Consistent Multi-Label Classification from Noisy Labels

Mingyuan Zhang
University of Pennsylvania
Philadelphia, PA 19104
myz@seas.upenn.edu

Shivani Agarwal
University of Pennsylvania
Philadelphia, PA 19104
ashivani@seas.upenn.edu

Abstract

In many applications of machine learning, the training data comes with noisy labels; this issue is even more pronounced in multi-label problems, where multiple labels/tags can be active in an instance simultaneously. In recent years, many consistent noise-corrected algorithms have been designed for binary and multiclass learning under class-conditional noise (CCN) and other noise models; however, relatively few consistent algorithms exist for multi-label learning, and those that do are under the very simple independent flipping noise (IFN) model. In this paper, we develop three consistent noise-corrected multi-label learning algorithms: *Noise-Corrected Plug-in* (NCPLUG) algorithm for Hamming loss under IFN; *Noise-Corrected Exact F-measure Plug-in* (NCEFP) algorithm for multi-label F_1-measure under general CCN; and *Noise-Corrected Output Coding* (NCOC) algorithm for general low-rank multi-label losses under general CCN. We provide quantitative regret transfer bounds for all three algorithms to establish their consistency. We also propose a new family of structured multi-label noise models that we term *Similar-Tag Switching Noise* (STSN) models; STSN models are a special case of CCN that require fewer parameters and enable fast computation, and moreover, unlike IFN, they also capture some correlations among tags. Our experiments confirm the effectiveness of our algorithms in correcting for multi-label noise.

1 Introduction

In many applications of machine learning, accurate labels are difficult or expensive to obtain; therefore, in practice, one often receives noisy labels. This problem is even more pronounced in multi-label classification (MLC) settings, where multiple labels/tags can be active in an instance simultaneously. In recent years, there has been much interest in developing learning algorithms that can learn good classifiers from data with noisy labels [10, 13, 32]. While there has been much work in this area for binary and multiclass problems, there has been relatively limited work on multi-label learning from noisy labels. In this paper, we develop principled noise-corrected multi-label learning algorithms for a variety of performance measures under the general class-conditional noise (CCN) model.

The key challenge in learning from noisy labels is to develop algorithms that can produce accurate classifiers for the true/clean distribution despite noisy labels; in particular, a desirable goal is that the algorithms should be *(Bayes) consistent*, meaning that as the size of the (noisy) training sample increases, the performance of the learned classifier converges to the Bayes optimal performance under the clean (non-noisy) distribution. For binary and multiclass learning, many such consistent algorithms have been designed under CCN and related noise models [23, 30, 29, 22, 19, 26, 12, 35, 27, 24, 38, 41, 20, 48, 18, 47]. However, for multi-label learning, only a few consistent noise-corrected algorithms have been designed, and the consistency guarantees that do currently exist are for specific performance measures under the relatively simple independent flipping noise (IFN) model (a special case of CCN) that fails to capture correlations among tags [16, 44]. In this paper, we develop provably Bayes consistent noise-corrected multi-label algorithms for a broad family of multi-label performance measures under the general CCN model.

Preprint. Under review.

Noise model	Multi-label losses	Noise-corrected algorithms
Symmetric IFN	Hamming	[16]
IFN	Hamming, Ranking	CCMN [44]
IFN	Hamming	NCPLUG (this work)
CCN	F_1-measure	NCEFP (this work)
CCN	General low-rank $\mathbf{L}$	NCOC (this work)

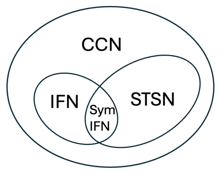


Figure 1: Summary of consistent noise-corrected multi-label algorithms and associated noise models.

Our contributions include the following (see also Figure 1):

1. Algorithms. We provide the following three consistent noise-corrected multi-label algorithms, all of which work by identifying a small set of what we call 'Bayes-sufficient' statistics for the target loss and estimating these reliably from the given noisy training sample:

- *Noise-Corrected Plug-in* (NCPLUG) algorithm for Hamming loss under IFN;
- *Noise-Corrected Exact F-measure Plug-in* (NCEFP) algorithm for multi-label F_1-measure under general CCN;
- *Noise-Corrected Output Coding* (NCOC) algorithm for general low-rank multi-label losses under general CCN.

2. Regret transfer bounds and consistency. For all these algorithms, we provide quantitative regret transfer bounds to establish consistency. The bounds suggest that as the amount of label noise increases, the (noisy) sample size needed to reach a given level of performance generally increases.

3. Similar-Tag Switching Noise (STSN) model. While our NCEFP and NCOC algorithms are provably consistent under general CCN models, in multi-label settings, general CCN models involve extremely large noise matrices that make computation prohibitive. We propose a new family of structured multi-label noise models that we term *Similar-Tag Switching Noise* (STSN) models. STSN models are a special case of CCN that require fewer parameters and enable fast computation for NCEFP and NCOC; moreover, unlike IFN models, they also capture some correlations among tags.

4. Experimental validation. Finally, we evaluate our algorithms on both synthetic and real data.

Related work. Below we briefly review some works that are most closely related to our study.

• **Consistent algorithms for standard (non-noisy) multi-label learning.** Bayes optimal multi-label classifiers and consistent algorithms for MLC performance measures, including Hamming loss and F_1-measure, have been studied by [4, 6, 11, 5, 21, 49, 40, 39] and others. These works do not deal with noisy labels. A detailed survey on multi-label learning can be found in [46].

• **Consistent algorithms for binary/multiclass learning from noisy labels for CCN models.** Many consistent noise-corrected algorithms have been designed for binary/multiclass learning [23, 30, 29, 22, 19, 26, 12, 35, 27, 24, 38, 41, 20, 48, 18, 47]. But when applied to multi-label problems in a straightforward way (by treating each label vector as a class), these algorithms need exponential (in the number of tags) number of parameters and suffer from slow computation. In essence, they are not designed for multi-label problems.

• **Multi-label learning from noisy labels.** Two types of noise models have been studied: statistical and non-statistical. For statistical noise models, [16] studied a special 'symmetric' case of IFN and focused on loss functions satisfying certain conditions (e.g., Hamming loss). [44] showed consistent algorithms for Hamming and Ranking losses under IFN. [17] proposed a way to estimate noise matrices under IFN. For non-statistical noise models, partial multi-label learning (PML) is a prominent example, where for each instance, its noisy label contains all active tags from the clean label, as well as some non-active tags. Some algorithms have been proposed to deal with PML [42, 9, 37, 33, 43]; however it is unclear whether a Bayes optimal classifier can be recovered under PML. Other empirical studies of multi-label learning from noisy labels include [36, 14, 3, 50].

Organization. After preliminaries and background in Section 2, we provide intuition for our algorithms in Section 3, followed by our three noise-corrected algorithms in Section 4. Section 5 gives regret transfer bounds for our algorithms. Section 6 describes the STSN model. Section 7 summarizes our experiments. Finally, Section 8 concludes the paper. All proofs are in the Appendix.

Notation. For an integer n, we denote by $[n]$ the set of integers $\{1, \ldots, n\}$, and by Δ_n the probability simplex $\{\mathbf{p} \in \mathbb{R}_+^n : \sum_{y=1}^n p_y = 1\}$. For a vector $\mathbf{a}$, we denote by $\|\mathbf{a}\|_p$ the p-norm of $\mathbf{a}$, and by a_j the j-indexed entry of $\mathbf{a}$. For a matrix $\mathbf{A}$, we denote by $\|\mathbf{A}\|_p$ the induced p-norm of $\mathbf{A}$, by $\mathbf{a}_y$ the

y-indexed column vector of $\mathbf{A}$, and by $a_{j,y}$ the (j,y)-indexed element of $\mathbf{A}$. We use $\mathbf{1}(\cdot)$ to denote the indicator function and $\xrightarrow{p}$ to denote convergence in probability.

2 Preliminaries and background

Multi-label classification (MLC) with noisy labels. In an MLC problem, there is an instance space $\mathcal{X}$, and a set of s tags $\mathcal{T} = [s] := \{1, \ldots, s\}$ that can be associated with each instance in $\mathcal{X}$. For example, in image tagging, $\mathcal{X}$ is the set of possible images, and $\mathcal{T}$ is a set of s pre-defined tags (such as `sky`, `cloud`, `water` etc.) that can be associated with each image. The label space $\mathcal{Y} \subseteq \{0,1\}^s$ consists of label vectors $\mathbf{y} \in \{0,1\}^s$ that indicate which of the s tags are active (specifically, $y_j = 1$ denotes that tag j is active, and $y_j = 0$ denotes it is inactive). Let $|\mathcal{Y}|$ denote the size of $\mathcal{Y}$.

There is a (unknown) joint probability distribution D on $\mathcal{X} \times \mathcal{Y}$ from which labeled examples $(X, \mathbf{Y})$ are drawn. In the standard (non-noisy) MLC problem, the learner would be given training examples drawn directly from D. However, when learning from noisy labels, the learner instead sees only noisy examples $(X, \widetilde{\mathbf{Y}})$, where $\widetilde{\mathbf{Y}}$ is a noisy version of $\mathbf{Y}$. Given a noisy training sample $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, the goal is to learn a multi-label classifier $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$ that performs well with respect to the clean distribution D.

Class-conditional noise (CCN). The label noise models we consider here belong to the well-known CCN model that has been widely studied in binary and multiclass learning from noisy labels [23, 35, 27], in which a label $\mathbf{y}$ is randomly flipped to a label $\widetilde{\mathbf{y}}$ with some probability $c_{\mathbf{y},\widetilde{\mathbf{y}}}$ that depends on $\mathbf{y}$ and $\widetilde{\mathbf{y}}$, but not on the features. Specifically, the CCN model is characterized by a row-stochastic noise matrix $\mathbf{C} \in [0,1]^{|\mathcal{Y}| \times |\mathcal{Y}|}$ with entries $c_{\mathbf{y},\widetilde{\mathbf{y}}}$, such that for each $\mathbf{y}, \widetilde{\mathbf{y}} \in \mathcal{Y}$, $c_{\mathbf{y},\widetilde{\mathbf{y}}} = \mathbf{P}(\widetilde{\mathbf{Y}} = \widetilde{\mathbf{y}} \,|\, \mathbf{Y} = \mathbf{y})$. The noisy training examples can therefore be viewed as being drawn i.i.d. from a 'noisy' distribution $\widetilde{D}$ on $\mathcal{X} \times \mathcal{Y}$: an example $(X, \mathbf{Y})$ is first drawn randomly according to D, and then noise is injected according to the noise matrix $\mathbf{C}$ to produce $(X, \widetilde{\mathbf{Y}})$. In MLC, $|\mathcal{Y}|$ can be as large as 2^s; therefore, a fully general noise matrix $\mathbf{C}$ requires too many parameters (exponential in s). This necessitates considering more structured noise models well-suited to MLC problems.

Independent flipping noise (IFN). So far, most previous work has considered only the very simple multi-label IFN model (a special case of CCN) where each tag is flipped independently from active to inactive or vice versa; this involves only $2s$ parameters defined as $c^{(j)}_{0,1} = \mathbf{P}(\widetilde{Y}_j = 1 | Y_j = 0)$ and $c^{(j)}_{1,0} = \mathbf{P}(\widetilde{Y}_j = 0 | Y_j = 1)$, $\forall j \in [s]$.

Multi-label performance measures/loss matrices L. We will consider multi-label loss matrices of the form $\mathbf{L} \in \mathbb{R}_+^{|\mathcal{Y}| \times |\mathcal{Y}|}$, with entries $\ell_{\mathbf{y},\widehat{\mathbf{y}}}$ indicating the loss incurred on predicting $\widehat{\mathbf{y}}$ when the clean label is $\mathbf{y}$. Two specific examples we will use throughout the paper are the following:

- **(Normalized) Hamming loss $\mathbf{L}^{\text{Ham}}$:** $\quad \ell^{\text{Ham}}_{\mathbf{y},\widehat{\mathbf{y}}} = \dfrac{1}{s}\displaystyle\sum_{j=1}^{s} \mathbf{1}(\widehat{y}_j \neq y_j)\,, \qquad (1)$
- **F_1-measure $\mathbf{L}^{F_1}$ (specified as a loss [5, 49]):** $\quad \ell^{F_1}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - \dfrac{2\sum_{j=1}^{s} y_j \widehat{y}_j}{\|\mathbf{y}\|_1 + \|\widehat{\mathbf{y}}\|_1}$, where we take $\dfrac{0}{0} = 1\,. \qquad (2)$

L-generalization error, L-regret, and Bayes consistency. Given a multi-label loss matrix $\mathbf{L}$, the $\mathbf{L}$-*generalization* error of a multi-label classifier $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$ under the clean distribution D is defined as $\text{er}^{\mathbf{L}}_D[\mathbf{h}] = \mathbf{E}_{(X,\mathbf{Y}) \sim D}[\ell_{\mathbf{Y},\mathbf{h}(X)}]$, its $\mathbf{L}$-*regret* is defined as $\text{regret}^{\mathbf{L}}_D[\mathbf{h}] = \text{er}^{\mathbf{L}}_D[\mathbf{h}] - \inf_{\mathbf{h}' : \mathcal{X} \to \mathcal{Y}} \text{er}^{\mathbf{L}}_D[\mathbf{h}']$. We will say a noise-corrected algorithm that maps a noisy training sample $\widetilde{S}$ to a classifier $\widehat{\mathbf{h}}$ is *Bayes consistent* for $\mathbf{L}$ under D if $\text{regret}^{\mathbf{L}}_D[\widehat{\mathbf{h}}] \xrightarrow{p} 0$ as the noisy sample size $m \to \infty$.

Multi-label class probability functions. We will denote by $\boldsymbol{\eta}, \widetilde{\boldsymbol{\eta}} : \mathcal{X} \to \Delta_{|\mathcal{Y}|}$ the multi-label class probability functions associated with the clean distribution D and the noisy distribution $\widetilde{D}$, respectively, defined as $\eta_{\mathbf{y}}(x) = \mathbf{P}(\mathbf{Y} = \mathbf{y} | X = x)$ and $\widetilde{\eta}_{\mathbf{y}}(x) = \mathbf{P}(\widetilde{\mathbf{Y}} = \mathbf{y} | X = x)$.

