned}
\mathrm{tr}(\mathbb{C}) &= \sum_{i\in[K]} \boldsymbol{P}(\hat{y} = i, y = i) \\
&= \int_{\Omega_X} dx \sum_{i\in[K]} \boldsymbol{P}(\hat{y} = i, y = i, x) \\
&= \int_{\Omega_X} dx \sum_{i\in[K]} \boldsymbol{P}(\hat{y} = i | y = i, x) \boldsymbol{P}(x, y = i) \\
&= \int_{\Omega_X} dx \sum_{i\in[K]} \boldsymbol{P}(\hat{y} = i | x) \boldsymbol{P}(x, y = i),
\end{aligned}$$

where Ω_X is the domain of the feature x, and $[K] = \{1, 2, \cdots, K\}$ refers to the K classes. In the last step, we use the fact that the output classifier only depends on the input feature x. Here, we emphasize agian the distinction between the prediction of a neural network model $h_y(x)$ and the prediction of the corresponding classifier $cl(x)$. $h_y(x)$ models the conditional probability of getting label y given the feature x, which we will denote as $\boldsymbol{Q}(y|x)$. On the other hand, $cl(x)$ only assigns a specific label l^* for any input x. In the above equation, $\boldsymbol{P}(\hat{y} = i|x)$ is the conditional probability corresponds to $cl(x)$ rather than $h_y(x)$. We have

$$\boldsymbol{P}(\hat{y} = i|x) = \delta_{i;l^*},$$

namely, $\boldsymbol{P}(\hat{y} = i|x) = 1$ if $i = l^*$ and 0 otherwise. Hence, the summation in the above integrand becomes

$$\sum_{i\in[K]} \boldsymbol{P}(\hat{y} = i|x)\boldsymbol{P}(x, y = i) = \boldsymbol{P}(x, y = l^*).$$

Recall that there is a special classifier called the Bayes optimal classifier, which is defined as

$$cl^*_{Bayes}(x) = \arg\max_{i\in[K]} \boldsymbol{P}(x, y = i).$$

The Bayes optimal classifier chooses the label with the highest probability at each x as its prediction. The accuracy of the Bayes optimal classifier is the highest one among all possible classifiers. In this sense, we say that *the Bayes optimal classifier matches the accuracy measure*.

To learn the Bayes optimal classifier, the most direct way is to apply 0-1 loss

$$\mathbb{1}[cl(x), y] := \begin{cases} 1, & \text{if } cl(x) \neq y, \\ 0, & \text{otherwise.} \end{cases}$$

In expectation over the dataset distribution $\mathcal{D}$, we have $\mathbb{E}_{\mathcal{D}}[\mathbb{1}[cl(x), y]]$ equals to $1-\text{tr}(\mathbb{C})$. Thus, minimizing $\mathbb{1}[cl(x), y]$ over $\mathcal{D}$ will lead to the Bayes optimal classifer. However, an optimization problem contains 0-1 loss can not be solved easily as it is not differentiable. To resolve this difficulty, people seeks a class of differentiable convex surrogates which are classification-calibrated [25]. For those classification-calibrated functions $\ell(\cdot, \cdot)$, minimizing its population risk $\mathbb{E}_{\mathcal{D}}[\ell(h(x), y)]$ leads to Bayes optimal classifier. Normally used convex loss functions, such as $-\log$, are classification calibrated.

2.3.2 The correlation measure

The Correlated Agreement (CA) matrix Δ is introduced by Shnayder et al. [24], which is defined as $\Delta_{ij} := \boldsymbol{P}(\hat{y} = i, y = j) - \boldsymbol{P}(\hat{y} = i)\boldsymbol{P}(y = j)$. We define the correlation measure as the trace of the Δ matrix,

$$\mathrm{tr}(\Delta) = \sum_{i \in [K]} \boldsymbol{P}(\hat{y} = i, y = i) - \boldsymbol{P}(\hat{y} = i)\boldsymbol{P}(y = i). \tag{2.5}$$

We can further expand the above expression as

$$\begin{aligned}
\mathrm{tr}(\Delta) &= \int_{\Omega_X} dx \sum_{i \in [K]} [\boldsymbol{P}(\hat{y} = i, y = i, x) - \boldsymbol{P}(\hat{y} = i, x)\boldsymbol{P}(y = i)] \\
&= \int_{\Omega_X} dx \sum_{i \in [K]} \boldsymbol{P}(\hat{y} = i|x)[\boldsymbol{P}(x, y = i) - \boldsymbol{P}(x)\boldsymbol{P}(y = i)],
\end{aligned}$$

where $\boldsymbol{P}(\hat{y} = i|x)$ share the same meaning as in the accuracy measure case. We can define a correlation optimal classifier as follows

$$cl^*_{corr}(x) = \arg\max_{i \in [K]} [\boldsymbol{P}(x, y = i) - \boldsymbol{P}(x)\boldsymbol{P}(y = i)].$$

This correlation optimal classifier maximize the correlation measure. In other word, *the correlation optimal classifier matches the correlation measure.*

Properties of the correlation measure and its usefulness

The first term in the correlation measure is just the accuracy, while the second term refers to a trade-off between accuracy and a balanced prediction. To see this, let us rewrite the second term in definition (2.5) as

$$-\sum_i a_i p_i, \quad s.t. \sum_i a_i = 1, a_i \geq 0 \text{ and } \sum_i p_i = 1, p_i \geq 0.$$

We adopt $a_i = \boldsymbol{P}(y = i)$ and $p_i = \boldsymbol{P}(\hat{y} = i)$. Let a_{min} be the minimum value of a_i. Maximizing this term alone gives

$$p_i = \begin{cases} 1, & \text{if } a_i = a_{min}, \\ 0, & \text{otherwise,} \end{cases}$$

and the maximum value is $-a_{min}$. Here, for simplicity, we assume a_i does not degenerate. We find that this term pushes the prediction to emphasize the least populated class in the data. This tendency is contradicted to the first term, i.e., the accuracy.

Therefore, we conclude that compared to the accuracy measure, the correlation measures weight more of the model performance on minor classes, and can be useful in the fields such as fairness in machine learning. Besides, we will show in later sections and chapter 3 that the correlation risk provides a systematic way of maximizing the correlation measure, while there does not exist such systematic method to maximize many other measures, say the F1 score.

2.3.3 The correlation risk and dynamical calibration

The correlation risk is inspired by the expectation of peer loss over the data distribution, which is

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}\left[\ell_{peer}(h(X), Y)\right] &= \mathbb{E}_{\mathcal{D}}\left[\ell_{CE}(h(X), Y)\right] - \mathbb{E}_{\mathcal{D}_Y}\left[\mathbb{E}_{\mathcal{D}_X}\left[\ell_{CE}\left(h\left(X\right), Y\right)\right]\right] \\ &= -\int_{\Omega_X} dx \sum_{y\in[K]} \boldsymbol{P}(x, y) \log \boldsymbol{Q}(y|x) + \\ &\qquad \int_{\Omega_X} dx \sum_{y\in[K]} \boldsymbol{P}(x)\boldsymbol{P}(y) \log \boldsymbol{Q}(y|x) \\ &= \int_{\Omega_X} dx \sum_{y\in[K]} -\log \boldsymbol{Q}(y|x)[\boldsymbol{P}(x, y) - \boldsymbol{P}(x)\boldsymbol{P}(y)]. \end{aligned}$$

Comparing to the definition of the population risk, we can define the correlation risk for a differentiable loss function ℓ as

$$\begin{aligned} R_c(\ell) :=& \mathbb{E}_c[\ell(h(x), y)] \\ =& \int_{\Omega_X} dx \sum_{y \in [K]} \ell(h(x), y)[\boldsymbol{P}(x, y) - \boldsymbol{P}(x)\boldsymbol{P}(y)]. \end{aligned}$$

Note that the weight is no longer the probability $\boldsymbol{P}(x, y)$ but the correlation term $\boldsymbol{P}(x, y) - \boldsymbol{P}(x)\boldsymbol{P}(y)$. Usually, the correlation term takes both positive and negative values. We find that the new risk is in the form as the difference between two convex functions (assuming ℓ is convex), which breaks the proof of classification calibration property in [25].

To justify the name of correlation risk, we need to show that optimizing this risk will maximize the correlation measure defined in the previous subsection. Apart from the 0-1 loss case, this property is very hard to establish using conventional methods. The difficulties root in the nonconvexity nature of the objective. To address these difficulties, we shall take a dynamic view. More specifically, we will study the stable steady states in the searching space, through which we can show that optimizing the correlation riks indeed encourage the model to grasp the positive correlations in the data. We defer a full description of the dynamical analysis to chapter 3.

For the completion of the discussion in this section, we will state several main results from chapter 3 here for reference. Training with the correlation risk, in the best case, for all x we have the network output to be

$$h_y(x) = \begin{cases} \mathcal{N}[\boldsymbol{P}(y|x) - \boldsymbol{P}(y)], & \text{if} \quad \boldsymbol{P}(y|x) - \boldsymbol{P}(y) > 0, \\ 0, & \text{otherwise}, \end{cases}$$

where $\mathcal{N}$ is a normalization factor. Note that this $h_y(x)$ only captures the positive correlation between the feature x and label y. It can be shown that if we include the negative correlation part in the above definition, then the resulting $h_y(x)$ cannot be optimal, i.e., one can find an update $\Delta_y(x)$ such that the correlation risk decreases. For a detailed proof, please refer to chapter 3.

It is straightforward to see that the corresponding classifier of the above $h_y(x)$ is equivalent to the correlation optimal classifier. This property validates the name of correlation risk. For the general case, even though there are enormous sub stable steady state, we have the following stronger conjecture

Conjecture 1. *In practical training, under the dynamics induced by the Peer Loss, the model state tends to converge to the stable steady states in the positively correlated case.*

In this conjecture, the positively correlated case refers to a specific type of $h_y(x)$. Such $h_y(x)$ takes positive values on those x and y such that $\boldsymbol{P}(y|x)-\boldsymbol{P}(y) > 0$, and takes value 0 otherwise. For a detailed discussion, please refer to the section 3.4.2.

Within the correlation risk framework, we can provide a more quantitative explanation of the α tuning. We will introduce a α correlation risk

$$\mathbb{E}_{c;\alpha}[\ell(h(x), y)] = \int_{\Omega_X} dx \sum_{y\in[K]} \ell(h(x), y)[\boldsymbol{P}(x, y) - \alpha \boldsymbol{P}(x)\boldsymbol{P}(y)],$$

which corresponds to the expectation of the α peer loss. Tuning α is to tune the threshold of positive correlation. If we properly increase α during the training, then eventually, only the label with the largest correlation can have positive weight $\boldsymbol{P}(x, y) - \alpha \boldsymbol{P}(x)\boldsymbol{P}(y)$. If the conjecture mentioned above holds, then we are more likely to learn the largest positive correlation rather than other sub-optimal solutions.

Remark 4. *Note that even though we include α in the correlation risk, there is no need to include α in the correlation measure.*

Our dynamical analysis is based on the CE loss instead of the 0-1 loss; hence it can also be viewed as a calibration argument, which we named the dynamical calibration.

Remark 5. *The correlation risk viewpoint also provides a new way to look at the label noise tolerance property of peer loss. Actually, it is of interest to compare with the population risk case:*

- *If $\boldsymbol{P}(x, y') > \boldsymbol{P}(x, y), \forall y \neq y'$ holds in both the clean and noisy distribution with the same y', we say the order of $\boldsymbol{P}(x, y)$ is preserved. Any noise that does not change the order of $\boldsymbol{P}(x, y)$ will not affect the Bayes optimal classifier.*

- *Similarly, any noise that does not change the order of $\boldsymbol{P}(x, y) - \boldsymbol{P}(x)\boldsymbol{P}(y)$ will not affect the correlation optimal classifier.*

- *It is possible to find a noise* [3] *that changes the order of $\boldsymbol{P}(x, y)$ while preserving the order of $\boldsymbol{P}(x, y) - \boldsymbol{P}(x)\boldsymbol{P}(y)$. Likewise, one can also find a noise that changes the order of $\boldsymbol{P}(x, y) - \boldsymbol{P}(x)\boldsymbol{P}(y)$ while preserving the order of $\boldsymbol{P}(x, y)$.*

Thus, we find it is hard to say which of the correlation risk and the population risk is better in tolerating data noise in the general case. We would like to further

[3] As an example, let us consider a binary classification task on noisy dataset with noise rates $e_+ = 0.1$ and $e_- = 0.8$. Suppose in the clean dataset $\boldsymbol{P}(x_0, y = +1) = \boldsymbol{P}(x_0, y = -1)$ at x_0. Then, in the noisy dataset, we find

$$\boldsymbol{P}(x_0, \tilde{y} = +1) = 0.9\boldsymbol{P}(x_0, y = +1) + 0.8\boldsymbol{P}(x_0, y = -1)$$

and

$$\boldsymbol{P}(x_0, \tilde{y} = -1) = 0.2\boldsymbol{P}(x_0, y = -1) + 0.1\boldsymbol{P}(x_0, y = +1).$$

Thus, $\boldsymbol{P}(x_0, \tilde{y} = +1) > \boldsymbol{P}(x_0, \tilde{y} = -1)$, and the order of $\boldsymbol{P}(x, y)$ has been changed.

ask about the difference between the types of noise that are tolerant under the population riks and the correlation risk, respectively. This is one of our ongoing research projects.

