Beyond Images: Label Noise Transition Matrix Estimation for Tasks with Lower-Quality Features

Zhaowei Zhu 1, Jialu Wang 1, Yang Liu 1

Abstract
The label noise transition matrix, denoting the transition probabilities from clean labels to noisy labels, is crucial for designing statistically robust solutions. Existing estimators for noise transition matrices, e.g., using either anchor points or clusterability, focus on computer vision tasks that are relatively easier to obtain high-quality representations. We observe that tasks with lower-quality features fail to meet the anchor-point or clusterability condition, due to the coexistence of both uninformative and informative representations. To handle this issue, we propose a generic and practical information-theoretic approach to down-weight the less informative parts of the lower-quality features. This improvement is crucial to identifying and estimating the label noise transition matrix. The salient technical challenge is to compute the relevant information-theoretical metrics using only noisy labels instead of clean ones. We prove that the celebrated $f$-mutual information measure can often preserve the order when calculated using noisy labels. We then build our transition matrix estimator using this distilled version of features. The necessity and effectiveness of the proposed method are also demonstrated by evaluating the estimation error on a varied set of tabular data and text classification tasks with lower-quality features. Code is available at github.com/UCSC-REAL/BeyondImages.

1. Introduction
When a feature is not properly annotated, the returned noisy label may differ from the ground-truth one (Wei et al., 2022d). One popular and useful statistical information about noisy labels is the label noise transition matrix (Liu, 2022), which characterizes the transition probability from the ground-truth label to a particular noisy label. The noise transition matrix has been demonstrated to be crucial in various tasks, e.g., learning noise-tolerant classifiers (Natarajan et al., 2013), recovering unbiased estimates of fairness constraints (Lamy et al., 2019; Wang et al., 2021b), identifying mislabeled data (Northcutt et al., 2021b), and aggregating multiple labels in crowdsourcing (Liu & Liu, 2015). These applications often postulate the true knowledge of the noise transition matrix, which is generally unattainable in real-world data. Problems may arise when the noise transition matrix, acquired through an estimation process, incurs misspecification. It has been shown that the blind application of such a noise transition matrix may in fact lead to higher errors (Liu & Wang, 2021). In consequence, it is important to estimate the label noise transition matrix accurately.

Most existing approaches to estimating the label noise transition matrix presuppose high-quality features. For example, a series of research relies on finding the anchor points (Scott, 2015; Menon et al., 2015; Patrini et al., 2017; Liu & Tao, 2015; Xia et al., 2019; Yao et al., 2020; Zhang et al., 2021a; Yang et al., 2021), defined as the instances that belong to a true label class almost surely. Without high-quality features, instances belonging to different true label classes may not be well-separated in any hidden space such that anchor points rarely exist. Likewise, estimating the noise transition matrix with confident learning (Northcutt et al., 2021a) also requires high-quality features to estimate the confident points accurately.

Recent training-free estimator requires 2-Nearest-Neighbor (2-NN) clusterability (Zhu et al., 2021c) to estimate the noise transition matrix, where one feature and its 2-NN should have the same true label. This method inevitably depends on the quality of features since the likelihood that a data instance satisfies the clusterability condition will naturally decrease with lower-quality features.

Despite the above approaches achieving huge success on image benchmark datasets, such as CIFAR-10 (Krizhevsky et al., 2009), it remains questionable how these noise estimation approaches perform when it is hard to obtain high-
quality features. In particular, for some tabular data with categorical features like UCI datasets (Dua & Graff, 2017), popular feature learning or representation learning techniques seem unusable to improve the quality due to the sparseness of features. In more sophisticated natural language processing (NLP) tasks, such as text classification, some stop words (e.g., “a/an”, “the”, “is”, and “are”) are less informative but may be encoded in textual representations. For the learning-based methods, these less informative components may be overrated on noisy labels and harmful to finding anchor points or converging to the global optimum. For the training-free methods, these uninformative variables are likely to obscure the useful counterpart and result in mistakes in finding the nearest neighbors.

Unfortunately, we have observed consistent performance drops for the existing estimators, as shown in Figure 1. We evaluate several baselines (Xia et al., 2019; Northcutt et al., 2021a; Zhu et al., 2021c) and find the estimation errors of most approaches are around or larger than 0.1 for binary classifications with lower-quality features. As a rough comparison, the estimation errors on CIFAR-10 reported by these methods are around 0.05. Note the average noise rates of binary classification and a 10-class classification are within the range $[0, 0.5]$ and $[0, 0.9]$, respectively, showing the “same” error for binary classifications is effectively worse than a 10-class one, not to mention 0.1 for binary versus an even lower 0.05 for 10-class. Thus there is a non-negligible performance drop. We defer more detailed observations and discussions to Section 2.3.

We propose a generally practical information-theoretic approach to address the label noise matrix estimation for tasks with lower-quality features in response to the failures. Our approach builds on HOC (Zhu et al., 2021c), since we seek to avoid heavy task-specific hyperparameter fine-tuning and make it a generically applicable and light estimator. In particular, we divide the input features into several exclusively uncorrelated parts and down-weight the less informative parts by the $f$-mutual information between each part and the noisy labels. This operation allows us to improve the clusterability of features which proves to be a crucial property for identifying label noise transition matrix (Zhu et al., 2021c; Liu, 2022). Our main contributions are:

- Based on the $f$-mutual information, we propose a novel reweighting mechanism to firstly decode features by projecting to its eigenspace and then down-weight the less-informative parts with only access to noisy labels. The mechanism intuitively disentangles features but it does not require training and can be applied efficiently.
- The salient technical challenge is to compute $f$-mutual information using only noisy labels instead of clean ones. We prove that calculating the total-variation-based mutual information with noisy labels preserves the same order as using clean labels (Theorem 4.4), and the traditional mutual information preserves the order when the absolute gap between two noisy measurements is larger than a guaranteed threshold (Theorem 4.5).
- We empirically demonstrate that our approach helps return a more accurate estimate of the transition matrix and reduce the classification errors of downstream learning tasks on datasets with lower-quality or more sophisticated features, including UCI datasets with tabular data and text classification benchmarks.

1.1. Related Works

The noise transition matrix $T$ is important in several communities (Han et al., 2020). For example, it helps build noise-consistent classifiers when learning with noisy labels, such as loss correction (Natarajan et al., 2013; Patrini et al., 2017; Wei et al., 2022e), loss reweighting (Liu & Tao, 2015), and capturing the imbalance caused by heterogeneous noise rates (Zhu et al., 2021b). It also helps tune hyperparameters for label smoothing (Lukasik et al., 2020; Wei et al., 2021b), set thresholds for sample selection (Han et al., 2018; Wei et al., 2020; Zhang et al., 2021b) and detect label mistakes (Zhu et al., 2021a; Northcutt et al., 2021b). Additionally, $T$ contributes to evaluating (Awasthi et al., 2021) or improving (Lamy et al., 2019) the model fairness when the sensitive attribute is protected, or mitigating bias in treating different groups (e.g., racial, gender) when the label quality for different groups is different (Wang et al., 2021b). It also has medical applications such as evaluating the physician variability (McCormick et al., 2016). All of the above applications require an accurate estimate of $T$. 

In addition to the $T$ estimators introduced in Section 1, there are related works in crowdsourcing (Liu et al., 2012; Zhang et al., 2014; Liu & Liu, 2015) and peer prediction (Liu & Chen, 2017; Liu et al., 2020). However, these works require redundant noisy labels. For a general machine learning task, the datasets that have only one noisy label for each feature are more common. Compared with these approaches, the 2-NN clusterability of HOC (Zhu et al., 2021c) can be treated as a proxy of two redundant noisy labels, where a better proxy requires well-extracted features. According to the analyses on the identifiability of the label noise transition matrix (Liu, 2022), the disentangled and informative features are crucial. It has been demonstrated that mutual information helps select more informative features (Battiti, 1994; Estévez et al., 2009; Vergara & Estévez, 2014) with clean data. Recent work (Wei & Liu, 2020) finds that some $f$-mutual information metrics are robust to label noise, which makes the feature selection on noisy data promising.

2. Preliminaries

In this section, we first formally define the noise transition matrix $T$ and pose the problem in Section 2.1. As a comple-
ment to related works, we introduce more technical details of two lines of most relevant $T$ estimators in Section 2.2 and numerically show the possible failures of these approaches in Section 2.3.

2.1. The Definition of Noise Transition Matrix

**Clean/Noisy distribution** Consider a $K$-class classification task with a dataset $D := \{x_n, y_n\}_{n \in [N]}$, where $x_n$ is the feature, $y_n$ is the noisy label that may come from human annotations (Wei et al., 2022c; Luo et al., 2020), sensors (Wang et al., 2021a) and machine pseudo labels (Zhu et al., 2022). $N$ is the number of instances, $[N] := \{1, 2, \ldots, N\}$. Suppose $x$ is $d$-dimensional, i.e., $x = [x_1, \ldots, x_d]^\top$. We denote the $\mu$-th element by feature variable $x_\mu$. Note the feature vector $x$ is not necessary to be the raw input for some complicated tasks that require deep neural networks. In these tasks, the feature vector $x$ should be the output of some feature extractors (Devlin et al., 2019; Radford et al., 2021). We will not discuss methods to get appropriate feature extractors in this paper since the focus is on the data processing given $D$. The clean label associated with the noisy label $\tilde{y}$ is denoted by $y$. Both clean labels and noisy labels are in the same label space, i.e., $y \in [K], \tilde{y} \in [K]$ where $[K] := \{1, 2, \ldots, K\}$. The random variable forms of the above realizations of features and labels are: feature vector $x \sim X = [X_1, \ldots, X_d]^\top$, feature variable $x_\mu \sim X_\mu$, clean label $y \sim Y$, noisy label $\tilde{y} \sim \tilde{Y}$. The (unobservable) clean dataset is denoted by $D$.

**Noise transition matrix $T$** The relationship between $(X, Y)$ and $(X, \tilde{Y})$ is denoted by a noise transition matrix $T(X)$, where each element $T_{ij}(X) := \mathbb{P}(\tilde{Y} = j | Y = i, X)$ stands for the probability of mislabeling a clean label $Y = i$ as the noisy label $\tilde{Y} = j$ given feature $X$. A majority line of works assume the noise transition matrix is class-dependent (Natarajan et al., 2013; Liu & Tao, 2015; Patrini et al., 2017; Liu & Guo, 2020), i.e., $T(X) \equiv T$, implying the following independency:

$$\mathbb{P}(\tilde{Y} = j | Y = i, X) = \mathbb{P}(\tilde{Y} = j | Y = i), \forall i, j \in [K], \forall X.$$

**Problem Statement** We formally frame the problem of label noise transition matrix estimation. The learner can only access noisy examples $\tilde{D}$ in this setting. The noisy label $\tilde{Y}$ satisfies an underlying transition probability characterized by $T$. The goal is to minimize the estimation error calculated by the average total variation of the true $T$ and the estimated $\hat{T}$ (Zhang et al., 2021a):

$$\text{Error}(T, \hat{T}) = \sum_{i \in [K], j \in [K]} |T_{ij} - \hat{T}_{ij}| / (2K).$$

2.2. Existing Estimation Approaches

There are mainly two categories of prior works on estimating the noise transition matrix $T$. One popular solution pipeline is to estimate $T$ with the confidence/predictions of the model trained on the noisy data distribution. As an alternative to the learning-based methods, one recent work (Zhu et al., 2021c) proposes a training-free pipeline when clusterable features are given. We introduce more details as follows.