Binary and multiclass class probability estimation (CPE), and logistic losses. Our multi-label algorithms will involve solving various binary and multiclass CPE sub-problems, which in turn involve estimating the class probability functions associated with the corresponding binary/multiclass problems. For binary CPE problems, we will use the binary logistic loss $\phi_{\log} : \{0,1\} \times \mathbb{R} \to \mathbb{R}_+$ and associated inverse link function $\gamma^{-1}_{\log} : \mathbb{R} \to [0,1]$ defined by $\phi_{\log}(y,u) = \ln(1 + e^{-(2y-1)u})$ and $\gamma^{-1}_{\log}(u) = \frac{1}{1+\exp(-u)}$, respectively; similarly for n-class CPE problems, we will use the multiclass logistic loss $\phi_{\text{mlog}} : [n] \times \mathbb{R}^{n-1} \to \mathbb{R}_+$ and associated inverse link function $\boldsymbol{\gamma}^{-1}_{\text{mlog}} : \mathbb{R}^{n-1} \to \Delta_n$

defined by $\phi_{\text{mlog}}(y, \mathbf{u}) = -\ln\big(\frac{\exp(u_y)}{1+\sum_{i=1}^{n-1}\exp(u_i)}\big)$ if $y \in [n-1]$ and $\ln\big(1+\sum_{i=1}^{n-1}\exp(u_i)\big)$ if $y = n$, and $(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\mathbf{u}))_y = \frac{\exp(u_y)}{1+\sum_{i=1}^{n-1}\exp(u_i)}$ if $y \in [n-1]$ and $\frac{1}{1+\sum_{i=1}^{n-1}\exp(u_i)}$ if $y = n$, respectively.

3 Key ideas and intuition

Bayes optimal classifier for L and 'Bayes-sufficient' statistics $\mathbf{q}(x)$. Given a multi-label loss matrix $\mathbf{L}$, the Bayes optimal classifier for $\mathbf{L}$ under the clean distribution D (i.e., the classifier with smallest $\mathbf{L}$-generalization error under D) is given by

$$\mathbf{h}^*(x) \in \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \boldsymbol{\eta}(x)^\top \boldsymbol{\ell}_{\widehat{\mathbf{y}}} \,.$$

Our goal will be to construct an approximation to $\mathbf{h}^*$ from $\widetilde{S}$. There are two main challenges: (1) for multi-label problems, $\boldsymbol{\eta}(x) \in \Delta_{|\mathcal{Y}|}$ is potentially a very large vector; (2) we have access to only the noisy training sample $\widetilde{S}$. In order to overcome these challenges, the key ideas in all our algorithms will be to (1) identify a small set of statistics $\mathbf{q}(x)$ of the class probability vector $\boldsymbol{\eta}(x)$ – which we will refer to as 'Bayes-sufficient' statistics for $\mathbf{L}$ – that suffice to construct the Bayes optimal classifier for $\mathbf{L}$; and (2) estimate the statistics $\mathbf{q}(x)$ reliably from the given noisy training sample $\widetilde{S}$.

4 Algorithms

4.1 Hamming loss under IFN: NCPLUG algorithm

Let us start with the simplest case: Hamming loss under the independent flipping noise (IFN) model. This is also the main setting for which consistent noise-corrected algorithms have previously been developed [16, 44]. Under this setting, the loss and noise model both involve independent components for the s tags, and the problem reduces to solving s independent binary noisy label problems, one for each tag; indeed, the CCMN algorithm of [44] essentially solves each of these binary problems using the binary unbiased estimator method of [23]. Our *Noise-Corrected Plug-in* (NCPLUG) algorithm will also solve s independent binary problems, but will do so in a way that illustrates in this simple setting the essence of the approach that we will build on for more complex settings later.

Under the Hamming loss, the Bayes optimal classifier requires only the s Bayes-sufficient statistics

$$q_j(x) = \mathbf{P}(Y_j = 1|x) \quad \forall j \in [s] \,.$$

Indeed, the Bayes optimal classifier for $\mathbf{L}^{\text{Ham}}$ can be written as

$$h_j^*(x) = \mathbf{1}(q_j(x) \geq \tfrac{1}{2}) \quad \forall j \in [s] \,.$$

The key idea then is to estimate the statistics $q_j(x)$, associated with the clean distribution D, reliably from the noisy training sample $\widetilde{S}$. For this, we use a very simple approach. In particular, we first apply a standard binary CPE learner to $\widetilde{S}$ to obtain estimates of the statistics $q'_j(x) = \mathbf{P}(\widetilde{Y}_j = 1|x)$ associated with the noisy distribution $\widetilde{D}$. Next, under the IFN model, assuming $c_{0,1}^{(j)} + c_{1,0}^{(j)} < 1$, we have $q_j(x)$ and $q'_j(x)$ are related by $q'_j(x) = (1 - c_{1,0}^{(j)}) \cdot q_j(x) + c_{0,1}^{(j)} \cdot (1 - q_j(x))$, or equivalently $q_j(x) = \frac{q'_j(x) - c_{0,1}^{(j)}}{1 - c_{0,1}^{(j)} - c_{1,0}^{(j)}}$, $\forall j \in [s]$. Therefore, given $\widehat{q}'_j(x)$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCPLUG algorithm is given by

$$\widehat{h}_j(x) = \mathbf{1}\Big(\frac{\widehat{q}'_j(x) - c_{0,1}^{(j)}}{1 - c_{0,1}^{(j)} - c_{1,0}^{(j)}} \geq \tfrac{1}{2}\Big) \quad \forall j \in [s] \,.$$

Estimating $\mathbf{q}'(x)$. Our implementation of NCPLUG uses a binary logistic loss minimizer for the CPE learner. In particular, we first learn a vector of s real-valued functions $\widehat{\mathbf{f}} : \mathcal{X} \to \mathbb{R}^s$ by minimizing the s-dimensional convex surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^s \to \mathbb{R}_+$ defined as $\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^s \phi_{\log}(y_j, u_j)$ over the noisy training sample $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of real-valued vector functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X} \to \mathbb{R}^s\}$. The estimated statistics are then given by $\widehat{q}'_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$. Detailed pseudocode is in Appendix A.

4.2 F_1-measure under general CCN: NCEFP algorithm

Next, we consider multi-label learning with F_1-measure under general class-conditional noise (CCN). In this section, we will build on the Exact F-measure Plug-in (EFP) algorithm of [5], which is consistent for multi-label F_1-measure in the non-noisy setting. We will develop a noise-corrected version of this algorithm that we will call the *Noise-Corrected Exact F-measure Plug-in* (NCEFP) algorithm. Again, our approach will be to reliably estimate suitable Bayes-sufficient statistics $\mathbf{q}(x)$ associated with the clean distribution D from the noisy training sample.

As shown in [5], for the multi-label F_1-measure, the following s^2+1 statistics are Bayes-sufficient:

$$q_0(x) = \mathbf{P}(\|\mathbf{Y}\|_1 = 0|x)\,; \qquad q_{jk}(x) = \mathbf{P}(Y_j = 1, \|\mathbf{Y}\|_1 = k|x) \quad \forall j,k \in [s]\,.$$

In particular, under the F_1-measure, the Bayes optimal classifier is given by

$$\mathbf{h}^*(x) \in \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{ 1 - q_0(x)\cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^{s}\sum_{k=1}^{s} q_{jk}(x)\cdot \frac{2\cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \Big\}\,.$$

For the standard (non-noisy) setting, Dembczynski et al. [5] showed how to estimate the s^2 statistics $\{q_{jk}(x) : j,k \in [s]\}$ by solving s multiclass ($(s+1)$-class) CPE problems, and statistic $q_0(x)$ by solving a binary CPE problem. Here we develop noise-corrected versions of these procedures for estimating these statistics from the noisy training sample.

Let us first define a matrix $\mathbf{A} \in [0,1]^{(s^2+1)\times|\mathcal{Y}|}$ as follows:

$$a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)\,; \qquad a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot y_j \quad \forall j,k \in [s]\,.$$

Then it can be seen that the Bayes-sufficient statistics above can be written as $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x)$. Under the general CCN model, the clean class probability function $\boldsymbol{\eta}(x)$ is related to the noisy class probability function $\widetilde{\boldsymbol{\eta}}(x)$ via $\widetilde{\boldsymbol{\eta}}(x) = \mathbf{C}^\top\boldsymbol{\eta}(x)$. Therefore, if $\mathbf{C}$ is invertible, then the desired statistics $\mathbf{q}(x)$ can be written in terms of $\widetilde{\boldsymbol{\eta}}(x)$ as $\mathbf{q}(x) = \mathbf{A}(\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x) = \widetilde{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)$, where $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$. Now unlike the standard (non-noisy) setting, where the statistics $\mathbf{q}(x)$ expressed directly in terms of $\boldsymbol{\eta}(x)$ naturally decomposed into a set of multiclass (and one binary) CPE problems, these statistics expressed in terms of $\widetilde{\boldsymbol{\eta}}(x)$ no longer naturally decompose this way. Nevertheless, we will show how to estimate these statistics from the noisy training sample by solving s suitably weighted multiclass CPE problems together with a suitably weighted binary CPE problem. To do so, we will use a shifted and scaled matrix $\widetilde{\mathbf{A}}'$ to estimate related statistics $\mathbf{q}'(x)$ and then factor back in the scaling and shifting when using the estimated statistics to make a final prediction.[1] Towards this, define $\widetilde{a}_{\min} = \min(\min_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \min_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$ and $\widetilde{a}_{\max} = \max(\max_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \max_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$, and let the entries of $\widetilde{\mathbf{A}}' \in [0,1]^{(s^2+1)\times|\mathcal{Y}|}$ be defined as

$$\widetilde{a}'_{0,\mathbf{y}} = \frac{\widetilde{a}_{0,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1]\,; \qquad \widetilde{a}'_{jk,\mathbf{y}} = \frac{\widetilde{a}_{jk,\mathbf{y}} - \widetilde{a}_{\min}}{s\cdot(\widetilde{a}_{\max} - \widetilde{a}_{\min})} \in [0,1] \quad \forall j,k \in [s]\,.$$

It can be verified that $\sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}} \le 1$ for all $j \in [s], \mathbf{y} \in \mathcal{Y}$. Next, define $\mathbf{q}'(x) = \widetilde{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x)$. Then, for each $j \in [s]$, we set up a weighted multiclass ($(s+1)$-class) CPE problem with weights $(\widetilde{a}'_{j1,\mathbf{y}}, ..., \widetilde{a}'_{js,\mathbf{y}}, (1 - \sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}}))$ to estimate the statistics $q'_{j1}(x), ..., q'_{js}(x)$, and a weighted binary CPE problem with weights $(\widetilde{a}'_{0,\mathbf{y}}, (1 - \widetilde{a}'_{0,\mathbf{y}}))$ to estimate $q'_0(x)$. Finally, given $\widehat{\mathbf{q}}'(x)$ estimated in this way from the noisy training sample, our NCEFP algorithm outputs the multi-label classifier[2]

$$\begin{aligned}\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{ & 1 - [(\widetilde{a}_{\max} - \widetilde{a}_{\min})\cdot \widehat{q}'_0(x) + \widetilde{a}_{\min}]\cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) \\ & - \sum_{j=1}^{s}\sum_{k=1}^{s} [s\cdot(\widetilde{a}_{\max} - \widetilde{a}_{\min})\cdot \widehat{q}'_{jk}(x) + \widetilde{a}_{\min}]\cdot \frac{2\cdot\widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \Big\}\,.\end{aligned}$$

Estimating $\mathbf{q}'(x)$. Our implementation of NCEFP uses weighted multiclass and binary logistic loss minimizers for the weighted CPE learners. In particular, we first learn a vector of s^2+1 real-valued functions $\widehat{\mathbf{f}} : \mathcal{X} \to \mathbb{R}^{s^2+1}$ by minimizing the (s^2+1)-dimensional convex surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^{s^2+1} \to \mathbb{R}_+$ defined as $\psi(\mathbf{y}, \mathbf{u}) = \widetilde{a}'_{0,\mathbf{y}} \cdot \phi_{\text{log}}(1, u_0) + (1 - \widetilde{a}'_{0,\mathbf{y}}) \cdot \phi_{\text{log}}(0, u_0) + \sum_{j=1}^{s} \big[\sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}} \phi_{\text{mlog}}(k, (u_{j1}, ..., u_{js})) + (1 - \sum_{k=1}^{s} \widetilde{a}'_{jk,\mathbf{y}}) \phi_{\text{mlog}}(s+1, (u_{j1}, ..., u_{js})) \big]$ over the noisy sample $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of real-valued vector functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X} \to \mathbb{R}^{s^2+1}\}$. The estimated statistics are then given by $\widehat{q}'_0(x) = \gamma^{-1}_{\text{log}}(\widehat{f}_0(x))$ and $\widehat{q}'_{jk}(x) = \big(\boldsymbol{\gamma}^{-1}_{\text{mlog}}(\widehat{f}_{j1}(x), ..., \widehat{f}_{js}(x))\big)_k$. Detailed pseudocode is in Appendix A.

[1]Entries of $\widetilde{\mathbf{A}}$ cannot be used directly as they may be negative and/or not add up to one.

[2]The combinatorial optimization problem involved in producing $\widehat{\mathbf{h}}(x)$ has a similar functional form as its non-noisy counterpart, and for $\mathcal{Y} = \{0,1\}^s$, it can be solved in order $O(s^3)$ time using a procedure of [6] (if the label vectors are sparse with at most K nonzero entries each, then the optimization can be solved in order $O(sK^2)$ time; a special case of this scenario is discussed in Section 6 and Appendix C).

