Are Anchor Points Really Indispensable in Label-Noise Learning?

Xiaobo Xia\textsuperscript{1,2} Tongliang Liu\textsuperscript{1} Nannan Wang\textsuperscript{2}
Bo Han\textsuperscript{3} Chen Gong\textsuperscript{4} Gang Niu\textsuperscript{3} Masashi Sugiyama\textsuperscript{3,5}

\textsuperscript{1}University of Sydney \textsuperscript{2}Xidian University \textsuperscript{3}RIKEN
\textsuperscript{4}Nanjing University of Science and Technology \textsuperscript{5}University of Tokyo

Abstract

In label-noise learning, the noise transition matrix, denoting the probabilities that clean labels flip into noisy labels, plays a central role in building statistically consistent classifiers. Existing theories have shown that the transition matrix can be learned by exploiting anchor points (i.e., data points that belong to a specific class almost surely). However, when there are no anchor points, the transition matrix will be poorly learned, and those previously consistent classifiers will significantly degenerate. In this paper, without employing anchor points, we propose a transition-revision (T-Revision) method to effectively learn transition matrices, leading to better classifiers. Specifically, to learn a transition matrix, we first initialize it by exploiting data points that are similar to anchor points, having high noisy class posterior probabilities. Then, we modify the initialized matrix by adding a slack variable, which can be learned and validated together with the classifier by using noisy data. Empirical results on benchmark-simulated and real-world label-noise datasets demonstrate that without using exact anchor points, the proposed method is superior to state-of-the-art label-noise learning methods.

1 Introduction

Label-noise learning can be dated back to [1] but becomes a more and more important topic recently. The reason is that, in this era, datasets are becoming bigger and bigger. Often, large-scale datasets are infeasible to be annotated accurately due to the expensive cost, which naturally brings us cheap datasets with noisy labels.

Existing methods for label-noise learning can be generally divided into two categories: algorithms that result in statistically inconsistent/consistent classifiers. Methods in the first category usually employ heuristics to reduce the side-effect of noisy labels. For example, many state-of-the-art approaches in this category are specifically designed to, e.g., select reliable examples [45, 14, 24], reweight examples [33, 15], correct labels [23, 17, 37, 32], employ side information [39, 21], and (implicitly) add regularization [13, 12, 43, 39, 21]. All those methods were reported to work empirically very well. However, the differences between the learned classifiers and the optimal ones for clean data are not guaranteed to vanish, i.e., no statistical consistency has been guaranteed.

The above issue motivates researchers to explore algorithms in the second category: risk-/classifier-consistent algorithms. In general, risk-consistent methods possess statistically consistent estimators to the clean risk (i.e., risk w.r.t. the clean data), while classifier-consistent methods guarantee the classifier learned from the noisy data is consistent to the optimal classifier (i.e., the minimizer of the clean risk) [42]. Methods in this category utilize the noise transition matrix, denoting the probabilities that clean labels flip into noisy labels, to build consistent algorithms. Let $Y$ denote the variable for the clean label, $\bar{Y}$ the noisy label, and $X$ the instance/feature. The basic idea is that given the

Given risk-consistent estimators, one stream to learn the transition matrix is the cross-validation method (using only noisy data) for binary classification [26]. However, it is prohibited for multi-class problems as its computational complexity grows exponentially to the number of classes. Besides, the current risk-consistent estimators involve the inverse of the transition matrix, making tuning the transition matrix inefficient and also leading to performance degeneration [30], especially when the transition matrix is non-invertible. Independent of risk-consistent estimators, another stream to learn the transition matrix is closely related to mixture proportion estimation [40]. A series of assumptions [36, 22, 35, 31] were proposed to efficiently learn transition matrices (or mixture parameters) by only exploiting the noisy data. All those assumptions require anchor points, i.e., instances belonging to a specific class with probability exactly one or close to one. Nonetheless, without anchor points, the transition matrix could be poorly learned, which will degenerate the accuracies of existing consistent algorithms.

Therefore, in this paper, to handle the applications where the anchor-point assumptions are violated [46, 41], we propose a transition-revision (T-Revision) method to effectively learn transition matrices, leading to better classifiers. In a high level, we design a deep-learning-based risk-consistent estimator to tune the transition matrix accurately. Specifically, we first initialize the transition matrix by exploiting examples that are similar to anchor points, namely, those having high estimated noisy class posterior probabilities. Then, we modify the initial matrix by adding a slack variable, which will be learned and validated together with the classifier by using noisy data only. Note that given true transition matrix, the proposed estimator will converge to the classification risk w.r.t. clean data by increasing the size of noisy training examples. Our heuristic for tuning the transition matrix is that a favorable transition matrix would make the classification risk w.r.t. clean data small. We empirically show that the proposed T-Revision method will enable tuned transition matrices to be closer to the ground truths, which explains why T-Revision is much superior to state-of-the-art algorithms in classification.

The rest of the paper is organized as follows. In Section 2 we review label-noise learning with anchor points. In Section 3, we discuss how to learn the transition matrix and classifier without anchor points. Experimental results are provided in Section 4. Finally, we conclude the paper in Section 5.

2 Label-Noise Learning with Anchor Points

In this section, we briefly review label-noise learning when there are anchor points.

Preliminaries Let $D$ be the distribution of a pair of random variables $(X, Y) \in \mathcal{X} \times \{1, 2, \ldots, C\}$, where the feature space $\mathcal{X} \subseteq \mathbb{R}^d$ and $C$ is the size of label classes. Our goal is to predict a label $y$ for any given instance $x \in \mathcal{X}$. However, in many real-world classification problems, training examples drawn independently from distribution $D$ are unavailable. Before being observed, their true labels are independently flipped and what we can obtain is a noisy training sample $\{(x_i, y_i)\}_{i=1}^n$, where $\bar{Y}$ denotes the noisy label. Let $\bar{D}$ be the distribution of the noisy random variables $(\bar{X}, \bar{Y}) \in \mathcal{X} \times \{1, 2, \ldots, C\}$.

Transition matrix The random variables $\bar{Y}$ and $Y$ are related through a noise transition matrix $T \in \{0, 1\}^{C \times C}$ [8]. Generally, the transition matrix depends on instances, i.e., $T_{ij}(X = x) = P(\bar{Y} = j|Y = i, X = x)$. Given only noisy examples, the instance-dependent transition matrix is non-identifiable without any additional assumption. For example, $P(\bar{Y} = j|X = x) = \sum_{i=1}^{C} T_{ij}(X = x)P(Y = i|X = x)$ is both valid, when $T_{ij}(X = x) = T_{ij}(X = x)P(\bar{Y} = j|X = x)/P(\bar{Y} = j|X = x)$. In this paper, we study the class-dependent and instance-independent transition matrix, i.e., $P(\bar{Y} = j|Y = i, X = x) = P(\bar{Y} = j|Y = i)$, which is identifiable under mild conditions and on which the vast majority of current methods focus [14, 13, 30, 29, 26].

noisy class posterior probability $P(\bar{Y}|X = x) = [P(\bar{Y} = 1|X = x), \ldots, P(\bar{Y} = C|X = x)]^\top$ (which can be learned using noisy data) and the transition matrix $T(X = x)$ where $T_{ij}(X = x) = P(\bar{Y} = j|Y = i, X = x)$, the clean class posterior probability $P(Y|X = x)$ can be inferred, i.e., $P(\bar{Y}|X = x) = (T(X = x))^\top P(Y|X = x)$. For example, loss functions are modified to ensure risk consistency, e.g., [49, 17, 22, 29, 35, 36]; a noise adaptation layer is added to deep neural networks to design classifier-consistent deep learning algorithms [9, 30, 38, 47]. Those algorithms are strongly theoretically grounded but heavily rely on the success of learning transition matrices.