**Definition 2.1** (Anchor Point (Liu & Tao, 2015; Scott, 2015; Xia et al., 2019)). An instance $x$ is an anchor point for class $i$ if $\mathbb{P}(Y = i | X = x) = 1$. 

**Using model predictions** In the learning-based methods, the anchor point, which is the instance that belongs to a particular class almost surely as defined in Definition 2.1, plays a significant role. Particularly, if $\mathbb{P}(Y = i | X = x) = 1$ holds for some class $i$, we have $\mathbb{P}(\tilde{Y} = j | X = x) = \sum_{k \in [K]} T_{kj} \mathbb{P}(Y = k | X = x) = T_{ij}$. Note the anchor point is defined on the clean data. If the features $X$ are of lower quality, the condition $\mathbb{P}(Y = i | X = x) = 1$ is hard to satisfy (Zhu et al., 2021a), not to mention finding anchor points accurately. Although recent advances relax the requirement of anchor points (Xia et al., 2019; Zhang et al., 2021a; Li et al., 2021), they tend to design a specific learning pipeline with a particular regularizer or objective related to $T$, which is sensitive to hyperparameters and ultimately the quality of features.

**Using clusterability** Recent work HOC (Zhu et al., 2021c) studies this problem from a novel data-centric perspective, which proposes a statistical solution based on clusterability without fitting the data distribution. The main idea is to utilize a set of High-Order Consensus (HOC) information aggregated on the nearest neighbors’ noisy labels and solve a series of equations for the noise transition matrix $T$ and clean label prior probability $p$. The effectiveness of this approach relies on the $k$-NN ($k$-Nearest-Neighbor) label clusterability, which is defined as:

**Definition 2.2** ($k$-NN label clusterability (Zhu et al., 2021c)). A dataset $D$ satisfies $k$-NN label clusterability if $\forall n \in [N]$, the feature $x_n$ and its $k$-Nearest-Neighbor $x_{n_1}, \ldots, x_{n_k}$ belong to the same true label class.

The distance between two features $x$ and $x'$ can be measured by $1 - \text{Sim}(x, x')$, where $\text{Sim}(x, x')$ could be the cosine similarity. It has been proved by (Zhu et al., 2021c) that the $2$-NN label clusterability is sufficient for uniquely getting the true $T$. However, it is likely that the clusterability is not sufficiently satisfied for lower-quality features.

2.3. Failures on Lower-Quality Features

The above approaches to estimating $T$ are demonstrated to perform well on some image classification datasets, such as MNIST (LeCun et al., 1998) and CIFAR (Krizhevsky et al., 2009), which usually enjoy better representation learning tools (Chen et al., 2020; Wang et al., 2022a) that would return clusterable features, as compared to text sequence
and tabular data. When facing other tasks, high-quality features may not always be available. In this subsection, we explore how these existing methods fare on other datasets, possibly with lower-quality features.

**Observations** In Figure 1, we implement two learning-based methods built on anchor points (T-revision (Xia et al., 2019)) or confident points (Confident Learning (Northcutt et al., 2021a)), and one training-free method based on the clusterability (HOC (Zhu et al., 2021c)). The average noise rate, i.e., $\sum_{i \in [K]} (1 - T_{ni}) / K$, is around 0.3. Figure 1 shows the estimation error of these three methods on most of the datasets are around or larger than 0.1, and some methods may even approximate to 0.2. Note an error of 0.2 is excessive for binary classification when the noise rate is 0.3. For example, when $T_{11} = T_{22} = 0.3$, a random guess of $T$ has an error of exactly 0.2. Recall an error of 0.05 for binary classification is worse than the same error for a 10-class one as explained in Introduction. Compared with an estimation error of $\approx 0.05$ on CIFAR-10 with a similar noise rate as reported in these baselines, Figure 1 shows a serious performance drop: an error of 0.05 for 83% of tests and an error of 0.1 for 57% of tests. Therefore, it is crucial to design an estimator which is also robust on datasets with lower-quality features.

**Discussions** Before diving into a concrete solution to lower-quality features, we discuss the advantages and disadvantages of both existing lines of work. On one hand, the learning-based methods (Northcutt et al., 2021a; Xia et al., 2019) could take full advantage of deep neural networks. During supervised training, different parts of features are weighted differently, thus the informative parts could weigh more than the less informative ones. The optimal weight combinations could induce a model that accurately fits the data distribution, which further help estimate $T$. On the other hand, due to various factors such as the model capacity, the quality of features, the number of instances, and the setting of hyperparameters, the learning-based models are often infeasible to converge to the global optimum in practice. For example, with the existence of label noise, deep neural networks (DNN) tend to be overconfident (Wei et al., 2022b) and memorize wrong feature-label patterns (Cheng et al., 2021b; Liu, 2021; Wei et al., 2021a). When unintended memorization occurs, the weights for combining different parts will be non-optimal and some uninformative parts may be mis-specified with high weights. Alternatively, the training-free method (Zhu et al., 2021c) will not be affected by the wrong memorization since it is a fully-statistical solution without any training procedures. Nevertheless, the problem of not employing training is also severe: blindly treating different parts of features equally important may cause failures. The above observations and discussions motivate us to find a solution that compromises between the learning-based and the training-free approaches.

**3. An Information-Theoretic Approach**

We propose an information-theoretic approach to distinguish the importance of different features. To avoid complicated hyperparameters tuning and make it a light tool for more general applications, the solution is built on HOC. See more detailed rationale in Appendix B.1.

We now briefly introduce HOC (Zhu et al., 2021c). Algorithm 1 summarizes the key steps, where the high-level idea is that, when 2-NN label clusterability holds, the frequency of consensus patterns of the three grouped noisy labels $\hat{y}_n, \hat{y}_{n1}, \hat{y}_{n2}$ encodes $T$. For instance, a triplet $\hat{y}_n = 0, \hat{y}_{n1} = 0, \hat{y}_{n2} = 1$ will add 1 to the count of the consensus pattern $(0, 0, 1)$. With the estimated frequency of patterns $\{ (\hat{y}_n, \hat{y}_{n1}, \hat{y}_{n2}), \forall n \}$, we can simply solve equations by gradient descents. Therefore, in Algorithm 1, the 2-NN label clusterability is critical, which depends heavily on calculating the distance or similarity between features.

**Overview of our approach:** The main idea is to decouple features (Step 1) and down-weight the less informative parts (Step 2) when measuring the distances by HOC, which is summarized in Algorithm 2. Firstly, we motivate the

<table>
<thead>
<tr>
<th>Algorithm 1 Key Steps of HOC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Noisy dataset: $D = {(x_n, y_n)}_{n \in [N]}$.</td>
</tr>
<tr>
<td><strong>Find 2-NN:</strong> $\text{Sim}(x, x') \rightarrow \text{Sim}_\text{2NN}(z, z')$ in our approach.</td>
</tr>
<tr>
<td>1: With $1 - \text{Sim}(x, x')$ as the distance metric:</td>
</tr>
<tr>
<td>$(\hat{y}<em>n, \hat{y}</em>{n1}, \hat{y}_{n2}), \forall n \leftarrow \text{Get2NN}(D)$;</td>
</tr>
<tr>
<td><strong>Count first-, second-, and third-order consensus patterns:</strong></td>
</tr>
<tr>
<td>$(c^{[1]}, c^{[2]}, c^{[3]}) \leftarrow \text{CountFreq}({(\hat{y}<em>n, \hat{y}</em>{n1}, \hat{y}_{n2}), \forall n})$</td>
</tr>
<tr>
<td><strong>Solve equations:</strong></td>
</tr>
<tr>
<td>3: Find $T$ such that match the counts $(c^{[1]}, c^{[2]}, c^{[3]})$.</td>
</tr>
</tbody>
</table>

---

**Figure 1.** Existing methods may suffer from failures. **Red** horizontal dashed line shows the error of random guessing $T$ in binary classifications. Tasks on the left side of the dotted line are binary.
Algorithm 2 Our Information-Theoretic Approach

1. **Input:** Noisy dataset: $D = \{(x_n, y_n)\}_{n \in [N]}$.

2. **Step 1:** Remove correlation (Section 3.3).

3. **Step 2:** Estimate the weight matrix $W$ (Section 3.4).

4. **Output:** The estimated noise transition matrix $\hat{T}$.

To capture more information, we incorporate a different from $W$ matrix. In practice, the true and unknown $\hat{W}$ may cause the off-diagonal values of $W$ to capture the off-diagonal values of $W$ (the correlations between $x_\mu$) and violates the clusterability.

Consider the following three $W$'s: $W_1 = I,$

$W_2 = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.1 \end{pmatrix}, \quad W_3 = \begin{pmatrix} 1.0 & -0.2 & -0.5 \\ -0.5 & 1.0 & 0.5 \\ -0.5 & 0.5 & 1.0 \end{pmatrix}.$

**Example 1:** $W_1$ (uncorrelated and equally important)

The following hard cosine similarity shows: $\text{Sim}_{W_1}(x_1, x_3) \approx 0.73 > \text{Sim}_{W_1}(x_2, x_3) \approx 0.69 \Rightarrow y = 1.$

**Example 2:** $W_2$ (not equally important)

The following soft cosine similarity shows: $\text{Sim}_{W_2}(x_1, x_3) \approx 0.64 < \text{Sim}_{W_2}(x_2, x_3) \approx 0.77 \Rightarrow y = 2,$

which is different from the inferred $y$ in Example 1. If $W_2$ defines the true clusterability, this example shows that simply using $W = I$ fails to capture the diagonal values of $W$ (the weights of $x_\mu, \mu \in [d]$) and violates the clusterability.

**Example 3:** $W_3$ (correlated)

The following soft cosine similarity shows: $\text{Sim}_{W_3}(x_1, x_3) \approx 0.75 < \text{Sim}_{W_3}(x_2, x_3) \approx 0.85 \Rightarrow y = 2,$

which is different from the inferred $y$ in Example 1. If $W_3$ is true, this example shows that simply using $W = I$ fails to capture the off-diagonal values of $W$ (the correlations between $x_\mu$) and violates the clusterability.

The above three examples exemplify that the less informative parts of features may damage the clusterability, where the definition of each “part” depends on both the diagonal elements and off-diagonal elements of $W$. We propose an information-theoretic approach to construct a proxy of $W$.