4.3 General low-rank multi-label losses under general CCN: NCOC algorithm

We now consider multi-label learning with a general low-rank loss matrix (encompassing the Hamming loss and F_1-measure as special cases) under general class-conditional noise (CCN). In this section, we will build on the Output Coding (OC) algorithm of [49] which was developed for the multi-label F_1-measure in the standard (non-noisy) setting and was shown to be consistent for that setting. The approach applies more broadly to low-rank loss matrices in general, and we will develop a noise-corrected version of this algorithm for the general setting that we will call the *Noise-Corrected Output Coding* (NCOC) algorithm. Again, our approach will be to reliably estimate suitable Bayes-sufficient statistics $\mathbf{q}(x)$ associated with the clean distribution D from the noisy training sample. In the special cases of Hamming loss and F_1-measure, these statistics are the same as those discussed in Section 4.1 and Section 4.2, but the estimation procedures will be different.

Output coding is a general term that refers to the solution of multiclass or multi-label problems by decomposing them into a set of binary prediction problems [7, 2, 28]. The OC algorithm of [49] breaks down a multi-label prediction problem with a low-rank loss into a small number of weighted binary CPE problems. In particular, consider a multi-label loss matrix $\mathbf{L}$ that can be written as $\mathbf{L} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ for some $\mathbf{A} \in [0,1]^{r\times|\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r\times|\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ (so that $\text{rank}(\mathbf{L}) \le r+1$). Then it turns out that the r-dimensional vector statistic $\mathbf{q}(x)$ defined as

$$\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x)$$

is Bayes-sufficient for $\mathbf{L}$. Indeed, the Bayes optimal classifier for $\mathbf{L}$ can be written as

$$\mathbf{h}^*(x) \in \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \boldsymbol{\ell}_{\widehat{\mathbf{y}}}^\top \boldsymbol{\eta}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \mathbf{b}_{\widehat{\mathbf{y}}}^\top (\mathbf{A}\boldsymbol{\eta}(x)) + t_{\widehat{\mathbf{y}}} = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \mathbf{b}_{\widehat{\mathbf{y}}}^\top \mathbf{q}(x) + t_{\widehat{\mathbf{y}}}\,.$$

In the standard (non-noisy) setting, the OC algorithm of [49] estimates these statistics $\mathbf{q}(x)$ by decomposing the multi-label problem into r weighted binary CPE problems. Again, we will develop noise-corrected versions of these procedures to estimate the statistics from the noisy training sample.

As before, under the general CCN model, we have $\widetilde{\boldsymbol{\eta}}(x) = \mathbf{C}^\top \boldsymbol{\eta}(x)$. Therefore, if $\mathbf{C}$ is invertible, then $\mathbf{q}(x)$ can be written in terms of $\widetilde{\boldsymbol{\eta}}(x)$ as $\mathbf{q}(x) = \widetilde{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)$, where $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$. Again, in order to set up suitably weighted binary CPE problems that can allow us to estimate these statistics from the noisy training sample, we will use a shifted and scaled matrix $\widetilde{\mathbf{A}}'$ to estimate related statistics $\mathbf{q}'(x)$, and then factor back in the scaling and shifting when making a final prediction. Towards this, define $\widetilde{a}_{\min} = \min_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$ and $\widetilde{a}_{\max} = \max_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$, and define the entries of $\widetilde{\mathbf{A}}' \in [0,1]^{r\times|\mathcal{Y}|}$ as

$$\widetilde{a}'_{j,\mathbf{y}} = \frac{\widetilde{a}_{j,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1] \quad \forall j \in [r]\,.$$

We note that scaling for the terms $\widetilde{a}'_{j,\mathbf{y}}$ is different from that used for the NCEFP algorithm in Section 4.2, as now we are decomposing the problem into r binary problems rather than multiclass. Next, define $\mathbf{q}'(x) = \widetilde{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x)$. Then, for each $j \in [r]$, we set up a weighted binary CPE problem with weights $(\widetilde{a}'_{j,\mathbf{y}}, (1-\widetilde{a}'_{j,\mathbf{y}}))$ to estimate $q'_j(x)$. Finally, given estimated statistics $\widehat{\mathbf{q}}'(x)$ estimated in this way from the noisy training sample, our NCOC algorithm outputs the multi-label classifier

$$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{ t_{\widehat{\mathbf{y}}} + \sum_{j=1}^{r} [(\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \widehat{q}'_j(x) + \widetilde{a}_{\min}] \cdot b_{j,\widehat{\mathbf{y}}} \Big\}\,.$$

Estimating $\mathbf{q}'(x)$. Our implementation of NCOC uses weighted binary logistic loss minimizers for the weighted CPE learners. In particular, we first learn a vector of r real-valued functions $\widehat{\mathbf{f}} : \mathcal{X}\to\mathbb{R}^r$ by minimizing the r-dimensional convex surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^r \to \mathbb{R}_+$ defined as $\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^r \big(\widetilde{a}'_{j,\mathbf{y}}\phi_{\log}(1, u_j) + (1 - \widetilde{a}'_{j,\mathbf{y}})\phi_{\log}(0, u_j)\big)$ over $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X}\to\mathbb{R}^r\}$. The estimated statistics are then given by $\widehat{q}'_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$. Detailed pseudocode is in Appendix A.

The above approach can be applied to any low-rank loss matrix that can be written in the form described above, including both Hamming loss and F_1-measure as discussed below.

Example 1 (Low-rank decomposition for $\mathbf{L}^{\text{Ham}}$). *The Hamming loss in Eq.* (1) *can be written as*

$$\ell^{\text{Ham}}_{\mathbf{y},\widehat{\mathbf{y}}} = \frac{1}{s}\sum_{j=1}^{s} \mathbf{1}(\widehat{y}_j \neq y_j) = \sum_{j=1}^{s} \frac{1-2\widehat{y}_j}{s} y_j + \sum_{j=1}^{s} \frac{\widehat{y}_j}{s}\,. \tag{3}$$

In other words, we have $\mathbf{L}^{\text{Ham}} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ *where* $\mathbf{A} \in [0,1]^{s \times |\mathcal{Y}|}$ *with* $a_{j,\mathbf{y}} = y_j$, $\mathbf{B} \in \mathbb{R}^{s \times |\mathcal{Y}|}$ *with* $b_{j,\widehat{\mathbf{y}}} = \frac{1-2\widehat{y}_j}{s}$, *and* $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ *with* $t_{\widehat{\mathbf{y}}} = \sum_{j=1}^s \frac{\widehat{y}_j}{s} = \frac{1}{s}\|\widehat{\mathbf{y}}\|_1$. *Thus, for Hamming loss,* $r = s$ *and the statistics* $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x) \in [0,1]^s$ *turn out to be the same as in Section 4.1.*

Example 2 (**Low-rank decomposition for** $\mathbf{L}^{F_1}$). *The* F_1*-measure loss in Eq.* (2) *can be written as*

$$\ell^{F_1}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - \mathbf{1}(\|\mathbf{y}\|_1 = 0) \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^{s}\sum_{k=1}^{s} \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \cdot \frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} . \quad (4)$$

In other words, we have $\mathbf{L}^{F_1} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ *where* $\mathbf{A} \in [0,1]^{(s^2+1) \times |\mathcal{Y}|}$ *with* $a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)$ *and* $a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j$, $\mathbf{B} \in \mathbb{R}^{(s^2+1)\times|\mathcal{Y}|}$ *with* $b_{0,\widehat{\mathbf{y}}} = -\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0)$ *and* $b_{jk,\widehat{\mathbf{y}}} = -\frac{2\cdot\widehat{y}_j}{k+\|\widehat{\mathbf{y}}\|_1}$, *and* $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ *with* $t_{\widehat{\mathbf{y}}} = 1$. *Thus, for* F_1*-measure,* $r = s^2 + 1$ *and the statistics* $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x) \in [0,1]^{s^2+1}$ *turn out to be the same as in Section 4.2.*

Remark on computation for NCEFP and NCOC, and fast NCOC-Ham-IFN algorithm. We note that the NCEFP and NCOC algorithms above both require storing the noise matrix $\mathbf{C}$ and computing $(\mathbf{C}^\top)^{-1}$. For a general multi-label noise matrix $\mathbf{C}$, this can be prohibitively expensive. Therefore, these algorithms are practical when either the number of tags s is small or the noise matrix $\mathbf{C}$ is suitably structured. We describe one such structure, namely the STSN model, in Section 6, that enables fast computation. We also note that under the previously well-studied IFN model, noise matrices $\mathbf{C}$ – even though relatively simple with few parameters – are (to our knowledge) expensive to invert. For the special case of Hamming loss under IFN, in Appendix A.4, we present an alternative faster noise-corrected output coding algorithm – that we call NCOC-Ham-IFN – that decomposes the problem of estimating statistics $\mathbf{q}(x)$ into a different set of s binary CPE problems obtained using a different coding matrix $\widetilde{\mathbf{A}}''$ that does not require inverting $\mathbf{C}^\top$.

5 Regret transfer bounds and consistency

Below we provide quantitative regret transfer bounds for each of the three algorithms above that upper bound the target $\mathbf{L}$-regret of the learned classifier $\widehat{\mathbf{h}}$ under the clean distribution D in terms of the surrogate ψ*-regret* (defined below) of the associated real-valued vector function $\widehat{\mathbf{f}}$ obtained by minimizing the corresponding convex surrogate loss ψ (defined for each algorithm in the corresponding section above) under the noisy distribution $\widetilde{D}$. In each case, if the surrogate loss ψ is minimized over a suitably rich function class, then ψ-regret under $\widetilde{D}$ converges in probability to 0 as $m \to \infty$. Therefore, this also implies Bayes consistency of these algorithms for the target loss $\mathbf{L}$ under D.

ψ**-generalization error and** ψ**-regret.** For any positive integer r, an r-dimensional surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^r \to \mathbb{R}_+$ and vector-valued function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^r$, define the ψ*-generalization error* of $\mathbf{f}$ under the noisy distribution $\widetilde{D}$ as $\text{er}^\psi_{\widetilde{D}}[\mathbf{f}] = \mathbf{E}_{(X,\widetilde{\mathbf{Y}})\sim\widetilde{D}}[\psi(\widetilde{\mathbf{Y}}, \mathbf{f}(X))]$, and its ψ-regret of under $\widetilde{D}$ as $\text{regret}^\psi_{\widetilde{D}}[\mathbf{f}] = \text{er}^\psi_{\widetilde{D}}[\mathbf{f}] - \inf_{\mathbf{f}':\mathcal{X}\to\mathbb{R}^r} \text{er}^\psi_{\widetilde{D}}[\mathbf{f}']$.

Theorem 1 (**Regret bound for NCPLUG**). *Consider Hamming loss* $\mathbf{L}^{\text{Ham}}$ *(Eq.* (1)*) under IFN model. Assume* $c^{(j)}_{0,1} + c^{(j)}_{1,0} < 1$ *for all* $j \in [s]$. *Let* D *be any distribution on* $\mathcal{X} \times \mathcal{Y}$ *with corresponding noisy distribution* $\widetilde{D}$. *Suppose NCPLUG (Section 4.1) is run with noisy training sample* $\widetilde{S}$ *(in which examples are sampled i.i.d. from* $\widetilde{D}$*), and let* $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ *be as defined in Section 4.1. Then we have*

$$\text{regret}^{\mathbf{L}^{\text{Ham}}}_D[\widehat{\mathbf{h}}] \le \frac{1}{\sqrt{s}} \max_i \frac{1}{1 - c^{(i)}_{0,1} - c^{(i)}_{1,0}} \sqrt{2\text{regret}^\psi_{\widetilde{D}}[\widehat{\mathbf{f}}]} .$$

Theorem 2 (**Regret bound for NCEFP**). *Consider* F*-measure* $\mathbf{L}^{F_1}$ *(Eq.* (2)*) under the general CCN model. Assume noise matrix* $\mathbf{C}$ *is invertible. Let* D *be any distribution on* $\mathcal{X} \times \mathcal{Y}$ *with corresponding noisy distribution* $\widetilde{D}$. *Suppose NCEFP (Section 4.2) is run with noisy training sample* $\widetilde{S}$ *(in which examples are sampled i.i.d. from* $\widetilde{D}$*). Let* $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ *be as defined in Section 4.2, and let* $\mathbf{A} \in [0,1]^{(s^2+1)\times|\mathcal{Y}|}$, $\mathbf{B} \in \mathbb{R}^{(s^2+1)\times|\mathcal{Y}|}$, *and* $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ *be as defined in Example 2. Then we have*

$$\text{regret}^{\mathbf{L}^{F_1}}_D[\widehat{\mathbf{h}}] \le 4s \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \|\mathbf{A}\|_1 \|(\mathbf{C}^\top)^{-1}\|_1 \sqrt{2\text{regret}^\psi_{\widetilde{D}}[\widehat{\mathbf{f}}]} .$$

Theorem 3 (**Regret bound for NCOC**). *Consider a general low-rank loss matrix* $\mathbf{L}$ *written as* $\mathbf{L} = \mathbf{A}^\top\mathbf{B} + \mathbf{1}\mathbf{t}^\top$ *for some* $\mathbf{A} \in [0,1]^{r\times|\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r\times|\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$, *under the general CCN model. Assume noise matrix* $\mathbf{C}$ *is invertible. Let* D *be any distribution on* $\mathcal{X} \times \mathcal{Y}$ *with corresponding noisy distribution* $\widetilde{D}$. *Suppose NCOC (Section 4.3) is run with noisy training sample* $\widetilde{S}$ *(in which examples*

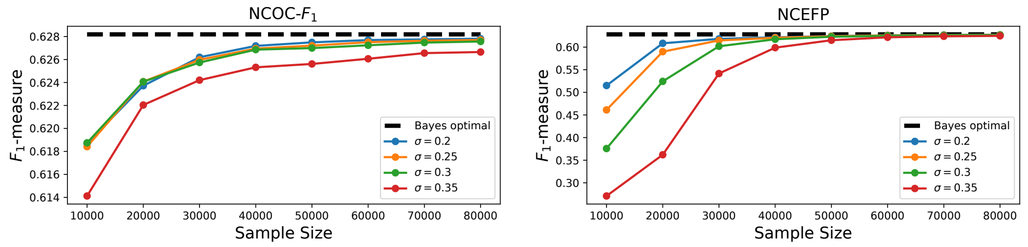


Figure 2: Sample complexity behavior of NCOC-F_1 and NCEFP on synthetic multi-label data for 4 noise parameters under the STSN model. Performance measure is F_1-measure (specified as a gain).

are sampled i.i.d. from $\widetilde{D}$), and let $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ be as defined in Section 4.3. Then we have

$$\text{regret}_D^{\mathbf{L}}[\widehat{\mathbf{h}}] \leq 2 \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \|\mathbf{A}\|_1 \|(\mathbf{C}^\top)^{-1}\|_1 \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]}\,.$$

The quantity $\|(\mathbf{C}^\top)^{-1}\|_1$ can be viewed as capturing the amount of label noise in $\mathbf{C}$. The bounds therefore suggest that as the amount of label noise increases (larger $\|(\mathbf{C}^\top)^{-1}\|_1$), the sample size needed to reach a given level of performance generally increases.