2.4 Chapter summary

Section 2.1 considers the extension of peer loss to the multi-class classification tasks, emphasizing the ability to recover clean Bayes optimal classifier and the exact cancellation of certain types of label noise. For those properties to hold in multi-class classification, we need to make more restricted assumptions on the label noise than in binary classification case. Furthermore, it is hard to explain why peer loss works in practical applications within this formulation. To address these difficulties, we developed a divergence formulation of peer loss in section 2.2, which provides an intuitive explanation of the working mechanism of peer loss in practice. Primarily, we recognize the peer term as a correlation regularizer that encourages the model prediction to encodes more correlation between the features and labels. However, there lack accurate descriptions about the properties of the resulting optimal classifier. In other words, the divergence formulation tells us where peer loss pushes the model but provides little information about the optimal classifier's concrete properties. Hence, in section 2.3, we turn to the third viewpoint, identifying peer loss as a correlation risk. It is demonstrated that training with peer loss, we indeed aim at a different target comparing to the population risk. The full analysis requires techniques from the evolutionary game theory [2], whose details are deferred to the next chapter.

Chapter 3

Dynamical Analysis

In the previous chapter, we discussed the extension of the original peer loss to the multi-class classification tasks and studied the intuition and working mechanism behind peer loss. Especially in section 2.3, we investigated the possibility of interpreting peer loss as an instance of correlation risk. To complete the correlation risk framework, we need a calibration property which links the correlation risk and the optimal correlation classifier. This is the focus of the present chapter. As already discussed, the nonconvex feature of the correlation prevents us from establishing the calibration property by conventional method [25]. We will instead adopt the method from the evolutionary game theory [2], studying the stable steady states in the searching space. In this way, we can get a dynamical version of the calibration property. We also emphasize that the present dynamical analysis is not complete. However, we believe it provides a promising way to understand the training dynamics using peer loss (hence, the correlation risk).

In the following, we will use the terms *peer loss* and *correlation risk* interchangeably, as their formulations are identical.

3.1 Assumptions

In this section, we only consider the correlation risk with CE loss, i.e.

$$\mathbb{E}_c[\ell_{CE}(h(x), y)] = -\int_{\Omega_X} dx \sum_{y\in[K]} \log h_y(x)[\boldsymbol{P}(x, y) - \boldsymbol{P}(x)\boldsymbol{P}(y)].$$

This can be regarded as the limit of the empirical correlation risk when the number of samples tends to infinite. Further, we have the following three assumptions:

A1. The model capacity is infinite (i.e., it can realize arbitrary variation).

A2. The model is updated using SGD algorithm (i.e. updates follow the decreasing $\mathbb{E}_c[\ell_{CE}(h(x), y)]$ direction).

A3. The derivative of network function $\frac{\partial f_y(x;w)}{\partial w_i}$ is smooth (i.e. the network function has no singular point).

The infinite capacity assumption is somehow justified in practice as we usually have an over-parametrized neural network model comparing to the scale of the underlying task.

3.2 Searching space geometry

At each feature x, the model prediction $h_y(x)$ satisfies $\sum_y h_y(x) = 1$. That is to say, assuming there are K classes, the model preidction is a point in a $(K-1)$-dimension simplex $\mathcal{S}_{K-1}$. Let Ω_X denote the set of features, the whole searching space will be $\Omega_X \times \mathcal{S}_{K-1}$. We call this searching space a hypersimplex as shown in figure 3.1 below.

Let Int_{K-1} and V_{K-1} denote the interior and vertices of $\mathcal{S}_{K-1}$, respectively. The interior and veritices of the hypersimplex are defined as the union of the

interior and vertices of all the $(K-1)$-simplex, i.e. $\text{Int}_{K-1}^{\Omega_X} := \bigcup\{\text{Int}_{K-1}(x)\}_{x\in\Omega_X}$ and $V_{K-1}^{\Omega_X} := \bigcup\{V_{K-1}(x)\}_{x\in\Omega_X}$. Then, the edges of the hypersimplex are defined as the complementory set of $\text{Int}_{K-1}^{\Omega_X} \bigcup V_{K-1}^{\Omega_X}$ in $\Omega_X \times \mathcal{S}_{K-1}$.

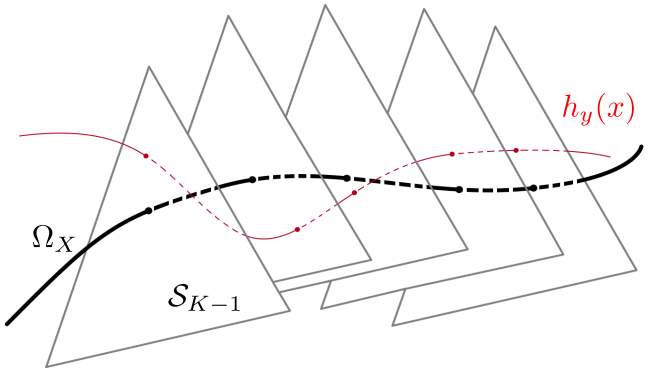


Figure 3.1: A sketch of the hypersimplex and $h_y(x)$ as a "curve"/"point" in it.

3.3 Preliminaries

In this chapter, we investigate the training dynamics of peer loss (treating as a special instance of the correlation risk). Thus, we care about the update rule and the corresponding variations of $h_y(x)$ and $\mathbb{E}_c[\ell_{CE}(h(x), y)]$. We summarize those quantities in this section for future reference. The full deduction can be found in the appendix.

When referring to the update rule, we mean SGD, i.e.

$$\delta w_i = -\eta \frac{\partial \mathbb{E}_c[\ell_{CE}(h(x;\mathbf{w}), y)]}{\partial w_i},$$

where η is the learning rate and $\mathbf{w}$ (and its i-th element w_i) refers to the model parameters, e.g. the weights and bias in each layer of a neural network model. Note in the above expression, we explicitly write down the dependence of the model prediction h on $\mathbf{w}$. In later content, we will suppress this explicit depen-

dency when it makes no misunderstanding. Corresponding to this change in w, the variations of $h_y(x)$ and $\mathbb{E}_c[\ell_{CE}]$ are

$$\begin{aligned}\delta h_y(x) &:= \Delta_y(x) \\ &= h_y(x) \cdot \eta \int_{\Omega_X} dx' \sum_{y' \in [K]} [\boldsymbol{P}(x', y') - \boldsymbol{P}(x')\boldsymbol{P}(y')] \sum_i G_i(x, y) G_i(x', y')\end{aligned} \tag{3.1}$$

and

$$\delta \mathbb{E}_c[\ell_{CE}] = -\int_{\Omega_X} dx\, \boldsymbol{P}(x) \sum_{y \in [K]} \Delta_y(x) \frac{\boldsymbol{P}(y|x) - \boldsymbol{P}(y)}{h_y(x)}, \tag{3.2}$$

respectively. In equation (3.1), we have used abbreviation

$$G_i(x, y) = -\frac{\partial g_y(x)}{\partial w_i} + \sum_{y' \in [K]} h_{y'}(x) \frac{\partial g_{y'}(x)}{\partial w_i},$$

where $g_y(x)$ is the network output before the softmax activation, i.e. $h_y(x) = \exp(g_y(x)) / \sum'_y \exp(g_{y'}(x))$. We will assume $g_y(x)$ and $\frac{\partial g_y(x)}{\partial w_i}$ are smooth functions.

There is an important implication of equation (3.1), namely, if $h_y(x) = 0$ then corresponding $\Delta_y(x) = 0$ as well. Note that this property holds even when we use the Cross-Entropy loss with cut-off (i.e. use $-\log(h_y(x) + \epsilon)$ rather than $-\log(h_y(x))$).

Directly solving the evolution of the model prediction $h_y(x)$ in the searching space is infeasible. Hence, we turn to look for those steady states, in which $\delta \mathbb{E}_c[\ell_{CE}] = 0$ for arbitrary variation $\Delta_y(x)$ that satisfies $\sum_{y \in [K]} \Delta_y(x) = 0$. In other states, there exist an update that reduces the correlation risk $\mathbb{E}_c[\ell_{CE}]$, and, thus, cannot be optimal. Moreover, we are interested in the stable steady states, which are the local minima in the hypersimplex. They are the only reachable states in practical training.

3.4 Steady states and their stability

3.4.1 Steady states in the interior of the hypersimplex

Proposition 1. *There is no steady state in the interior of the hypersimplex.*

At a specific x_0, the summation in the integrand of equation (3.2) reads

$$F(x_0) := \sum_{y\in[K]} \Delta_y(x_0)\frac{\boldsymbol{P}(y|x_0) - \boldsymbol{P}(y)}{h_y(x_0)}. \tag{3.3}$$

Let us split the labels y into the following two sets (without loss of generality, we ignore the $\boldsymbol{P}(y|x_0) - \boldsymbol{P}(y) = 0$ cases):

$$\mathbb{Y}_{x_0;-} = \{y : \boldsymbol{P}(y|x_0) - \boldsymbol{P}(y) < 0\}$$

and

$$\mathbb{Y}_{x_0;+} = \{y : \boldsymbol{P}(y|x_0) - \boldsymbol{P}(y) > 0\}.$$

Now, by assigning $\Delta_y(x_0) = a_y < 0, \forall y \in \mathbb{Y}_{x_0;-}$ and $\Delta_y(x_0) = b_y > 0, \forall y \in \mathbb{Y}_{x_0;+}$, one finds $F(x_0) > 0$ since $h_y(x_0) > 0$ for interior states. To accommodate the constraint $\sum_y \Delta_y(x_0) = 0$, we can multiply every b_y a normalization factor $\mathcal{N}_b$ so that

$$\sum_{y\in\mathbb{Y}_-} a_y + \mathcal{N}_b \cdot \sum_{y\in\mathbb{Y}_+} b_y = 0,$$

which is always possible. Let $B_\epsilon(x_0)$ be a ϵ-neighbourhood of x_0. Because $h_y(x)$ is continuous, we can set[1] $\Delta_y(x) = \frac{1}{2}(1 + \cos\frac{\pi\|x-x_0\|}{\epsilon})\Delta_y(x_0), \forall x \in B_\epsilon(x_0)$ and 0 otherwise. This choice will lead to $\delta\mathbb{E}_c[\ell_{CE}] < 0$.

The above argument demonstrates the existence of a variation $\Delta_y(x)$, which leads to a decrease in $\mathbb{E}_c[\ell_{CE}]$ for all interior states of the hypersimplex. Under

[1]The coefficient $\frac{1}{2}(1 + \cos\frac{\pi\|x-x_0\|}{\epsilon})$ is added so that the continuity of $h_y(x)$ preserves.

the assumption of infinite model capacity, SGD is always able to discover such an update. The trajectory of the model state will approach the edges and vertices of the hypersimplex under the dynamics induced by peer loss.

3.4.2 Steady states in the edges and vertices of the hypersimplex

As the model state approaches the edges and vertices of the hypersimplex, there must be at least one x_0, at which $h_y(x_0) \to 0$ for some y. Equation (3.1) tells us

Proposition 2. *For any* (x, y), $h_y(x) = 0$ *implies* $\Delta_y(x) = 0$.

Thus, the model state can never escape from an edge nor vertex after arriving at it, as the corresponding component $h_y(x) = 0$ will always be 0. In this way, the effective dimensionality of the simplex at x_0 is reduced. Now, if in this simplex $\boldsymbol{P}(y|x) - \boldsymbol{P}(y)$ still has both positive and negative values, then we can repeat the argument in section 3.4.1 and conclude that the state is not stable. Such process can be performed recurrently until for all $x \in \Omega_X$ either $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) < 0$ or $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) > 0$ for all y such that $h_y(x) \neq 0$.

More rigorously, we define the set of nonzero components of the model prediction at x

$$\mathcal{Y}_x := \{y | h_y(x) \neq 0\}.$$

Then, one can distinguish the following three cases of states in the hypersimplex that can support steady states:

1. For all $x \in \Omega_X$, $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) > 0, \forall y \in \mathcal{Y}_x$, which is called the *positively correlated* case.

2. For all $x \in \Omega_X$, $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) < 0, \forall y \in \mathcal{Y}_x$, which is called the *negatively correlated* case.

3. This case is the mixture of the case 1 and 2, namely for some x one has $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) > 0, \forall y \in \mathcal{Y}_x$, while for the other x one has $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) < 0, \forall y \in \mathcal{Y}_x$. We call this the *mixed* case.

Next, we give the explicit expression of the steady states in the above three cases. At any x, we have a special assignment for $h_y(x)$

$$h_y(x) = \begin{cases} \mathcal{N}|\boldsymbol{P}(y|x) - \boldsymbol{P}(y)|, & \forall y \in \mathcal{Y}_x \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathcal{N}$ is the normalization factor that ensures $h_y(x)$ summed to 1. By equation (3.3), it is straightforward to see $F(x) = 0, \forall x \in \Omega_X$ for arbitrary $\Delta_y(x)$. Hence, $\delta\mathbb{E}_c[\ell_{CE}] = 0$ for arbitrary $\Delta_y(x)$. Based on the above arguments, these are all the possible steady states in the hypersimplex.