3.2. Proxy of $W$

Noting $W$ is a square matrix of order $d$, it requires $O(d^2)$ operations to estimate all elements. Each operation incurs an estimation error, and the accumulated errors may not be bounded. We propose to find a proxy of $W$. Intuitively, we expect the following properties (Battiti, 1994):

- **Symmetric:** $\forall \mu, \nu \in [d], W_{\mu\nu} = W_{\nu\mu}$.
- **Information monotone:** For every two feature variables $X_\mu, X_\nu$, if $X_\mu$ is less informative with respect to $Y$ than $X_\nu$, then $X_\mu$ will be less important than $X_\nu$, i.e., $I_f(X_\mu; Y) \leq I_f(X_\nu; Y) \Rightarrow W_{\mu\nu} \leq W_{\nu\mu}, \mu, \nu \in [d],$

where $I_f(X_\mu; Y)$ measures the $f$-MI ($f$-Mutual Information) (Csiszár, 1967) between $X_\mu$ and $Y$.
- **Correlation monotone:** Given two feature variables $X_\mu, X_\nu$, for any other feature variable $X_\nu'$ ($X_\mu, X_\nu, X_\nu'$ are equally informative), if $X_\mu$ is less correlated to $X_\nu$ than $X_\nu'$, $X_\nu$ will have a lower weight than $X_\nu'$ in mea-

---

3.1. Vanilla Similarity Measures Are Not Sufficient

Our following analyses focus on the cosine similarity as originally implemented by HOC (Zhu et al., 2021c). The larger cosine similarity implies the smaller distances of features. We first analyze possible problems in evaluating $k$-NN with the vanilla (hard) cosine similarity.

Consider the cosine similarity of two feature vectors $x$ and $x'$, which is denoted by $\text{Sim}(x, x') = \frac{x^T x'}{||x||_2 ||x'||_2}$, where $|| \cdot ||_2$ denotes the vector $\ell_2$ norm. The above measure inherently assumes different elements of $x$ are 1) equally important and 2) uncorrelated to each other, thus may underestimate the true similarity between imperfect feature vectors. To capture more information, we incorporate a change-of-basis matrix $\sqrt{W}$ to obtain a soft cosine measure defined as follows.

**Definition 3.1** (Soft cosine similarity (Sidorov et al., 2014)),

$$\text{Sim}_W(x, x') = \frac{(\sqrt{W x})^T (\sqrt{W x'})}{||\sqrt{W x}||_2 ||\sqrt{W x'}||_2}.$$  \hspace{1cm} (2)

Hereby, the symmetric matrix $W$ encodes the pairwise similarity between features. Note that the soft cosine similarity measure $\text{Sim}_W(x, x')$ recovers the (hard) cosine similarity when $W = I$, where $I$ denotes a $K \times K$ identity matrix. In practice, the true and unknown $W$ may be different from $I$. Thus simply letting $W = I$ may cause severe problems in using clusterability. For example, when $K = 2$, consider three instances $(x_1, y_1) = ([1, 0, 1]^T, 1)$, $(x_2, y_2) = ([0, 1, 0]^T, 2)$, and $(x_3, y_3) = ([0.8, 1, 0.7]^T, y)$. Based on 1-NN label clusterability, we infer label $y$ following the rule below:

$$y = \begin{cases} 1 & \text{if } \text{Sim}(x_1, x_3) > \text{Sim}(x_2, x_3); \\ 2 & \text{if } \text{Sim}(x_1, x_3) \leq \text{Sim}(x_2, x_3). \end{cases}$$

Consider the following three $W$'s: $W_1 = I,$

$W_2 = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.1 \end{pmatrix}, \quad W_3 = \begin{pmatrix} 1.0 & -0.2 & -0.5 \\ -0.5 & 1.0 & 0.5 \\ -0.5 & 0.5 & 1.0 \end{pmatrix}.$

Example 1: $W_1$ (uncorrelated and equally important)

The following hard cosine similarity shows: $\text{Sim}_{W_1}(x_1, x_3) \approx 0.73 > \text{Sim}_{W_1}(x_2, x_3) \approx 0.69 \Rightarrow y = 1.$

Example 2: $W_2$ (not equally important)

The following soft cosine similarity shows: $\text{Sim}_{W_2}(x_1, x_3) \approx 0.64 < \text{Sim}_{W_2}(x_2, x_3) \approx 0.77 \Rightarrow y = 2,$

which is different from the inferred $y$ in Example 1. If $W_2$ defines the true clusterability, this example shows that simply using $W = I$ fails to capture the diagonal values of $W$ (the weights of $x_\mu, \mu \in [d]$) and violates the clusterability.

Example 3: $W_3$ (correlated)

The following soft cosine similarity shows: $\text{Sim}_{W_3}(x_1, x_3) \approx 0.75 < \text{Sim}_{W_3}(x_2, x_3) \approx 0.85 \Rightarrow y = 2,$

which is different from the inferred $y$ in Example 1. If $W_3$ is true, this example shows that simply using $W = I$ fails to capture the off-diagonal values of $W$ (the correlations between $x_\mu$) and violates the clusterability.

The above three examples exemplify that the less informative parts of features may damage the clusterability, where the definition of each “part” depends on both the diagonal elements and off-diagonal elements of $W$. We propose an information-theoretic approach to construct a proxy of $W$. 


---

Label Noise Transition Matrix Estimation for Tasks with Lower-Quality Features
where \( X \) we can transform \( \Lambda_0 = \rho \) split \( X \) into diagonal values of \( \rho \) being non-correlated form to achieve \( \rho \) estimate only the diagonal elements by non-correlated form to achieve \( \rho \) might not be computation-efficient. If \( \rho \) becomes uncorrelated parts such that \( E \). Inspired by the principle component analysis (PCA) (Wold et al., 1987), we adopt \( \Lambda^{-1/2} P^T \) as the matrix to remove the correlation between \( x_\mu, x_\nu, \mu \neq \nu \), \( P \) is an orthogonal matrix whose columns are eigenvectors of \( E[X X^T] \), \( \Lambda \) is a diagonal matrix where diagonal values are eigenvalues and off-diagonal values are 0. Let \( Z = \Lambda^{-1/2} P^T X = [Z_1, \ldots, Z_d]^T \), (3) where \( E[Z] = \Lambda^{-1/2} P^T E[X] = 0 \). We have \( E[ZZ^T] = \Lambda^{-1/2} P^T E[X X^T] P \Lambda^{-1/2} = I \). Therefore, after transforming \( X \) to \( Z \) by \( \Lambda^{-1/2} P^T \), we can split \( X \) into \( d \) uncorrelated parts such that \( E[Z_\mu Z_\nu] \) satisfies

\[
E[Z_\mu Z_\nu] = E[(ZZ^T)_{\mu,\nu}] = \begin{cases} 0 & \text{if } \mu \neq \nu, \\ 1 & \text{if } \mu = \nu. \end{cases}
\]

Noting (zero-mean) \( \rho(Z_\mu, Z_\nu) = \frac{E[Z_\mu Z_\nu]}{\sqrt{E[Z_\mu^2]} \sqrt{E[Z_\nu^2]}} \) we know \( \rho = I \) after this transformation.

3.4. Estimate \( I_f(Z;Y) \) With Only Noisy Labels

After transforming \( X \) to \( Z \), the off-diagonal elements of the proxy of \( W \) are expected to be 0. Thus we only need to estimate the diagonal elements of \( W_{\mu\mu} = I_f(Z_\mu;Y) \). Define an \( f \)-MI metric with \( f \)-MI divergence \( I_f(Z;Y) := D_f(Z \otimes Y; Z \otimes Y) \), where \( D_f(P||Q) \) measures the \( f \)-divergence between two distributions \( P \) and \( Q \) with probability density function \( p \) and \( q \): 

\[
D_f(P||Q) = \int_{\mathcal{V}} q(v) f \left( \frac{p(v)}{q(v)} \right) dv.
\]

Note the variable \( v \) in the domain \( \mathcal{V} \) is a realization of random variable \( V = (Z, Y) \), which follows either distribution \( P \) or distribution \( Q \) depending on where it is used. In our setting, \( P \) and \( Q \) are the joint distribution \( P := Z \oplus Y = \mathbb{P}(Z = z, Y = y) \) and the marginal product distribution \( Q := Z \otimes Y = \mathbb{P}(Z = z) \cdot \mathbb{P}(Y = y) \), respectively. There are lots of choices of \( f \). Specially, when \( f(v) = v \log v \), the \( f \)-divergence becomes the celebrated KL divergence and \( I_f \) is exactly the mutual information.

Alternatively, we can calculate the \( f \)-divergence using the variational form (Nowozin et al., 2016; Wei & Liu, 2020). Denote the variational difference between \( P \) and \( Q \) by

\[
VD_f(g) = E_{V \sim P}[g(V)] - E_{V \sim Q}[f^*(g(V))], \forall g,
\]

where \( g : \mathcal{V} \rightarrow \text{domain}(f^*) \) is a variational function, and \( f^* \) is the conjugate function of \( f(\cdot) \). We instantiate some prominent variational-conjugate \((g^*, f^*)\) pairs in Appendix A.1 (Table 4). The \( f \)-MI is calculated by

\[
I_f(Z;Y) = D_f(P||Q) = VD_f(g^*) = \sup_g VD_f(g).
\]

However, the above calculation, built on the clean distributions, will be intractable when we can only access the noisy data distribution. Let \( \bar{P} := Z \oplus \bar{Y} \) and \( \bar{Q} := Z \otimes \bar{Y} \). One tractable approach is to calculate \( D_f(\bar{P}||\bar{Q}) \) following

\[
I_f(Z;\bar{Y}) = D_f(\bar{P}||\bar{Q}) = \bar{V}D_f(g^*) = \sup_g \bar{V}D_f(g), \quad (4)
\]

where \( \bar{V}D_f(g) = E_{V \sim \bar{P}}[g(\bar{V})] - E_{V \sim \bar{Q}}[f^*(g(\bar{V}))], \forall g \). Generally, there will be a gap between our calculated \( I_f(Z;\bar{Y}) \) and the real \( I_f(Z;Y) \). We defer detailed analyses of the gap to Section 4.

4. Theoretical Guarantees

The quality of our \( T \) estimator relies on the following steps:

(a) Noise-resistant estimates of \( f \)-MI using noisy labels;
(b) Accurate estimates of clean \( f \)-MI;
(c) Down-weighting less informative features with \( f \)-MI;
(d) Robust distance/similarity calculation;
(e) Satisfying clusterability (Definition 2.2);
(f) Accurate estimates of the noise transition matrix $T$.

The most critical step in the above chain is (a)$\rightarrow$(b), which is the key ingredient to Step (c). This step will be the focus of the theoretical results in this section. Steps (c)$\rightarrow$(e) are explained in Section 3. Steps (e)$\rightarrow$(f) is guaranteed by HOC (Zhu et al., 2021c).