Experimental results are provided in Section 4. Finally, we conclude the paper in Section 5.
**Consistent algorithms** The transition matrix bridges the class posterior probabilities for noisy and clean data, i.e., \( P(\bar{Y} = j | X = x) = \sum_{i=1}^{C} T_{ij} P(Y = i | X = x) \). Thus, it has been exploited to build consistent algorithms. Specifically, it has been used to modify loss functions to build risk-consistent estimators, e.g., [26, 35, 30], and has been used to correct hypotheses to build classifier-consistent algorithms, e.g., [9, 30, 47].

Note that an estimator is risk-consistent if, by increasing the size of noisy examples, the empirical risk calculated by noisy examples and the modified loss function will converge to the expected risk calculated by clean examples and the original loss function. Similarly, an algorithm is classifier-consistent if, by increasing the size of noisy examples, the learned classifier will converge to the optimal classifier learned by clean examples. Definitions of the expected and empirical risks can be found in Appendix B, where we further discuss how consistent algorithms work.

**Anchor points** The successes of consistent algorithms rely on firm bridges, i.e., accurately learned transition matrices. To learn transition matrices, the concept of anchor point was proposed [22, 35]. Anchor points are defined in the clean data domain, i.e., an instance \( x \) is an anchor point for the class \( i \) if \( P(Y = i | X = x) \) is equal to one or close to one\(^1\). Given an \( x \), if \( P(Y = i | X = x) = 1 \), we have that for \( k \neq i \), \( P(Y = k | X = x) = 0 \). Then, we have

\[
P(\bar{Y} = j | X = x) = \sum_{k=1}^{C} T_{kj} P(Y = k | X = x) = T_{ij}.
\]

Namely, \( T \) can be obtained via estimating the noisy class posterior probabilities for anchor points [47]. However, the requirement of given anchor points is a bit strong. Thus, anchor points are assumed to exist but unknown in datasets, which can be identified either theoretically [22] or heuristically [30].

Transition matrix learning is also closely related to mixture proportion estimation [40], which is independent of classification. By giving only noisy data, to ensure the learnability and efficiency of learning transition matrices (or mixture parameters), a series of assumptions were proposed, e.g., irreducibility [36], anchor point [22, 35], and separability [31]. All those assumptions require anchor points or instances belonging to a specific class with probability one or approaching one.

When there are no anchor points in datasets/data distributions, all the above mentioned methods will lead to inaccurate transition matrices, which will degenerate the performances of current consistent algorithms. This motivates us to investigate how to maintain the efficacy of those consistent algorithms without using exact anchor points.

### 3 Label-Noise Learning without Anchor Points

This section presents a deep-learning-based risk-consistent estimator for the classification risk w.r.t. clean data. We employ this estimator to tune the transition matrix effectively without using anchor points, which finally leads to better classifiers.

---

\(^1\)In the literature, the assumption \( \inf_{x} P(Y = i | X = x) \rightarrow 1 \) was introduced as irreducibility [5] to ensure the transition matrix is identifiable; an anchor point \( x \) for class \( i \) is defined by \( P(Y = i | X = x) = 1 \) [35, 22] to ensure a fast convergence rate. In this paper, we generalize the definition for the anchor point family, including instances whose class posterior probability \( P(Y = i | X = x) \) is equal to or close to one.

---
3.1 Motivation

According to Eq. (1), to learn the transition matrix, \( P(\hat{Y}|X = x) \) needs to be estimated and anchor points need to be given. Note that learning \( P(\hat{Y}|X = x) \) may introduce error. Even worse, when there are no anchor points, it will be problematic if we use existing methods [36, 22, 35, 31] to learn transition matrices. For example, let \( P(Y|X = x^i) \) be the \( i \)-th column of a matrix \( L \), \( i = 1,\ldots,C \). If \( x^i \) is an anchor point for the \( i \)-th class, then \( L \) is an identity matrix. According to Eq. (1), if we use \( x^i \) as an anchor point for the \( i \)-th class while \( P(Y = i|X = x^i) \neq 1 \) (e.g., the identified instances in [30] are not guaranteed to be anchor points), the learned transition matrix would be \( TL \), where \( L \) is a non-identity matrix. This means that transition matrices will be inaccurately estimated.

Based on inaccurate transition matrices, the accuracy of current consistent algorithms will significantly degenerate. To demonstrate this, Figure 1 shows that given a noisy class posterior probability \( P(\hat{Y}|X = x) \), even if the transition matrix changes slightly by two entries, e.g., \( \|T - \bar{T}\|_1/\|T\|_1 = 0.02 \) where \( T \) and \( \bar{T} \) are defined in Figure 1 and \( \|T\|_1 = \sum_{i,j} |T_{ij}| \), the inferred class posterior probability for the clean data may lead to an incorrect classification. Since anchor points require clean class posterior probabilities to be or approach one, which is quite strong to some real-world applications [46, 41], we would like to study how to maintain the performances of current consistent algorithms when there are no anchor points and then transition matrices are inaccurately learned.

3.2 Risk-consistent estimator

Intuitively, the entries of transition matrix can be tuned by minimizing the risk-consistent estimator, since the noisy class posterior probability \( P(\hat{Y}|X = x) \) does not have an identity matrix. This means that transition matrices will be inaccurately estimated.

Based on inaccurate transition matrices, the accuracy of current consistent algorithms will significantly degenerate. To demonstrate this, Figure 1 shows that given a noisy class posterior probability \( P(\hat{Y}|X = x) \), even if the transition matrix changes slightly by two entries, e.g., \( \|T - \bar{T}\|_1/\|T\|_1 = 0.02 \) where \( T \) and \( \bar{T} \) are defined in Figure 1 and \( \|T\|_1 = \sum_{i,j} |T_{ij}| \), the inferred class posterior probability for the clean data may lead to an incorrect classification. Since anchor points require clean class posterior probabilities to be or approach one, which is quite strong to some real-world applications [46, 41], we would like to study how to maintain the performances of current consistent algorithms when there are no anchor points and then transition matrices are inaccurately learned.

The inverse of transition matrix is involved in risk-consistent estimators, since the noisy class posterior probability \( P(\hat{Y}|X = x) \) is defined in Figure 1 and \( \|T\|_1 = \sum_{i,j} |T_{ij}| \), the inferred class posterior probability for the clean data may lead to an incorrect classification. Since anchor points require clean class posterior probabilities to be or approach one, which is quite strong to some real-world applications [46, 41], we would like to study how to maintain the performances of current consistent algorithms when there are no anchor points and then transition matrices are inaccurately learned.
Algorithm 1 Reweight $T$-Revision (Reweight-R) Algorithm.

**Input:** Noisy training sample $\mathcal{D}_t$; Noisy validation set $\mathcal{D}_v$.

**Stage 1: Learn $\hat{T}$**
1: Minimize the unweighted loss to learn $\hat{P}(\bar{Y}|X = x)$ without a noise adaption layer;
2: Initialize $\hat{T}$ according to Eq. (1) by using instances with the highest $\hat{P}(\bar{Y} = i|X = x)$ as anchor points for the $i$-th class;

**Stage 2: Learn the classifier $f$ and $\Delta T$**
3: Initialize the neural network by minimizing the weighted loss with a noisy adaption layer $\hat{T}^T$;
4: Minimize the weighted loss to learn $f$ and $\Delta T$ with a noisy adaption layer $(\hat{T} + \Delta T)^T$;

//Stopping criterion for learning $\hat{P}(\bar{Y}|X = x)$, $f$ and $\Delta T$: when $\hat{P}(\bar{Y}|X = x)$ yields the minimum classification error on the noisy validation set $\mathcal{D}_v$.

**Output:** $\hat{T}$, $\Delta T$, and $f$.

In the rest of the paper, we have omitted the subscript for $P$ when no confusion is caused. Since $P(\bar{Y}|X = x) = T^T P(\bar{Y}|X = x)$ and that the diagonal entries of (learned) transition matrices for label-noise learning are all much larger than zero, $P_D(\bar{Y} = i|X = x) \neq 0$ implies $P_D(\bar{Y} = i|X = x) \neq 0$, which also makes the proposed importance reweighting method stable without truncating the importance ratios.