Remark 6. *Since $\boldsymbol{P}(y|x) - \boldsymbol{P}(y)$ represent the correlation between x and y relative to the average label distribution. We will call those y with $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) > 0$ positively correlated labels and those y with $\boldsymbol{P}(y|x) - \boldsymbol{P}(y) < 0$ negatively correlated labels.*

Remark 7. *These three cases include both the steady states on the edges as well as the vertices. This is because we only require $\boldsymbol{P}(y|x) - \boldsymbol{P}(y)$ have the same sign for $y \in \mathcal{Y}_x$, thus, $\mathcal{Y}_x$ may not include all the positively/negatively correlated labels.*

Remark 8. *It is possible to relate the correlation optimal classifier to the Bayes optimal classifier. We know that the uniform off-diagonal noise preserves the order*

of the correlation term $\boldsymbol{P}(x,y) - \boldsymbol{P}(x)\boldsymbol{P}(y)$*, i.e.,*

$$\boldsymbol{P}(x,\tilde{y}) - \boldsymbol{P}(x)\boldsymbol{P}(\tilde{y}) = \left(1 - \sum_{\ell} e_\ell\right)[\boldsymbol{P}(x,y) - \boldsymbol{P}(x)\boldsymbol{P}(y)].$$

At the **best situation** *of the positively correlated case, the classifier* $h_y(x)$ *will fully capture the positive correlation in the dataset. Then, one can successfully recover the label with the largest correlation by prediction rule*

$$y_x^* = \arg\max_{y\in[K]} h_y(x).$$

Further, when the clean prior is balanced, the order of correlation $\boldsymbol{P}(x,y) - \boldsymbol{P}(x)\boldsymbol{P}(y)$ *is the same as the joint probability* $\boldsymbol{P}(x,y)$*. Thus, we recover the Bayes optimal classifier.*

3.4.3 Stability of the steady states

The positively correlated case: In this case, all the nonzero components of the model prediction correspond to the positively correlated labels. On such edges of the hypersimplex, the situation is much similar to the conventional Cross-Entropy loss case. It follows these steady states are stable. Recall the assignment

$$h_y(x) = \begin{cases} \mathcal{N}|\boldsymbol{P}(y|x) - \boldsymbol{P}(y)|, & \forall y \in \mathcal{Y}_x \\ 0, & \text{otherwise,} \end{cases}$$

Particularly, in the best situation, where $\mathcal{Y}_x$ includes all the positively correlated labels for all $x \in \Omega_X$, we see the stable steady state reflects the CA apart from all the negative values are cut off. In this sense, $\arg\max_y h_y(x)$ will give the component corresponding to the maximum correlation, i.e., being the correlation

optimal classifier.

We also need to look at the value of the correlation risk, which is

$$\mathbb{E}_c[\ell_{CE}] = \int_{\Omega_X} dx \boldsymbol{P}(x) \left\{ - \sum_{y\in[K]} [\boldsymbol{P}(y|x) - \boldsymbol{P}(y)] \log h_y(x) \right\}. \tag{3.4}$$

Easiely, one find at such stable steady state $\mathbb{E}_c[\ell_{CE}] \to -\infty$.

Remark 9. *In the above argument, we treat ∞ as a normal number so that even when $\mathbb{E}_c[\ell_{CE}] \to -\infty$ the model state can continue evolving and approach the steady state at the edge. This is somehow justified as, in practice, we usually use $\log(x + \epsilon)$ in computing the loss. Hence, one would get a negative value with a large magnitude rather than $-\infty$. We will adopt this convention for all arguments in the present subsection.*

The negatively correlated case: This case only differs from the positively correlated case by a minus sign. It almost changes everything. Now the steady state is a maximum on edge and thus is not stable. The model state will tend to converge to a vertex of the edge. Mathematically, this means for all $x \in \Omega_X$

$$h_y(x) = \begin{cases} 1, & \text{for one } y^* \text{ such that } \boldsymbol{P}(y^*|x) - \boldsymbol{P}(y^*) < 0 \\ 0, & \text{otherwise.} \end{cases}$$

Note that y^* will depend on the details of the training process, such as the initialization. Note that $\mathbb{E}_c[\ell_{CE}] \to \infty$ even for the above solution. We conclude the negatively correlated case will **not** be favored during practical training.

The mixed case: To analyse this case, we must rely on equation (3.4), i.e. the explicit expression for $\mathbb{E}_c[\ell_{CE}]$. Now, at different x, the integrand may contribute

∞ as well $-\infty$. We observe that only when the contribution from the positively correlated part dominates, the equation (3.4) gives $\mathbb{E}_c[\ell_{CE}] \to -\infty$. Those are all the steady states that could be reached during practical training. (Similar to the local minima)

It may be argued that the mixed case is not as favorable as the positively correlated case in practice. This is because we need a delicate balance between the positively correlated part and the negatively correlated part so that the steady states in the mixed case can be reachable

More concretely, let's consider a training process that just begins. The model state will gradually approach the edges. Let x_0 be the first point such that $h_{y_0}(x_0) = 0$. We claim $\boldsymbol{P}(y_0|x_0) - \boldsymbol{P}(y_0) < 0$, since otherwise the integrand of equation (3.4) at x_0 will contribute a ∞ and cause a significant increase in $\mathbb{E}_c[\ell_{CE}]$. From a different viewpoint, to make $\boldsymbol{P}(y_0|x_0) - \boldsymbol{P}(y_0) > 0$ possible, we must have at least another $h_{y_1}(x_1) = 0$ for which $\boldsymbol{P}(y_1|x_1) - \boldsymbol{P}(y_1) < 0$, so that their contributions to the integrand in equation (3.4) can be balanced and, thus, give rise to a decreasing $\mathbb{E}_c[\ell_{CE}]$. Such coincidences would be less possible. Hence, we have the following conjecture:

Conjecture 1. *In practical training, under the dynamics induced by peer loss, the model state tends to converge to the stable steady states in the positively correlated case.*

3.4.4 The α peer loss case

The above arguments can be extended to the α peer loss, which corresponds to the α correlation risk. In that case, the correlation term is defined as $\boldsymbol{P}(x, y) - \alpha\boldsymbol{P}(x)\boldsymbol{P}(y)$. The definition of the positively correlated case also need to be updated as for all $\boldsymbol{x} \in \Omega_X$, $\boldsymbol{P}(x, y) - \alpha\boldsymbol{P}(x)\boldsymbol{P}(y) > 0, \forall y \in \mathcal{Y}_X$. A similar update

applies to the definition of the negative correlation.

We note that the arguments in this section are not affected as long as $\boldsymbol{P}(x, y) - \alpha \boldsymbol{P}(x)\boldsymbol{P}(y)$ can still take both positive and negative values at any x. This provides a rough guide to the upper bound of the magnitude of α:

$$\alpha < \min_{x,y} \frac{\boldsymbol{P}(x, y)}{\boldsymbol{P}(x)\boldsymbol{P}(y)}.$$

Also, we see that tuning α is to tune a threshold, which determines the labels that can be considered as positively correlated. In the extreme case, for any x there is only one y being positively correlated, and the magnitudes of the positive values are small compared to those negative values. We argue that, in this extreme situation, the mixed case is *less possible*. This is because the negative values in the term $\boldsymbol{P}(x, y) - \alpha \boldsymbol{P}(x)\boldsymbol{P}(y)$ are now dominant, making it harder to get a balance between the positively correlated part and the negatively correlated part. Even if the model indeed converges to the mixed case, it will be positively correlated at most of the x. We thus can expect the model tends to grasp the highest correlated label in the data distribution.

Remark 10. *We note the above mentioned upper bound is not a hard condition on the magnitude of α in practice. In practical training, there are many other factors that matter, e.g., the learning rate. Even if we have the α exceed the upper bound, as long as the learning rate is low enough, the resulting model will not be affected much. Thus, we would like to see such upper bound as guidance for us to tune the hyper-parameter α.*

Remark 11. *As an example of the suggested upper bound of α, let us consider a well-separated dataset of* 10 *classes. We further assume the label distribution is balanced. By well-separated, we refer to the data distribution whose probability*

satisfies $\boldsymbol{P}(y|x) \approx 1$ *for only one* y *and* ≈ 0 *otherwise. In this case, we see that the upper bound is approximately* 10.

Chapter 4

Experiments

In this chapter, we collectively present all the relevant experimental results. The first section on benchmark image datasets is included to show the effectiveness of peer loss in multi-class classification tasks with label noise. It is demonstrated that peer loss outperforms other state-of-the-art methods in almost all situations of our experiments. In section 4.2, we consider the experiments on the 2-dimension synthetic dataset, which endows a clear visualization of the decision boundary. We see that training with peer loss will encourage the model to learn a harder decision boundary.

4.1 Experiments on benchmark image datasets

In this section, we implement the multi-class peer loss to image classification with noisy labels tasks on MNIST [26], Fashion-MNIST [27], CIFAR-10 and CIFAR-100 [28], and Clothing1M [29]. Those are five standard datasets for image classification, whose contents range from small scale hand-written digits to huge scale real-life clothes. Different types of noise settings are considered, including the sophisticated synthetic noise for both label-independent and label-dependent

cases and the real-life human-level noise (feature-dependent). The robustness and advantages of the proposed peer loss are verified via testing on those noise settings and by comparing with other benchmark methods. Throughout the experiments, we adopt the CE loss for ℓ in peer loss. We now explain the details.

4.1.1 Baseline methods

Our experiments focus on the comparison with three baseline methods: **Cross-Entropy (CE)**, the **Backward (BLC) and Forward Loss Correction (FLC)** method as introduced in [17], and the **determinant-based mutual information (DMI)** method introduced in[18]. The forward and backward loss correction method is introduced in [17], which relies on the estimation of the transition matrix T. Thus the accuracy of estimating matrix T, which is restricted by the size of the dataset, could be a bottleneck of performance and robustness. DMI [18] implements an information-theoretic function into deep neural networks. Its implementation builds on the estimates of the joint distribution of classifier output and the noisy labels. For [18] and [17], we use the codes shared by the authors and adopt their reporting best parameters for performance comparison. [1]

It is reported that both the loss correction method [17] and DMI [18] work well when the noise in labels is reasonably low and sparse. In our experiments, we further test relatively dense noise settings for their frameworks.

[1]Some other recent approaches, e.g., [20, 19, 21], provide semi-supervised learning based solution which requires engineering efforts at different parts of the learning pipeline. We compare with the method that focuses on loss functions and believe the performance can be further enhanced by combining our loss function with appropriate pipeline designs, which is beyond the scope of this paper and left for future works.

4.1.2 Image datasets with synthetic label noise

In synthesizing noisy labels, we consider three different types of transition matrix T, i.e., sparse noise matrix, the uniform off-diagonal noise matrix, and random noise matrix. The first two types correspond to the label noise models mentioned in section 2.1.3, while in the last one, we flip the label of each sample to a randomly chosen one from all possible classes with a constant probability. The original clean labels are flipped according to those transition matrices. All the transition matrices can be found in the appendix. Note that we will reserve 20% randomly selected samples from the noisy training data for validation if a stand-alone validate dataset is not provided.

It is worth pointing out that, compared with the noise settings in [17] and [18], our noise settings are more disturbing to the learning problem. On the one hand, [17, 18] primarily considered sparse transition matrices with low overall noise rates, while we take the uniform off-diagonal noise into consideration. Note in this case $T_{i,j} \neq 0$ holds for every element in T now.

Furthermore, our sparse noise setting is potentially harder to handle compared to the ones documented in the state-of-the-arts results, e.g., [17, 18]. This is because we test cases where there is no class with clean labels (every class is contaminated with noisy labels), while the transition matrices considered in [17, 18] have a good number of classes containing only clean labels.

MNIST, Fashion-MNIST and CIFAR-10

We test all three types of synthetic noise settings on these three datasets. Each type includes a high-level noise setting and a low-level noise setting. The noise rates of low-level sparse and uniform noise are both about 0.2. The noise rates of high-level sparse and uniform noise are about 0.4 and 0.55, respectively. The

Dataset	Noise	CE (%)	BLC†(%)	FLC†(%)	DMI‡(%)	Peer loss (%)
MNIST	Sparse, Low	97.21	95.23	97.37	97.76	**98.82 (98.76 ± 0.04)**
	Sparse, High	48.55	55.86	49.67	49.61	**97.34 (97.14 ± 0.25)**
	Uniform, Low	97.14	94.27	95.51	97.72	**98.73 (98.69 ± 0.01)**
	Uniform, High	93.25	85.92	87.75	95.50	**98.41 (98.37 ± 0.04)**
Fashion MNIST	Sparse, Low	84.36	86.02	**88.15**	85.65	87.74 (87.63 ± 0.10)
	Sparse, High	43.33	46.97	47.63	47.16	**78.40 (77.81 ± 0.35)**
	Uniform, Low	82.98	84.48	86.58	83.69	**87.25 (87.10 ± 0.13)**
	Uniform, High	79.52	78.10	82.41	77.94	**83.50 (83.25 ± 1.09)**
CIFAR-10	Sparse, Low	87.20	86.01	87.83	89.50	**91.72 (91.67 ± 0.04)**
	Sparse, High	61.81	49.39	54.63	84.50	**88.94 (88.59 ± 0.34)**
	Uniform, Low	85.68	84.80	87.78	86.26	**89.29 (89.24 ± 0.25)**
	Uniform, High	71.38	68.19	81.33	72.98	**82.84 (82.69 ± 0.06)**
	Random, Low	78.40	72.49	77.12	79.57	**83.74 (83.64 ± 0.06)**
	Random, High	68.26	37.44	68.68	71.97	**74.16 (73.96 ± 0.16)**

† use a fully-connected network for MNIST and Fashion MNIST; ResNet-32 for CIFAR10.
‡ use a self-defined convolutional neural network for MNIST and Fashion MNIST; ResNet-18 for CIFAR10.

Table 4.1: Experiment results of five models: CE: Cross-Entropy, BLC: backward loss-correction , FLC: forward loss-correction , and peer loss, on MNIST, Fashion-MNIST, and CIFAR-10. In the peer loss column, we report the maximum accuracy (outside number) as well as the (mean ± standard deviation). For other methods, we report the maximum accuracy. The best performance in each row is highlighted in bold.

random noise is generated by randomly flipping a class to one of 10 classes w.p. 0.5 (low-level) or 0.7 (high-level).

We have carried out experiments on MNIST and Fashion-MNIST datasets using LeNet [26], the convolutional neural network used in DMI [18] as well as the fully-connected neural network used in loss-correction [17]. All the experiments are performed with batch size 128 and an initial learning rate of 1e-4. For LeNet and DMI's convolutional neural network, Adam [30] with default parameters is used as the optimizer, while for loss-correction's fully-connected neural network case we use AdaGrad [31] in order to be consistent with their works.