Based on our intuition for constructing the proxy of $W$ that the less informative parts should be assigned with lower weights, the order between two parts with different informativeness is crucial. Thus in this section, we study whether the noisy $f$-MI calculated using noisy labels, i.e., $I_f(Z;\tilde{Y})$, preserves the order of the clean $f$-MI $I_f(Z;Y)$. Note the order-preservation property distinguishes from the robustness of $f$-MI between optimal classifier prediction $h^*(X)$ and noisy label $\tilde{Y}$ by Wei & Liu (2020) since 1) $Z$ does not have class-specific meaning; 2) $Z$ is not optimizable and its ranking is concerned, while $h^*(X)$ is only a special (optimal) case of $Z$. All proofs are deferred to Appendix A. We define $\epsilon$-order-preserving as follows.

**Definition 4.1** ($\epsilon$-Order-Preserving Under Label Noise). $I_f(Z;\tilde{Y})$ is called $\epsilon$-order-preserving under label noise if

$$
\forall \mu \in [d], \nu \in [d], \text{given } |I_f(Z_\mu;\tilde{Y}) - I_f(Z_\nu;\tilde{Y})| > \epsilon,
$$

we have

$$
\text{sign}[I_f(Z_\mu;\tilde{Y}) - I_f(Z_\nu;\tilde{Y})] = \text{sign}[I_f(Z_\mu;Y) - I_f(Z_\nu;Y)].
$$

The smaller $\epsilon$ is, the stricter the requirement is. The following analyses focus on the binary classification. Define $e_1 := \mathbb{P}(Y = 2|Y = 1)$, $e_2 := \mathbb{P}(Y = 1|Y = 2)$. To show an $f$-MI metric is $\epsilon$-order-preserving under label noise, we need to study how $\Delta VD_f(\hat{g}^*)$ differs from the order of $VD_f(g^*)$.

### 4.1. Total-Variation is $0$-Order-Preserving

When $f(v) = \frac{1}{2}|v - 1|$, we get the Total-Variation (TV). To analyze the order-preserving property of TV, we first build the relationship between $VD_{TV}(g)$ and $VD_f(g)$, $\forall g$ by the following lemma:

**Lemma 4.2** (Linear relationship (Wei & Liu, 2020)).

$$
\Delta VD_{TV}(g) = (1 - e_1 - e_2) \Delta VD_f(g), \forall g.
$$

Lemma 4.2 shows that there is a linear relationship between $VD_{TV}(g)$ and $VD_f(g)$. The constant only depends on the noise rates. With this lemma, we only need to study the difference between $VD_{TV}(\hat{g}^*)$ and $VD_f(g^*)$, which is summarized in the following lemma:

**Lemma 4.3.** When $e_1 + e_2 < 1$, $VD_f(g^*) = \Delta VD_{TV}(\hat{g}^*)$.

Note the condition $e_1 + e_2 < 1$ indicates the label noise is not too large to be dominant (Liu & Guo, 2020; Natarajan et al., 2013; Liu & Chen, 2017). With Lemma 4.2 and Lemma 4.3, we can conclude that TV is 0-order-preserving.

**Theorem 4.4.** When $e_1 + e_2 < 1$, total-variation is 0-order-preserving under class-dependent label noise.

Theorem 4.4 shows the total-variation-based mutual information under class-dependent label noise preserves the order of the original clean results.

### 4.2. KL Divergence is $\epsilon$-Order-Preserving

Unfortunately, Lemma 4.2 and Lemma 4.3 do not hold for KL divergence. Recall the $f$-MI is exactly the standard mutual information when KL divergence is adopted. Denote by $H(\cdot)$ the entropy. We can start from the definition of mutual information $H(Z, \tilde{Y}) = H(Z) + H(Y) - H(Z, Y)$ and decouple the effect of noisy labels as:

$$
I(Z, \tilde{Y}) = (1 - e_1 - e_2) \cdot [H(Z) - H(Y)] + H(\tilde{Y}) + \int z \Delta_{\text{bias}}(\beta_z, e_1, e_2) dz,
$$

where $\beta_z$ is a function of $z$, and $\Delta_{\text{bias}}(\beta_z, e_1, e_2)$ is the bias caused by label noise specified in Eqn. (7). We then find the lower and upper bounds for $\Delta_{\text{bias}}(\beta_z, e_1, e_2)$ and summarize the result as follows:

**Theorem 4.5.** Assume $e_1 = \delta e_2$, $\delta \in [0, 1]$ and $e_1 + e_2 < 1$. KL divergence (mutual information) is $\epsilon$-order-preserving under class-dependent label noise, where

$$
\epsilon = e_1 \log(1 + \delta) + H(e_1),
$$

and $H(e_1) := -e_1 \log(1 - e_1) + (1 - e_1) \log(1 - e_1)$.

For the symmetric label noise, we have:

**Corollary 4.6.** When $e_1 = e_2 < 0.5$, KL divergence (mutual information) is $[H(e_1) - 2e_1 \log 2]$-order-preserving under class-dependent label noise.

We evaluate the bound in the following remark.
Remark 4.7. The value of $\epsilon$ is illustrated in Figure 3, where the left figure shows the worst-case bound in Theorem 4.5, and the right figure shows the case when $\Pr(Y = 1 | Z = 2) \in \left[ \frac{1}{2}, 5 \right]$. This is a more practical case for lower-quality features since a single feature variable cannot infer the clean label with high confidence. Noting the $\log_2$-based mutual information is within range $[0, 1]$ and $\epsilon \leq 0.34$ happens in 90% of the cases, it is reasonable to believe the mutual information has a good $\epsilon$-order-preserving ability.

5. Evaluations

We evaluate our approaches on datasets with possibly lower-quality features in this section. The evaluation metric is the average total variation between the true $T$ and the estimated $\hat{T}$ (Zhang et al., 2021a) as Eqn. (1).

Baselines We mainly compare our methods with three baselines: T-revision (T-REV) (Xia et al., 2019), Confident Learning (CL) (Northcutt et al., 2021a), and the clusterability-based approach (Zhu et al., 2021c) (HOC). Some recent learning-based methods (Li et al., 2021; Zhang et al., 2021a) jointly optimize $T$ and the model during learning. We find their methods are unstable compared to the above baseline approaches on non-image datasets and defer their results to Appendix B.2.

Our approach To evaluate each component of our approach, we test four variants: OURS-X-KL, OURS-X-TV, OURS-A-KL, and OURS-A-TV. Prefix OURS-X indicates that we directly use the original input features and substitute the soft cosine similarity for the original hard one employed by HOC. This setting checks the performance of ignoring correlations among different feature variables. Prefix OURS-A indicates that we firstly transform $X$ to $A$ as Section 3.2 then apply soft cosine similarity. Suffixes -KL and -TV denote using the $f$-mutual information when $f(v) = v \log(v)$ and $f(v) = \frac{1}{2}v - 1$, respectively. The matrix $W$ for soft cosine similarity is: $W_{\mu \nu} = \phi(\bar{I}_f(X_\mu, Y_\nu))$, $W_{\mu \nu} = 0, \forall \nu \neq \mu$, where $\phi(x)$ is an order-preserving activation function. In our evaluations, we set $\phi(x) = \left[x\right]_{0}^{1}$ for tabular benchmarks and $\phi(x) = \left[\log(x)\right]_{0}^{1}$ for natural language benchmarks, where $\left[x\right]_{0}^{1}$ represents normalizing $x$ to range $(0, 1)$.

Datasets and models We examine our approaches on two different application domains other than images: the tabular benchmarks including 7 tabular datasets from the UCI machine learning repository (Dua & Graff, 2017) and 4 natural language benchmarks including AG’s news (Zhang et al., 2015), DBpedia (Auer et al., 2007), Yelp-5 (Yelp, 2015), and Jigsaw (Jigsaw, 2018). We use the raw features for tabular benchmarks. Following the same preprocessing procedure as Wang et al. (2022b), we use a pre-trained BERT model (Devlin et al., 2019) to extract 768 dimensional embedding vectors for natural language benchmarks.

Noise type We synthesize the noisy data distribution by injecting class-dependent noise. Particularly, on tabular benchmarks, we test both the symmetric and the asymmetric label noise in Table 1. On natural language benchmarks, we randomly generate the diagonal-dominant label noise following the Dirichlet distribution. Particularly, suppose the average noise rate is $e$. For each row of $T$, we randomly sample a diagonal element following $T_{ii} = e + \text{Unif}(-0.05, 0.05)$ and set the off-diagonal elements following $\text{Dir}(1)$, where $\text{Unif}(a, b)$ denotes the uniform distribution bounded by $(a, b)$, $\text{Dir}(1)$ denotes the Dirichlet distribution with parameter $1 := [1, \cdots, 1]$ ($K - 1$ values).

5.1. Evaluation on Tabular Benchmarks

Table 1 shows the performance comparisons on tabular datasets. By counting the number of bold (top-2 performance) results, we know all the four variants of our proposed method perform better than three baselines statistically. Additionally, based on the counting results, OURS-X wins in 16 settings while OURS-A wins in 20 settings, indicating decoupling different parts by eigen decomposition is not statistically significantly useful. This may be due to the property of tabular data: the original correlation in $X$ may not be strong and the decoupling operations will involve extra errors and make the results even worse.

5.2. Evaluation on Natural Language Benchmarks

We also evaluate our methods on more sophisticated text classifications tasks. We use a heuristic method to set the average noise rate $e$. The aim is to make the ratio between diagonal elements and off-diagonal elements of $T$ consistent. According to our rule in the caption of Table 2, the low, medium, and high noise rates for binary, 5-class, and 10-class classifications are $(0.11, 0.2, 0.4)$, $(0.2, 0.33, 0.57)$, and $(0.27, 0.43, 0.67)$, respectively, which align with the general cognition of the community (Cheng et al., 2021a). Comparing Table 2 with Table 1, we find the direct reweighting by $f$-mutual information (OURS-X) performs similar to the hard cosine similarity adopted by HOC, while the information-theoretic reweighting after projecting $X$ to its eigen space (OURS-A) performs consistently better than other methods. Intuitively, the correlations of BERT embeddings are more stronger than those of tabular data. Thus we observe a clear performance improvement by removing correlations. Besides, it is interesting to see the estimation error may decrease for higher noise rate setting. We conjecture the reasons may comprise: 1) The original dataset may be noisy, e.g., the original Yelp dataset contains lots of noisy reviews (Luca, 2016). Consider a binary classification with inherent 20% noise. Then adding 10% (low) noise will make the average noise rate to 0.26, where the gap between the real noise rate and the hypothesized noise rate is 0.26 − 0.1 = 0.16. Similarly, the gap of adding
Table 1. The estimation error (×100) on tabular benchmarks. Feature dimension: \(d\). Number of clean instances in class-1 (or class-2): \(N_1\) (or \(N_2\)). Noise Rate: LOW: \(e_1 = 0.2, e_2 = 0.1\). MEDIUM: \(e_1 = 0.4, e_2 = 0.2\). HIGH: \(e_1 = 0.4, e_2 = 0.4\). Top-2 of each row are bold.