In Appendix B, we provide proofs of more general versions of the above theorems that allow one to use an estimated noise matrix $\widehat{\mathbf{C}}$ when the true noise matrix $\mathbf{C}$ may be unknown.

6 Similar-Tag Switching Noise (STSN) model

As noted earlier, general CCN models can require too many (up to order $O(4^s)$) parameters and make computation prohibitive. Here we propose a new family of structured multi-label noise models that we term *Similar-Tag Switching Noise* (STSN) models; STSN models are a special case of CCN that require fewer parameters and enable fast computation, and moreover, unlike IFN, they also capture some correlations among tags. The idea behind STSN models is that tags are partitioned into several groups, each of which contains similar/related tags, and independently within each group, an active tag can be switched with another tag in the group (i.e., similar tags can be switched with each other).

Specifically, let the set of s tags $\mathcal{T} = [s]$ be partitioned into K ($K \leq s$) groups of tags $G_1, ..., G_K$, such that the tags within any group are similar/related to each other (for example, G_1 could contain tags `lion`, `tiger`; G_2 could contain tags `river`, `lake`; etc.). We will assume that within each group, at most one tag is active in any label vector $\mathbf{y}$; this gives $|\mathcal{Y}| = \prod_{k=1}^{K}(1 + |G_k|) \ll 2^s$, and $\|\mathbf{y}\|_1 \leq K$ for all $\mathbf{y} \in \mathcal{Y}$ (indeed, this is in line with many real multi-label datasets in which labels are very sparse). The STSN model involves K noise parameters: $\sigma_k \in [0, 1]$ for $k \in [K]$. Specifically, for a label $\mathbf{y} \in \mathcal{Y} \subseteq \{0, 1\}^s$, let $\mathbf{y}_{G_k} \in \{0, 1\}^{|G_k|}$ denote the sub-label restricted to tags in G_k. Recall that $\|\mathbf{y}_{G_k}\| \leq 1$. For groups G_k with $|G_k| \geq 2$, the noise process within G_k is as follows: if $\mathbf{y}_{G_k} = \mathbf{0}$, then $\widetilde{\mathbf{y}}_{G_k} = \mathbf{0}$; if $\mathbf{y}_{G_k} \neq \mathbf{0}$, then with probability σ_k, $\mathbf{y}_{G_k}$ is changed to $\widetilde{\mathbf{y}}_{G_k}$ by switching its (only) active tag with one of the remaining $|G_k| - 1$ non-active tags chosen uniformly at random, and with probability $1 - \sigma_k$, $\widetilde{\mathbf{y}}_{G_k}$ is the same as $\mathbf{y}_{G_k}$. For groups G_k with $|G_k| = 1$, $\mathbf{y}_{G_k}$ is flipped to the opposite with probability σ_k. The above noise process is applied independently to each group. (For the special case when there are $K = s$ groups each of size one, the STSN model reduces to the symmetric IFN model studied by [16], but in the general case, it can capture much richer structure.) Appendix C summarizes various small changes/simplifications to our algorithms under STSN. We also include a way to estimate STSN parameters under the anchor point assumption [19, 27, 45, 48].

7 Experiments

7.1 Synthetic data: sample complexity behavior

We tested the sample complexity behavior of our algorithms. Here we report results for the F_1-measure under the STSN model; results for Hamming loss under IFN are in Appendix D.

F_1-measure under STSN. We generated a multi-label dataset with instances x in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 10$ tags partitioned into $K = 5$ groups $\mathcal{G} = \{\{1, 2, 3\}, \{4, 5, 6\}, \{7, 8\}, \{9\}, \{10\}\}$, so $|\mathcal{Y}| = 192$ and $\|\mathbf{y}\|_1 \leq K = 5\ \forall \mathbf{y} \in \mathcal{Y}$ (details in Appendix D). We then added label noise using 4 single-parameter STSN noise matrices respecting this partition: specifically, we let $\sigma_1, ..., \sigma_5 = \sigma$, and chose 4 values of σ: $0.2, 0.25, 0.3, 0.35$. We ran NCOC-F_1 and NCEFP (with a linear function class) to learn multi-label classifiers from increasingly large noisy training samples generated in this way, and measured the F_1-measure on a test set of $10,000$ clean data points. The results are shown in

Figure 3: Examples of the selected 17 tags in Mediamill dataset. Pictures were taken from [31].

Table 1: Hamming loss on (modified) Mediamill data with IFN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter $(c_{0,1}, c_{1,0})$	**Noise level (%)**	**NCPLUG**	**NCOC-Ham-IFN**	**CCMN**	**OC-Ham/BR**
(0.1,0.2)	85.62	**7.73±0.0**	**7.73±0.0**	7.77±0.0	7.98±0.0
(0.15,0.4)	96.17	**7.8±0.0**	**7.8±0.0**	8.03±0.07	8.32±0.0
(0.25,0.45)	99.48	**7.9±0.0**	**7.9±0.0**	8.29±0.02	8.33±0.0

Table 2: Hamming loss on (modified) Mediamill data with STSN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCOC-Ham**	**OC-Ham/BR**
0.1	13.47	**7.66±0.0**	7.84±0.0
0.2	26.31	**7.68±0.0**	8.09±0.0
0.3	38.18	**7.68±0.0**	8.24±0.0
0.4	49.01	**7.65±0.0**	8.31±0.0
0.6	68.21	**7.75±0.0**	8.37±0.0

Figure 2. We see that, as suggested by our regret bounds, as noise parameter σ increases, the sample size needed to reach a given level of performance generally increases.

7.2 Real data: comparison with other methods

Next, we evaluated our algorithms on two real multi-label datasets: Mediamill and Multi-MNIST [31, 34]. Here we report results on the Mediamill data; results for Multi-MNIST are in Appendix D.

Mediamill dataset. The original Mediamill dataset has 30,993 training examples and 12,914 test examples with 101 tags, and the images have been processed into 120 features [31]. We selected a subset of 17 tags that contains naturally groupable tags, and divided them into 7 groups: { {`Duo-anchor`, `Anchor`}, {`People`, `People marching`, `People walking`}, {`Split screen`, `Screen`}, {`Sky`, `Cloud`}, {`Religious leader`, `Monologue`}, {`Court`, `Meeting`}, {`Tower`, `Government building`, `Urban`, `Building`} }. (See also Figure 3 for visual impressions.) For consistency with the STSN model assumption, we removed instances that have more than one active tag in any group; we also removed instances that do not have any active tag among the 17 tags. Our modified Mediamill dataset has 20,141 training examples and 7,737 test examples with 17 tags.

Hamming loss under IFN. For IFN, we let $c_{1,0}^{(j)} = c_{1,0}$ and $c_{0,1}^{(j)} = c_{0,1}$ for all j, and chose noise parameters of the form $(c_{0,1}, c_{1,0})$. We compared our NCPLUG and NCOC-Ham-IFN algorithms with CCMN [44] and basic OC-Ham/BR [46]. All algorithms were trained to learn linear models with regularization (details in Appendix D). The results are shown in Table 1. As seen, our NCPLUG and NCOC-Ham-IFN algorithms generally outperform other baselines.

Hamming loss and F_1-measure under STSN. For STSN, we used single-parameter noise matrices respecting the partition into $K = 7$ groups described above, with $\sigma_1, ..., \sigma_7 = \sigma$. We compared our NCOC-Ham algorithm with basic OC-Ham/BR [46], as well as our NCOC-F_1 and NCEFP algorithms with basic OC-F_1 [49] and EFP [5]. All algorithms were trained to learn linear models with regularization (details in Appendix D). The results are shown in Table 2 and Table 3. Again, our noise-corrected algorithms generally outperform other baselines.

8 Conclusion

We have developed three consistent noise-corrected multi-label learning algorithms (NCPLUG, NCEFP, and NCOC), encompassing a variety of multi-label performance measures and general class-conditional noise (CCN) models. We have provided quantitative regret transfer bounds for all three algorithms to establish their consistency. We have also proposed a new family of structured multi-label noise models that we term similar-tag switching noise (STSN) models; STSN models are a special case of CCN that require fewer parameters and enable fast computation, and moreover,

Table 3: F_1-measure on (modified) Mediamill data with STSN model (*higher* values are better). Performance values are in %, and are reported as Mean$\pm$SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCEFP**	**NCOC-F_1**	**EFP**	**OC-F_1**
0.1	13.47	42.29±0.03	**42.86±0.02**	41.84±0.03	41.76±0.04
0.2	26.31	42.22±0.02	**42.95±0.02**	41.42±0.01	41.44±0.03
0.3	38.18	41.36±0.03	**43.03±0.03**	41.01±0.02	40.96±0.03
0.4	49.01	39.04±0.09	**43.1±0.05**	40.81±0.03	40.71±0.03
0.6	68.21	31.81±0.03	**42.83±0.06**	6.57±0.03	6.57±0.03

unlike IFN, they also capture some correlations among tags. Our experiments have confirmed the effectiveness of our algorithms in correcting for multi-label noise. Future work includes developing ways to estimate STSN models from noisy data, and exploring the design of other structured noise models that could be suitable for multi-label settings.

References

[1] Shivani Agarwal. Surrogate regret bounds for bipartite ranking via strongly proper losses. *Journal of Machine Learning Research*, 15:1653–1674, 2014.

[2] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1:113–141, 2000.

[3] Junwen Bai, Shufeng Kong, and Carla P. Gomes. Disentangled variational autoencoder based multi-label classification with covariance-aware multivariate probit model. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4313–4321, 2020.

[4] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 279–286, 2010.

[5] Krzysztof Dembczynski, Arkadiusz Jachnik, Wojciech Kotlowski, Willem Waegeman, and Eyke Hüllermeier. Optimizing the F-measure in multi-label classification: Plug-in rule approach versus structured loss minimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1130–1138, 2013.

[6] Krzysztof Dembczynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. An exact algorithm for F-measure maximization. In *Advances in Neural Information Processing Systems 24*, pages 1404–1412, 2011.

[7] Thomas Dietterich and G Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[8] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

[9] Jun-Peng Fang and Min-Ling Zhang. Partial multi-label learning via credible label elicitation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 3518–3525. AAAI Press, 2019.

[10] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 25(5):845–869, 2014.

[11] Wei Gao and Zhi-Hua Zhou. On the consistency of multi-label learning. In *Conference on Learning Theory*, 2011.

[12] Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017*, pages 1919–1925. AAAI Press, 2017.

[13] Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, Ivor W. Tsang, James T. Kwok, and Masashi Sugiyama. A survey of label-noise representation learning: Past, present and future. *CoRR*, abs/2011.04406, 2020.

[14] Mengying Hu, Hu Han, Shiguang Shan, and Xilin Chen. Multi-label learning from noisy labels with non-linear feature transformation. In *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, 2018*, volume 11365 of *Lecture Notes in Computer Science*, pages 404–419. Springer, 2018.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015.

[16] Himanshu Kumar, Naresh Manwani, and P. S. Sastry. Robust learning of multi-label classifiers under label noise. In *CoDS-COMAD 2020: 7th ACM IKDD CoDS and 25th COMAD*, pages 90–97. ACM, 2020.

[17] Shikun Li, Xiaobo Xia, Hansong Zhang, Yibing Zhan, Shiming Ge, and Tongliang Liu. Estimating noise transition matrix with label correlations for noisy multi-label learning. In *Advances in Neural Information Processing Systems*, 2022.

[18] Xuefeng Li, Tongliang Liu, Bo Han, Gang Niu, and Masashi Sugiyama. Provably end-to-end label-noise learning without anchor points. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 6403–6413. PMLR, 2021.

[19] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(3):447–461, 2016.

[20] Yang Liu and Hongyi Guo. Peer loss functions: Learning from noisy labels without knowing noise rates. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119, pages 6226–6236. PMLR, 2020.

[21] Aditya Krishna Menon, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Multilabel reductions: what is my loss optimising? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 10599–10610, 2019.

[22] Aditya Krishna Menon, Brendan van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37, pages 125–134. JMLR.org, 2015.

[23] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 1196–1204, 2013.

[24] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Cost-sensitive learning with noisy labels. *J. Mach. Learn. Res.*, 18:155:1–155:33, 2017.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[26] Giorgio Patrini, Frank Nielsen, Richard Nock, and Marcello Carioni. Loss factorization, weakly supervised learning and label noise robustness. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 708–717. JMLR.org, 2016.

[27] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 2233–2241. IEEE Computer Society, 2017.

[28] Harish G. Ramaswamy, Balaji Srinivasan Babu, Shivani Agarwal, and Robert C. Williamson. On the consistency of output code based learning algorithms for multiclass learning problems. In *Proceedings of the 27th Conference on Learning Theory (COLT)*, pages 885–902, 2014.

[29] Clayton Scott. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015*, volume 38. JMLR.org, 2015.

[30] Clayton Scott, Gilles Blanchard, and Gregory Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In *COLT 2013 - The 26th Annual Conference on Learning Theory*, volume 30, pages 489–511. JMLR.org, 2013.

[31] Cees Snoek, Marcel Worring, Jan C. van Gemert, Jan-Mark Geusebroek, and Arnold W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM International Conference on Multimedia, 2006*, pages 421–430. ACM, 2006.

[32] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, 34(11):8135–8153, 2023.