Eq. (2) shows that the expected risk w.r.t. clean data and the loss $\ell(f(x), i)$ is equivalent to an expected risk w.r.t. noisy data and a reweighted loss, i.e., $P_D(Y = i|X = x)\ell(f(x), i)$. The empirical counterpart of the risk in the rightmost-hand side of Eq. (2) is therefore a risk-consistent estimator for label-noise learning. We exploit a deep neural network to build this counterpart. As shown in Figure 2, we use the output of the softmax function $g(x)$ to approximate $P(\bar{Y}|X = x)$, i.e., $g(x) = \hat{P}(\bar{Y}|X = x) \approx P(\bar{Y}|X = x)$. Then, $T^T g(x)$ (or $(\hat{T} + \Delta T)^T g(x)$ in the figure) is an approximation for $P(Y|X = x)$, i.e., $T^T g(x) = \hat{P}(\bar{Y}|X = x) \approx P(\bar{Y}|X = x)$. By employing $P(Y = y|X = x)/\hat{P}(\bar{Y} = y|X = x)$ as weight, we build the risk-consistent estimator as

$$
\tilde{R}_{n,w}(T, f) = \frac{1}{n} \sum_{i=1}^{n} \frac{g_{\bar{Y}_i}(X_i)}{(T^T g)_{\bar{Y}_i}(X_i)} \ell(f(X_i), \bar{Y}_i),
$$

where $f(X) = \arg \max_{j \in \{1, \ldots, C\}} g_j(X)$, $g_j(X)$ is an estimate for $P(Y = j|X)$, and the subscript $w$ denotes that the loss function is weighted. Note that if the true transition matrix $T$ is given, $\tilde{R}_{n,w}(T, f)$ only has one argument $g$ to learn.

### 3.3 Implementation and the $T$-revision method

When the true transition matrix $T$ is unavailable, we propose to use $\tilde{R}_{n,w}(\hat{T} + \Delta T, f)$ to approximate $R(f)$, as shown in Figure 2. To minimize $\tilde{R}_{n,w}(\hat{T} + \Delta T, f)$, a two-stage training procedure is proposed. Stage 1: first learn $P(\bar{Y}|X = x)$ by minimizing the unweighted loss without a noise adaption layer and initialize $\hat{T}$ by exploiting examples that have the highest learned $\hat{P}(\bar{Y}|X = x)$; Stage 2: modify the initialization $\hat{T}$ by adding a slack variable $\Delta T$ and learn the classifier and $\Delta T$ by minimizing the weighted loss. The procedure is called the Weighted $T$-Revision method and is summarized in Algorithm 1. It is worthwhile to mention that all anchor points based consistent estimators for label-noise learning have a similar two-stage training procedure. Specifically, with one stage to learn $P(\bar{Y}|X = x)$ and the transition matrix and a second stage to learn the classifier for the clean data.
The proposed $T$-revision method works because we learn $\Delta T$ by minimizing the risk-consistent estimator, which is asymptotically equal to the expected risk w.r.t. clean data. The learned slack variable can also be validated on the noisy validation set, i.e., to check if $P(\hat{Y}|X = x)$ fits the validation set. The philosophy of our approach is similar to that of the cross-validation method. However, the proposed method does not need to try different combinations of parameters ($\Delta T$ is learned) and thus is much more computationally efficient. Note that the proposed method will also boost the performances of consistent algorithms even there are anchor points as the transition matrices and classifiers are jointly learned. Note also that if a clean validation set is available, it can be used to better initialize the transition matrix, to better validate the slack variable $\Delta T$, and to fine-tune the deep network.

### 3.4 Generalization error

While we have discussed the use of the proposed estimator for evaluating the risk w.r.t. clean data, we theoretically justify how it generalizes for learning classifiers. Assume the neural network has $d$ layers, parameter matrices $W_1, \ldots, W_d$, and activation functions $\sigma_1, \ldots, \sigma_{d-1}$ for each layer. Let denote the mapping of the neural network by $h : x \mapsto W_d\sigma_{d-1}(W_{d-1}\sigma_{d-2}(\ldots \sigma_1(W_1x))) \in \mathbb{R}^C$.

Then, the output of the softmax is defined by $g_i(x) = \exp(h_i(x))/\sum_{k=1}^C \exp(h_k(x)), i = 1, \ldots, C$.

Let $\hat{f} = \arg \max_{\ell \in \{1, \ldots, C\}} \hat{g}_\ell$ be the classifier learned from the hypothesis space $F$ determined by the real-valued parameters of the neural network, i.e., $\hat{f} = \arg \min_f \bar{R}_{n,w}(f)$.

To derive a generalization bound, as the common practice [6, 25], we assume that instances are upper bounded by $B$, i.e., $\|x\| \leq B$ for all $x \in \mathcal{X}$, and that the loss function is $L$-Lipschitz continuous w.r.t. $f(x)$ and upper bounded by $M$, i.e., for any $f_1, f_2 \in F$ and any $x, \bar{y}$, $|\ell(f_1(x), \bar{y}) - \ell(f_2(x), \bar{y})| \leq L|f_1(x) - f_2(x)|$, and for any $x, \bar{y}$, $\ell(f(x), \bar{y}) \leq M$.

**Theorem 1** Assume the Frobenius norm of the weight matrices $W_1, \ldots, W_d$ are at most $M_1, \ldots, M_d$. Let the activation functions be 1-Lipschitz, positive-homogeneous, and applied element-wise (such as the ReLU). Let the loss function be the cross-entropy loss, i.e., $\ell(f(x), \bar{y}) = -\sum_{i=1}^C 1(\bar{y}_i = 1) \log(g_i(x))$. Let $\hat{f}$ and $\Delta \hat{T}$ be the learned classifier and slack variable. Assume $\Delta \hat{T}$ is searched from a space of $\Delta T$ constituting valid transition matrices\textsuperscript{2}, i.e., $\forall \Delta T$ and $\forall i \neq j$, $\hat{T}_{ij} + \Delta T_{ij} \geq 0$ and $\hat{T}_{ii} + \Delta T_{ii} > \hat{T}_{ij} + \Delta T_{ij}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$,

$$
\mathbb{E}[\bar{R}_{n,w}((\hat{T} + \Delta \hat{T}, \hat{f})) - \bar{R}_{n,w}(\hat{T} + \Delta \hat{T}, \hat{f})] \leq \frac{2BCL(\sqrt{2d\log 2} + 1)|\Pi_{i=1}^d M_i|}{\sqrt{n}} + CM \sqrt{\frac{\log 1/\delta}{2n}}.
$$

A detailed proof is provided in Appendix C. The factor $(\sqrt{2d\log 2} + 1)|\Pi_{i=1}^d M_i|$ is induced by the hypothesis complexity of the deep neural network [10] (see Theorem 1 therein), which could be improved [27, 48, 16]. Although the proposed reweighted loss is more complex than the traditional unweighted loss function, we have derived a generalization error bound not larger than those derived for the algorithms employing the traditional loss [25] (can be seen by Lemma 2 in the proof of the theorem). This shows that the proposed Algorithm 1 does not need a larger training sample to achieve a small difference between training error ($\bar{R}_{n,w}(\hat{T} + \Delta \hat{T}, \hat{f})$) and test error ($\mathbb{E}[\bar{R}_{n,w}(T + \Delta T, f)]$). Also note that deep learning is powerful in yielding a small training error. If the training sample size $n$ is large, then the upper bound in Theorem 1 is small, which implies a small $\mathbb{E}[\bar{R}_{n,w}(T + \Delta T, f)]$ and justifies why the proposed method will have small test errors in the experiment section. Meanwhile, in the experiment section, we show that the proposed method is much superior to the state-of-the-art methods in classification accuracy, implying that the small generalization error is not obtained at the cost of enlarging the approximation error.