| TABULAR DATASETS \((d,\{N_1, N_2\})\) | NOISE RATE | T-REV | CL | HOCE | OURS-X-KL | OURS-X-TV | OURS-A-KL | OURS-A-TV |
|---|---|---|---|---|---|---|---|
| Heart | LOW | 9.25 | 8.00 | 9.82 | 8.09 | 8.70 | 8.88 | 9.18 |
| | MEDIUM | 11.94 | 11.48 | 7.85 | 9.55 | 1.48 | 3.98 | 5.51 |
| | HIGH | 6.74 | 6.54 | 4.71 | 9.78 | 14.91 | 1.33 | 4.21 |
| Banana | LOW | 30.89 | 30.53 | 14.96 | 10.74 | 11.49 | 10.57 | 9.26 |
| | MEDIUM | 20.77 | 20.81 | 8.97 | 4.90 | 3.98 | 6.41 | 6.97 |
| | HIGH | 6.71 | 7.58 | 4.78 | 12.11 | 8.89 | 9.78 | 11.26 |
| Titanic | LOW | 21.40 | 20.62 | 11.24 | 10.83 | 9.96 | 11.95 | 11.60 |
| | MEDIUM | 10.83 | 10.31 | 9.97 | 9.82 | 9.65 | 9.61 | 9.75 |
| | HIGH | 6.93 | 6.75 | 1.94 | 1.97 | 1.97 | 1.89 | 1.92 |
| Splice | LOW | 10.63 | 9.32 | 7.32 | 2.29 | 1.87 | 3.00 | 3.65 |
| | MEDIUM | 10.35 | 9.84 | 5.38 | 2.21 | 3.90 | 1.26 | 2.15 |
| | HIGH | 7.86 | 7.74 | 17.43 | 4.26 | 2.94 | 4.41 | 5.81 |
| Twonorm | LOW | 1.79 | 1.63 | 2.12 | 2.34 | 2.80 | 0.54 | 0.65 |
| | MEDIUM | 1.86 | 1.42 | 1.38 | 1.42 | 1.30 | 2.14 | 1.73 |
| | HIGH | 1.67 | 1.22 | 5.18 | 5.88 | 3.47 | 1.46 | 2.12 |
| Waveform | LOW | 10.59 | 10.89 | 7.93 | 6.68 | 6.78 | 5.67 | 5.79 |
| | MEDIUM | 8.45 | 7.82 | 3.51 | 2.50 | 3.09 | 2.34 | 2.72 |
| | HIGH | 5.04 | 4.76 | 3.84 | 2.72 | 1.71 | 2.29 | 5.42 |
| Flare-solar | LOW | 19.28 | 18.43 | 15.24 | 14.60 | 14.84 | 15.63 | 14.71 |
| | MEDIUM | 16.57 | 16.38 | 4.58 | 4.39 | 5.05 | 4.64 | 4.82 |
| | HIGH | 8.35 | 8.25 | 4.71 | 4.47 | 4.74 | 3.87 | 3.30 |

Table 2. The estimation error (×100) on natural language benchmarks. Feature dimension: \(d\). Number of clean instances in class-\(k\): \(N_k\). \([30k \times 4]\): \(N_1 = N_2 = N_3 = N_4 = 30k\). Average noise rate follows \(c = 1/(1 + r/\sqrt{K - 1})\). LOW: \(r = 8\). MEDIUM: \(r = 4\). HIGH: \(r = 1.5\). Top-2 of each row are bold.

<table>
<thead>
<tr>
<th>TEXT DATASETS ((d,{N_1, \ldots, N_d}))</th>
<th>NOISE RATE</th>
<th>T-REV</th>
<th>CL</th>
<th>HOCE</th>
<th>OURS-X-KL</th>
<th>OURS-X-TV</th>
<th>OURS-A-KL</th>
<th>OURS-A-TV</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG’s news (BERT) ((768, [30k \times 4]))</td>
<td>LOW</td>
<td>10.38</td>
<td>11.41</td>
<td>13.32</td>
<td>12.65</td>
<td>12.75</td>
<td>8.36</td>
<td>8.35</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>10.71</td>
<td>10.63</td>
<td>10.62</td>
<td>10.13</td>
<td>10.45</td>
<td>6.44</td>
<td>6.52</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>13.97</td>
<td>13.82</td>
<td>6.80</td>
<td>6.83</td>
<td>6.69</td>
<td>4.54</td>
<td>4.19</td>
</tr>
<tr>
<td>DBpedia (BERT) ((768, [40k \times 14]))</td>
<td>LOW</td>
<td>8.60</td>
<td>8.51</td>
<td>7.57</td>
<td>6.76</td>
<td>6.94</td>
<td>2.52</td>
<td>2.52</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>14.91</td>
<td>14.40</td>
<td>6.30</td>
<td>5.66</td>
<td>5.78</td>
<td>2.33</td>
<td>2.28</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>24.23</td>
<td>23.28</td>
<td>6.00</td>
<td>5.18</td>
<td>5.22</td>
<td>2.42</td>
<td>2.43</td>
</tr>
<tr>
<td>YELP-5 (BERT) ((768, [30k \times 5]))</td>
<td>LOW</td>
<td>38.49</td>
<td>38.75</td>
<td>40.87</td>
<td>40.71</td>
<td>40.58</td>
<td>37.37</td>
<td>37.19</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>35.46</td>
<td>36.05</td>
<td>33.63</td>
<td>34.23</td>
<td>33.88</td>
<td>31.79</td>
<td>31.94</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>21.20</td>
<td>20.88</td>
<td>19.09</td>
<td>18.56</td>
<td>20.13</td>
<td>18.11</td>
<td>18.06</td>
</tr>
<tr>
<td>Jigawat (BERT) ((768, [144,277, 15,294]))</td>
<td>LOW</td>
<td>17.10</td>
<td>16.44</td>
<td>11.28</td>
<td>11.80</td>
<td>12.23</td>
<td>7.45</td>
<td>7.66</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>7.19</td>
<td>6.81</td>
<td>4.84</td>
<td>4.85</td>
<td>3.43</td>
<td>0.78</td>
<td>1.02</td>
</tr>
</tbody>
</table>

Table 3. The last/best epoch clean test accuracies (%) when training with high-level noise defined in Table 2.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>AG’s news</th>
<th>DBpedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOCE ((Zhu et al., 2021c))</td>
<td>82.17</td>
<td>83.08</td>
</tr>
<tr>
<td>OURS-A-TV</td>
<td>85.01</td>
<td>85.17</td>
</tr>
</tbody>
</table>

40\% (high) noise is 0.44 – 0.4 = 0.04. Therefore, even though \(T\) is accurately estimated, the absolute error under our current metric will be higher for the HIGH-noise case. 2) As analyzed in Section 2.3, the error of random guess for low-noise (10\%) and high-noise (40\%) settings are 0.4 and 0.1, respectively, indicating a small error may cause more severe problems in higher-noise settings. We leave more detailed discussions to Appendix B.3.

Downstream learning error Liu & Wang (2021) showed the additional learning risk is positively related to estimation error. To further consolidate the estimation error reduction of our method, we feed the estimated \(T\) into forward loss correction (Patrini et al., 2017) and check the clean test accuracy. Table 3 shows our approach significantly improves both the best and last epoch test accuracy by simply changing the \(T\) in loss correction from HOCE estimates to ours.

6. Conclusions

This work has studied the problem of estimating noise transition matrix on application domains apart from images. We have proposed an information-theoretic approach to down-weight the less informative parts of features with only noisy labels for tasks with lower-quality features. Future directions include implementing and delivering this approach in real-world label noise settings, e.g., long-tail and open-set (Wei et al., 2022a; Hu et al., 2019).

Acknowledgment This work is partially supported by the National Science Foundation (NSF) under grants IIS-2007951, IIS-2143895 and CCF-2023495.
References


Scott, C. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In AISTATS, 2015.


Wei, H., Xie, R., Feng, L., Han, B., and An, B. Deep learning from multiple noisy annotators as a union. *IEEE Transactions on Neural Networks and Learning Systems*, 2022c.


Appendix

We show more theoretical details in Section A and more empirical details in Section B. Particularly,

- Section A.1 numerates some popular $f$-divergence functions and the corresponding optimal variational-conjugate pair $(g^*, f^*)$.
- Section A.2 shows the order-preserving property of using total variation.
- Section A.3 shows the order-preserving property of using KL divergence.
- Section B.1 explains why we build our method on HOC.
- Section B.2 compares two more baselines.
- Section B.3 discuss an interesting observation shown in Table 2 that high-noise settings may have lower errors.

A. Theorems

A.1. Common $f$-Divergence Functions

Following (Nowozin et al., 2016; Wei & Liu, 2020), we show some common $f$-divergence functions in Table 4.

<table>
<thead>
<tr>
<th>Name</th>
<th>$f(v)$</th>
<th>$g^*$</th>
<th>dom$_{f^*}$</th>
<th>$f^*(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Variation</td>
<td>$\frac{1}{2}</td>
<td>v - 1</td>
<td>$</td>
<td>$\frac{1}{2} \text{sign} \left( \frac{p(v)}{q(v)} - 1 \right)$</td>
</tr>
<tr>
<td>KL</td>
<td>$v \log v$</td>
<td>$1 + \log \frac{p(v)}{q(v)}$</td>
<td>$\mathbb{R}$</td>
<td>$e^{u-1}$</td>
</tr>
<tr>
<td>Jenson-Shannon</td>
<td>$-(1 + v) \log \frac{1 + u}{2} + v \log v$</td>
<td>$\log \frac{p(v) + q(v)}{p(v)q(v)}$</td>
<td>$u &lt; \log 2$</td>
<td>$-\log (2 - e^u)$</td>
</tr>
<tr>
<td>Squared Hellinger</td>
<td>$(\sqrt{v} - 1)^2$</td>
<td>$1 - \sqrt{\frac{q(v)}{p(v)}}$</td>
<td>$u &lt; 1$</td>
<td>$\frac{u}{1 - u}$</td>
</tr>
<tr>
<td>Pearson $\chi^2$</td>
<td>$(1 - v)^2$</td>
<td>$2 \left( \frac{p(v)}{q(v)} - 1 \right)$</td>
<td>$\mathbb{R}$</td>
<td>$\frac{1}{4}u^2 + u$</td>
</tr>
<tr>
<td>Neyman $\chi^2$</td>
<td>$\frac{(1-v)^2}{v}$</td>
<td>$1 - \left( \frac{q(v)}{p(v)} \right)^2$</td>
<td>$u &lt; 1$</td>
<td>$2 - 2\sqrt{1-u}$</td>
</tr>
<tr>
<td>Reverse KL</td>
<td>$-\log v$</td>
<td>$-\frac{q(v)}{p(v)}$</td>
<td>$\mathbb{R}_-$</td>
<td>$-1 - \log (-u)$</td>
</tr>
</tbody>
</table>
A.2. Total-Variation