[33] Lijuan Sun, Songhe Feng, Tao Wang, Congyan Lang, and Yi Jin. Partial multi-label learning by low-rank and sparse decomposition. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 5016–5023. AAAI Press, 2019.

[34] Shao-Hua Sun. Multi-digit mnist for few-shot learning. *GitHub repository, https://github.com/shaohua0116/MultiDigitMNIST*, 2019.

[35] Brendan van Rooyen and Robert C. Williamson. A theory of learning with corrupted labels. *J. Mach. Learn. Res.*, 18:228:1–228:50, 2017.

[36] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge J. Belongie. Learning from noisy large-scale datasets with minimal supervision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 6575–6583. IEEE Computer Society, 2017.

[37] Haobo Wang, Weiwei Liu, Yang Zhao, Chen Zhang, Tianlei Hu, and Gang Chen. Discriminative and correlative partial multi-label learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3691–3697. ijcai.org, 2019.

[38] Ruxin Wang, Tongliang Liu, and Dacheng Tao. Multiclass learning with partially corrupted labels. *IEEE Trans. Neural Networks Learn. Syst.*, 29(6):2568–2580, 2018.

[39] Guoqiang Wu, Chongxuan Li, Kun Xu, and Jun Zhu. Rethinking and reweighting the univariate losses for multi-label ranking: Consistency and generalization. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 14332–14344, 2021.

[40] Guoqiang Wu and Jun Zhu. Multi-label classification: do hamming loss and subset accuracy really conflict with each other? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[41] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 6835–6846, 2019.

[42] Ming-Kun Xie and Sheng-Jun Huang. Partial multi-label learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018*, pages 4302–4309. AAAI Press, 2018.

[43] Ming-Kun Xie and Sheng-Jun Huang. Partial multi-label learning with noisy label identification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 6454–6461. AAAI Press, 2020.

[44] Ming-Kun Xie and Sheng-Jun Huang. CCMN: A general framework for learning with class-conditional multi-label noise. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1):154–166, 2023.

[45] Yu Yao, Tongliang Liu, Bo Han, Mingming Gong, Jiankang Deng, Gang Niu, and Masashi Sugiyama. Dual T: reducing estimation error for transition matrix in label-noise learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[46] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.*, 26(8):1819–1837, 2014.

[47] Mingyuan Zhang and Shivani Agarwal. Multiclass learning from noisy labels for non-decomposable performance measures. In *International Conference on Artificial Intelligence and Statistics 2024, AISTATS 2024*, volume 238 of *Proceedings of Machine Learning Research*, pages 2170–2178. PMLR, 2024.

[48] Mingyuan Zhang, Jane Lee, and Shivani Agarwal. Learning from noisy labels with no change to the training process. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12468–12478. PMLR, 2021.

[49] Mingyuan Zhang, Harish Guruprasad Ramaswamy, and Shivani Agarwal. Convex calibrated surrogates for the multi-label f-measure. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 11246–11255. PMLR, 2020.

[50] Wenting Zhao and Carla P. Gomes. Evaluating multi-label classifiers with noisy labels. *CoRR*, abs/2102.08427, 2021.

Consistent Multi-Label Classification from Noisy Labels

Appendix

A Supplement to Section 4

A.1 Detailed pseudocode for NCPLUG algorithm

See Algorithm 1.

Algorithm 1 Noise-Corrected Plug-in (NCPLUG) for Hamming loss under IFN

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Noise rates $c^{(j)}_{0,1}, c^{(j)}_{1,0} \quad \forall j \in [s]$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^s$
3: Compute $\widehat{\mathbf{f}} \in \text{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is defined as
   $$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^s \phi_{\log}(y_j, u_j)$$
4: Output:
   Multi-label classifier
   $$\widehat{h}_j(x) = \mathbf{1}\Big(\frac{\gamma^{-1}_{\log}(\widehat{f}_j(x)) - c^{(j)}_{0,1}}{1 - c^{(j)}_{0,1} - c^{(j)}_{1,0}} \geq \frac{1}{2}\Big) \quad \forall j \in [s]$$
```

A.2 Detailed pseudocode for NCEFP algorithm

See Algorithm 2.

A.3 Detailed pseudocode for NCOC algorithm

See Algorithm 3.

A.4 NCOC-Ham-IFN algorithm

As noted in Section 4.3, under the IFN model, noise matrices $\mathbf{C}$ are (to our knowledge) computationally expensive to invert, which makes it difficult to run the NCOC algorithm for such noise matrices in practice. For the special case of Hamming loss under the IFN model, we present an alternative faster noise-corrected output coding algorithm – that we call NCOC-Ham-IFN – that decomposes the problem of estimating statistics $\mathbf{q}(x)$ into a different set of s binary CPE problems obtained using a different coding matrix $\widetilde{\mathbf{A}}''$ that does not require inverting $\mathbf{C}^\top$.

Recall from Example 1 that $\mathbf{L}^{\text{Ham}} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ where $\mathbf{A} \in [0,1]^{s \times |\mathcal{Y}|}$ with $a_{j,\mathbf{y}} = y_j$, $\mathbf{B} \in \mathbb{R}^{s \times |\mathcal{Y}|}$ with $b_{j,\widehat{\mathbf{y}}} = \frac{1-2\widehat{y}_j}{s}$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ with $t_{\widehat{\mathbf{y}}} = \sum_{j=1}^s \frac{\widehat{y}_j}{s} = \frac{1}{s}\|\widehat{\mathbf{y}}\|_1$. Recall also that the s-dimensional vector statistic $\mathbf{q}(x)$ defined as

$$\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x)$$

is Bayes-sufficient for $\mathbf{L}^{\text{Ham}}$. We will express the desired statistics $\mathbf{q}(x)$ in terms of $\widetilde{\boldsymbol{\eta}}(x)$ using a matrix $\widetilde{\widetilde{\mathbf{A}}}$ that does not require inverting $\mathbf{C}^\top$, and will then use a shifted and scaled version of this matrix to obtain $\widetilde{\mathbf{A}}''$.

We have that

$$q_j(x) = \mathbf{P}(Y_j = 1|x) \quad \forall j \in [s]\,.$$

Algorithm 2 Noise-Corrected Exact F-measure Plug-in (NCEFP) for F_1-measure

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Noise matrix $\mathbf{C} \in [0,1]^{|\mathcal{Y}| \times |\mathcal{Y}|}$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^{s^2+1}$
3: Let $\mathbf{A} \in [0,1]^{(s^2+1) \times |\mathcal{Y}|}$ be
   $$a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)\,; \qquad a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \quad \forall j, k \in [s]$$
4: Let $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$, and define $\widetilde{a}_{\min} = \min(\min_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \min_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$ and $\widetilde{a}_{\max} = \max(\max_{\mathbf{y}} \widetilde{a}_{0,\mathbf{y}}, \max_{\mathbf{y},jk} \widetilde{a}_{jk,\mathbf{y}})$
5: Construct $\widetilde{\mathbf{A}}' \in [0,1]^{(s^2+1) \times |\mathcal{Y}|}$ by shifting and scaling $\widetilde{\mathbf{A}}$ as
   $$\widetilde{a}'_{0,\mathbf{y}} = \frac{\widetilde{a}_{0,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1]\,; \qquad \widetilde{a}'_{jk,\mathbf{y}} = \frac{\widetilde{a}_{jk,\mathbf{y}} - \widetilde{a}_{\min}}{s \cdot (\widetilde{a}_{\max} - \widetilde{a}_{\min})} \in [0,1] \quad \forall j, k \in [s]$$
6: Compute $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is as defined as
   $$\psi(\mathbf{y}, \mathbf{u}) = \widetilde{a}'_{0,\mathbf{y}} \cdot \phi_{\log}(1, u_0) + (1 - \widetilde{a}'_{0,\mathbf{y}}) \cdot \phi_{\log}(0, u_0) + \sum_{j=1}^s \Big[ \sum_{k=1}^s \widetilde{a}'_{jk,\mathbf{y}} \phi_{\text{mlog}}(k, (u_{j1}, \ldots, u_{js})) + (1 - \sum_{k=1}^s \widetilde{a}'_{jk,\mathbf{y}}) \phi_{\text{mlog}}(s+1, (u_{j1}, \ldots, u_{js})) \Big]$$
7: Output:
   Multi-label classifier
   $$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} \Big\{ 1 - [(\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \widehat{q}'_0(x) + \widetilde{a}_{\min}] \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^s \sum_{k=1}^s [s \cdot (\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \widehat{q}'_{jk}(x) + \widetilde{a}_{\min}] \cdot \frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1} \Big\}$$
   where $\widehat{q}'_0(x) = \gamma_{\log}^{-1}(\widehat{f}_0(x))$ and $\widehat{q}'_{jk}(x) = \big(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\widehat{f}_{j1}(x), \ldots, \widehat{f}_{js}(x))\big)_k$
```

Algorithm 3 Noise-Corrected Output Coding (NCOC)

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Target loss $\mathbf{L} \in \mathbb{R}_+^{|\mathcal{Y}| \times |\mathcal{Y}|}$ factorized as $\mathbf{L} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ for some $\mathbf{A} \in [0,1]^{r \times |\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r \times |\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$
   (3) Noise matrix $\mathbf{C} \in [0,1]^{|\mathcal{Y}| \times |\mathcal{Y}|}$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^r$
3: Let $\widetilde{\mathbf{A}} = \mathbf{A}(\mathbf{C}^\top)^{-1}$, and define $\widetilde{a}_{\min} = \min_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$ and $\widetilde{a}_{\max} = \max_{\mathbf{y},j} \widetilde{a}_{j,\mathbf{y}}$
4: Construct $\widetilde{\mathbf{A}}' \in [0,1]^{r \times |\mathcal{Y}|}$ by shifting and scaling $\widetilde{\mathbf{A}}$ as
   $$\widetilde{a}'_{j,\mathbf{y}} = \frac{\widetilde{a}_{j,\mathbf{y}} - \widetilde{a}_{\min}}{\widetilde{a}_{\max} - \widetilde{a}_{\min}} \in [0,1] \quad \forall j \in [r]$$
5: Compute $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is defined as
   $$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^r \Big( \widetilde{a}'_{j,\mathbf{y}} \phi_{\log}(1, u_j) + (1 - \widetilde{a}'_{j,\mathbf{y}}) \phi_{\log}(0, u_j) \Big)$$
6: Output:
   Multi-label classifier
   $$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} \Big\{ t_{\widehat{\mathbf{y}}} + \sum_{j=1}^r [(\widetilde{a}_{\max} - \widetilde{a}_{\min}) \cdot \gamma_{\log}^{-1}(\widehat{f}_j(x)) + \widetilde{a}_{\min}] \cdot b_{j,\widehat{\mathbf{y}}} \Big\}$$
```

Algorithm 4 NCOC-Ham-IFN for Hamming loss under IFN