### 4 Experiments

**Datasets** We verify the effectiveness of the proposed method on three synthetic noisy datasets, i.e., MNIST [19], CIFAR-10 [18], and CIFAR-100 [18], and one real-world noisy dataset, i.e., clothing1M.

\textsuperscript{2}During the training, $T + \Delta T$ can be ensured to be a valid transition matrix by first projecting their negative entries to be zero and then performing row normalization. In the experiments, $\Delta T$ is initialized to be a zero matrix and we haven’t pushed $T + \Delta T$ to be a valid matrix when tuning $\Delta T$. 


Since the transition matrix used is revised by a revision $\Delta \hat{T}$, the datasets may contain light label noise and lead to around 20% of instances to have noisy labels. Note that the pair flipping $T$ data. We corrupted the training and validation sets manually according to true transition matrices $T$. Specifically, we employ the symmetry flipping setting defined in Appendix D. Sym-50 generates heavy label noise and leads around 20% of instances to have noisy labels, while Sym-20 generates light label noise and leads around 20% of instances to have label noise. Note that the pair flipping setting [14], where each row of the transition matrix only has two non-zero entries, has also been widely studied. However, for simplicity, we do not pose any constraint on the slack variable $\Delta T$ to achieve specific speculation of the transition matrix, e.g., sparsity [13]. We leave this for future work.

Besides reporting the classification accuracy on test set, we also report the discrepancy between the learned transition matrix $\hat{T} + \Delta \hat{T}$ and the true one $T$. All experiments are repeated five times on those three datasets. ClothingLM consists of 1M images with real-world noisy labels, and additional 50k, 14k, 10k images with clean labels for training, validation, and testing. We use the 50k clean data to help initialize the transition matrix as did in the baseline [30].

**Network structure and optimization** For fair comparison, we implement all methods with default parameters by PyTorch on NVIDIA Tesla V100. We use a LeNet-5 network for MNIST, a ResNet-18 network for CIFAR-10, and a ResNet-34 network for CIFAR-100. For learning the transition matrix $\hat{T}$ in the first stage, we follow the optimization method in [30]. During the second stage, we first use SGD with momentum 0.9, weight decay $10^{-4}$, batch size 128, and an initial learning rate of $10^{-2}$ to initialize the network. The learning rate is divided by 10 after the 40th epoch and 80th epoch. 200 epochs are set in total. Then, the optimizer and learning rate are changed to Adam and $5 \times 10^{-7}$ to learn the classifier and slack variable. For CIFAR-10 and CIFAR-100, we perform data augmentation by horizontal random flips and $32 \times 32$ random crops after padding 4 pixels on each side. For clothingLM, we use a ResNet-50 pre-trained on ImageNet. Follow [30], we also exploit the 1M noisy data and 50k clean data to initialize the transition matrix. In the second stage, for initialization, we use SGD with momentum 0.9, weight decay $10^{-3}$, batch size 32, and run with learning rates $10^{-3}$ and $10^{-4}$ for 5 epochs each. For learning the classifier and slack variable, Adam is used and the learning rate is changed to $5 \times 10^{-7}$.

### Table 1: Means and standard deviations (percentage) of classification accuracy. Methods with “-A” means that they run on the intact datasets without removing possible anchor points; Methods with “-R” means that the transition matrix used is revised by a revision $\Delta \hat{T}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sym-20%</td>
<td>Sym-50%</td>
<td>Sym-20%</td>
</tr>
<tr>
<td>Decoupling-A</td>
<td>95.39 ± 0.29</td>
<td>81.52 ± 0.29</td>
<td>79.85 ± 0.30</td>
</tr>
<tr>
<td>MentorNet-A</td>
<td>96.57 ± 0.18</td>
<td>90.13 ± 0.09</td>
<td>80.49 ± 0.52</td>
</tr>
<tr>
<td>Co-teaching-A</td>
<td>97.22 ± 0.18</td>
<td>91.68 ± 0.21</td>
<td>82.38 ± 0.11</td>
</tr>
<tr>
<td>Forward-A</td>
<td>98.75 ± 0.08</td>
<td>97.86 ± 0.22</td>
<td>85.63 ± 0.52</td>
</tr>
<tr>
<td>Reweight-A</td>
<td>98.71 ± 0.11</td>
<td>98.13 ± 0.19</td>
<td>86.77 ± 0.40</td>
</tr>
<tr>
<td>Forward-A-R</td>
<td>98.84 ± 0.09</td>
<td>98.12 ± 0.22</td>
<td>88.10 ± 0.21</td>
</tr>
<tr>
<td>Reweight-A-R</td>
<td>98.91 ± 0.04</td>
<td>98.38 ± 0.21</td>
<td>89.63 ± 0.13</td>
</tr>
</tbody>
</table>

### Table 2: Means and standard deviations (percentage) of classification accuracy. Methods with “-N/A” means instances with high estimated $P(Y|X)$ are removed from the dataset; Methods with “-R” means that the transition matrix used is revised by a revision $\Delta \hat{T}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sym-20%</td>
<td>Sym-50%</td>
<td>Sym-20%</td>
</tr>
<tr>
<td>Decoupling-N/A</td>
<td>95.93 ± 0.21</td>
<td>82.55 ± 0.39</td>
<td>75.37 ± 1.24</td>
</tr>
<tr>
<td>MentorNet-N/A</td>
<td>97.11 ± 0.09</td>
<td>91.44 ± 0.25</td>
<td>78.51 ± 0.31</td>
</tr>
<tr>
<td>Co-teaching-N/A</td>
<td>97.69 ± 0.23</td>
<td>93.58 ± 0.49</td>
<td>81.72 ± 0.14</td>
</tr>
<tr>
<td>Forward-N/A</td>
<td>98.64 ± 0.12</td>
<td>97.74 ± 0.13</td>
<td>84.75 ± 0.81</td>
</tr>
<tr>
<td>Reweight-N/A</td>
<td>98.69 ± 0.08</td>
<td>98.05 ± 0.22</td>
<td>85.53 ± 0.26</td>
</tr>
<tr>
<td>Forward-N/R-A</td>
<td>98.80 ± 0.06</td>
<td>97.96 ± 0.13</td>
<td>86.93 ± 0.39</td>
</tr>
<tr>
<td>Reweight-N/R-A</td>
<td>98.85 ± 0.02</td>
<td>98.37 ± 0.17</td>
<td>88.90 ± 0.22</td>
</tr>
</tbody>
</table>

[44]. MNIST has 10 classes of images including 60,000 training images and 10,000 test images. CIFAR-10 has 10 classes of images including 50,000 training images and 10,000 test images. CIFAR-100 also has 50,000 training images and 10,000 test images, but 100 classes. For all the datasets, we leave out 10% of the training examples as a validation set. The three datasets contain clean data. We corrupted the training and validation sets manually according to true transition matrices $T$. Specifically, we employ the symmetry flipping setting defined in Appendix D. Sym-50 generates heavy label noise and leads almost half of the instances to have noisy labels, while Sym-20 generates light label noise and leads around 20% of instances to have label noise. Note that the pair flipping setting [14], where each row of the transition matrix only has two non-zero entries, has also been widely studied. However, for simplicity, we do not pose any constraint on the slack variable $\Delta T$ to achieve specific speculation of the transition matrix, e.g., sparsity [13]. We leave this for future work.
Table 3: Means and standard deviations (percentage) of classification accuracy on \textit{MNIST} with different label noise levels. Methods with “-A” means that they run on the intact datasets without removing possible anchor points; Methods with “-R” means that the transition matrix used is revised by a revision $\Delta \hat{T}$; Methods with “-N/A” means instances with high estimated $P(Y|X)$ are removed from the dataset.