A.2.1. Proof for Lemma 4.3

Proof. Consider TV, we have:

\[
\text{VD}_f(\tilde{g}^*) = \mathbb{E}_{V \sim P}[\widetilde{g}^*(V)] - \mathbb{E}_{P \sim Q}[\tilde{f}^*(\tilde{g}^*(V))]
\]

\[
= \frac{1}{2} \left[ \mathbb{E}_{V \sim P} \left[ \frac{\text{sign}(\tilde{p}(V))}{q(V)} - 1 \right] \right] - \mathbb{E}_{P \sim Q} \left[ \text{sign} \left( \frac{\tilde{p}(V)}{q(V)} - 1 \right) \right]
\]

\[
= \frac{1}{2} \int \sum_{z \in \{1, 2\}} \left[ \mathbb{P}(Z = z, Y = 1) - \mathbb{P}(Z = z) \mathbb{P}(Y = 1) \right] \cdot \text{sign} \left[ \mathbb{P}(Z = z, \tilde{Y} = 1) - \mathbb{P}(Z = z) \mathbb{P}(\tilde{Y} = 1) \right] dz
\]

\[
+ \frac{1}{2} \int \sum_{z \in \{1, 2\}} \left[ \mathbb{P}(Z = z, Y = 2) - \mathbb{P}(Z = z) \mathbb{P}(Y = 2) \right] \cdot \text{sign} \left[ \mathbb{P}(Z = z, \tilde{Y} = 2) - \mathbb{P}(Z = z) \mathbb{P}(\tilde{Y} = 2) \right] dz
\]

Note (assume class-dependent label noise)

\[
\text{sign} \left[ \sum_{z \in \{1, 2\}} \left( \mathbb{P}(\tilde{Y} = 1 | Z = z, Y = j) \mathbb{P}(Z = z, Y = j) - \mathbb{P}(Z = z) \mathbb{P}(\tilde{Y} = 1 | Y = j) \mathbb{P}(Y = j) \right) \right]
\]

= \text{sign} \left[ (1 - e_1) \mathbb{P}(Z = z, Y = 1) + e_2 \mathbb{P}(Z = z, Y = 2) - (1 - e_1) \mathbb{P}(Z = z) \mathbb{P}(Y = 1) - e_2 \mathbb{P}(Z = z) \mathbb{P}(Y = 2) \right]

Thus

\[
\text{VD}_f(\tilde{g}^*)
\]

\[
= \frac{1}{2} \int \left[ \mathbb{P}(Z = z, Y = 1) - \mathbb{P}(Z = z) \mathbb{P}(Y = 1) \right] \cdot \text{sign}(1 - e_1 - e_2) \cdot \text{sign}(\mathbb{P}(Z = z, Y = 1) - \mathbb{P}(Z = z) \mathbb{P}(Y = 1))
\]

\[
+ \left[ \mathbb{P}(Z = z, Y = 2) - \mathbb{P}(Z = z) \mathbb{P}(Y = 2) \right] \cdot \text{sign}(1 - e_1 - e_2) \cdot \text{sign}(\mathbb{P}(Z = z, Y = 2) - \mathbb{P}(Z = z) \mathbb{P}(Y = 2)) dz.
\]

When \(1 - e_1 - e_2 = 1\), i.e. \(e_1 + e_2 < 1\), we have

\[
\text{VD}_f(\tilde{g}^*)
\]

\[
= \frac{1}{2} \int \sum_{z \in \{1, 2\}} \left[ \mathbb{P}(Z = z, Y = i) - \mathbb{P}(Z = z) \mathbb{P}(Y = i) \right] \cdot \text{sign}(\mathbb{P}(Z = z, Y = i) - \mathbb{P}(Z = z) \mathbb{P}(Y = i)) dz
\]

\[
= \text{VD}_f(\tilde{g}^*).
\]

A.2.2. Proof for Theorem 4.4

Proof. Recall that, to show an \(f\)-mutual information metric is \(\epsilon\)-order-preserving under label noise, we need to study how \(\text{VD}_f(\tilde{g}^*)\) differs from the order of \(\text{VD}_f(\tilde{g}^*)\).
For total variation, with Lemma 4.2 and Lemma 4.3, we know
\[ \tilde{\text{VD}}_{\text{TV}}(\tilde{g}^*) = (1 - e_1 - e_2)\text{VD}_{\text{TV}}(\tilde{g}^*) = (1 - e_1 - e_2)\text{VD}_{\text{TV}}(g^*). \]

Therefore, when \( e_1 + e_2 < 1 \), \( \tilde{\text{VD}}_{\text{TV}}(\tilde{g}^*) \) always preserves the order of \( \text{VD}_{\text{TV}}(g^*) \), indicating the total-variation-based mutual information is 0-order-preserving under class-dependent label noise.

### A.3. KL Divergence

The definition of MI is
\[
I(Z, \tilde{Y}) = \sum_{j \in \{1, 2\}} \int \mathbb{P}(Z = z, \tilde{Y} = j) \log \left( \frac{\mathbb{P}(Z = z, \tilde{Y} = j)}{\mathbb{P}(Z = z) \mathbb{P}(\tilde{Y} = j)} \right) dz
\]

\[
= \sum_{j \in \{1, 2\}} \int \mathbb{P}(Z = z, \tilde{Y} = j) \log \mathbb{P}(Z = z) dz - \sum_{j \in \{1, 2\}} \int \mathbb{P}(Z = z, \tilde{Y} = j) \left( \log \mathbb{P}(Z = z) + \log \mathbb{P}(\tilde{Y} = j) \right) dz
\]

\[
= \sum_{j \in \{1, 2\}} \int \mathbb{P}(Z = z, \tilde{Y} = j) \log \mathbb{P}(Z = z, \tilde{Y} = j) dz - \sum_{j \in \{1, 2\}} \mathbb{P}(\tilde{Y} = j) \log \mathbb{P}(\tilde{Y} = j) dz,
\]

where
\[
-H(Z, \tilde{Y}) = \sum_{j \in \{1, 2\}} \int \mathbb{P}(Z = z, \tilde{Y} = j) \log \mathbb{P}(Z = z, \tilde{Y} = j) dz
\]

\[
= \int \mathbb{P}(Z = z, \tilde{Y} = 1) \log \mathbb{P}(Z = z, \tilde{Y} = 1) + \mathbb{P}(Z = z, \tilde{Y} = 2) \log \mathbb{P}(Z = z, \tilde{Y} = 2) dz.
\]

We first decouple term-1. Note
\[
\int \mathbb{P}(Z = z, \tilde{Y} = 1) \log \mathbb{P}(Z = z, \tilde{Y} = 1) dz
\]

\[
= \int \left[ \mathbb{P}(\tilde{Y} = 1|Z = z, Y = 1)\mathbb{P}(Z = z, Y = 1) + \mathbb{P}(\tilde{Y} = 1|Z = z, Y = 2)\mathbb{P}(Z = z, Y = 2) \right]
\]

\[
\cdot \log \left[ \mathbb{P}(\tilde{Y} = 1|Z = z, Y = 1)\mathbb{P}(Z = z, Y = 1) + \mathbb{P}(\tilde{Y} = 1|Z = z, Y = 2)\mathbb{P}(Z = z, Y = 2) \right] dz
\]

\[
= \int \left[ (1 - e_1)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z, Y = 2) \right]
\]

\[
\cdot \log \left[ (1 - e_1)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z, Y = 2) \right] dz
\]

\[
= \int \left[ (1 - e_1 - e_2)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z) \right]
\]

\[
\cdot \log \left[ (1 - e_1 - e_2)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z) \right] dz
\]

\[
+ \left[ (1 - e_1 - e_2)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z) \right] \log \left[ \frac{(1 - e_1 - e_2)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z)}{\mathbb{P}(Z = z, Y = 1)} \right] dz
\]

\[
= \int \left[ (1 - e_1 - e_2)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z) \right] \log \mathbb{P}(Z = z, Y = 1)
\]

\[
+ \left[ (1 - e_1 - e_2)\mathbb{P}(Z = z, Y = 1) + e_2\mathbb{P}(Z = z) \right] \log \left[ 1 - e_1 + e_2 \frac{\mathbb{P}(Z = z, Y = 2)}{\mathbb{P}(Z = z, Y = 1)} \right] dz.
\]
Let $\alpha = \mathbb{P}(Z = z, Y = 1)/\mathbb{P}(Z = z, Y = 2) \in [0, +\infty)$ (note $\alpha$ is actually a function of $(Z, Y)$). Then

$$
\int_z \mathbb{P}(Z = z, \bar{Y} = 1) \log \mathbb{P}(Z = z, \bar{Y} = 1) \, dz \\
= \int_z (1 - e_1 - e_2) \mathbb{P}(Z = z, Y = 1) \log \mathbb{P}(Z = z, Y = 1) \, dz + \int_z e_2 \mathbb{P}(Z = z) \log(\alpha + \log \mathbb{P}(Z = z, Y = 2)) \\
+ [(1 - e_1 - e_2) \alpha \mathbb{P}(Z = z, Y = 2) + e_2 \mathbb{P}(Z = z)] \log (1 - e_1 + \frac{e_2}{\alpha}) \, dz
$$

and

$$
\int_z \mathbb{P}(Z = z, \bar{Y} = 2) \log \mathbb{P}(Z = z, \bar{Y} = 2) \, dz \\
= \int_z (1 - e_1 - e_2) \mathbb{P}(Z = z, Y = 2) \log(1 - e_2 + e_1 \alpha) \, dz
$$

Thus

$$
\sum_{j \in \{1, 2\}} \int_z \mathbb{P}(Z = z, \bar{Y} = j) \log \mathbb{P}(Z = z, \bar{Y} = j) \, dz \\
= (1 - e_1 - e_2) \sum_{i \in \{1, 2\}} \int_z \mathbb{P}(Z = z, Y = i) \log \mathbb{P}(Z = z, Y = i) \, dz \\
+ \int_z e_2 \mathbb{P}(Z = z) \log(1 - e_2 + e_1 \alpha) + e_2 \mathbb{P}(Z = z) \log(1 - e_1 + \frac{e_2}{\alpha}) \, dz \\
+ (1 - e_1 - e_2) \mathbb{P}(Z = z) \left[ \alpha \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) + \log(1 - e_2 + e_1 \alpha) \right] \, dz \\
= (1 - e_1 - e_2) \sum_{i \in \{1, 2\}} \int_z \mathbb{P}(Z = z, Y = i) \log \mathbb{P}(Z = z, Y = i) \, dz \\
+ \int_z e_2 \mathbb{P}(Z = z) \log(1 - e_2 + e_1 \alpha) + e_2 \mathbb{P}(Z = z) \log(1 - e_1 + \frac{e_2}{\alpha}) \, dz \\
+ \mathbb{P}(Z = z) \left[ \alpha \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) + \log(1 - e_2 + e_1 \alpha) \right] \, dz \\
= (1 - e_1 - e_2) \sum_{i \in \{1, 2\}} \int_z \mathbb{P}(Z = z, Y = i) \log \mathbb{P}(Z = z, Y = i) \, dz \\
+ \int_z (e_1 + e_2) \mathbb{P}(Z = z) \log \mathbb{P}(Z = z) + \mathbb{P}(Z = z) \Delta_{\text{Bias}}(\alpha, e_1, e_2) \, dz, \\
\text{(Term 1.1)}
$$

where in Term 1.2:

$$
\Delta_{\text{Bias}}(\alpha, e_1, e_2) = e_2 \log \alpha - (e_1 + e_2) \log(1 + \alpha) + \frac{(1 - e_1 - e_2)}{\alpha + 1} \left[ \alpha \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) + \log(1 - e_2 + e_1 \alpha) \right] \\
+ \left[ e_1 \log(1 - e_2 + e_1 \alpha) + e_2 \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) \right]. \\
(5)
$$
In Term 1.1, recalling $\alpha = \mathbb{P}(Z = z, Y = 1)/\mathbb{P}(Z = z, Y = 2)$, we have

$$(1 - e_1 - e_2) \sum_{i \in\{1,2\}} \int_z \mathbb{P}(Z = z, Y = i) \log \mathbb{P}(Z = z, Y = i) \, dz$$

$$= (1 - e_1 - e_2) \int_z \mathbb{P}(Z = z) \log \frac{\mathbb{P}(Z = z)}{\alpha + 1} + \frac{\mathbb{P}(Z = z)\alpha}{\alpha + 1} \log \frac{\alpha}{\alpha + 1} \, dz$$

$$= (1 - e_1 - e_2) \int_z \mathbb{P}(Z = z) \log \mathbb{P}(Z = z) + \mathbb{P}(Z = z)\log \frac{1}{\alpha + 1} + \frac{\mathbb{P}(Z = z)\alpha}{\alpha + 1} \log \frac{\alpha}{\alpha + 1} \, dz$$

$$= (1 - e_1 - e_2) \int_z \mathbb{P}(Z = z) \log \mathbb{P}(Z = z) \, dz + (1 - e_1 - e_2) \int_z \mathbb{P}(Z = z) \left[ \frac{\alpha}{\alpha + 1} \log \frac{\alpha}{\alpha + 1} + \frac{1}{\alpha + 1} \log \frac{1}{\alpha + 1} \right] \, dz. $$

Denote the effective part of MI by

$$\Delta_{\text{MI}}(\alpha, e_1, e_2) = (1 - e_1 - e_2) \left[ \frac{\alpha}{\alpha + 1} \log \frac{\alpha}{\alpha + 1} + \frac{1}{\alpha + 1} \log \frac{1}{\alpha + 1} \right].$$

We have

$$-H(Z, \bar{Y}) = \sum_{j \in\{1,2\}} \int_z \mathbb{P}(Z = z, \bar{Y} = j) \log \mathbb{P}(Z = z, \bar{Y} = j) \, dz$$

$$= \int_z \mathbb{P}(Z = z) \log \mathbb{P}(Z = z) \, dz + \int_z \mathbb{P}(Z = z) \left[ \Delta_{\text{MI}}(\alpha, e_1, e_2) + \Delta_{\text{Bias}}(\alpha, e_1, e_2) \right] \, dz,$$

and

$$I(Z, \bar{Y}) = \int_z \mathbb{P}(Z = z) \left[ \Delta_{\text{MI}}(\alpha, e_1, e_2) + \Delta_{\text{Bias}}(\alpha, e_1, e_2) \right] \, dz + H(\bar{Y}).$$

Define $\Delta_{\text{Bias,MI}}(\alpha, e_1, e_2) = \Delta_{\text{MI}}(\alpha, e_1, e_2) + \Delta_{\text{Bias}}(\alpha, e_1, e_2)$. Then

$$\Delta_{\text{Bias,MI}}(\alpha, e_1, e_2) = \Delta_{\text{MI}}(\alpha, e_1, e_2) + \Delta_{\text{Bias}}(\alpha, e_1, e_2)$$

$$= (1 - e_1 - e_2) \left[ \frac{\alpha}{\alpha + 1} \log \frac{\alpha}{\alpha + 1} + \frac{1}{\alpha + 1} \log \frac{1}{\alpha + 1} \right] + e_2 \log \alpha - (e_1 + e_2) \log(\alpha + 1)$$

$$+ \frac{1 - e_1 - e_2}{\alpha + 1} \left[ \alpha \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) + \log(1 - e_1 + e_2) \right]$$

$$+ \left[ e_1 \log(1 - e_2 + e_1\alpha) + e_2 \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) \right]$$

$$= (1 - e_1 - e_2) \left[ \frac{\alpha}{\alpha + 1} \log \alpha + \log \frac{1}{\alpha + 1} \right] + e_2 \log \alpha + (e_1 + e_2) \log \frac{1}{\alpha + 1} +$$

$$\left[ \frac{1 - e_1 - e_2}{\alpha + 1} \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) + \frac{1 - e_1 - e_2}{\alpha + 1} \log(1 - e_2 + e_1\alpha) \right]$$

$$+ \left[ \frac{1 - e_1 - e_2}{\alpha + 1} \log \alpha + \log \frac{1}{\alpha + 1} +$$

$$\left[ \frac{1 - e_1 - e_2}{\alpha + 1} \log \left( 1 - e_1 + \frac{e_2}{\alpha} \right) + \frac{1 - e_1 - e_2}{\alpha + 1} \log(1 - e_2 + e_1\alpha) \right]$$

$$= \log \frac{1}{\alpha + 1} + \frac{1}{\alpha + 1} \left[ (1 - e_1 - e_2)\alpha + e_2(\alpha + 1) \right] \log(\alpha(1 - e_1) + e_2)$$

$$+ \frac{1}{\alpha + 1} \left[ (1 - e_1 - e_2)\alpha + e_2(\alpha + 1) \right] \log(1 - e_2 + e_1\alpha)$$

$$= \frac{\alpha(1 - e_1) + e_2}{\alpha + 1} + \frac{\alpha + 1 - (\alpha(1 - e_1) + e_2)}{\alpha + 1} \log \frac{1}{\alpha + 1} + \frac{1}{\alpha + 1} \left[ \alpha(1 - e_1) + e_2 \right] \log(\alpha(1 - e_1) + e_2)$$

$$+ \frac{1}{\alpha + 1} \left[ (\alpha + 1 - (\alpha(1 - e_1) + e_2)) \log((\alpha + 1) - (\alpha(1 - e_1) + e_2)) \right]$$

$$= \frac{\alpha(1 - e_1) + e_2}{\alpha + 1} \log \frac{\alpha(1 - e_1) + e_2}{\alpha + 1} + \frac{1}{\alpha + 1} \left[ \alpha(1 - e_1) + e_2 \right] \log \left[ 1 - \frac{\alpha(1 - e_1) + e_2}{\alpha + 1} \right].$$
A looser bound that holds for all the possible $\beta$. Assume $\beta$ is a function of $z$. We drop notation $z$ for ease of notation. Then

$$\Delta_{\text{Bias}}(\beta, e_1, e_2) = ((1 - e_1 - e_2)\beta + e_2) \log ((1 - e_1 - e_2)\beta + e_2) + 1 - ((1 - e_1 - e_2)\beta + e_2) \log [1 - ((1 - e_1 - e_2)\beta + e_2)].$$

The bias caused by label noise is

$$\Delta_{\text{Bias}}(\beta, e_1, e_2) = \Delta_{\text{Bias}, \text{MI}}(\beta, e_1, e_2) - \Delta_{\text{MI}}(\beta, e_1, e_2) = (1 - e_1 - e_2)\beta + e_2 \log ((1 - e_1 - e_2)\beta + e_2) + 1 - ((1 - e_1 - e_2)\beta + e_2) \log [1 - ((1 - e_1 - e_2)\beta + e_2)] - (1 - e_1 - e_2) [\beta \log \beta + (1 - \beta) \log (1 - \beta)].$$

To get $\arg \max_{\beta \in [0, 1]}$, we check the first derivative:

$$\frac{\partial \Delta_{\text{Bias}}(\beta, e_1, e_2)}{\partial \beta} = (1 - e_1 - e_2) \left[ \log \frac{(1 - e_1 - e_2)\beta + e_2}{1 - (1 - e_1 - e_2)\beta - e_2} - \log \frac{\beta}{1 - \beta} \right].$$

Let $\frac{\partial \Delta_{\text{Bias}}(\beta, e_1, e_2)}{\partial \beta} = 0$, we have

$$\beta^* = \frac{e_2}{e_1 + e_2}.$$

By checking $\frac{\partial^2 \Delta_{\text{Bias}}(\beta, e_1, e_2)}{\partial \beta^2}$, we can find that $\Delta_{\text{Bias}}(\beta, e_1, e_2)$ is increasing when $\beta \in [0, \beta^*]$, and decreasing when $\beta \in [\beta^*, 1]$. Thus $\beta^* = \frac{e_2}{(e_1 + e_2)}$ is the global maximum and the upper bound for $\Delta_{\text{Bias}}(\beta, e_1, e_2)$ is

$$\Delta_{\text{Bias}}(\beta, e_1, e_2) \leq e_1 \log e_1 + e_2 \log e_2 - (e_1 + e_2) \log (e_1 + e_2).$$

Assume $\beta \in [\bar{\beta}, \beta]$ in practice. The lower bound for $\Delta_{\text{Bias}}(\beta, e_1, e_2)$ is

$$\Delta_{\text{Bias}}(\beta, e_1, e_2) \geq \min(\Delta_{\text{Bias}}(\beta, e_1, e_2), \Delta_{\text{Bias}}(\bar{\beta}, e_1, e_2)).$$

A looser bound that holds for all the possible $\beta \in [0, 1]$ is:

$$\Delta_{\text{Bias}}(\beta, e_1, e_2) \geq \min(e_1 \log e_1 + (1 - e_1) \log (1 - e_1), e_2 \log e_2 + (1 - e_2) \log (1 - e_2)).$$

Note (when $e_1 = e_2 = 0$)

$$I(Z, Y) = \int_z \mathbb{P}(Z = z) [\Delta_{\text{MI}}(\alpha, 0, 0)] \, dz + H(Y),$$

and $\Delta_{\text{MI}}(\beta, e_1, e_2) = (1 - e_1 - e_2)\Delta_{\text{MI}}(\alpha, 0, 0)$.