```
1: Inputs:
   (1) Noisy training sample, $\widetilde{S} = ((x_1, \widetilde{\mathbf{y}}_1), \ldots, (x_m, \widetilde{\mathbf{y}}_m)) \in (\mathcal{X} \times \mathcal{Y})^m$
   (2) Noise rates $c^{(j)}_{0,1}, c^{(j)}_{1,0} \quad \forall j \in [s]$
2: Parameters:
   (1) Class $\mathcal{F}$ of functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^s$
3: Let $\mathbf{c}_{0,1} = [c^{(1)}_{0,1}, ..., c^{(s)}_{0,1}]^\top \in [0,1)^s$, and define a diagonal matrix $\mathbf{\Lambda}$ of size $s \times s$ such that its
   $i$-th diagonal element is $\frac{1}{1-c^{(i)}_{0,1}-c^{(i)}_{1,0}}$
4: Let $\widetilde{\widetilde{\mathbf{A}}} = \mathbf{\Lambda}(\mathbf{A} - \mathbf{c}_{0,1}\mathbf{1}^\top)$, and define $\widetilde{\widetilde{a}}_{\min} = \min_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$ and $\widetilde{\widetilde{a}}_{\max} = \max_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$
5: Construct $\widetilde{\mathbf{A}}'' \in [0,1]^{s \times |\mathcal{Y}|}$ by shifting and scaling $\widetilde{\widetilde{\mathbf{A}}}$ as
   $$\widetilde{a}''_{j,\mathbf{y}} = \frac{\widetilde{\widetilde{a}}_{j,\mathbf{y}} - \widetilde{\widetilde{a}}_{\min}}{\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}} \in [0,1] \quad \forall j \in [s]\,.$$
6: Compute $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f} \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^m \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$, where $\psi$ is defined as
   $$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^s \Big( \widetilde{a}''_{j,\mathbf{y}} \phi_{\log}(1, u_j) + (1 - \widetilde{a}''_{j,\mathbf{y}}) \phi_{\log}(0, u_j) \Big)$$
7: Output:
   Multi-label classifier
   $$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} \Big\{ \frac{1}{s} \|\widehat{\mathbf{y}}\|_1 + \sum_{j=1}^s [(\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}) \cdot \gamma^{-1}_{\log}(\widehat{f}_j(x)) + \widetilde{\widetilde{a}}_{\min}] \cdot \frac{1 - 2\widehat{y}_j}{s} \Big\}$$
```

Next, as in Section 4.1, define $\mathbf{q}'(x) = \mathbf{A}\widetilde{\boldsymbol{\eta}}(x)$ so that

$$q'_j(x) = \mathbf{P}(\widetilde{Y}_j = 1|x) \quad \forall j \in [s]\,.$$

Under the IFN model where $c^{(j)}_{0,1} + c^{(j)}_{1,0} < 1 \; \forall j \in [s]$, $q_j(x)$ and $q'_j(x)$ are related by

$$q_j(x) = \frac{q'_j(x) - c^{(j)}_{0,1}}{1 - c^{(j)}_{0,1} - c^{(j)}_{1,0}} \quad \forall j \in [s]\,.$$

Let $\mathbf{c}_{0,1} = [c^{(1)}_{0,1}, ..., c^{(s)}_{0,1}]^\top \in [0,1)^s$. Define a diagonal matrix $\mathbf{\Lambda}$ of size $s \times s$ such that its i-th diagonal element is $\frac{1}{1-c^{(i)}_{0,1}-c^{(i)}_{1,0}}$. Then we can write

$$\mathbf{q}(x) = \mathbf{\Lambda}(\mathbf{q}'(x) - \mathbf{c}_{0,1})\,.$$

Now, define

$$\widetilde{\widetilde{\mathbf{A}}} = \mathbf{\Lambda}(\mathbf{A} - \mathbf{c}_{0,1}\mathbf{1}^\top)\,,$$

where $\mathbf{1}$ here is a $|\mathcal{Y}| \times 1$ vector. Then $\widetilde{\widetilde{\mathbf{A}}}$ also has size $s \times |\mathcal{Y}|$, and we have

$$\begin{aligned}
\widetilde{\widetilde{\mathbf{A}}}\widetilde{\boldsymbol{\eta}}(x) &= \mathbf{\Lambda}(\mathbf{A} - \mathbf{c}_{0,1}\mathbf{1}^\top)\widetilde{\boldsymbol{\eta}}(x) \\
&= \mathbf{\Lambda}(\mathbf{A}\widetilde{\boldsymbol{\eta}}(x) - \mathbf{c}_{0,1}\mathbf{1}^\top\widetilde{\boldsymbol{\eta}}(x)) \\
&= \mathbf{\Lambda}(\mathbf{q}'(x) - \mathbf{c}_{0,1}) \\
&= \mathbf{q}(x)\,.
\end{aligned}$$

Thus we have that statistics $\mathbf{q}(x)$ can be expressed in terms of $\widetilde{\boldsymbol{\eta}}(x)$ as $\mathbf{q}(x) = \widetilde{\widetilde{\mathbf{A}}}\widetilde{\boldsymbol{\eta}}(x)$, where $\widetilde{\widetilde{\mathbf{A}}}$ does not require inverting $\mathbf{C}^\top$.

Again, in order to set up suitably weighted binary CPE problems that can allow us to estimate these statistics from the noisy training sample, we will use a shifted and scaled matrix $\widetilde{\mathbf{A}}''$ to estimate related statistics $\mathbf{q}''(x)$, and then factor back in the scaling and shifting when making a final prediction. Towards this, define $\widetilde{\widetilde{a}}_{\min} = \min_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$ and $\widetilde{\widetilde{a}}_{\max} = \max_{\mathbf{y},j} \widetilde{\widetilde{a}}_{j,\mathbf{y}}$, and define the entries of $\widetilde{\mathbf{A}}'' \in [0,1]^{s\times|\mathcal{Y}|}$ as

$$\widetilde{a}''_{j,\mathbf{y}} = \frac{\widetilde{\widetilde{a}}_{j,\mathbf{y}} - \widetilde{\widetilde{a}}_{\min}}{\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}} \in [0,1] \quad \forall j \in [s]\,.$$

We note again that matrix $\widetilde{\mathbf{A}}''$ here is different from that used for the NCOC algorithm in Section 4.3; in particular, now we do not need to compute $(\mathbf{C}^\top)^{-1}$. Next, define $\mathbf{q}''(x) = \widetilde{\mathbf{A}}''\widetilde{\boldsymbol{\eta}}(x)$. Then, for each $j \in [s]$, we set up a weighted binary CPE problem with weights $(\widetilde{a}''_{j,\mathbf{y}}, (1-\widetilde{a}''_{j,\mathbf{y}}))$ to estimate $q''_j(x)$. Finally, given estimated statistics $\widehat{\mathbf{q}}''(x)$ estimated in this way from the noisy training sample, our NCOC-Ham-IFN algorithm outputs the multi-label classifier

$$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \left\{ \frac{1}{s}\|\widehat{\mathbf{y}}\|_1 + \sum_{j=1}^{s} [(\widetilde{\widetilde{a}}_{\max} - \widetilde{\widetilde{a}}_{\min}) \cdot \widehat{q}''_j(x) + \widetilde{\widetilde{a}}_{\min}] \cdot \frac{1-2\widehat{y}_j}{s} \right\}.$$

Estimating $\mathbf{q}''(x)$. Our implementation of NCOC-Ham-IFN uses weighted binary logistic loss minimizers for the weighted CPE learners. In particular, we first learn a vector of s real-valued functions $\widehat{\mathbf{f}} : \mathcal{X}\to\mathbb{R}^s$ by minimizing the s-dimensional convex surrogate loss $\psi : \mathcal{Y} \times \mathbb{R}^s \to \mathbb{R}_+$ defined as

$$\psi(\mathbf{y},\mathbf{u}) = \sum_{j=1}^{s} \Big(\widetilde{a}''_{j,\mathbf{y}} \phi_{\log}(1,u_j) + (1-\widetilde{a}''_{j,\mathbf{y}})\phi_{\log}(0,u_j) \Big)$$

over the noisy training sample $\widetilde{S}$. Specifically, $\widehat{\mathbf{f}} \in \operatorname{argmin}_{\mathbf{f}\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} \psi(\widetilde{\mathbf{y}}_i, \mathbf{f}(x_i))$ for a suitable class of real-valued vector functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X}\to\mathbb{R}^s\}$. The estimated statistics are then given by $\widehat{q}''_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$.

Detailed pseudocode is in Algorithm 4.

B Supplement to Section 5

B.1 Proof of Theorem 1

In this section, we prove a more general version of Theorem 1 that can handle the case when estimated noise rates $\widehat{c}_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)}$ are used in place of the true noise rates $c_{0,1}^{(j)}$ and $c_{1,0}^{(j)}$. Therefore, setting $\widehat{c}_{0,1}^{(j)} = c_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)} = c_{1,0}^{(j)}$ for $j \in [s]$ in the theorem below proves Theorem 1.

Theorem 4 (More general version of Theorem 1). *Consider Hamming loss $\mathbf{L}^{\text{Ham}}$ (Eq. (1)) under IFN model. Assume $c_{0,1}^{(j)} + c_{1,0}^{(j)} < 1$ for all $j \in [s]$. Let D be any distribution on $\mathcal{X} \times \mathcal{Y}$ with corresponding noisy distribution $\widetilde{D}$. Let $\widehat{c}_{0,1}^{(j)} \geq 0$ and $\widehat{c}_{1,0}^{(j)} \geq 0$ be estimated noise rates such that $\widehat{c}_{0,1}^{(j)} + \widehat{c}_{1,0}^{(j)} < 1$ for $j \in [s]$. Suppose NCPLUG (Section 4.1) is run with noisy training sample $\widetilde{S}$ (in which examples are sampled i.i.d. from $\widetilde{D}$) and estimated noise rates $\widehat{c}_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)}$ in place of $c_{0,1}^{(j)}$ and $c_{1,0}^{(j)}$, for all $j \in [s]$. Let $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ be as defined in Section 4.1. Then we have*

$$\text{regret}_D^{\mathbf{L}^{\text{Ham}}}[\widehat{\mathbf{h}}] \leq \frac{1}{\sqrt{s}} \max_i \frac{1}{1 - \widehat{c}_{0,1}^{(i)} - \widehat{c}_{1,0}^{(i)}} \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]} +$$
$$\frac{2}{s} \sum_{j=1}^{s} \frac{|\widehat{c}_{0,1}^{(j)} + \widehat{c}_{1,0}^{(j)} - c_{0,1}^{(j)} - c_{1,0}^{(j)}| + |c_{0,1}^{(j)}(1 - \widehat{c}_{1,0}^{(j)}) - \widehat{c}_{0,1}^{(j)}(1 - c_{1,0}^{(j)})|}{(1 - c_{0,1}^{(j)} - c_{1,0}^{(j)})(1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)})} .$$

Moreover, if $\widehat{c}_{0,1}^{(j)} = c_{0,1}^{(j)}$ and $\widehat{c}_{1,0}^{(j)} = c_{1,0}^{(j)}$ for all $j \in [s]$, the above bound can be simplified to

$$\text{regret}_D^{\mathbf{L}^{\text{Ham}}}[\widehat{\mathbf{h}}] \leq \frac{1}{\sqrt{s}} \max_i \frac{1}{1 - c_{0,1}^{(i)} - c_{1,0}^{(i)}} \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]} .$$

We will need the following lemma from [1] in the proof.

Lemma 5 (Property of the binary logistic loss [1]). *Recall that the binary logistic loss $\phi_{\log} : \{0,1\} \times \mathbb{R} \rightarrow \mathbb{R}_+$ is defined as*

$$\phi_{\log}(y, u) = \ln(1 + e^{-(2y-1)u}) ,$$

and the invertible link function $\gamma_{\log} : [0,1] \rightarrow \mathbb{R}$ together with its inverse $\gamma_{\log}^{-1} : \mathbb{R} \rightarrow [0,1]$ is given by

$$\gamma_{\log}(p) = \ln(\frac{p}{1-p}) ;$$
$$\gamma_{\log}^{-1}(u) = \frac{1}{1 + \exp(-u)} .$$

Then for all $q \in [0,1]$ and $u \in \mathbb{R}$:

$$\mathbf{E}_{Y \sim \text{Bernoulli}(q)} \Big[\phi_{\log}(Y, u) - \phi_{\log}(Y, \gamma_{\log}(q))\Big] \;\geq\; 2\Big(\gamma_{\log}^{-1}(u) - q\Big)^2 ,$$

where $Y \sim$ Bernoulli(q) denotes a Bernoulli random variable that takes value 1 with probability q and value 0 with probability $1 - q$.

Proof of Theorem 4.

Proof. Recall that the multi-label surrogate $\psi : \mathcal{Y} \times \mathbb{R}^s \rightarrow \mathbb{R}_+$ here is given by

$$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^{s} \phi_{\log}(y_j, u_j) . \tag{5}$$

Also, given $\widehat{q}_j'(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCPLUG algorithm is given by

$$\widehat{h}_j(x) = \mathbf{1}\Big(\frac{\widehat{q}_j'(x) - \widehat{c}_{0,1}^{(j)}}{1 - \widehat{c}_{0,1}^{(j)} - \widehat{c}_{1,0}^{(j)}} \geq \tfrac{1}{2}\Big) \quad \forall j \in [s] . \tag{6}$$

To simplify notations, in what follows, we let $\mathbf{L} = \mathbf{L}^{\text{Ham}}$, $\phi = \phi_{\log}$, $\gamma = \gamma_{\log}$, $\mathbf{f} = \widehat{\mathbf{f}}$, and $\mathbf{h} = \widehat{\mathbf{h}}$. We also let $q_j(x) = \mathbf{P}(Y_j = 1|x)$ and $\widetilde{q}_j(x) = \mathbf{P}(\widetilde{Y}_j = 1|x)$ for $j \in [s]$.

$$\begin{aligned}
&\text{regret}_D^{\mathbf{L}}[\mathbf{h}] \\
&= \frac{1}{s}\sum_{j=1}^{s} \mathbf{E}_x\left[(1-2q_j(x))\cdot[\mathbf{1}(\frac{\gamma^{-1}(f_j(x)) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} \geq \frac{1}{2}) - \mathbf{1}(q_j(x) \geq \frac{1}{2})]\right] \\
&\leq \frac{2}{s}\sum_{j=1}^{s} \mathbf{E}_x\left[|\frac{\gamma^{-1}(f_j(x)) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - q_j(x)|\right] \\
&= \frac{2}{s}\sum_{j=1}^{s} \mathbf{E}_x\left[|\frac{\gamma^{-1}(f_j(x)) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - \frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} + \frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - q_j(x)|\right] \\
&\leq \frac{2}{s}\sum_{j=1}^{s} \mathbf{E}_x\left[|\frac{\gamma^{-1}(f_j(x)) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - \frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}}| + |\frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - q_j(x)|\right] \\