For CIFAR-10 dataset, ResNet-18 [32] is used as the backbone. In the experiments, stochastic gradient descent (SGD) is used as the optimizer with a weight decay of 1e-4. The batch size is set at 128. The learning rate starts at 0.1. For

Dataset	Noise	CE†(%)	BLC‡(%)	FLC‡(%)	DMI†(%)	Peer loss†(%)
CIFAR-100	Uniform	63.87	51.40	60.04	63.08	**67.94 (67.73 ± 0.13)**
	Sparse	40.45	36.57	43.39	38.54	**56.36 (56.13 ± 0.24)**
	Random (0.2)	65.84	61.21	61.52	66.03	**69.17 (68.96 ± 0.15)**
	Random (0.5)	56.92	22.21	55.88	57.27	**60.90 (60.72 ± 0.15)**
	Random (0.7)	40.80	4.06	43.16	41.27	**48.53 (48.36 ± 0.12)**

† use PreResNet-18. ‡ use ResNet-50.

Table 4.2: Experiment results of CIFAR-100. For Random (0.5) and Random (0.7) noise setting of CIFAR100, we provide BLC and FLC with the *ground truth transition matrix*.

every 40 epochs, it decays by a factor of 0.1. In all experiments for other methods, we use the best set of hyper-parameters they provided in similar settings.

We document the best performance for our baseline competitors, but we also take the average of peer loss's performance over five runs and record the standard variance. From Table 4.1, we observe that peer loss ranked as the best performing method for the majority of the time (except for one setting, where peer loss ranks as the second-best one). What of particular interests is that peer loss is robust across different noise models and noise rates, and looks particularly robust in high noise regime. For instance, in the MNIST dataset, with high sparse noise, the best performance of the rest baselines is about 55.86%, nonetheless peer loss is able to achieve 97.34% in accuracy.

CIFAR-100

For CIFAR-100 [28], all three types of synthetic noise are tested as well. We do not distinguish high- and low-level settings as the number of classes are large. The uniform type refers to the uniform off-diagonal noise defined previously, with an average noise rate of 0.25. The sparse label noise is generated by randomly dividing 100 classes into 50 pairs, and the flipping probability (T_{ji}, T_{ij}) in each pair is randomly choosing from $(0.05, 0.75)$, $(0.1, 0.70)$, $(0.15, 0.65)$, $(0.2, 0.6)$. The

Dataset	Noise	CE (%)	BLC (%)	FLC (%)	DMI (%)	Peer loss (%)
CIFAR-10	Manual Pair	47.02	58.82	78.08	77.55	**88.13 (87.93 ± 0.16)**
Clothing1M	Human Noise	68.94	69.13	69.84	72.46	**72.60**

Table 4.3: Results on manual-pair synthetic noise and real human-level noise. Milestones: [20, 50, 120], α-list: [0.0, 2.0, 5.0].

random noise is synthesized by randomly flipping each class to one of 100 classes w.p. 0.2, 0.5, or 0.7.

We use an 18-layer PreAct Resnet [33] and train it using SGD with a momentum of 0.9, a weight decay of 0.0005, and a batch size of 128. For uniform noise and random noise, we train the network with CE as a warm-up for 100 epochs [2], then apply peer loss with a learning rate of 0.0001 and $\alpha = 0.95$. For sparse noise, we adopt the following setting for α: Milestones: [10, 30, 100, 150]. α-list [0.0, 2.0, 10.0, 20.0].

Table 4.2 compares performance with different methods under different noise settings. We do observe that, again, peer loss consistently achieves the best performance. Since it is hard for BLC and FLC to achieve good performance with estimated transition matrix under the Random (0.5) and Random (0.7) noise setting of CIFAR-100, we use the *ground truth transition matrix* to train BLC and FLC which is calculated according to clean training labels and noisy training labels.

4.1.3 CIFAR-10 with manual-pair noise and Clothing1M

We are also very interested in the performance of peer loss on datasets with human-level noise. Hence, we defined manual-pair noise. Unlike the sparse label noise, we flip only within elaborately chosen pairs in the manual-pair noise case. Each pair contains two "similar" classes that are likely to be wrongly labeled by

[2]The learning rate starts at 0.1 and decays by a factor of 10 after 60 epochs.

humans, such as cat vs. dog, horse vs. deer. Besides, we are also interested in the performance of the proposed method with human-level label noise. For a more sophisticated test, we also carry out experiments on the Clothing1M dataset. Clothing1M [29] is a large-scale dataset with real-world noisy labels, which consist of one million noisy training images collected from online shopping websites.

We use the same setting as basic experiments for CIFAR-10, and use a ResNet-50 with ImageNet pre-trained weights for Clothing1M. Table 4.3 shows that peer loss works reasonably well in both of the manual-pair noise and the feature-dependent human-level noise.

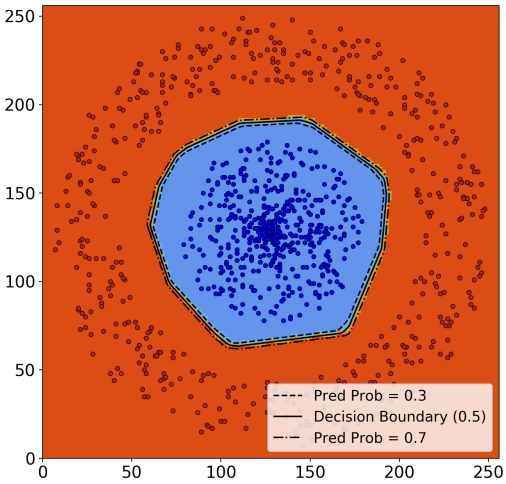
4.2 Experiments on synthetic dataset

The experiments on the benchmark image datasets support the effectiveness of peer loss when applying to problems of learning with noisy labels. However, it is not clear how the resulting classifiers differ from the ones trained using conventional CE loss. Knowing such differences can help us to understand how and why peer loss works. In order to achieve a clear visualization, we carry out experiments on a binary-class synthetic dataset, which we call the circle dataset [34][3]. The circle datasets contain 1000 points, 500 for each class. We adopt a 3-layer fully-connected neural network, which is trained using the Adam [30] optimizer, and the initial learning rate is set to be 0.1. We adopt the following settings for α: Milestones: [20, 40, 50, 100]. α-list = [0.0, 1.0, 2.0, 5.0].

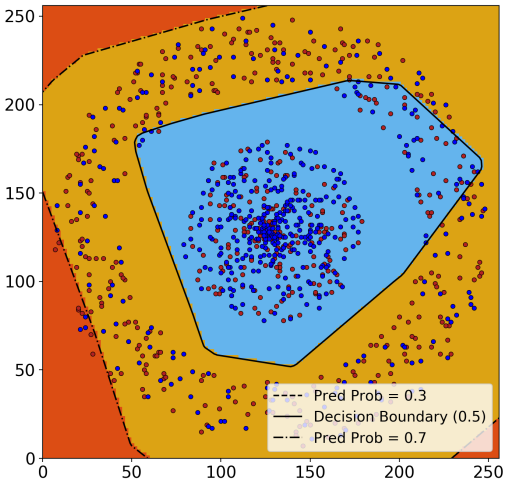
We have tested three types of random noise: the uniform random noise, the high margin noise, and the low margin noise. For detailed settings, please refer to appendix B.3. Because the resulting conclusions are similar, here we only include one characteristic case of the uniform random noise, where each sample's

[3]We thanks Prof. Manfred K. Warmuth for suggesting this dataset.

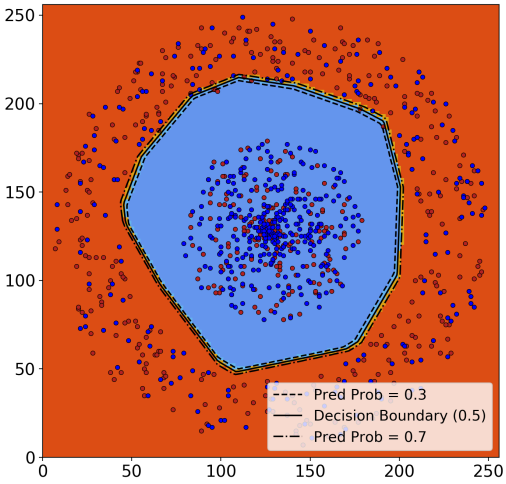
label is flipped independently according to the noise rate. For the other results, please refer to the appendix. Figure 4.1 summarizes our findings. It is clear from the figures that with label noise, the model trained with CE loss becomes quite uncertain with its predictions, while the model trained with peer loss can still learn a hard decision boundary and retain the performance almost the same as in the clean case. We actually expect this phenomenon as peer loss will encourage the model to make more confident predictions. For detailed analysis, please refer to the section 2.2.3.



(a) Decision boundary trained with CE loss using clean data.



(b) Decision boundary trained with CE loss under uniform random label nois.



(c) Decision boundary trained with peer loss under uniform random label nois.

Figure 4.1: Experiments on the circle dataset. Using a 3-layer fully-connected neural network. The noise rate is 0.4.

Chapter 5

Conclusion and Future Work

In this thesis, we first present the multi-class extension of the original peer loss [1] and investigate its noise tolerance properties. Further, we explore two distinct and novel views toward a better understanding of peer loss's working mechanism. In the first one, we demonstrate that peer loss can be formulated as the difference between two KL divergence. This provides us an intuitive explanation of the peer term in peer loss; namely, it encourages the model to encode more correlation between the feature and the label. The second view is more profound than the first one. We show that peer loss is related to a new risk called the correlation risk rather than the normally used population risk. Even though presently incomplete, this potentially provides us a different framework in the decision theory. We also analyzed the training dynamics of peer loss, exploiting the methods from the evolutionary game theory. This dynamical analysis complements our discussion about the correlation risk. In addition to those theoretical investigations, we also carry out numerical experiment studies that empirically demonstrate the effectiveness of peer loss in dealing with the learning with noisy label problem. Further, they also reveal interesting behaviors of the Peer Loss that help to guides our theoretical reasoning.

In the future, we plan to focus on developing the correlation risk framework. The present method and results are promising yet incomplete. Specifically, we have two primary directions:

1. We need both theoretical and experimental justifications about the usefulness of the correlation measure to motivate the use of correlation risk. We can probably find some clues in the field of fairness in machine learning.

2. We need to develop a firm theoretical description of the training dynamics with the correlation risk, For example, proving the conjecture 1. In other words, we need stronger results that guarantee the model will be able to grasp the positive correlations in the data distribution.

Appendix A

Proofs supplementary materials

A.1 Proof for Lemma 1

Lemma 1. *Suppose in the sparse noise setting there are C disjoint pairs, $(i_c, j_c)_{c\in[C]}$, whose corresponding error rates are $(T_{i_c j_c}, T_{j_c i_c})$. If $T_{i_c j_c} + T_{j_c i_c} < 1$ and $T^*_{i_c j_c} + T^*_{j_c i_c} < 1, \forall c \in [C]$, then $M = I_{K\times K}$.*

Proof. First, we observe that there exists a permutation matrix that transforms T into a block diagonal matrix. Without losing generality, we assume $i_c > j_c$. Focusing on a specific pair c, the transition matrix looks like

$$T = \begin{pmatrix} \ddots & & & & \\ & 1 - T_{i_c j_c} & \dots & T_{j_c i_c} & \\ & \vdots & \ddots & \vdots & \\ & T_{i_c j_c} & \dots & 1 - T_{j_c i_c} & \\ & & & & \ddots \end{pmatrix}.$$

Note there are C pairs in total. By the construction of the sparse noise, we have

$$\begin{cases} T_{i_c m} = 0 & \text{unless } m = i_c \text{ or } m = j_c, \\ T_{j_c m} = 0 & \text{unless } m = j_c \text{ or } m = i_c, \\ T_{n i_c} = 0 & \text{unless } n = i_c \text{ or } n = j_c, \\ T_{n j_c} = 0 & \text{unless } n = j_c \text{ or } n = i_c. \end{cases}$$

Thus, by permuting rows $\{1, j_c\}$ and rows $\{2, i_c\}$ as well as permuting columns $\{1, j_c\}$ and columns $\{2, i_c\}$, we can put the original T into the following form:

$$\left(\begin{array}{cc|c} 1 - T_{i_c j_c} & T_{j_c i_c} & 0 \cdots 0 \\ T_{i_c j_c} & 1 - T_{j_c i_c} & 0 \cdots 0 \\ \hline 0 & 0 & \\ \vdots & \vdots & T' \\ 0 & 0 & \end{array}\right),$$

where T' is a smaller transition matrix that contains the remaining pairs. We can apply permutation to T' and put it into a similar form. Clearly, repeating this procedure will result in a block diagonal matrix, whose block is of size 2×2 and has the form

$$\begin{pmatrix} 1 - T_{i_c j_c} & T_{j_c i_c} \\ T_{i_c j_c} & 1 - T_{j_c i_c} \end{pmatrix}, \forall c \in [C].$$

Note that along the way to achieve this block diagonal matrix, we apply a series of permutations, which can be combined as a permutation matrix. Denote this permutation matrix as $\mathcal{P}$. We then show matrix Δ can be put into a block diagonal form by applying the same permutation $\mathcal{P}$ as well.