<table>
<thead>
<tr>
<th></th>
<th>Sym-60%</th>
<th>Sym-70%</th>
<th>Sym-80%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward-A</td>
<td>97.10±0.08</td>
<td>96.06±0.41</td>
<td>91.46±1.03</td>
</tr>
<tr>
<td>Forward-A-R</td>
<td>97.65±0.11</td>
<td>96.42±0.35</td>
<td>91.77±0.22</td>
</tr>
<tr>
<td>Reweight-A</td>
<td>97.39±0.27</td>
<td>96.25±0.26</td>
<td>93.79±0.52</td>
</tr>
<tr>
<td>Reweight-A-R</td>
<td>97.83±0.18</td>
<td>97.13±0.08</td>
<td>94.19±0.45</td>
</tr>
<tr>
<td>Forward-N/A</td>
<td>96.82±0.14</td>
<td>94.61±0.28</td>
<td>85.95±1.01</td>
</tr>
<tr>
<td>Forward-N/A-R</td>
<td>96.99±0.16</td>
<td>95.02±0.17</td>
<td>86.04±1.03</td>
</tr>
<tr>
<td>Reweight-N/A</td>
<td>97.01±0.20</td>
<td>95.94±0.14</td>
<td>91.59±0.70</td>
</tr>
<tr>
<td>Reweight-N/A-R</td>
<td>97.81±0.12</td>
<td>96.59±0.15</td>
<td>91.91±0.65</td>
</tr>
</tbody>
</table>

Table 4: Classification accuracy (percentage) on \textit{Clothing1M}.

<table>
<thead>
<tr>
<th>Method</th>
<th>Decoupling</th>
<th>MentorNet</th>
<th>Co-teaching</th>
<th>Forward</th>
<th>Reweight</th>
<th>Forward-R</th>
<th>Reweight-R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>53.98</td>
<td>56.77</td>
<td>58.68</td>
<td>71.79</td>
<td>70.95</td>
<td>72.25</td>
<td>74.18</td>
</tr>
</tbody>
</table>

\textbf{Baselines} We compare the proposed method with state-of-the-art approaches. Specifically, we compare with the following three inconsistent but well-designed algorithms: Decoupling [24], MentorNet [15], and Co-teaching [14], which free the learning of transition matrices. To compare with consistent estimators, we set Forward [30], a classifier-consistent algorithm, and the importance reweighting method (Reweight), a risk-consistent algorithm, as baselines. The risk-consistent estimator involving the inverse of transition matrix, e.g., Backward in [30], has not been included in the comparison, because it has been reported to perform worse than the Forward method [30].

4.1 \textbf{Comparison for classification accuracy}

\textbf{The importance of anchor points} To show the importance of anchor points, we modify the datasets by moving possible anchor points, i.e., instances with large estimated class posterior probability $P(Y|X)$, before corrupting the training and validation sets. As the \textit{MNIST} dataset is simple, we removed 40\% of the instances with the largest estimated class posterior probabilities in each class. For \textit{CIFAR-10} and \textit{CIFAR-100}, we removed 20\% of the instances with the largest estimated class posterior probabilities in each class. To make it easy for distinguishing, we mark a “-A” in the algorithm’s name if it runs on the original intact datasets, and mark a “-N/A” in the algorithm’s name if it runs on those modified datasets.

Comparing Decoupling-A, MentorNet-A, and Co-teaching-A in Table 1 with Decoupling-N/A, MentorNet-N/A, and Co-teaching-N/A in Table 2, we can find that on \textit{MNIST}, the methods with “-N/A” work better; while on \textit{CIFAR-10} and \textit{CIFAR-100}, the methods with “-A” work better. This is because those methods are independent of transition matrices but dependent of dataset properties. Removing possible anchors points may not always lead to performance degeneration.

Comparing Forward-A and Reweight-A with Forward-N/A and Reweight-N/A, we can find that the methods without anchor points, i.e., with “-N/A”, degenerate clearly. The degeneration on \textit{MNIST} is slight because the dataset can be well separated and many instances have high class posterior probability even in the modify dataset. Those results show that, without anchor points, the consistent algorithms will have performance degeneration. Specifically, on \textit{CIFAR-100}, the methods with “-N/A” have much worse performance than the ones with “-A”, with accuracy dropping at least 4\%.

To discuss the model performances on \textit{MNIST} with more label noise, we raise the noise rates to 60\%, 70\%, 80\%. Other experiment settings are unchanged. The results are presented in Table 3. We can see that the proposed model outperforms the baselines more significantly as the noise rate grows.

\textbf{Risk-consistent estimator vs. classifier-consistent estimator} Comparing Forward-A with Reweight-A in Table 1 and comparing Forward-N/A with Reweight-N/A in Table 2, it can be seen that the proposed Reweight method, a risk-consistent estimator not involving the inverse of transition matrix, works slightly better than or is comparable to Forward, a classifier-consistent
algorithm. Note that in [30], it is reported that Backward, a risk-consistent estimator which involves the inverse of the transition matrix, works worse than Forward, the classifier-consistent algorithm.

The importance of T-revision Note that for fair comparison, we also set it as a baseline to modify the transition matrix in Forward. As shown in Tables 1 and 2, methods with “-R” means that they use the proposed T-revision method, i.e., modify the learned $\hat{T}$ by adding $\Delta \hat{T}$. Comparing the results in Tables 1 and 2, we can find that the T-revision method significantly outperforms the others. Among them, the proposed Reweight-R works significantly better than the baseline Forward-R. We can find that the T-Revision method boosts the classification performance even without removing possible anchor points. The rationale behind this may be that the network, transition matrix, and classifier are jointly learned and validated and that the identified anchor points are not reliable.

Comparison on real-world dataset The proposed T-revision method significantly outperforms the baselines as shown in Table 4, where the highest accuracy is bold faced.

4.2 Comparison for estimating transition matrices

To show that the proposed risk-consistent estimator is more effective in modifying the transition matrix, we plot the estimation error for the transition matrix, i.e., $\| T - \hat{T} - \Delta \hat{T} \|_1 / \| T \|_1$. In Figure 4, we can see that for all cases, the proposed risk-consistent-estimator-based revision leads to smaller estimator errors than the classifier-consistent algorithm based method (Forward-R), showing that the risk-consistent estimator is more powerful in modifying the transition matrix. This also explains why the proposed method works better. We provide more discussions about Figure 4 in Appendix E.

5 Conclusion

This paper presents a risk-consistent estimator for label-noise learning without involving the inverse of transition matrix and a simple but effective learning paradigm called T-revision, which trains deep neural networks robustly under noisy supervision. The aim is to maintain effectiveness and efficiency of current consistent algorithms when there are no anchor points and then the transition matrices are poorly learned. The key idea is to revise the learned transition matrix and validate the revision by exploiting a noisy validation set. We conduct experiments on both synthetic and real-world label noise data to demonstrate that the proposed T-revision can significantly help boost the performance of label-noise learning. In the future, we will extend the work in the following aspects. First, how to incorporate some prior knowledge of the transition matrix, e.g., sparsity, into the end-to-end learning system. Second, how to recursively learn the transition matrix and classifier as our experiments show that transition matrices can be refined.
Acknowledgments

TLL was supported by Australian Research Council Project DP180103424 and DE190101473. NNW was supported by National Natural Science Foundation of China under Grants 61922066, 61876142, and the CCF-Tencent Open Fund. CG was supported by NSF of China under Grants 61602246, 61973162, NSF of Jiangsu Province under Grants BK20171430, the Fundamental Research Funds for the Central Universities under Grants 30918011319, and the “Young Elite Scientists Sponsorship Program” by CAST under Grants 2018QNRC001. MS was supported by the International Research Center for Neurointelligence (WPI-IRCN) at The University of Tokyo Institutes for Advanced Study. XBX and TLL would give special thanks to Haifeng Liu and Brain-Inspired Technology Co., Ltd. for their support of GPUs used for this research.