&= \frac{2}{s}\sum_{j=1}^{s} \mathbf{E}_x\left[|\frac{\gamma^{-1}(f_j(x)) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - \frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}}|\right] + \frac{2}{s}\sum_{j=1}^{s} \mathbf{E}_x\left[|\frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - q_j(x)|\right].
\end{aligned} \tag{7}$$

We bound the first sum as follows:

$$\begin{aligned}
&\frac{2}{s}\sum_{j=1}^{s} \mathbf{E}_x\left[|\frac{\gamma^{-1}(f_j(x)) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}} - \frac{\widetilde{q}_j(x) - \widehat{c}_{0,1}^{(j)}}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}}|\right] \\
&= \frac{2}{s}\sum_{j=1}^{s} \frac{1}{1-\widehat{c}_{0,1}^{(j)}-\widehat{c}_{1,0}^{(j)}}\mathbf{E}_x\left[|\gamma^{-1}(f_j(x)) - \widetilde{q}_j(x)|\right] \\
&\leq \frac{2}{s}\max_i \frac{1}{1-\widehat{c}_{0,1}^{(i)}-\widehat{c}_{1,0}^{(i)}}\mathbf{E}_x\left[\sum_{j=1}^{s}|\gamma^{-1}(f_j(x)) - \widetilde{q}_j(x)|\right] \\
&\leq \frac{2}{s}\max_i \frac{1}{1-\widehat{c}_{0,1}^{(i)}-\widehat{c}_{1,0}^{(i)}}\mathbf{E}_x\left[\sqrt{s}\sqrt{\sum_{j=1}^{s}\Big(\gamma^{-1}(f_j(x)) - \widetilde{q}_j(x)\Big)^2}\right] \\
&= \frac{2}{\sqrt{s}}\max_i \frac{1}{1-\widehat{c}_{0,1}^{(i)}-\widehat{c}_{1,0}^{(i)}}\mathbf{E}_x\left[\sqrt{\sum_{j=1}^{s}\Big(\gamma^{-1}(f_j(x)) - \widetilde{q}_j(x)\Big)^2}\right].
\end{aligned} \tag{8}$$

By Lemma 5, we have

$$\begin{aligned}
&\mathbf{E}_x\left[\sum_{j=1}^{s}\Big(\gamma^{-1}(f_j(x)) - \widetilde{q}_j(x)\Big)^2\right] \\
&\leq \frac{1}{2}\mathbf{E}_x\left[\sum_{j=1}^{s}\mathbf{E}_{y\sim\text{Bernoulli}(\widetilde{q}_j(x))}\Big[\phi(y, f_j(x)) - \phi(y, \gamma(\widetilde{q}_j(x)))\Big]\right] \\
&= \frac{1}{2}\mathbf{E}_x\left[\mathbf{E}_{\mathbf{y}|x\sim\widetilde{\boldsymbol{\eta}}(x)}\Big[\psi(\mathbf{y}, \mathbf{f}(x)) - \inf_{\mathbf{u}\in\mathbb{R}^r}\psi(\mathbf{y}, \mathbf{u})\Big]\right] \\
&= \frac{1}{2}\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}].
\end{aligned} \tag{9}$$

Next, we bound the second sum in Eq. (7) as follows:

$$
\begin{aligned}
&\frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\left[\left|\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-q_j(x)\right|\right]\\
&=\frac{2}{s}\sum_{j=1}^{s}\mathbf{E}_x\left[\left|\frac{\widetilde{q}_j(x)-\widehat{c}^{(j)}_{0,1}}{1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0}}-\frac{\widetilde{q}_j(x)-c^{(j)}_{0,1}}{1-c^{(j)}_{0,1}-c^{(j)}_{1,0}}\right|\right]\\
&=\frac{2}{s}\sum_{j=1}^{s}\frac{\mathbf{E}_x\left[|\widetilde{q}_j(x)(\widehat{c}^{(j)}_{0,1}+\widehat{c}^{(j)}_{1,0}-c^{(j)}_{0,1}-c^{(j)}_{1,0})+\widehat{c}^{(j)}_{0,1}(c^{(j)}_{1,0}-1)+c^{(j)}_{0,1}(1-\widehat{c}^{(j)}_{1,0})|\right]}{(1-c^{(j)}_{0,1}-c^{(j)}_{1,0})(1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0})}\\
&\le\frac{2}{s}\sum_{j=1}^{s}\frac{|\widehat{c}^{(j)}_{0,1}+\widehat{c}^{(j)}_{1,0}-c^{(j)}_{0,1}-c^{(j)}_{1,0}|+|c^{(j)}_{0,1}(1-\widehat{c}^{(j)}_{1,0})-\widehat{c}^{(j)}_{0,1}(1-c^{(j)}_{1,0})|}{(1-c^{(j)}_{0,1}-c^{(j)}_{1,0})(1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0})}\,. \qquad (10)
\end{aligned}
$$

Combining Eqs. (7), (8), (9) and (10) and applying Jensen's inequality (to the convex function $x\mapsto x^2$), we have

$$
\begin{aligned}
\mathrm{regret}^{\mathbf{L}}_D[\mathbf{h}]\le\;&\frac{1}{\sqrt{s}}\max_i\frac{1}{1-\widehat{c}^{(i)}_{0,1}-\widehat{c}^{(i)}_{1,0}}\sqrt{2\mathrm{regret}^{\psi}_{\widetilde{D}}[\mathbf{f}]}+\\
&\frac{2}{s}\sum_{j=1}^{s}\frac{|\widehat{c}^{(j)}_{0,1}+\widehat{c}^{(j)}_{1,0}-c^{(j)}_{0,1}-c^{(j)}_{1,0}|+|c^{(j)}_{0,1}(1-\widehat{c}^{(j)}_{1,0})-\widehat{c}^{(j)}_{0,1}(1-c^{(j)}_{1,0})|}{(1-c^{(j)}_{0,1}-c^{(j)}_{1,0})(1-\widehat{c}^{(j)}_{0,1}-\widehat{c}^{(j)}_{1,0})}\,.
\end{aligned}
$$

$\square$

B.2 Proof of Theorem 2

In this section, we prove a more general version of Theorem 2 that can handle the case when an estimated noise matrix $\widehat{\mathbf{C}}$ is used in place of the true noise matrix $\mathbf{C}$. Therefore, setting $\widehat{\mathbf{C}}=\mathbf{C}$ in the theorem below proves Theorem 2.

Theorem 6 (More general version of Theorem 2). *Consider F-measure $\mathbf{L}^{F_1}$ (Eq. (2)) under the general CCN model. Assume noise matrix $\mathbf{C}$ is invertible. Let D be any distribution on $\mathcal{X}\times\mathcal{Y}$ with corresponding noisy distribution $\widetilde{D}$. Let $\widehat{\mathbf{C}}$ be an estimated (invertible) noise matrix for the noise matrix $\mathbf{C}$. Suppose NCEFP (Section 4.2) is run with noisy training sample $\widetilde{S}$ (in which examples are sampled i.i.d. from $\widetilde{D}$) and estimated noise matrix $\widehat{\mathbf{C}}$ in place of $\mathbf{C}$. Let $\psi,\widehat{\mathbf{f}},\widehat{\mathbf{h}}$ be as defined in Section 4.2, and let $\mathbf{A}\in[0,1]^{(s^2+1)\times|\mathcal{Y}|}$, $\mathbf{B}\in\mathbb{R}^{(s^2+1)\times|\mathcal{Y}|}$, and $\mathbf{t}\in\mathbb{R}^{|\mathcal{Y}|}$ be as defined in Example 2. Then we have*

$$
\mathrm{regret}^{\mathbf{L}^{F_1}}_D[\widehat{\mathbf{h}}]\le 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2\cdot\left(\|\mathbf{A}\|_2\|(\mathbf{C}^\top)^{-1}-(\widehat{\mathbf{C}}^\top)^{-1}\|_2+2\|\mathbf{A}\|_1\|(\widehat{\mathbf{C}}^\top)^{-1}\|_1 s\sqrt{2\mathrm{regret}^{\psi}_{\widetilde{D}}[\widehat{\mathbf{f}}]}\right).
$$

Further, if $\widehat{\mathbf{C}}=\mathbf{C}$, the above bound can be simplified to

$$
\mathrm{regret}^{\mathbf{L}^{F_1}}_D[\widehat{\mathbf{h}}]\le 4s\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2\cdot\|\mathbf{A}\|_1\|(\mathbf{C}^\top)^{-1}\|_1\sqrt{2\mathrm{regret}^{\psi}_{\widetilde{D}}[\widehat{\mathbf{f}}]}\,.
$$

We will need the following lemma from [48] in the proof.

Lemma 7 (Property of the multiclass logistic loss [48]). *Recall that the multiclass logistic loss $\phi_{\text{mlog}}:[n]\times\mathbb{R}^{n-1}\to\mathbb{R}_+$ is defined as*

$$
\phi_{\text{mlog}}(y,\mathbf{u})=\begin{cases}-\ln\left(\frac{\exp(u_y)}{1+\sum_{i=1}^{n-1}\exp(u_i)}\right) & \text{if } y\in[n-1]\\ \ln\left(1+\sum_{i=1}^{n-1}\exp(u_i)\right) & \text{if } y=n\end{cases},
$$

and the invertible link function $\boldsymbol{\gamma}_{\text{mlog}} : \Delta_n \to \mathbb{R}^{n-1}$ together with its inverse $\boldsymbol{\gamma}_{\text{mlog}}^{-1} : \mathbb{R}^{n-1} \to \Delta_n$ is given by

$$\boldsymbol{\gamma}_{\text{mlog}}(\mathbf{p}) = \begin{pmatrix} \ln(\frac{p_1}{p_n}) \\ \vdots \\ \ln(\frac{p_{n-1}}{p_n}) \end{pmatrix}; \quad \boldsymbol{\gamma}_{\text{mlog}}^{-1}(\mathbf{u}) = \begin{pmatrix} \frac{\exp(u_1)}{1+\sum_{i=1}^{n-1}\exp(u_i)} \\ \vdots \\ \frac{\exp(u_n)}{1+\sum_{i=1}^{n-1}\exp(u_i)} \\ \frac{1}{1+\sum_{i=1}^{n-1}\exp(u_i)} \end{pmatrix}.$$

Then for all $\mathbf{p} \in \Delta_n$ and $\mathbf{u} \in \mathbb{R}^{n-1}$,

$$\mathbf{E}_{Y\sim\mathbf{p}}\Big[\phi_{\text{mlog}}(Y, \mathbf{u}) - \phi_{\text{mlog}}(Y, \boldsymbol{\gamma}_{\text{mlog}}(\mathbf{p}))\Big] \geq \frac{1}{2}\big\|\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\mathbf{u}) - \mathbf{p}\big\|_2^2.$$

Proof of Theorem 6.

Proof. For clarity, we redefine here all quantities involving $\widehat{\mathbf{C}}$. In particular, define $\widehat{\mathbf{A}} = \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}$, $\widehat{a}_{\min} = \min(\min_{\mathbf{y}} \widehat{a}_{0,\mathbf{y}}, \min_{\mathbf{y},jk} \widehat{a}_{jk,\mathbf{y}})$ and $\widehat{a}_{\max} = \max(\max_{\mathbf{y}} \widehat{a}_{0,\mathbf{y}}, \max_{\mathbf{y},jk} \widehat{a}_{jk,\mathbf{y}})$, and let

$$\widehat{a}'_{0,\mathbf{y}} = \frac{\widehat{a}_{0,\mathbf{y}} - \widehat{a}_{\min}}{\widehat{a}_{\max} - \widehat{a}_{\min}} \in [0,1],$$

and

$$\widehat{a}'_{jk,\mathbf{y}} = \frac{\widehat{a}_{jk,\mathbf{y}} - \widehat{a}_{\min}}{s\cdot(\widehat{a}_{\max} - \widehat{a}_{\min})} \in [0,1] \quad \forall j,k \in [s].$$

Here, the multi-label surrogate $\psi : \mathcal{Y} \times \mathbb{R}^{s^2+1} \to \mathbb{R}_+$ then becomes

$$\begin{aligned}\psi(\mathbf{y},\mathbf{u}) =\ & \widehat{a}'_{0,\mathbf{y}} \cdot \phi_{\log}(1, u_0) + (1 - \widehat{a}'_{0,\mathbf{y}}) \cdot \phi_{\log}(0, u_0) + \\ & \sum_{j=1}^{s}\Big[\sum_{k=1}^{s} \widehat{a}'_{jk,\mathbf{y}} \phi_{\text{mlog}}(k, (u_{j1}, ..., u_{js})) + (1 - \sum_{k=1}^{s} \widehat{a}'_{jk,\mathbf{y}}) \phi_{\text{mlog}}(s+1, (u_{j1}, ..., u_{js}))\Big].\end{aligned} \tag{11}$$

Also, given $\widehat{q}'_0(x) = \gamma_{\log}^{-1}(\widehat{f}_0(x))$ and $\widehat{q}'_{jk}(x) = \big(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(\widehat{f}_{j1}(x), ..., \widehat{f}_{js}(x))\big)_k$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCEFP algorithm is given by

$$\begin{aligned}\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}}\in\mathcal{Y}} \Big\{ & 1 + [(\widehat{a}_{\max} - \widehat{a}_{\min}) \cdot \widehat{q}'_0(x) + \widehat{a}_{\min}] \cdot b_{0,\widehat{\mathbf{y}}} \\ & + \sum_{j=1}^{s}\sum_{k=1}^{s} [s \cdot (\widehat{a}_{\max} - \widehat{a}_{\min}) \cdot \widehat{q}'_{jk}(x) + \widehat{a}_{\min}] \cdot b_{jk,\widehat{\mathbf{y}}} \Big\}.\end{aligned} \tag{12}$$

We use $\langle\cdot,\cdot\rangle$ to denote the standard inner product. Let $\widehat{\alpha} = (\widehat{a}_{\max} - \widehat{a}_{\min})$ and $\widehat{\beta} = \widehat{a}_{\min}$. To simplify notations, in what follows, we let $\mathbf{L} = \mathbf{L}^{F_1}$, $\mathbf{f} = \widehat{\mathbf{f}}$ and $\mathbf{h} = \widehat{\mathbf{h}}$.

We let

$$\begin{aligned} g(\mathbf{f}(x), \widehat{\mathbf{y}}) := & (\widehat{\alpha}\gamma_{\log}^{-1}(f_0(x)) + \widehat{\beta}) \cdot (b_{0,\mathbf{h}(x)} - b_{0,\widehat{\mathbf{y}}}) \\ & + \widehat{\alpha} s \sum_{j=1}^{s}\sum_{k=1}^{s} (\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k \cdot (b_{jk,\mathbf{h}(x)} - b_{jk,\widehat{\mathbf{y}}}) \\ & + \widehat{\beta} \sum_{j=1}^{s}\sum_{k=1}^{s} (b_{jk,\mathbf{h}(x)} - b_{jk,\widehat{\mathbf{y}}}) \\ & + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}).\end{aligned} \tag{13}$$

$$
\begin{aligned}
&\mathrm{regret}_D^{\mathbf{L}}[\mathbf{h}] \\
&= \mathbf{E}_x\left[\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)}\rangle - \min_{\widehat{\mathbf{y}}\in\mathcal{Y}}\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)} - \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \mathbf{A}^\top \mathbf{b}_{\mathbf{h}(x)} + t_{\mathbf{h}(x)}\mathbf{1} - \mathbf{A}^\top \mathbf{b}_{\widehat{\mathbf{y}}} - t_{\widehat{\mathbf{y}}}\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}) + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}})\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\right]\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\right]\right] \quad \text{(by property of adjoint)} \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}) - g(\mathbf{f}(x), \widehat{\mathbf{y}}) + g(\mathbf{f}(x), \widehat{\mathbf{y}})\right]\right] \\
&\leq \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}}\left[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}) - g(\mathbf{f}(x), \widehat{\mathbf{y}})\right]\right] \\
&\quad \Big(\text{Since by Eq. (12), } g(\mathbf{f}(x), \widehat{\mathbf{y}}) \leq 0 \text{ for all } \widehat{\mathbf{y}}\ \Big) \\
&= \mathbf{E}_x\Bigg[\max_{\widehat{\mathbf{y}}}\Big[[(\mathbf{A}\boldsymbol{\eta}(x))_0 - \widehat{\alpha}\gamma_{\mathrm{log}}^{-1}(f_0(x)) - \widehat{\beta}] \cdot (b_{0,\mathbf{h}(x)} - b_{0,\widehat{\mathbf{y}}}) \\
&\qquad + \sum_{j=1}^{s}\sum_{k=1}^{s}[(\mathbf{A}\boldsymbol{\eta}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k - \widehat{\beta}] \cdot (b_{jk,\mathbf{h}(x)} - b_{jk,\widehat{\mathbf{y}}})\Big]\Bigg] \\