Recall that

$$\Delta_{ij} = \sum_{k}^{K} T^*_{ik} \cdot T_{jk} \cdot \boldsymbol{P}(y = k) \\ - \sum_{l}^{K} T^*_{il} \cdot \boldsymbol{P}(y = l) \sum_{m}^{K} T_{jm} \boldsymbol{P}(y = m).$$

Let's introduce two new symbols

$$\mathcal{M} = \begin{pmatrix} \boldsymbol{P}(y=1) & 0 & \cdots & 0 \\ 0 & \boldsymbol{P}(y=2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{P}(y=K) \end{pmatrix} \quad \text{and} \quad v = \begin{pmatrix} \boldsymbol{P}(y=1) \\ \boldsymbol{P}(y=2) \\ \vdots \\ \boldsymbol{P}(y=K) \end{pmatrix},$$

with which we can rewrite the above expression for Δ in the following matrix form

$$\Delta = T^* \mathcal{M} T^{\mathrm{T}} - T^* v v^{\mathrm{T}} T^{\mathrm{T}}.$$

The same permutation can also be applied to Δ, i.e.,

$$\begin{aligned} \mathcal{P}\Delta\mathcal{P}^{\mathrm{T}} &= \mathcal{P}T^*\mathcal{P}^{\mathrm{T}}\mathcal{P}\mathcal{M}\mathcal{P}^{\mathrm{T}}\mathcal{P}T^{\mathrm{T}}\mathcal{P}^{\mathrm{T}} - \mathcal{P}T^*\mathcal{P}^{\mathrm{T}}\mathcal{P}vv^{\mathrm{T}}\mathcal{P}^{\mathrm{T}}\mathcal{P}T^{\mathrm{T}}\mathcal{P}^{\mathrm{T}} \\ &= (\mathcal{P}T^*\mathcal{P}^{\mathrm{T}})(\mathcal{P}\mathcal{M}\mathcal{P}^{\mathrm{T}})(\mathcal{P}T\mathcal{P}^{\mathrm{T}})^{\mathrm{T}} - (\mathcal{P}T^*\mathcal{P}^{\mathrm{T}})\mathcal{P}v(\mathcal{P}v)^{\mathrm{T}}(\mathcal{P}T\mathcal{P}^{\mathrm{T}})^{\mathrm{T}}, \end{aligned}$$

where we use the fact: $\mathcal{P}^{\mathrm{T}}\mathcal{P} = I_{K\times K}$. We see that T^*, T, and $\mathcal{M}$ undergo the same row and column permutations as for T, v undergoes the same row permutation as for T. Let $\Delta' = \mathcal{P}\Delta\mathcal{P}^{\mathrm{T}}$, $T^{*\prime} = \mathcal{P}T^*\mathcal{P}^{\mathrm{T}}$, $T' = \mathcal{P}T\mathcal{P}^{\mathrm{T}}$, $\mathcal{M}' = \mathcal{P}\mathcal{M}\mathcal{P}^{\mathrm{T}}$, and $v' = \mathcal{P}v$. We know from the previous discussion that T' is a block diagonal matrix. Additionally, it is easy to see $\mathcal{M}'$ remains a diagonal matrix.

Assume T^* share the same structure as T, i.e., the off-diagonal elements $T^*_{ij} \neq 0$ only if $i = i_c, j = j_c$ or $i = j_c, j = i_c$, $\forall c \in [C]$. Then $T^{*\prime}$ will be a block diagonal

matrix of the same structure as T'. With this assumption, Δ' will be a block diagonal matrix as well and the corresponding block for pair index c reads

$$\Delta'_c = T^{*\prime}_c \mathcal{M}_c T'^{\mathrm{T}}_c - T^{*\prime}_c v'_c v'^{\mathrm{T}}_c T'^{\mathrm{T}}_c.$$

More specifically,

$$\begin{aligned}\Delta'_c = &\begin{pmatrix} 1 - T^*_{i_c j_c} & T^*_{j_c i_c} \\ T^*_{i_c j_c} & 1 - T^*_{j_c i_c} \end{pmatrix} \begin{pmatrix} \boldsymbol{P}(y = j_c) & 0 \\ 0 & \boldsymbol{P}(y = i_c) \end{pmatrix} \begin{pmatrix} 1 - T_{i_c j_c} & T_{i_c j_c} \\ T_{j_c i_c} & 1 - T_{j_c i_c} \end{pmatrix} \\ &- \begin{pmatrix} 1 - T^*_{i_c j_c} & T^*_{j_c i_c} \\ T^*_{i_c j_c} & 1 - T^*_{j_c i_c} \end{pmatrix} \begin{pmatrix} \boldsymbol{P}(y = j_c)\boldsymbol{P}(y = j_c) & \boldsymbol{P}(y = j_c)\boldsymbol{P}(y = i_c) \\ \boldsymbol{P}(y = i_c)\boldsymbol{P}(y = j_c) & \boldsymbol{P}(y = i_c)\boldsymbol{P}(y = i_c) \end{pmatrix} \\ &\begin{pmatrix} 1 - T_{i_c j_c} & T_{i_c j_c} \\ T_{j_c i_c} & 1 - T_{j_c i_c} \end{pmatrix}.\end{aligned}$$

By setting $T^*_{i_c j_c} = e^*_{+1}, T^*_{i_c j_c} = e^*_{-1}$ and $T_{i_c j_c} = e_{+1}, T_{i_c j_c} = e_{-1}$, we effectively recover the noise settings for binary classification discussed in [1]. According to lemma 2 in [1], we have for the block Δ'_c that, if $T_{i_c j_c} + T_{j_c i_c} < 1$ and $T^*_{i_c j_c} + T^*_{j_c i_c} < 1$ then $\mathrm{sgn}(\Delta'_c) = I_{2\times 2}$. Similar conditions can be applied to the other blocks in Δ'. We can conclude:

If $T_{i_c j_c} + T_{j_c i_c} < 1$ and $T^*_{i_c j_c} + T^*_{j_c i_c} < 1$, $\forall c \in [C]$, then $\mathrm{sgn}(\Delta') = I_{K\times K}$.

Finally, since the operations of permutation and taking the matrix element's sign are commutable, we have

$$\mathrm{sgn}(\Delta') = I_{K\times K} = \mathrm{sgn}(\mathcal{P}\Delta\mathcal{P}^{\mathrm{T}}) = \mathcal{P}\mathrm{sgn}(\Delta)\mathcal{P}^{\mathrm{T}},$$

thus

$$sgn(\Delta) = \mathcal{P}^{\mathrm{T}} I_{K\times K} \mathcal{P} = I_{K\times K}.$$

By the definition $M = \text{sgn}(\Delta)$, we thus have proved Lemma 1.

□

A.2 Proof for Lemma 2

Lemma 2. *Assume* $T_{ij} = T_{ik} = e_i$, $\forall i \neq j \neq k$ *and* $T^*_{ij} = T^*_{ik} = e^*_i$, $\forall i \neq j \neq k$. *If* $\sum_{j\in[K]} e^*_j < 1$ *and* $\sum_{k\in[K]} e_k < 1$, *then* $M = I_{K\times K}$.

Proof. Recall that $M = \text{sgn}(\Delta)$ and

$$\Delta_{ij} = \boldsymbol{P}(h^*(x) = i, \tilde{y} = j) - \boldsymbol{P}(h^*(x) = i)\boldsymbol{P}(\tilde{y} = j). \tag{A.1}$$

For simplicity, we will use h^* instead $h^*(x)$ in the following proof. Note by our assumption and the conservation of probability, we have

$$T^*_{ii} = 1 - \sum_{i\neq j} e^*_j \quad \text{and} \quad T_{ii} = 1 - \sum_{i\neq j} e_j.$$

The terms in the r.h.s of equation (A.1) read

$$\begin{aligned}\boldsymbol{P}(h^* = i, \tilde{y} = j) &= \sum_{l=1}^{K} \boldsymbol{P}(h^* = i|y = l)\boldsymbol{P}(\tilde{y} = j|y = l)\boldsymbol{P}(y = l) \\ &= \sum_{l=1}^{K} T^*_{il}T_{jl}\boldsymbol{P}(y = l)\end{aligned}$$

and

$$\begin{aligned}\boldsymbol{P}(h^* = i)\boldsymbol{P}(\tilde{y} = j) &= \sum_{l=1}^{K} \boldsymbol{P}(h^* = i|y = l)\boldsymbol{P}(y = l) \sum_{m=1}^{K} \boldsymbol{P}(\tilde{y} = j|y = m)\boldsymbol{P}(y = m) \\ &= \sum_{l=1}^{K} T^*_{il}\boldsymbol{P}(y = l) \sum_{m=1}^{K} T_{jm}\boldsymbol{P}(y = m).\end{aligned}$$

In the above equations, we have used the conditional independence assumption

between the Bayes optimal classifier and the noisy label. Next, we will investigate the behavior of diagonal and off-diagonal elements of Δ separately.

Case 1: The diagonal terms, i.e. $i = j$.

$$\begin{aligned}
\boldsymbol{P}(h^* = i, \tilde{y} = i) &= T^*_{ii} T_{ii} \boldsymbol{P}(y = i) + \sum_{l=1; l\neq i}^{K} T^*_{il} T_{il} \boldsymbol{P}(Y = l) \\
&= \left(1 - \sum_{j\neq i} e^*_j\right)\left(1 - \sum_{k\neq i} e_k\right)\boldsymbol{P}(y = i) + \sum_{l=1; l\neq i}^{K} e^*_i e_i \boldsymbol{P}(y = l) \\
&= \left(1 - \sum_{j} e^*_j\right)\left(1 - \sum_{k} e_k\right)\boldsymbol{P}(y = i) \\
&\quad + e^*_i \left(1 - \sum_{k} e_k\right)\boldsymbol{P}(y = i) + e_i \left(1 - \sum_{j} e^*_j\right)\boldsymbol{P}(y = i) \\
&\quad + e^*_i e_i.
\end{aligned}$$

In the last step, we used the identity $\sum_{l=1}^{K} \boldsymbol{P}(y = l) = 1$. Likewise, for the second term

$$\begin{aligned}
\boldsymbol{P}(h^* = i)\boldsymbol{P}(\tilde{y} = i) &= \sum_{l=1}^{K} T^*_{il} \boldsymbol{P}(y = l) \sum_{m=1}^{K} T_{im} \boldsymbol{P}(y = m) \\
&= \left[\left(1 - \sum_{j} e^*_j\right)\boldsymbol{P}(y = i) + \sum_{l=1}^{K} e^*_i \boldsymbol{P}(y = l)\right] \\
&\quad \times \left[\left(1 - \sum_{k} e_k\right)\boldsymbol{P}(y = i) + \sum_{m=1}^{K} e_i \boldsymbol{P}(y = m)\right] \\
&= \left(1 - \sum_{j} e^*_j\right)\left(1 - \sum_{k} e_k\right)\boldsymbol{P}(y = i)\boldsymbol{P}(y = i) \\
&\quad + e^*_i \left(1 - \sum_{k} e_k\right)\boldsymbol{P}(y = i) + e_i \left(1 - \sum_{j} e^*_j\right)\boldsymbol{P}(y = i) \\
&\quad + e^*_i e_i.
\end{aligned}$$

Hence, we have

$$\begin{aligned}\Delta_{ii} &= \boldsymbol{P}(h^* = i, \tilde{y} = i) - \boldsymbol{P}(h^* = i)\boldsymbol{P}(\tilde{y} = i) \\ &= \left(1 - \sum_j e_j^*\right)\left(1 - \sum_k e_k\right)\boldsymbol{P}(y = i)(1 - \boldsymbol{P}(y = i)).\end{aligned}$$

We conclude $\Delta_{ii} > 0$ if $\sum_j e_j^* < 1$ and $\sum_k e_k < 1$.

Case 2: The off-diagonal terms, i.e. $i \neq j$.

$$\begin{aligned}\boldsymbol{P}(h^* = i, \tilde{y} = j) &= T_{ii}^* T_{ji}\boldsymbol{P}(y = i) + T_{ij}^* T_{jj}\boldsymbol{P}(y = j) + \sum_{l=1; l\neq i; l\neq j}^{K} T_{il}^* T_{jl}\boldsymbol{P}(y = l) \\ &= e_j\left(1 - \sum_{n\neq i} e_n^*\right)\boldsymbol{P}(y = i) + e_i^*\left(1 - \sum_{m\neq j} e_m\right)\boldsymbol{P}(y = j) \\ &\quad + \sum_{l=1; l\neq i; l\neq j}^{K} e_i^* e_j \boldsymbol{P}(y = l) \\ &= e_j\left(1 - \sum_n e_n^*\right)\boldsymbol{P}(y = i) + e_i^*\left(1 - \sum_m e_m\right)\boldsymbol{P}(y = j) + e_i^* e_j\end{aligned}$$

Similarly, for the second term

$$\begin{aligned}\boldsymbol{P}(h^* = i)\boldsymbol{P}(\tilde{y} = j) &= \left[\left(1 - \sum_n e_n^*\right)\boldsymbol{P}(y = i) + e_i^*\right]\left[\left(1 - \sum_m e_m\right)\boldsymbol{P}(y = j) + e_j\right] \\ &= \left(1 - \sum_n e_n^*\right)\left(1 - \sum_m e_m\right)\boldsymbol{P}(y = i)\boldsymbol{P}(y = j) \\ &\quad + e_j\left(1 - \sum_n e_n^*\right)\boldsymbol{P}(y = i) + e_i^*\left(1 - \sum_m e_m\right)\boldsymbol{P}(y = j) \\ &\quad + e_i^* e_j\end{aligned}$$

As a result, we find

$$\begin{aligned}\Delta_{ij} &= \boldsymbol{P}(h^* = i, \tilde{y} = j) - \boldsymbol{P}(h^* = i)\boldsymbol{P}(\tilde{y} = j) \\ &= -\left(1 - \sum_n e_n^*\right)\left(1 - \sum_m e_m\right)\boldsymbol{P}(y = i)\boldsymbol{P}(y = j).\end{aligned}$$

We thus conclude $\Delta_{ij} < 0$ if $\sum_n e_n^* < 1$ and $\sum_m e_m < 1$.