References


[24] Eran Malach and Shai Shalev-Shwartz. Decoupling ”when to update” from ”how to update”. In NeurIPS, pages 960–970, 2017.


A  Label-noise learning, or noisy-label learning, that is the point

Note that the title of this paper is a question “are anchor points really indispensable in label-noise learning” but not “are anchor points really indispensable in noisy-label learning”. Here, we explain why in order to make sense it must be label-noise learning. At first glance, label-noise learning may sound like we are learning the label noise, but this is exactly what we have implied in the title.

Generally speaking, the two names are synonyms of learning with noisy labels—this is the title of [26] where the first statistically consistent learning method was proposed for training classifiers with noisy labels. For the family of consistent learning methods, the estimation of the transition matrix $T$ is always necessary. In fact, any such method requires three components:

- a label corruption process parameterized by $T$,
- an estimator of $T$, and
- a statistical or algorithmic correction using the estimated $T$.

As a result, $T$ is also a target of learning, that is, the label noise is both a target to be learned and a source from which we learn. This is learning with noisy labels, more than learning from noisy labels.

For the first component, we assume the class-conditional noise model [26, 30]; for the third component, we rely on importance reweighting for learning with noisy labels [22]. Our novelty and major contribution is the second component, where we relax a requirement in existing consistent learning methods, namely we should have a certain amount of anchor points for estimating $T$ accurately, so that the correction using the estimated $T$ can be performed well.

While it is necessary to estimate $T$ for consistent learning methods, it is not the case for inconsistent learning methods. For instance, sample selection methods try to remove mislabeled data, and label correction methods try to fix the wrong labels of mislabeled data. None of them estimate $T$ so that none of them ever need the existence of anchor points.

Therefore, if we ask “are anchor points really indispensable in label-noise learning” where we are learning, modeling or estimating the label noise, the answer was yes previously and is no currently. Nevertheless, if we ask “are anchor points really indispensable in noisy-label learning” where some inconsistent learning method is employed without estimating the label noise, the answer has already been known to be no. That is the point.

B  How consistent algorithms work

The aim of multi-class classification is to learn a hypothesis $f$ that predicts labels for given instances. Typically, the hypothesis is of the following form: $f(x) = \arg \max_{i \in \{1, 2, \ldots, C\}} g_i(x)$, where $g_i(x)$ is an estimate of $P(Y = i | X = x)$. Let define the expected risk of employing $f$ as

$$R(f) = \mathbb{E}_{(X, Y) \sim D}[\ell(f(X), Y)].$$

The optimal hypothesis to learn is the one that minimizes the risk $R(f)$. Usually, the distribution $D$ is unknown. The optimal hypothesis is approximated by the minimizer of an empirical counterpart of $R(f)$, i.e., the empirical risk

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).$$

The empirical risk $R_n(f)$ is risk-consistent w.r.t. all loss functions, i.e., $R_n(f) \to R(f)$ as $n \to \infty$. Note that in the main paper, we have treated the training sample $\{X_i, Y_i\}_{i=1}^{n}$ as iid variables to derive the generalization bound.

If the loss function is zero-one loss, i.e., $\ell(f(x), y) = 1_{f(x) \neq y}$ where $1_{\cdot}$ is the indicator function and that the predefined hypothesis class [25] is large enough, the optimal hypothesis that minimizing $R(f)$ is identical to the Bayes classifier [3], i.e.,

$$f_\rho(x) = \arg \max_{i \in \{1, 2, \ldots, C\}} P(Y = i | X = x).$$

Many frequently used loss functions are proven to be classification-calibrated [3, 34], which means they will lead to classifiers having the same predictions as the classifier learned by using zero-one
loss if the training sample size is sufficiently large \cite{42,25}. In other words, the approximation, i.e., \( \arg\min R_n(f) \), could converge to the optimal hypothesis by increasing the sample size \( n \) and the corresponding estimator is therefore classifier-consistent. Note that risk-consistent algorithm is also classifier-consistent. However, a classifier-consistent algorithm may not be risk-consistent.

Given only the noisy training sample \( \{(X_i, \tilde{Y}_i)\}_{i=1}^{n} \), we have a noisy version of the empirical risk as

\[
\tilde{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), \tilde{Y}_i).
\]

The learned \( g(X) \) can be used to approximate \( P(\tilde{Y}|X) \). According to the definition of transition matrix, we have that \( P(\tilde{Y}|X) = T^\top P(Y|X) \), implying that if we let

\[
\hat{h}(X) = \arg \max_{i \in \{1,2,...,C\}} (T^\top g)_i(X),
\]

minimizing

\[
\tilde{R}_n(\hat{h}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\hat{h}(X_i), \tilde{Y}_i)
\]

by using only noisy data will lead to a classifier-consistent algorithm. In other words, \( \arg \max_{i \in \{1,2,...,C\}} g_i(x) \) in the algorithm will converge to the optimal classifier for clean data by increasing the noisy sample size. That’s why noise adaption layer has been widely used in deep learning to modify the softmax function (i.e., \( g(x) \)) \cite{9,30,38,47}.

If the transition matrix is invertable, the equation \( P(Y|X) = (T^\top)^{-1} P(\tilde{Y}|X) \) has been explored to design risk-consistent estimator for \( R(f) \), e.g., \cite{26,30}. The basic idea is to modify the loss function \( \ell(f(X), \tilde{Y}) \) to be \( \tilde{\ell}(f(X), \tilde{Y}) \) such that for \( X \) and \( Y \),

\[
E_{\tilde{Y}}[\tilde{\ell}(f(X), \tilde{Y})] = \ell(f(X), Y)
\]

and thus

\[
E_{(X,Y,\tilde{Y})}\tilde{\ell}(f(X), \tilde{Y}) = R(f).
\]

Specifically, let

\[
\mathcal{L}(f(X), Y) = [\ell(f(X), Y = 1), \ldots, \ell(f(X), Y = C)]^\top
\]

and

\[
\tilde{\mathcal{L}}(f(X), \tilde{Y}) = [\tilde{\ell}(f(X), \tilde{Y} = 1), \ldots, \tilde{\ell}(f(X), \tilde{Y} = C)]^\top = (T^\top)^{-1} \mathcal{L}(f(X), \tilde{Y}).
\]

The losses \( \tilde{\ell} \) will lead to risk-consistent estimator because

\[
E_{\tilde{Y}|Y}[\tilde{\mathcal{L}}(f(X), \tilde{Y})] = T^\top \tilde{\mathcal{L}}(f(X), \tilde{Y}) = \mathcal{L}(f(X), Y).
\]

Risk-consistent algorithms are also classifier-consistent, but have some unique properties than classifier-consistent algorithms, e.g., can be used to tune hyper-parameter. However, the current risk-consistent estimators contain the inverse of transition matrix, making parameter tuning inefficient and leading to performance degeneration. Our proposed risk-consistent estimator overcome the aforementioned issues.