Hence (note $\beta$ is actually a function of $Z$),

$$I(Z, \bar{Y}) = \int_z \mathbb{P}(Z = z) [\Delta_{\text{MI}}(\alpha, e_1, e_2) + \Delta_{\text{Bias}}(\alpha, e_1, e_2)] \, dz + H(\bar{Y})$$

$$= \int_z \mathbb{P}(Z = z) [(1 - e_1 - e_2)\Delta_{\text{MI}}(\beta, 0, 0) + \Delta_{\text{Bias}}(\beta, e_1, e_2)] \, dz + H(\bar{Y})$$

$$= (1 - e_1 - e_2) \int_z \mathbb{P}(Z = z) \Delta_{\text{MI}}(\beta, 0, 0) \, dz + \int_z \Delta_{\text{Bias}}(\beta, e_1, e_2) \, dz + H(\bar{Y})$$

$$= (1 - e_1 - e_2) I(Z, Y) + \int_z \Delta_{\text{Bias}}(\beta, e_1, e_2) \, dz - (1 - e_1 - e_2) H(Y) + H(\bar{Y})$$

$$= (1 - e_1 - e_2) I(Z, Y) + C(e_1, e_2, Y, \bar{Y}) + \Delta_Z(e_1, e_2),$$

where $C(e_1, e_2, Y, \bar{Y})$ is the conditional entropy.
where
\[ C(e_1, e_2, Y, \tilde{Y}) = \min(e_1 \log e_1 + (1 - e_1) \log(1 - e_1), e_2 \log e_2 + (1 - e_2) \log(1 - e_2)) - (1 - e_1 - e_2)H(Y) + H(\tilde{Y}) \]
is a constant for given \( Y \) and \( \tilde{Y} \). The other part is in the range \( \Delta_Z(e_1, e_2) \in [0, \text{Gap}_Z(e_1, e_2)] \), and
\[
\text{Gap}_Z(e_1, e_2) = e_1 \log e_1 + e_2 \log e_2 - (e_1 + e_2) \log(e_1 + e_2) \\
- \min(e_1 \log e_1 + (1 - e_1) \log(1 - e_1), e_2 \log e_2 + (1 - e_2) \log(1 - e_2)).
\]
Note \( \Delta_Z(e_1, e_2) \) may be different for \( Z_\mu \) and \( Z_\nu, \mu \neq \nu \).

Therefore, when
\[
|I_f(Z_\mu; \tilde{Y}) - I_f(Z_\nu; \tilde{Y})| > \text{Gap}_Z(e_1, e_2),
\]
we have
\[
\text{sign}[I_f(Z_\mu; Y) - I_f(Z_\nu; Y)] = \text{sign}[I_f(Z_\mu; \tilde{Y}) - I_f(Z_\nu; \tilde{Y})], \forall \mu \in [d] \nu \in [d].
\]

Now we take a further look at the gap \( \text{Gap}_Z(e_1, e_2) \). Assume \( e_1 \geq e_2 \Rightarrow e_2 = \delta e_1 \), where \( \delta \in [0, 1] \). Then \( H(e_1) \leq H(e_2) \), and
\[
\text{Gap}_Z(e_1, e_2) = e_1 \log e_1 + e_2 \log e_2 - (e_1 + e_2) \log(e_1 + e_2) - \min(H(e_1), H(e_2)) \\
= e_2 \log e_2 - (e_1 + e_2) \log(e_1 + e_2) - (1 - e_1) \log(1 - e_1) \\
= e_2 \log e_2 + e_1 \log \frac{1}{1 + \delta} + e_1 \log \frac{1 - e_1}{e_1(1 + \delta)} - \log(1 - e_1) \\
= e_1 \left[ \delta \log \frac{1 - e_1}{1 + \delta} \log \frac{1}{1 + \delta} + e_1 \log \frac{1 - e_1}{e_1} \right] + e_1 \log(1 - e_1) - \log(1 - e_1) \\
= e_1 [\delta \log (1 + 1) \log(1 + 1)] - (1 - e_1) \log(1 - e_1) - e_1 \log e_1 \\
= e_1 [\delta \log (1 + 1) \log(1 + 1)] + H(e_1)
\]

Note: We can also roughly estimate \( \delta \log \delta - (1 + \delta) \log(1 + \delta) \) by the best quadratic fit (rooted mean squared error \( \approx 0.02 \)) and get
\[
\delta \log \delta - (1 + \delta) \log(1 + \delta) \approx 0.9124\delta^2 - 2.14\delta - 0.1202.
\]

**B. More Discussions**

**B.1. Rationale for building on HOC**

The rationale for building our analyses on HOC is described as follows. Our major concern is that the learning-based approaches usually require more effort in tuning their hyperparameters. The effect of reweighing feature variables will be entangled with the training procedure, making things more complicated. On the other hand, the training-free approach seems to be more lightweight to employ the reweighing treatment. In particular, HOC consistently achieves lower estimation error, as shown in Figure 1.

**B.2. More Experimental Results**

We only use one linear layer as the model for the learning-based methods since it can achieve satisfying performance when training with clean data.

We compare our methods with two recent works (Li et al., 2021; Zhang et al., 2021a) in Table 5. Our method achieves an overall better performance than theirs. We search the learning rate from \([0.01, 0.001, 0.0001, 0.00001]\) and report the best result for their methods. During experiments, we find their methods tend to be sensitive to hyperparameter settings. On AG’s news and Jigsaw, we find T-Vol and T-TV estimate \( T \approx I \) for all three noise settings. Thus the low estimation error for the low-noise setting is more like a coincidence due to the trivial solution \( I \).
Table 5. The estimation error (×100) on natural language benchmarks. Feature dimension: d. Number of clean instances in class-k: N_k. [30K × 4]: N_1 = N_2 = N_3 = N_4 = 30k. Average noise rate follows e = 1/(1 + r/√K − 1). LOW: r = 8. MEDIUM: r = 4. HIGH: r = 1.5. Top-2 of each row are bold.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(d, [N_1, ⋯ , N_K])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AG's NEWS (BERT) (768, [30k × 4])</td>
<td>LOW</td>
<td>10.38</td>
<td>11.41</td>
<td>13.32</td>
<td>9.44</td>
<td>7.08</td>
<td>8.36</td>
<td>8.35</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>10.71</td>
<td>10.63</td>
<td>10.62</td>
<td>10.53</td>
<td>11.02</td>
<td>6.44</td>
<td>6.52</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>13.97</td>
<td>13.82</td>
<td>6.80</td>
<td>30.33</td>
<td>31.66</td>
<td>4.54</td>
<td>4.19</td>
</tr>
<tr>
<td>DBPEDIA (BERT) (768, [40k × 14])</td>
<td>LOW</td>
<td>6.80</td>
<td>5.31</td>
<td>7.57</td>
<td>34.23</td>
<td>34.00</td>
<td>2.52</td>
<td>2.52</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>14.91</td>
<td>14.40</td>
<td>6.30</td>
<td>26.94</td>
<td>27.62</td>
<td>2.33</td>
<td>2.28</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>24.23</td>
<td>23.28</td>
<td>6.00</td>
<td>30.72</td>
<td>31.35</td>
<td>2.42</td>
<td>2.43</td>
</tr>
<tr>
<td>YELP-5 (BERT) (768, [130k × 5])</td>
<td>LOW</td>
<td>35.46</td>
<td>36.05</td>
<td>33.63</td>
<td>13.16</td>
<td>12.46</td>
<td>37.37</td>
<td>37.19</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>38.49</td>
<td>38.75</td>
<td>40.87</td>
<td>12.30</td>
<td>12.48</td>
<td>31.79</td>
<td>31.94</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>42.90</td>
<td>41.88</td>
<td>19.09</td>
<td>30.90</td>
<td>33.20</td>
<td>18.11</td>
<td>18.06</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>17.10</td>
<td>16.44</td>
<td>11.28</td>
<td>12.23</td>
<td>12.21</td>
<td>7.45</td>
<td>7.66</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>7.19</td>
<td>6.81</td>
<td>4.84</td>
<td>32.39</td>
<td>32.39</td>
<td>0.78</td>
<td>1.02</td>
</tr>
</tbody>
</table>

B.3. More Experiments

High-noise settings may have lower errors Table 2 shows the estimation error may decrease for higher noise rate settings. This observations mainly due to two reasons:

- Reason-1: The original dataset may be noisy. Notably, the original Yelp dataset contains lots of noisy reviews (Luca, 2016). Now we analyze the issues caused by an originally noise dataset with a toy example.

**Example:** Consider a binary classification with inherent 20% noise. Define two noise settings: 1) Low noise: Add 10% symmetric label noise (e_1 = e_2 = 0.1). 2) High noise: Add 40% symmetric label noise (e_1 = e_2 = 0.4). For the low noise setting, the real average noise rate is:

\[ e_{\text{low, real}} = 0.1 \times 0.8 + 0.2 \times 0.9 = 0.26. \]

For the high noise setting, the real average noise rate is:

\[ e_{\text{high, real}} = 0.4 \times 0.8 + 0.2 \times 0.6 = 0.44. \]

Note e_{\text{low, synthetic}} = 0.1 and e_{\text{high, synthetic}} = 0.4. Thus the gap between the real noise rate and the synthetic noise rate is

\[ e_{\text{low, real}} - e_{\text{low, synthetic}} = 0.26 - 0.1 = 0.16, \quad e_{\text{high, real}} - e_{\text{high, synthetic}} = 0.44 - 0.4 = 0.04. \]

Therefore, with inherent label noise exists, the perfectly estimated \( \mathbf{T} \) will have an error of 0.16 for the low-noise setting and an error of 0.04 for the high-noise setting, which accounts for our current observation.

- Reason-2: The tolerance of noise rates for different settings are different. Consider a binary classification with symmetric label noise, i.e., e_1 = e_2 = e. Let the random guess be e_1 = e_2 = 0.5. Define the estimation error caused by the random guess as the tolerance. Thus the tolerances when e = 0.1 and e = 0.4 are 0.4 and 0.1, respectively. From this aspect, an error of 0.1 will not destroy the low-noise case since

\[ |\hat{e}_{1,\text{low}} - e_{1,\text{low}}| = 0.1 \Rightarrow \hat{e}_{1,\text{low}} = 0.2. \]

But an error of 0.1 may destroy the high-noise case since

\[ |\hat{e}_{1,\text{high}} - e_{1,\text{high}}| = 0.1 \Rightarrow \hat{e}_{1,\text{high}} = 0.3 \text{ or } 0.5. \]

Therefore, although the error of high-noise settings seems low, it may cause severe problems.
A preliminary test on calibrating inherent errors

We do the following experiment to help explain Reason-1 by calibrating the inherent label noise in Yelp-5. Note the label noise accumulation follows:

$$T_{\text{real}} = T_{\text{org}} T_{\text{synthetic}},$$

where $$T_{\text{synthetic}} = T$$. If we know $$T_{\text{org}}$$, we can calibrate $$T_{\text{synthetic}}$$ and evaluate the error by $$\text{Error}(T_{\text{org}}, T_{\text{org}}^{-1})$$. Unfortunately, we cannot find the ground-truth $$T_{\text{org}}$$ for Yelp-5. For a preliminary test, we estimate $$T_{\text{org}}$$ by applying OURS-A-KL on the original Yelp dataset. We show the calibrated error in Table 6, where we can find the high-noise settings are indeed more challenging (higher error) compared with the low-noise setting.

Table 6. The calibrated estimation error ($\times 100$) on Yelp-5. Average noise rate follows $e = 1/(1 + r/\sqrt{K-1})$. LOW: $r = 8$. MEDIUM: $r = 4$. HIGH: $r = 1.5$.

<table>
<thead>
<tr>
<th>TEXT DATASETS \ METHOD</th>
<th>LOW</th>
<th>MEDIUM</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>OURS-A-KL</td>
<td>3.56</td>
<td>4.01</td>
<td>8.59</td>
</tr>
<tr>
<td>OURS-A-TV</td>
<td>3.46</td>
<td>2.59</td>
<td>8.56</td>
</tr>
</tbody>
</table>