Based on above results and the definition $M = \text{sgn}(\Delta)$, we conclude that lemma 2 has been proved. □

A.3 Proof for Theorem 1

Theorem 1. *Under the uniform off-diagonal noise setting, the expected peer loss is invariant to label noise up to an affine transformation:*

$$\mathbb{E}_{\widetilde{\mathcal{D}}}[\ell_{peer}(f(X), \widetilde{Y})] = \left(1 - \sum_{i \in [K]} e_i\right) \mathbb{E}_{\mathcal{D}}[\ell_{peer}(f(X), Y)].$$

Proof. Recall that $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ refer to the joint distribution over (X, Y) and $(X, \widetilde{Y})$, respectively. We further denote the marginal distributions of X, Y, and $\widetilde{Y}$ by $\mathcal{D}_X$, $\mathcal{D}_Y$, and $\widetilde{\mathcal{D}}_Y$, respectively. Let $X_p \sim \mathcal{D}_X$, $\widetilde{Y}_p \sim \widetilde{\mathcal{D}}_Y$ be the random variables corresponding to the peer samples. The peer loss function is defined as

$$\ell_{\text{peer}}(f(x_n), \tilde{y}_n) = \ell(f(x_n), \tilde{y}_n) - \ell(f(x_{p,n}), \tilde{y}_{p.n}). \tag{A.2}$$

where $(x_n, \tilde{y}_n)$ is a normal training sample pair, $x_{p,n}$ and $\tilde{y}_{p,n}$ are corresponding peer samples.

Taking expectation for (A.2) yields

$$\mathbb{E}_{\widetilde{\mathcal{D}}}[\ell_{\text{peer}}(f(X),\widetilde{Y})] = \mathbb{E}_{\widetilde{\mathcal{D}}}[\ell(f(X),\widetilde{Y})] - \mathbb{E}_{\widetilde{\mathcal{D}}_Y}\left[\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),\widetilde{Y})]\right]. \tag{A.3}$$

The first term in (A.3) is

$$\begin{aligned}
&\mathbb{E}_{\widetilde{\mathcal{D}}}[\ell(f(X),\widetilde{Y})] \\
&= \sum_{j\in[K]}\sum_{i\in[K]} T_{ji}\mathbb{P}(Y=i)\mathbb{E}_{\mathcal{D}|Y=i}[\ell(f(X),j)] \\
&= \sum_{j\in[K]}\left[T_{jj}\mathbb{P}(Y=j)\mathbb{E}_{\mathcal{D}|Y=j}[\ell(f(X),j)] + \sum_{i\in[K],i\neq j} T_{ji}\mathbb{P}(Y=i)\mathbb{E}_{\mathcal{D}|Y=i}[\ell(f(X),j)]\right] \\
&= \sum_{j\in[K]}\left[\left(1-\sum_{i\neq j,i\in[K]} T_{ij}\right)\mathbb{P}(Y=j)\mathbb{E}_{\mathcal{D}|Y=j}[\ell(f(X),j)] + \right. \\
&\qquad\qquad \left. \sum_{i\in[K],i\neq j} T_{ji}\mathbb{P}(Y=i)\mathbb{E}_{\mathcal{D}|Y=i}[\ell(f(X),j)]\right]
\end{aligned}$$

Noting that X_p and $\widetilde{Y}_p$ are independent, the second term in (A.3) is

$$\begin{aligned}
&\mathbb{E}_{\widetilde{\mathcal{D}}_Y}\left[\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),\widetilde{Y})]\right] \\
&= \sum_{j\in[K]} \mathbb{P}(\widetilde{Y}_p=j)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X_p),j)] \\
&= \sum_{j\in[K]}\sum_{i\in[K]} T_{ji}\mathbb{P}(Y_p=i)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)] \\
&= \sum_{j\in[K]}\left[T_{jj}\mathbb{P}(Y_p=j)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)] + \sum_{i\in[K],i\neq j} T_{ji}\mathbb{P}(Y_p=i)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)]\right] \\
&= \sum_{j\in[K]}\left[\left(1-\sum_{i\neq j,i\in[K]} T_{ij}\right)\mathbb{P}(Y_p=j)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)] + \right. \\
&\qquad\qquad \left. \sum_{i\in[K],i\neq j} T_{ji}\mathbb{P}(Y_p=i)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)]\right]
\end{aligned}$$

In this case, we have $e_i = T_{ij}, \forall j \in [K], j \neq i$. The first term becomes

$$
\begin{aligned}
&\mathbb{E}_{\widetilde{\mathcal{D}}}[\ell(f(X), \widetilde{Y})] \\
&= \sum_{j\in[K]} \left[\left(1 - \sum_{i\neq j, i\in[K]} e_i\right) \mathbb{P}(Y=j)\mathbb{E}_{\mathcal{D}|Y=j}[\ell(f(X),j)] + \right. \\
&\qquad\qquad \left. \sum_{i\in[K], i\neq j} e_j \mathbb{P}(Y=i)\mathbb{E}_{\mathcal{D}|Y=i}[\ell(f(X),j)] \right] \\
&= \sum_{j\in[K]} \left[\left(1 - \sum_{i\in[K]} e_i\right) \mathbb{P}(Y=j)\mathbb{E}_{\mathcal{D}|Y=j}[\ell(f(X),j)] + \right. \\
&\qquad\qquad \left. \sum_{i\in[K]} e_j \mathbb{P}(Y=i)\mathbb{E}_{\mathcal{D}|Y=i}[\ell(f(X),j)] \right] \\
&= \left(1 - \sum_{i\in[K]} e_i\right) \mathbb{E}_{\mathcal{D}}[\ell(f(X),Y)] + \sum_{j\in[K]} e_j \mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)]
\end{aligned}
$$

The second term becomes

$$
\begin{aligned}
&\mathbb{E}_{\widetilde{\mathcal{D}}_Y}\left[\mathbb{E}_{\mathcal{D}_X}[\ell(f(X_p), \widetilde{Y}_p)]\right] \\
&= \sum_{j\in[K]} \left[\left(1 - \sum_{i\neq j, i\in[K]} e_i\right) \mathbb{P}(Y_p=j)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)] + \right. \\
&\qquad\qquad \left. \sum_{i\in[K], i\neq j} e_j \mathbb{P}(Y_p=i)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)] \right] \\
&= \sum_{j\in[K]} \left[\left(1 - \sum_{i\in[K]} e_i\right) \mathbb{P}(Y_p=j)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)] + \right. \\
&\qquad\qquad \left. \sum_{i\in[K]} e_j \mathbb{P}(Y_p=i)\mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)] \right] \\
&= \left(1 - \sum_{i\in[K]} e_i\right) \mathbb{E}_{\mathcal{D}_Y}\left[\mathbb{E}_{\mathcal{D}_X}[\ell(f(X_p),Y_p)]\right] + \sum_{j\in[K]} e_j \mathbb{E}_{\mathcal{D}_X}[\ell(f(X),j)].
\end{aligned}
$$

Comparing the above two terms we have

$$\mathbb{E}_{\widetilde{\mathcal{D}}}[\ell_{\text{peer}}(f(X),\widetilde{Y})] = \left(1 - \sum_{i\in[K]} e_i\right) \mathbb{E}_{\mathcal{D}}[\ell_{\text{peer}}(f(X),Y)].$$

□

A.4 Proof for Theorem 2

Theorem 2. *Under the uniform off-diagonal noise setting, when the true label Y has equal prior, i.e. $\boldsymbol{P}(y=k) = 1/K, \forall k \in [K]$, we have*

$$cl^*_{Bayes} = \arg\min_{cl\in\mathcal{F}} \ \mathbb{E}_{\widetilde{\mathcal{D}}}\Big[\mathbb{1}_{peer}(f(X),\widetilde{Y})\Big].$$

Proof. When the true label Y has equal prior $\mathbb{P}(y=k) = 1/K, \forall k \in [K]$, we have

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[\mathbb{1}_{\text{peer}}(f(X),Y)] &= \mathbb{E}_{\mathcal{D}}[\mathbb{1}(f(X),Y)] - \mathbb{E}_{\mathcal{D}_Y}\left[\mathbb{E}_{\mathcal{D}_X}[\mathbb{1}(f(X),Y)]\right] \\ &= \mathbb{E}_{\mathcal{D}}[\mathbb{1}(f(X),Y)] - \frac{1}{K}\sum_{i\in[K]} \mathbb{E}_{\mathcal{D}_X}[\mathbb{1}(f(X),i)]\end{aligned} \tag{A.4}$$

For the 0-1 loss, we have

$$\sum_{i\in[K]} \mathbb{E}_{\mathcal{D}_X}[\mathbb{1}(f(X_p),i)] = K-1.$$

Therefore, in the case of uniform off-diagonal noise, Theorem 1 can be further extended as

$$\mathbb{E}_{\widetilde{\mathcal{D}}}[\mathbb{1}_{\text{peer}}(f(X),\widetilde{Y})] = \left(1 - \sum_{i\in[K]} e_i\right)\left(\mathbb{E}_{\mathcal{D}}[\mathbb{1}_{\text{peer}}(f(X),Y)] - \frac{K-1}{K}\right),$$

which indicates

$$cl^*_{Bayes} = \arg\min_{f \in \mathcal{F}} \ \mathbb{E}_{\widetilde{\mathcal{D}}}\Big[\mathbb{1}_{\text{peer}}(cl(X), \widetilde{Y})\Big].$$

□

A.5 Proof for Propersition 2

Proposition 2. *For any pair* (x_0, y_0)*, if* $h_{y_0}(x_0) = 0$ *then* $\Delta_{y_0}(x_0) = 0$.

Proof. We need to take into account the actual form of activation function, i.e., the softmax function, as well as the SGD algorithm to demonstrate the correctness of this proposition. The variation $\Delta_{y_0}(x_0)$ is caused by the change in network parameters $\{w_i\}$, i.e.,

$$\Delta_{y_0}(x_0) = \sum_i \frac{\partial h_{y_0}(x_0)}{\partial w_i} \delta w_i,$$

where δw_i are determined by the SGD algorithm

$$\begin{aligned}\delta w_i =& -\eta \frac{\partial \mathbb{E}^{peer}_{\mathcal{D}}[\ell_{CE}]}{\partial w_i} \\ =& \eta \sum_{x,y} \frac{\boldsymbol{P}(x,y) - \boldsymbol{P}(x)\boldsymbol{P}(y)}{h_y(x)} \frac{\partial h_y(x)}{\partial w_i}.\end{aligned}$$

Plugging back to the expression for $\Delta_{y_0}(x_0)$, we have

$$\Delta_{y_0}(x_0) = \eta \sum_{x,y} \frac{\boldsymbol{P}(x,y) - \boldsymbol{P}(x)\boldsymbol{P}(y)}{h_y(x)} \sum_i \frac{\partial h_{y_0}(x_0)}{\partial w_i} \frac{\partial h_y(x)}{\partial w_i}.$$

To proceed further, we need to expand $\frac{\partial h_y(x)}{\partial w_i}$ in detail. Taking into account the activation function, one has

$$h_y(x) = \frac{e^{-f_y(x)}}{\sum_{y'} e^{-f_{y'}(x)}},$$

where $f_y(x)$ refers to the network output before passed to the activation function. Recall that, by our assumption, derivatives $\frac{\partial f_y(x;w)}{\partial w_i}$ are not singular. Now we have

$$\begin{aligned}\frac{\partial h_y(x)}{\partial w_i} =& \frac{\partial e^{-f_y(x)}}{\partial w_i}\frac{1}{\sum_{y'} e^{-f_{y'}(x)}} + e^{-f_y(x)}\frac{\partial}{\partial w_i}\left(\frac{1}{\sum_{y'} e^{-f_{y'}(x)}}\right) \\ =& \frac{-e^{-f_y(x)}}{\sum_{y'} e^{-f_{y'}(x)}}\frac{\partial f_y(x)}{\partial w_i} + \frac{e^{-f_y(x)}}{\left(\sum_{y''} e^{-f_{y''}(x)}\right)^2}\sum_{y'} e^{-f_{y'}(x)}\frac{\partial f_{y'}(x)}{\partial w_i} \\ =& h_y(x)\left[-\frac{\partial f_y(x)}{\partial w_i} + \sum_{y'} h_{y'}(x)\frac{\partial f_{y'}(x)}{\partial w_i}\right].\end{aligned}$$

For simplicity, we can rewrite the above result as

$$\frac{\partial h_y(x)}{\partial w_i} = h_y(x)G_i(x,y),$$

where

$$G_i(x,y) = -\frac{\partial f_y(x)}{\partial w_i} + \sum_{y'} h_{y'}(x)\frac{\partial f_{y'}(x)}{\partial w_i}$$

is a smooth function.

Combine all the above together, the $\Delta_{y_0}(x_0)$ reads

$$\Delta_{y_0}(x_0) = h_{y_0}(x_0)\cdot\eta\iint_{x,y}\left[\boldsymbol{P}(x,y) - \boldsymbol{P}(x)\boldsymbol{P}(y)\right]\sum_i G_i(x_0,y_0)G_i(x,y).$$

Now, it is straightforward to see $\Delta_{y_0}(x_0) = 0$ if $h_{y_0}(x_0) = 0$.