## C Proof of Theorem 1

We have defined

\[
\tilde{R}_{n,w}(\hat{T} + \Delta T, f) = \frac{1}{n} \sum_{i=1}^{n} \frac{g_{\tilde{Y}_i}(X_i)}{((\hat{T} + \Delta T)^\top g)_{\tilde{Y}_i}} \ell(f(X_i), \tilde{Y}_i),
\]

where \( f(X) = \arg \max_{i \in \{1,\ldots,C\}} g_i(X) \). Let \( S = \{(X_1, \tilde{Y}_1), \ldots, (X_n, \tilde{Y}_n)\} \), \( S^i = \{(X_1, \tilde{Y}_1), \ldots, (X_{i-1}, \tilde{Y}_{i-1}), (X_i, \tilde{Y}_i'), (X_{i+1}, \tilde{Y}_{i+1}), \ldots, (X_n, \tilde{Y}_n)\} \), and

\[
\Phi(S) = \sup_{\Delta T, f} (\tilde{R}_{n,w}(\hat{T} + \Delta T, f) - E_S[\tilde{R}_{n,w}(\hat{T} + \Delta T, f)]).
\]
Lemma 1 Let $\Delta \hat{T}$ and $\hat{f}$ be the learned slack variable and classifier respectively. Assume the learned transition matrix is valid, i.e., $T_{ij} + \Delta T_{ij} \geq 0$ for all $i, j$ and $T_{ii} + \Delta T_{ii} > T_{ij} + \Delta T_{ij}$ for all $j \neq i$. For any $\delta > 0$, with probability at least $1 - \delta$, we have

\[
E[\hat{R}_{n,w}(\hat{T} + \Delta \hat{T}, \hat{f})] - \hat{R}_{n,w}(\hat{T} + \Delta \hat{T}, \hat{f}) \leq E[\Phi(S)] + CM \sqrt{\frac{\log 1/\delta}{2n}}.
\]  

(17)

Detailed proof of Lemma 1 is provided in Section C.1.

Using the same trick to derive Rademacher complexity [4], we have

\[
E[\Phi(S)] \leq 2E \left[ \sup_{\Delta T, f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g_i(X_i) \left( (\hat{T} + \Delta T)\bar{\gamma}_i(Y_i) \right) \ell(f(X_i), \bar{Y}_i) \right],
\]  

where $\sigma_1, \ldots, \sigma_n$ are i.i.d. Rademacher random variables.

We can upper bound the right hand part of the above inequality by the following lemma.

Lemma 2

\[
E \left[ \sup_{\Delta T, f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g_i(X_i) \left( (\hat{T} + \Delta T)\bar{\gamma}_i(Y_i) \right) \ell(f(X_i), \bar{Y}_i) \right] \leq E \left[ \sup_{f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(f(X_i), \bar{Y}_i) \right].
\]  

(19)

Note that Lemma 2 is not an application of Talagrand Contraction Lemma [20]. Detailed proof of Lemma 2 is provided in Section C.2.

Recall that $f = \arg \max_{i \in \{1, \ldots, C\}} g_i$ is the classifier, where $g$ is the output of the softmax function, i.e., $g_i(X) = \exp(h_i(X))/\sum_{k=1}^{C} \exp(h_k(X))$, $i = 1, \ldots, C$, and $h(X)$ is defined by a $d$-layer neural network, i.e., $h : X \rightarrow W_d \sigma_{d-1} (W_{d-1} \sigma_{d-2} (\ldots \sigma_1 (W_1 X) \ldots)) \in \mathbb{R}^C$, $W_1, \ldots, W_d$ are the parameter matrices, and $\sigma_1, \ldots, \sigma_{d-1}$ are activation functions. To further upper bound the Rademacher complexity, we need to consider the Lipschitz continuous property of the loss function w.r.t. to $h(X)$.

To avoid more assumption, We discuss the widely used cross-entropy loss, i.e.,

\[
\ell(f(X), \bar{Y}) = -\sum_{i=1}^{C} 1_{\{Y_i = i\}} \log(g_i(X)).
\]  

(20)

We can further upper bound the Rademacher complexity by the following lemma.

Lemma 3

\[
E \left[ \sup_{f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(f(X_i), \bar{Y}_i) \right] \leq CLE \left[ \sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(X_i) \right],
\]  

(21)

where $H$ is the function class induced by the deep neural network.

Detailed proof of Lemma 3 is provided in Section C.3.

Note that $E \left[ \sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(X_i) \right]$ measures the hypothesis complexity of deep neural networks, which has been widely studied recently [27, 2, 10, 28]. Specifically, [10] proved the following theorem (Theorem 1 therein).

Theorem 2 Assume the Frobenius norm of the weight matrices $W_1, \ldots, W_d$ are at most $M_1, \ldots, M_d$. Let the activation functions be 1-Lipschitz, positive-homogeneous, and applied element-wise (such as the ReLU). Let $x$ is upper bounded by $B$, i.e., for any $x \in X$, $\|x\| \leq B$. Then,

\[
E \left[ \sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(X_i) \right] \leq \frac{B(\sqrt{2d \log 2} + 1)\Pi_{i=1}^{d} M_i}{\sqrt{n}}.
\]  

(22)

Theorem 1 follows by combining Lemmas 1, 2, 3, and Theorem 2.
C.1 Proof of Lemma 1

We employ McDiarmid’s concentration inequality [7] to prove the lemma. We first check the bounded difference property of $\Phi(S)$, e.g.,

$$
\Phi(S) - \Phi(S') \leq \sup_{\Delta T, f} \frac{1}{n} \left( \sum_{i=1}^{n} \sigma_i \frac{g_{T_i}(X_i)}{((T + \Delta T)^\top g)_{T_i}(X_i)} \ell(f(X_i), \tilde{Y}_i) \right).
$$

(23)

Before further upper bounding the above difference, we show that the weighted loss is upper bounded by $CM$. Specifically, we have assume the learned transition matrix is valid, i.e., $\hat{T}_{ii} + \Delta T_{ii} \geq 0$ for all $i, j$ and $\hat{T}_{ij} + \Delta T_{ij} \geq \hat{T}_{ij} + \Delta T_{ij}$ for all $j \neq i$. Thus $\frac{g_{T}(X)}{((T + \Delta T)^\top g)_{T}(X)} \leq 1/ \min_i (\hat{T}_{ii} + \Delta T_{ii}) \leq C$ for any $(X, \hat{Y})$ and $\tilde{g}$. Then, we can conclude that the weighted loss is upper bounded by $CM$ and that

$$
\Phi(S) - \Phi(S') \leq \frac{CM}{n}.
$$

(24)

Similarly, we could prove that $\Phi(S^i) - \Phi(S) \leq \frac{CM}{n}$.

By employing McDiarmid’s concentration inequality, for any $\delta > 0$ with probability at least $1 - \delta$, we have

$$
\Phi(S) - \mathbb{E}[\Phi(S)] \leq CM \sqrt{\frac{\log(1/\delta)}{2n}}.
$$

(25)

C.2 Proof of Lemma 2

Given the learned transition matrix is valid, we have shown that $\frac{g_{T}(X)}{((T + \Delta T)^\top g)_{T}(X)} \leq 1/ \min_i (\hat{T}_{ii} + \Delta T_{ii}) \leq C$ for all $(X, \hat{Y})$ in the proof of Lemma 1.

Lemma 2 holds of we could prove the following inequality

$$
\mathbb{E}_\sigma \left[ \sup_{\Delta T, f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \frac{g_{T_i}(X_i)}{((T + \Delta T)^\top g)_{T_i}(X_i)} \ell(f(X_i), \tilde{Y}_i) \right] \leq \mathbb{E}_\sigma \left[ \sup_{f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(f(X_i), \tilde{Y}_i) \right].
$$

(26)

Note that

$$
\mathbb{E}_\sigma \left[ \sup_{\Delta T, f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \frac{g_{T_i}(X_i)}{((T + \Delta T)^\top g)_{T_i}(X_i)} \ell(f(X_i), \tilde{Y}_i) \right] = \mathbb{E}_{\sigma_1, \ldots, \sigma_{n-1}} \left[ \mathbb{E}_{\sigma_n} \left[ \sup_{\Delta T, f} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \frac{g_{T_i}(X_i)}{((T + \Delta T)^\top g)_{T_i}(X_i)} \ell(f(X_i), \tilde{Y}_i) \right] \right].
$$

(27)