□

Appendix B

Experiments supplementary materials

B.1 Transition Matrices for MNIST Dataset and Fashion MNIST Dataset

Sparse-low noise matrix:

$$\begin{bmatrix}
0.7 & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0.3 & 0.8 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.7 & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.3 & 0.8 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.7 & 0.2 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.3 & 0.8 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.7 & 0.2 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.3 & 0.8 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.7 & 0.2 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.3 & 0.8
\end{bmatrix}$$

Sparse-high noise matrix:

$$\begin{bmatrix}
0.3 & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0.7 & 0.8 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.3 & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.7 & 0.8 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.3 & 0.2 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.7 & 0.8 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.3 & 0.2 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.7 & 0.8 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.3 & 0.2 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.7 & 0.8
\end{bmatrix}$$

Uniform-low noise matrix:

$$\begin{bmatrix}
0.258 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 \\
0.075 & 0.253 & 0.075 & 0.075 & 0.075 & 0.075 & 0.075 & 0.075 & 0.075 & 0.075 \\
0.09 & 0.09 & 0.268 & 0.09 & 0.09 & 0.09 & 0.09 & 0.09 & 0.09 & 0.09 \\
0.085 & 0.085 & 0.085 & 0.263 & 0.085 & 0.085 & 0.085 & 0.085 & 0.085 & 0.085 \\
0.07 & 0.07 & 0.07 & 0.07 & 0.248 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 \\
0.082 & 0.082 & 0.082 & 0.082 & 0.082 & 0.26 & 0.082 & 0.082 & 0.082 & 0.082 \\
0.077 & 0.077 & 0.077 & 0.077 & 0.77 & 0.077 & 0.255 & 0.077 & 0.077 & 0.077 \\
0.091 & 0.091 & 0.091 & 0.091 & 0.91 & 0.091 & 0.091 & 0.269 & 0.091 & 0.091 \\
0.092 & 0.092 & 0.092 & 0.092 & 0.92 & 0.092 & 0.092 & 0.092 & 0.27 & 0.092 \\
0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.258
\end{bmatrix}$$

Uniform-high noise matrix:

$$\begin{bmatrix}
0.58 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \\
0.045 & 0.0575 & 0.045 & 0.045 & 0.045 & 0.045 & 0.045 & 0.045 & 0.045 & 0.045 \\
0.047 & 0.047 & 0.577 & 0.047 & 0.047 & 0.047 & 0.047 & 0.047 & 0.047 & 0.047 \\
0.055 & 0.055 & 0.055 & 0.585 & 0.055 & 0.055 & 0.055 & 0.055 & 0.055 & 0.055 \\
0.053 & 0.053 & 0.053 & 0.053 & 0.583 & 0.053 & 0.053 & 0.053 & 0.053 & 0.053 \\
0.022 & 0.022 & 0.022 & 0.022 & 0.022 & 0.552 & 0.022 & 0.022 & 0.022 & 0.022 \\
0.068 & 0.068 & 0.068 & 0.068 & 0.068 & 0.068 & 0.598 & 0.068 & 0.068 & 0.068 \\
0.054 & 0.054 & 0.054 & 0.054 & 0.054 & 0.054 & 0.054 & 0.584 & 0.054 & 0.054 \\
0.056 & 0.056 & 0.056 & 0.056 & 0.056 & 0.056 & 0.056 & 0.056 & 0.586 & 0.056 \\
0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.55
\end{bmatrix}$$

B.2 Transition Matrices for CIFAR-10 Dataset

Sparse-low noise matrix:

$$\begin{bmatrix}
0.7 & 0.1 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0.3 & 0.9 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.7 & 0.1 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.3 & 0.9 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.7 & 0.1 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.3 & 0.9 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.7 & 0.1 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.3 & 0.9 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.7 & 0.1 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.3 & 0.9
\end{bmatrix}$$

Sparse-high noise matrix:

$$\begin{bmatrix}
0.4 & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0.6 & 0.8 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.4 & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0.6 & 0.8 & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.4 & 0.2 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.6 & 0.8 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.4 & 0.2 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.6 & 0.8 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.4 & 0.2 \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.6 & 0.8
\end{bmatrix}$$

Uniform-low noise matrix:

$$\begin{bmatrix}
0.82 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\
0.03 & 0.83 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\
0.01 & 0.01 & 0.81 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\
0.023 & 0.023 & 0.023 & 0.823 & 0.023 & 0.023 & 0.023 & 0.023 & 0.023 & 0.023 \\
0.017 & 0.017 & 0.017 & 0.017 & 0.817 & 0.017 & 0.017 & 0.017 & 0.017 & 0.017 \\
0.022 & 0.022 & 0.022 & 0.022 & 0.022 & 0.822 & 0.022 & 0.022 & 0.022 & 0.022 \\
0.021 & 0.021 & 0.021 & 0.021 & 0.021 & 0.021 & 0.821 & 0.021 & 0.021 & 0.021 \\
0.018 & 0.018 & 0.018 & 0.018 & 0.018 & 0.018 & 0.018 & 0.818 & 0.018 & 0.018 \\
0.019 & 0.019 & 0.019 & 0.019 & 0.019 & 0.019 & 0.019 & 0.019 & 0.819 & 0.019 \\
0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.82
\end{bmatrix}$$

Uniform-high noise matrix:

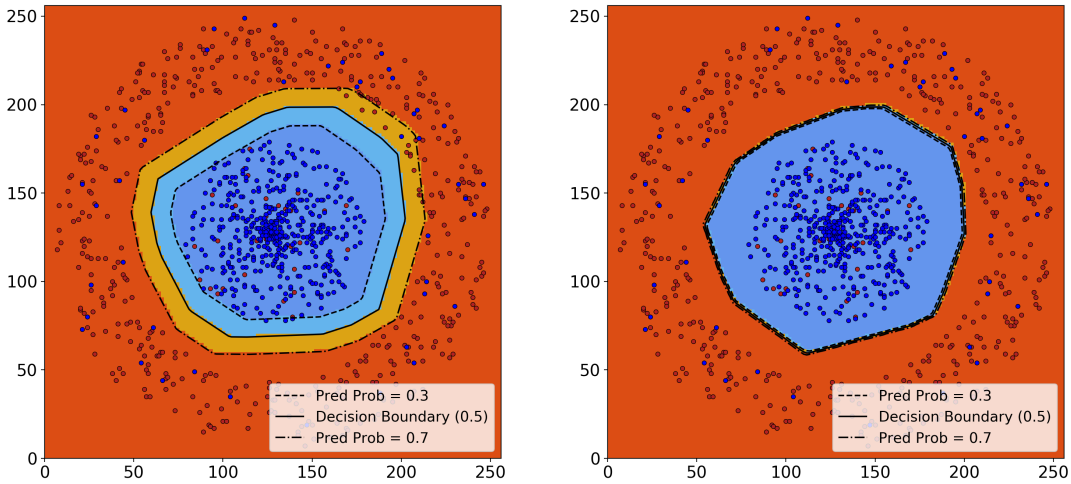
$$\begin{bmatrix}
0.46 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \\
0.07 & 0.48 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 \\
0.04 & 0.04 & 0.45 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.04 \\
0.05 & 0.05 & 0.05 & 0.46 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \\
0.06 & 0.06 & 0.06 & 0.06 & 0.47 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\
0.04 & 0.04 & 0.04 & 0.04 & 0.04 & 0.45 & 0.04 & 0.04 & 0.04 & 0.04 \\
0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.47 & 0.06 & 0.06 & 0.06 \\
0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.48 & 0.07 & 0.07 \\
0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.08 & 0.49 & 0.08 \\
0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.07 & 0.48
\end{bmatrix}$$

Manual Pair noise matrix:

$$\begin{bmatrix}
0.4 & 0. & 0.2 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0.4 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.2 \\
0.6 & 0. & 0.8 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0.4 & 0. & 0.2 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0.4 & 0. & 0. & 0.2 & 0. & 0. \\
0. & 0. & 0. & 0.6 & 0. & 0.8 & 0. & 0. & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.4 & 0. & 0.2 & 0. \\
0. & 0. & 0. & 0. & 0.6 & 0. & 0. & 0.8 & 0. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0.6 & 0. & 0.8 & 0. \\
0. & 0.6 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.8
\end{bmatrix}$$

B.3 Other experiment results on the 2-D synthetic dataset

We carry out experiments on three types of label noise. (1) uniform random noise. Each sample's label is flipped according to the noise rate. (2) high margin noise. Only those samples that are far away [1] from the boundary can be noisy. Their labels are flipped according to the noise rate. (3) low margin noise. Only those samples that are close [2] to the boundary can be noisy. Their labels are flipped according to the noise rate. We consider three noise rates: 0.1, 0.2, and 0.4 in our experiments. The results are summarised in the figures below.

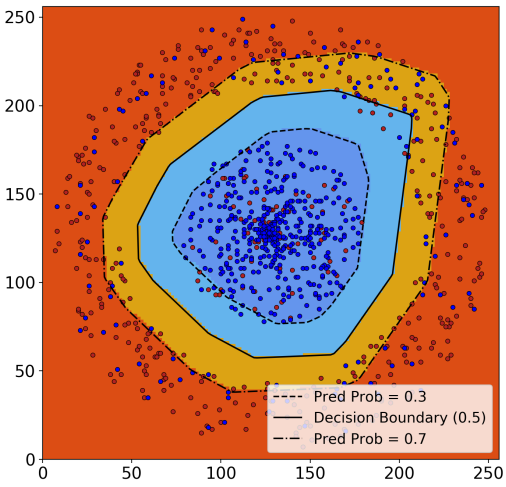


(a) Trained with the CE loss. **(b)** Trained with the Peer Loss.

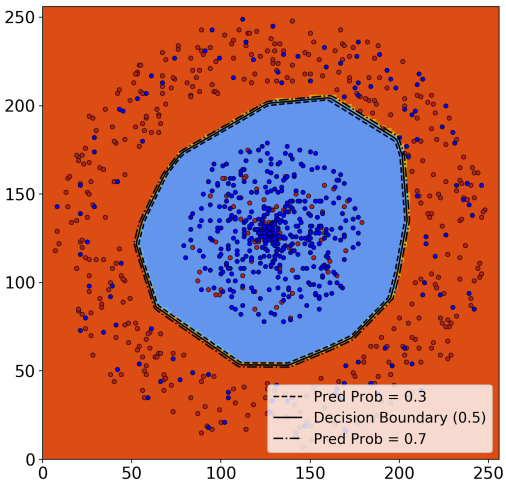
Figure B.1: Experiments on the circle dataset with uniform random noise. The noise rate is 0.1.

[1] Within a distance of $0.03r$ from the center or the outmost circle, where r is the radius of the outmost circle.

[2] Within a distance of $0.03r$ from the margin, where r is the radius of the outmost circle.

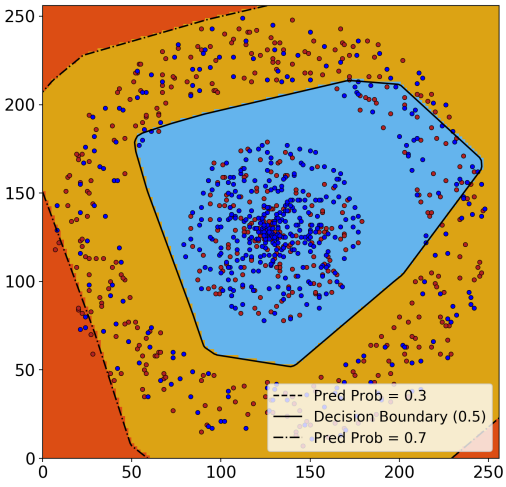


(a) Trained with the CE loss.

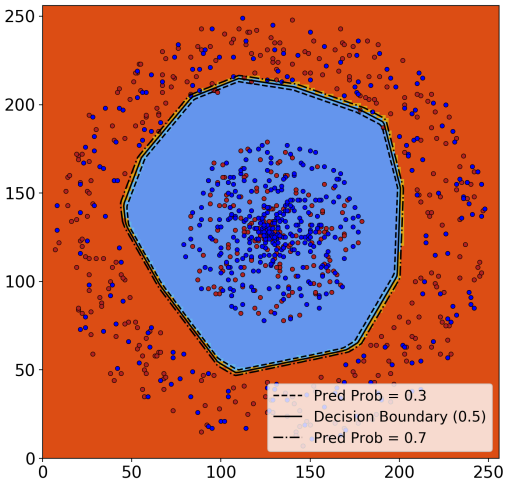


(b) Trained with the Peer Loss.

Figure B.2: Experiments on the circle dataset with uniform random noise. The noise rate is 0.2.

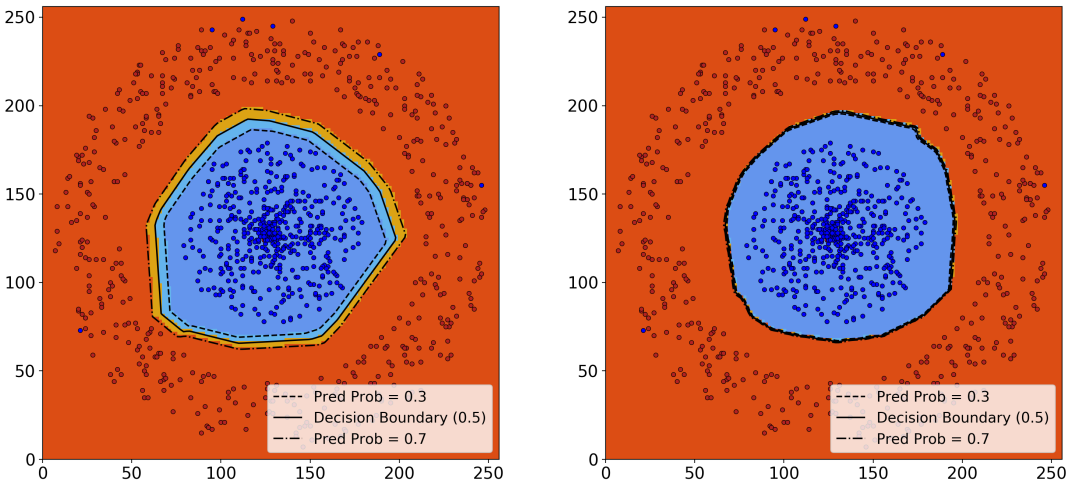


(a) Trained with the CE loss.



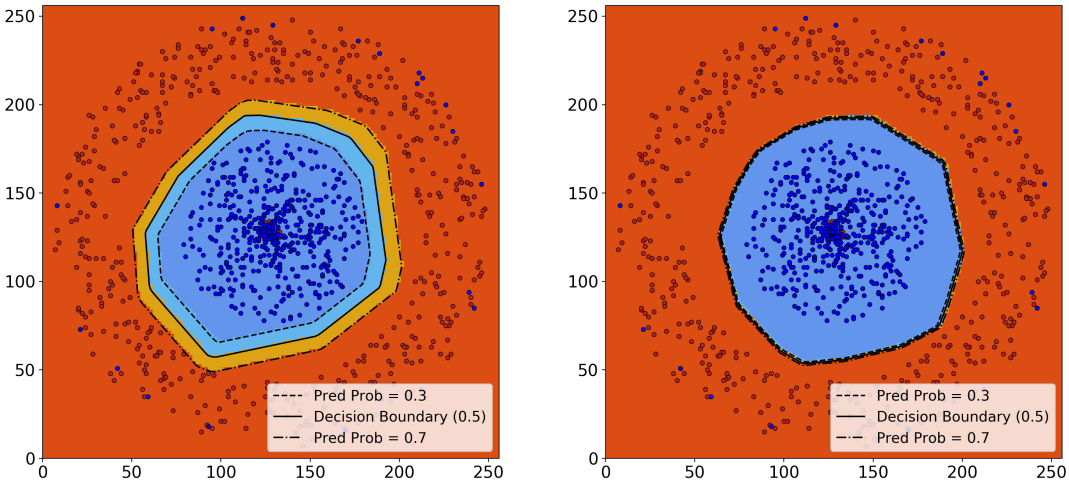
(b) Trained with the Peer Loss.

Figure B.3: Experiments on the circle dataset with uniform random noise. The noise rate is 0.4.



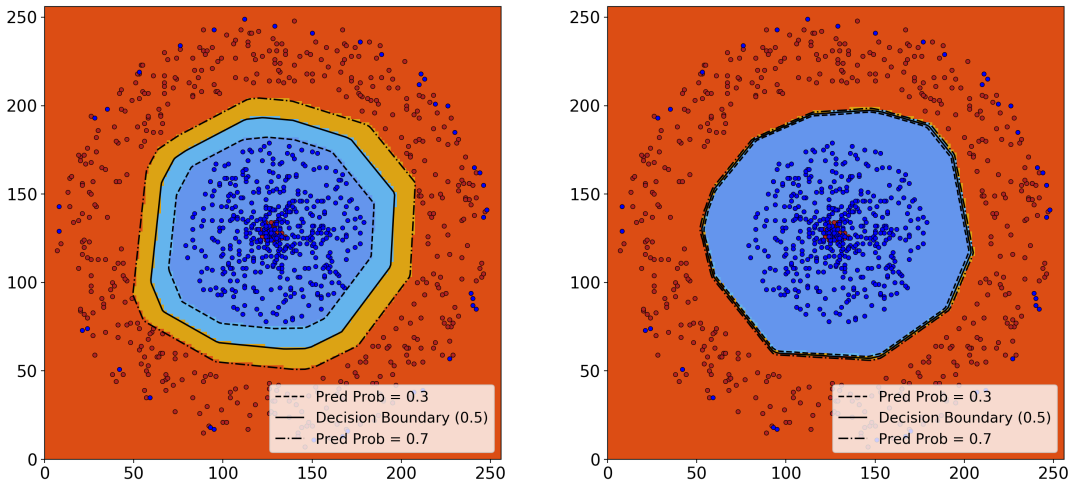
(a) Trained with the CE loss. **(b)** Trained with the Peer Loss.

Figure B.4: Experiments on the circle dataset with high margin noise. The noise rate is 0.1.



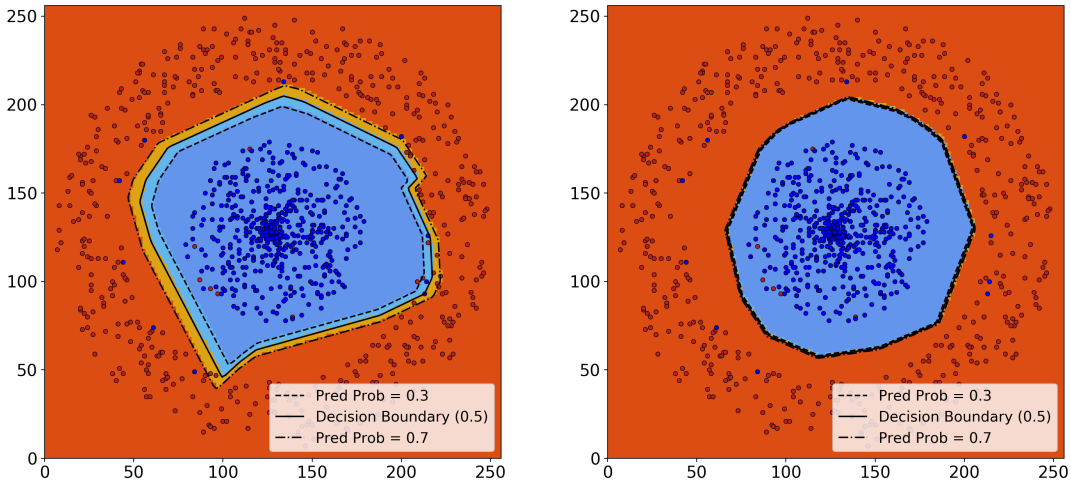
(a) Trained with the CE loss. **(b)** Trained with the Peer Loss.

Figure B.5: Experiments on the circle dataset with high margin noise. The noise rate is 0.2.



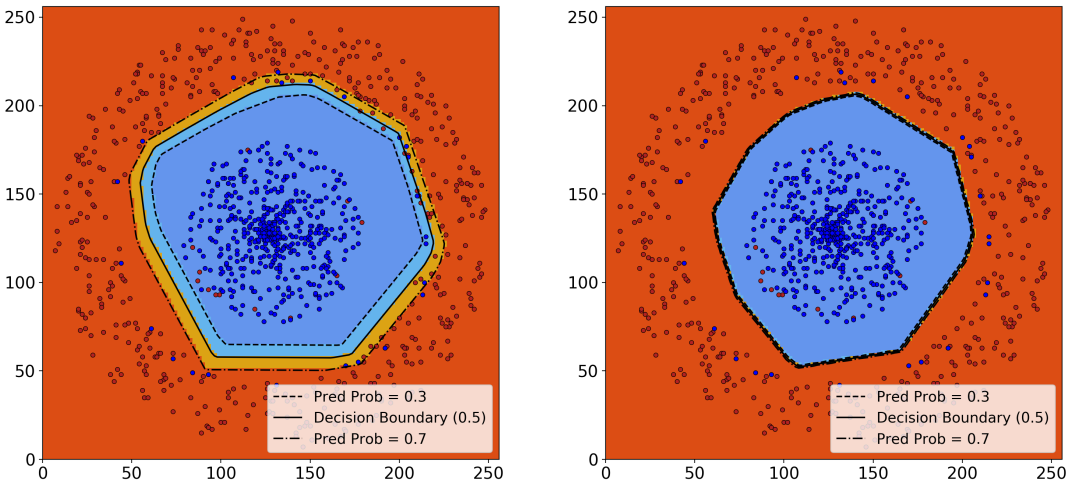
(a) Trained with the CE loss. **(b)** Trained with the Peer Loss.

Figure B.6: Experiments on the circle dataset with high margin noise. The noise rate is 0.4.



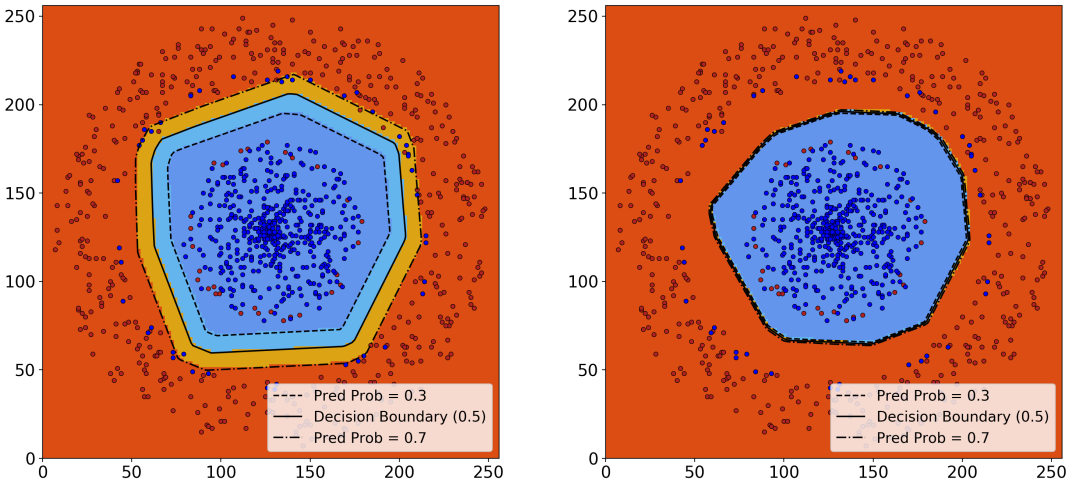
(a) Trained with the CE loss. **(b)** Trained with the Peer Loss.

Figure B.7: Experiments on the circle dataset with low margin noise. The noise rate is 0.1.



(a) Trained with the CE loss. **(b)** Trained with the Peer Loss.

Figure B.8: Experiments on the circle dataset with low margin noise. The noise rate is 0.2.



(a) Trained with the CE loss. **(b)** Trained with the Peer Loss.

Figure B.9: Experiments on the circle dataset with low margin noise. The noise rate is 0.4.

Bibliography

[1] Y. Liu and H. Guo, "Peer loss functions: Learning from noisy labels without knowing noise rates," 2019.

[2] D. Friedman and B. Sinervo, *Evolutionary Games in Natural, Social, and Virtual Worlds*, ser. OUP Catalogue. Oxford University Press, 2016, no. 9780199981151.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," pp. 1–15, nov 2016.

[5] T. Bylander, "Learning linear threshold functions in the presence of classification noise," in *Proceedings of the seventh annual conference on Computational learning theory*. ACM, 1994, pp. 340–347.

[6] N. Cesa-Bianchi, E. Dichterman, P. Fischer, E. Shamir, and H. U. Simon, "Sample-efficient strategies for learning in the presence of noise," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 684–719, 1999.

[7] N. Cesa-Bianchi, S. Shalev-Shwartz, and O. Shamir, "Online learning of noisy data," *IEEE Transactions on Information Theory*, vol. 57, no. 12, pp. 7907–7931, 2011.

[8] S. Ben-David, D. Pál, and S. Shalev-Shwartz, "Agnostic online learning." in *COLT 2009*.

[9] C. Scott, G. Blanchard, G. Handy, S. Pozzi, and M. Flaska, "Classification with asymmetric label noise: Consistency and maximal denoising." in *COLT*, 2013, pp. 489–511.

[10] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in neural information processing systems*, 2013, pp. 1196–1204.

[11] C. Scott, "A rate of convergence for mixture proportion estimation, with application to learning from noisy labels." in *AISTATS*, 2015.

[12] G. Stempfel and L. Ralaivola, "Learning svms from sloppily labeled data," in *International Conference on Artificial Neural Networks*. Springer, 2009, pp. 884–893.

[13] B. Van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," in *Advances in Neural Information Processing Systems*, 2015, pp. 10–18.

[14] A. Menon, B. Van Rooyen, C. S. Ong, and B. Williamson, "Learning from corrupted binary labels via class-probability estimation," in *International Conference on Machine Learning*, 2015, pp. 125–134.

[15] H. Ramaswamy, C. Scott, and A. Tewari, "Mixture proportion estimation via kernel embeddings of distributions," in *International Conference on Machine Learning*, 2016, pp. 2052–2060.

[16] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 447–461, 2015.

[17] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] Y. Xu, P. Cao, Y. Kong, and Y. Wang, "L_dmi: An information-theoretic noise-robust loss function," *NeurIPS, arXiv:1909.03388*, 2019.

[19] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *International Conference on Learning Representations*, 2020.

[20] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox, "Self: Learning to filter noisy labels with self-ensembling," *arXiv preprint arXiv:1910.01842*, 2019.

[21] Y. Kim, J. Yim, J. Yun, and J. Kim, "Nlnl: Negative learning for noisy labels," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 101–110.

[22] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou, "Empirical analysis of the hessian of over-parametrized neural networks," 2017.

[23] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[24] V. Shnayder, A. Agarwal, R. Frongillo, and D. C. Parkes, "Informed truthfulness in multi-task peer prediction," *EC 2016 - Proceedings of the 2016 ACM Conference on Economics and Computation*, pp. 179–196, 2016.

[25] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[26] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[28] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[29] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2691–2699.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[31] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *COLT 2010 - The 23rd Conference on Learning Theory*, vol. 12, pp. 257–269, 2010.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] ——, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[34] E. Amid, M. K. Warmuth, R. Anil, and T. Koren, "Robust bi-tempered logistic loss based on bregman divergences," 2019.

[35] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting," no. NeurIPS, pp. 1–23, 2019.

[36] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *arXiv preprint arXiv:1406.2080*, vol. 2, no. 3, p. 4, 2014.

[37] S. Jenni and P. Favaro, "Deep bilevel learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 618–633.

[38] K. Yi and J. Wu, "Probabilistic end-to-end noise correction for learning with noisy labels," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.