Let $s_{n-1}(\Delta T, f) = \sum_{i=1}^{n-1} \sigma_i \frac{g_{T_i}(X_i)}{((T + \Delta T)^\top g)_{T_i}(X_i)} \ell(f(X_i), \tilde{Y}_i)$. By definition of the supremum, for any $\epsilon > 0$, there exist $(\Delta T, f_1)$ and $(\Delta T, f_2)$ such that

$$
\frac{g_{T_n}(X_n)}{((T + \Delta T)^\top g)_{T_n}(X_n)} \ell(f_1(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f_1) \geq (1 - \epsilon) \sup_{\Delta T, f} \left( \frac{g_{T_n}(X_n)}{((T + \Delta T)^\top g)_{T_n}(X_n)} \ell(f(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f) \right)
$$

(28)

and

$$
\frac{-g_{T_n}(X_n)}{((T + \Delta T)^\top g)_{T_n}(X_n)} \ell(f_2(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f_2) \geq (1 - \epsilon) \sup_{\Delta T, f} \left( \frac{-g_{T_n}(X_n)}{((T + \Delta T)^\top g)_{T_n}(X_n)} \ell(f(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f) \right).
$$

(29)
Thus, for any $\epsilon$, we have
\begin{align*}
(1 - \epsilon)\mathbb{E}_{\sigma_n} & \left[ \sup_{\Delta_T, f} \left( \sigma_n \frac{g_{\tilde{Y}_n}(X_n)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_n(X_n)} \ell(f(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f) \right) \right] \\
& = \frac{(1 - \epsilon)}{2} \sup_{\Delta_T, f} \left( \frac{g_{\tilde{Y}_n}(X_n)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_n(X_n)} \ell(f_1(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f_1) \right) \\
& \quad + \frac{(1 - \epsilon)}{2} \sup_{\Delta_T, f} \left( - \frac{g_{\tilde{Y}_n}(X_n)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_n(X_n)} \ell(f_2(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f_2) \right) \\
& \leq \frac{1}{2} \left( \frac{g_{\tilde{Y}_n}(X_n)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_n(X_n)} \ell(f_1(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f_1) \right) \\
& \quad + \frac{1}{2} \left( s_{n-1}(\Delta T, f_1) + s_{n-1}(\Delta T, f_2) + C|\ell(f_1(X_n), \tilde{Y}_n) - \ell(f_2(X_n), \tilde{Y}_n)| \right)
\end{align*}

(30)

where the last inequality holds because \( \frac{g_{\tilde{Y}_n}(X)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_n(X)} \leq C \) for any \((X, Y), g, \) and valid \( \hat{T} + \Delta T \). Let \( s = \text{sgn}(\ell(f_1(X_n), \tilde{Y}_n) - \ell(f_2(X_n), \tilde{Y}_n)) \). We have
\begin{align*}
(1 - \epsilon)\mathbb{E}_{\sigma_n} & \left[ \sup_{\Delta_T, f} \left( \sigma_n \frac{g_{\tilde{Y}_n}(X_n)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_n(X_n)} \ell(f(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f) \right) \right] \\
& \leq \frac{1}{2} \left( s_{n-1}(\Delta T, f_1) + s_{n-1}(\Delta T, f_2) + sC(\ell(f_1(X_n), \tilde{Y}_n) - \ell(f_2(X_n), \tilde{Y}_n)) \right) \\
& = \frac{1}{2} \left( s_{n-1}(\Delta T, f_1) + sC\ell(f_1(X_n), \tilde{Y}_n) \right) + \frac{1}{2} \left( s_{n-1}(\Delta T, f_2) - sC\ell(f_2(X_n), \tilde{Y}_n) \right) \quad (31)
\end{align*}

Since the above inequality holds for any \( \epsilon > 0 \), we have
\begin{align*}
\mathbb{E}_{\sigma_n} \left[ \sup_{\Delta_T, f} \left( \sigma_n \frac{g_{\tilde{Y}_n}(X_n)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_n(X_n)} \ell(f(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f) \right) \right] \\
& \leq \mathbb{E}_{\sigma_n} \left[ \sup_{\Delta_T, f} \left( \sigma_n \ell(f(X_n), \tilde{Y}_n) + s_{n-1}(\Delta T, f) \right) \right] \quad (32)
\end{align*}

Proceeding in the same way for all other \( \sigma \), we have
\begin{align*}
\mathbb{E}_\sigma & \left[ \sup_{\Delta_T, f} \sum_{i=1}^n \sigma_i \frac{g_{\tilde{Y}_i}(X_i)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_i(X_i)} \ell(f(X_i), \tilde{Y}_i) \right] \\
& \leq \mathbb{E}_\sigma \left[ \sup_{f \in F} \sum_{i=1}^n \sigma_i \ell(f(X_i), \tilde{Y}_i) \right] . \quad (33)
\end{align*}

and thus
\begin{align*}
\mathbb{E} & \left[ \sup_{\Delta_T, f} \sum_{i=1}^n \sigma_i \frac{g_{\tilde{Y}_i}(X_i)}{((\hat{T} + \Delta T)^\top g) \tilde{Y}_i(X_i)} \ell(f(X_i), \tilde{Y}_i) \right] \\
& \leq \mathbb{E} \left[ \sup_{f \in F} \sum_{i=1}^n \sigma_i \ell(f(X_i), \tilde{Y}_i) \right] . \quad (34)
\end{align*}

C.3 Proof of Lemma 3

Before proving Lemma 3, we show that the loss function \( \ell(f(X), \tilde{Y}) \) is 1-Lipschitz-continuous w.r.t. \( h_i(X), i = \{1, \ldots, C\} \).
According to Eqs. (36) and (37), it is easy to conclude that
\[ C \] is number of the class.

The definition of symmetry flipping transition matrix is as follows, where \( C \) is number of the class.

\[
\text{sym-} \epsilon: \ T = \begin{bmatrix}
1 - \epsilon & \epsilon & \cdots & \epsilon \\
\epsilon & 1 - \epsilon & \cdots & \epsilon \\
\cdots & \cdots & \cdots & \cdots \\
\epsilon & \cdots & \epsilon & 1 - \epsilon \\
\end{bmatrix}
\]
Figure 4: Comparing the estimation error of the transition matrix by employing classifier-consistent and risk-consistent estimators. The first row is about sym-20 label noise while the second row is about sym-50 label noise. The error bar for STD in each figure has been highlighted as a shade.

E More discussions about Figure 3

We represent Figure 3 in the main paper as Figure 1 in this appendix.

From the figure, we can compare the transition matrices learned by the proposed T-revision method and the traditional anchor point based method. Specifically, as shown in Figure 1, at epoch 0, the estimation error corresponds to the estimation error of transition matrix learned by identifying anchor points [38] (the traditional method to learn transition matrix). Note that the method with "N/A" in its name means it runs on the modified datasets where instances with large clean class posterior probabilities are removed (anchor points are removed); while the method with "A" in its name means it runs the original intact dataset (may contain anchor points). Clearly, we can see that the estimation error will increase by removing possible anchor points, meaning that anchor points is crucial in the traditional transition matrix learning. Moreover, as the number of epochs grows, the figures show how the estimation error varies by running the proposed revision methods. We can see that the proposed Reweight method always leads to smaller estimation errors, showing that the proposed method works well in find a better transition matrix.

Figure 1 also shows the comparison of learning transition matrices between the risk-consistent estimator based method and the classifier-consistent method based method. For classifier-consistent algorithms, we can also modify the transition matrix by adding a slack variable and learning it jointly with the classifier, e.g., Forward-A-R and Forward-N/A-R. However, we can find that the classifier-consistent algorithm based method Forward-N/A-R may fail in learning a good transition matrix, e.g., Figure 1(a). This is because there is no reason to learn the transition matrix by minimizing the classifier-consistent objective function. It is reasonable to learn the transition matrix by minimizing the risk-consistent estimator because a favorable transition matrix should make the classification risk w.r.t. clean data small. This is verified by comparing Forward-A-R and Forward-N/A-R with the proposed Reweight-A-R and Reweight-N/A-R, we can find that the risk-consistent estimator Reweight always leads to smaller estimation errors for learning transition matrix.