&\leq \mathbf{E}_x\Bigg[\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \Big[[(\mathbf{A}\boldsymbol{\eta}(x))_0 - \widehat{\alpha}\gamma_{\mathrm{log}}^{-1}(f_0(x)) - \widehat{\beta}]^2 + \\
&\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[(\mathbf{A}\boldsymbol{\eta}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k - \widehat{\beta}]^2\Big]^{\frac{1}{2}}\Bigg] \\
&\quad \text{(by the Cauchy-Schwarz inequality)} \\
&\leq 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\Bigg[\Big[[(\mathbf{A}\boldsymbol{\eta}(x))_0 - \widehat{\alpha}\gamma_{\mathrm{log}}^{-1}(f_0(x)) - \widehat{\beta}]^2 + \\
&\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[(\mathbf{A}\boldsymbol{\eta}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k - \widehat{\beta}]^2\Big]^{\frac{1}{2}}\Bigg]. \qquad (14)
\end{aligned}
$$

For all $x \in \mathcal{X}$, let $\boldsymbol{\zeta}(x) \in \mathbb{R}^{s^2+1}$ be such that

$$\zeta_0(x) = \widehat{\alpha}\gamma_{\mathrm{log}}^{-1}(f_0(x)), \quad \zeta_{jk}(x) = \widehat{\alpha}s(\boldsymbol{\gamma}_{\mathrm{mlog}}^{-1}(f_{j1}(x), ..., f_{js}(x)))_k\,. \qquad (15)$$

Then Eq. (14) can be written as

$$
\begin{aligned}
\mathrm{regret}_D^{\mathbf{L}}[\mathbf{h}] &\leq 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2\right] \\
&= 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) + \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2\right] \\
&\leq 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2 + \|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2\right]. \qquad (16)
\end{aligned}
$$

Note that,

$$\begin{aligned}
&\mathbf{E}_x\Big[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&= \mathbf{E}_x\Big[\|\mathbf{A}(\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x) - \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&= \mathbf{E}_x\Big[\|\mathbf{A}\big((\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\big)\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&\le \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 \cdot \mathbf{E}_x\Big[\|\widetilde{\boldsymbol{\eta}}(x)\|_2\Big] \\
&\le \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2\,. \qquad (17)
\end{aligned}$$

Moreover,

$$\begin{aligned}
&\mathbf{E}_x\Big[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2^2\Big] \\
&= \mathbf{E}_x\Big[[(\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x))_0 - \widehat{\alpha}\gamma_{\text{log}}^{-1}(f_0(x)) - \widehat{\beta}]^2 + \\
&\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[(\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k - \widehat{\beta}]^2\Big] \\
&= \mathbf{E}_x\Big[[\sum_{\mathbf{y}}\widehat{a}_{0,\mathbf{y}}\cdot\widetilde{\eta}_{\mathbf{y}}(x) - \widehat{\alpha}\gamma_{\text{log}}^{-1}(f_0(x)) - \widehat{\beta}]^2 + \\
&\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[\sum_{\mathbf{y}}\widehat{a}_{jk,\mathbf{y}}\cdot\widetilde{\eta}_{\mathbf{y}}(x) - \widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k - \widehat{\beta}]^2\Big] \\
&= \mathbf{E}_x\Big[[\sum_{\mathbf{y}}(\widehat{\alpha}\cdot\widehat{a}'_{0,\mathbf{y}} + \widehat{\beta})\cdot\widetilde{\eta}_{\mathbf{y}}(x) - \widehat{\alpha}\gamma_{\text{log}}^{-1}(f_0(x)) - \widehat{\beta}]^2 + \\
&\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[\sum_{\mathbf{y}}(\widehat{\alpha}s\cdot\widehat{a}'_{jk,\mathbf{y}} + \widehat{\beta})\cdot\widetilde{\eta}_{\mathbf{y}}(x) - \widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k - \widehat{\beta}]^2\Big] \\
&= \mathbf{E}_x\Big[[\widehat{\alpha}(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0 - \widehat{\alpha}\gamma_{\text{log}}^{-1}(f_0(x))]^2 + \\
&\qquad \sum_{j=1}^{s}\sum_{k=1}^{s}[\widehat{\alpha}s(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - \widehat{\alpha}s(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k]^2\Big] \\
&= \mathbf{E}_x\Big[\widehat{\alpha}^2[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0 - \gamma_{\text{log}}^{-1}(f_0(x))]^2 + \widehat{\alpha}^2 s^2\sum_{j=1}^{s}\sum_{k=1}^{s}[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - (\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k]^2\Big]\,. \qquad (18)
\end{aligned}$$

Note that $(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0 \in [0,1]$. By Lemma 5, we have

$$\begin{aligned}
&\Big[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0 - \gamma_{\text{log}}^{-1}(f_0(x))\Big]^2 \\
&\le \frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)}\Big[\phi_{\text{log}}\Big(y, f_0(x)\Big) - \phi_{\text{log}}\Big(y, \gamma_{\text{log}}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)\Big)\Big]\,. \qquad (19)
\end{aligned}$$

By Lemma 7, we have

$$\begin{aligned}
&\sum_{k=1}^{s}\Big[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - (\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k\Big]^2 \\
&\le \Big[1 - \sum_{k=1}^{s}(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - 1 + \sum_{k=1}^{s}(\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k\Big]^2 \\
&\quad + \sum_{k=1}^{s}\Big[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - (\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k\Big]^2 \\
&\le 2\mathbf{E}_{y\sim\mathbf{p}_j(x)}\Big[\phi_{\text{mlog}}\Big(y,(f_{j1}(x),...,f_{js}(x))\Big) - \phi_{\text{mlog}}\Big(y,\boldsymbol{\gamma}_{\text{mlog}}\big(\mathbf{p}_j(x)\big)\Big)\Big], \qquad (20)
\end{aligned}$$

where $\mathbf{p}_j(x) \in \Delta_{s+1}$ is

$$\mathbf{p}_j(x) = \begin{bmatrix} (\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{j1} \\ \vdots \\ (\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{js} \\ 1 - \sum_{k=1}^{s}(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} \end{bmatrix}. \qquad (21)$$

Then, Eq. (18) becomes

$$\begin{aligned}
&\mathbf{E}_x\Big[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\zeta}(x) - \widehat{\beta}\mathbf{1}\|_2^2\Big] \\
&= \mathbf{E}_x\Big[\widehat{\alpha}^2[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0 - \gamma_{\text{log}}^{-1}(f_0(x))]^2 + \widehat{\alpha}^2 s^2\sum_{j=1}^{s}\sum_{k=1}^{s}[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_{jk} - (\boldsymbol{\gamma}_{\text{mlog}}^{-1}(f_{j1}(x),...,f_{js}(x)))_k]^2\Big] \\
&\le \mathbf{E}_x\Big[\widehat{\alpha}^2\frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)}\Big[\phi_{\text{log}}\Big(y,f_0(x)\Big) - \phi_{\text{log}}\Big(y,\gamma_{\text{log}}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)\Big)\Big] \\
&\quad + 2\widehat{\alpha}^2 s^2\sum_{j=1}^{s}\mathbf{E}_{y\sim\mathbf{p}_j(x)}\Big[\phi_{\text{mlog}}\Big(y,(f_{j1}(x),...,f_{js}(x))\Big) - \phi_{\text{mlog}}\Big(y,\boldsymbol{\gamma}_{\text{mlog}}\big(\mathbf{p}_j(x)\big)\Big)\Big]\Big] \\
&\le 2\widehat{\alpha}^2 s^2\cdot\mathbf{E}_x\Big[\mathbf{E}_{y\sim\text{Bernoulli}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)}\Big[\phi_{\text{log}}\Big(y,f_0(x)\Big) - \phi_{\text{log}}\Big(y,\gamma_{\text{log}}\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_0\big)\Big)\Big] \\
&\qquad + \sum_{j=1}^{s}\mathbf{E}_{y\sim\mathbf{p}_j(x)}\Big[\phi_{\text{mlog}}\Big(y,(f_{j1}(x),...,f_{js}(x))\Big) - \phi_{\text{mlog}}\Big(y,\boldsymbol{\gamma}_{\text{mlog}}\big(\mathbf{p}_j(x)\big)\Big)\Big]\Big] \\
&= 2\widehat{\alpha}^2 s^2\cdot\mathbf{E}_x\Big[\mathbf{E}_{\mathbf{y}|x\sim\widetilde{\boldsymbol{\eta}}(x)}\Big[\psi(\mathbf{y},\mathbf{f}(x)) - \inf_{\mathbf{u}\in\mathbb{R}^{s^2+1}}\psi(\mathbf{y},\mathbf{u})\Big]\Big] \\
&= 2\widehat{\alpha}^2 s^2\cdot\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]. \qquad (22)
\end{aligned}$$

Combining Eqs. (16), (17) and (22) and applying Jensen's inequality (to the convex function $x \mapsto x^2$), we have

$$\text{regret}_D^{\mathbf{L}}[\mathbf{h}] \le 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2\cdot\left(\|\mathbf{A}\|_2\|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 + \widehat{\alpha}s\sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]}\right).$$

We can further bound $\widehat{\alpha}$ by

$$\begin{aligned}
\widehat{\alpha} &= \widehat{a}_{\max} - \widehat{a}_{\min} \\
&\le 2\|\widehat{\mathbf{A}}\|_1 \\
&= 2\|\mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\|_1 \\
&\le 2\|\mathbf{A}\|_1\|(\widehat{\mathbf{C}}^\top)^{-1}\|_1.
\end{aligned}$$

□

B.3 Proof of Theorem 3

In this section, we prove a more general version of Theorem 3 that can handle the case when an estimated noise matrix $\widehat{\mathbf{C}}$ is used in place of the true noise matrix $\mathbf{C}$. Therefore, setting $\widehat{\mathbf{C}} = \mathbf{C}$ in the theorem below proves Theorem 3.

Theorem 8 (More general version of Theorem 3). *Consider a general low-rank loss matrix* $\mathbf{L}$ *written as* $\mathbf{L} = \mathbf{A}^\top \mathbf{B} + \mathbf{1}\mathbf{t}^\top$ *for some* $\mathbf{A} \in [0,1]^{r\times|\mathcal{Y}|}, \mathbf{B} \in \mathbb{R}^{r\times|\mathcal{Y}|}, \mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$*, under the general CCN model. Assume noise matrix* $\mathbf{C}$ *is invertible. Let* D *be any distribution on* $\mathcal{X} \times \mathcal{Y}$ *with corresponding noisy distribution* $\widetilde{D}$*. Let* $\widehat{\mathbf{C}}$ *be an estimated (invertible) noise matrix for the noise matrix* $\mathbf{C}$*. Suppose NCOC (Section 4.3) is run with noisy training sample* $\widetilde{S}$ *(in which examples are sampled i.i.d. from* $\widetilde{D}$*) and estimated noise matrix* $\widehat{\mathbf{C}}$ *in place of* $\mathbf{C}$*. Let* $\psi, \widehat{\mathbf{f}}, \widehat{\mathbf{h}}$ *be as defined in Section 4.3. Then we have*

$$\text{regret}_D^{\mathbf{L}}[\widehat{\mathbf{h}}] \le 2 \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \left(\|\mathbf{A}\|_2 \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 + \|\mathbf{A}\|_1 \|(\widehat{\mathbf{C}}^\top)^{-1}\|_1 \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]} \right).$$

Further, if $\widehat{\mathbf{C}} = \mathbf{C}$*, the above bound can be simplified to*

$$\text{regret}_D^{\mathbf{L}}[\widehat{\mathbf{h}}] \le 2 \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \|\mathbf{A}\|_1 \|(\mathbf{C}^\top)^{-1}\|_1 \sqrt{2\text{regret}_{\widetilde{D}}^{\psi}[\widehat{\mathbf{f}}]}\,.$$

Proof. For clarity, we redefine here all quantities involving $\widehat{\mathbf{C}}$. In particular, define $\widehat{\mathbf{A}} = \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}$, $\widehat{a}_{\min} = \min_{\mathbf{y},j} \widehat{a}_{j,\mathbf{y}}$ and $\widehat{a}_{\max} = \max_{\mathbf{y},j} \widehat{a}_{j,\mathbf{y}}$, and let

$$\widehat{a}'_{j,\mathbf{y}} = \frac{\widehat{a}_{j,\mathbf{y}} - \widehat{a}_{\min}}{\widehat{a}_{\max} - \widehat{a}_{\min}} \in [0,1] \quad \forall j \in [r]\,.$$

Here, the multi-label surrogate $\psi : \mathcal{Y} \times \mathbb{R}^r \to \mathbb{R}_+$ then becomes

$$\psi(\mathbf{y}, \mathbf{u}) = \sum_{j=1}^{r} \Big(\widehat{a}'_{j,\mathbf{y}} \phi(1, u_j) + (1 - \widehat{a}'_{j,\mathbf{y}}) \phi(0, u_j) \Big)\,. \tag{23}$$

Also, given $\widehat{q}'_j(x) = \gamma_{\log}^{-1}(\widehat{f}_j(x))$ estimated from $\widetilde{S}$, the multi-label classifier output by our NCOC algorithm is given by

$$\widehat{\mathbf{h}}(x) = \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \mathcal{Y}} \left\{ t_{\widehat{\mathbf{y}}} + \sum_{j=1}^{r} [(\widehat{a}_{\max} - \widehat{a}_{\min}) \cdot \widehat{q}'_j(x) + \widehat{a}_{\min}] \cdot b_{j,\widehat{\mathbf{y}}} \right\}. \tag{24}$$

We use $\langle \cdot, \cdot \rangle$ to denote the standard inner product. Let $\widehat{\alpha} = (\widehat{a}_{\max} - \widehat{a}_{\min})$ and $\widehat{\beta} = \widehat{a}_{\min}$. To simplify notations, in what follows, we let $\phi = \phi_{\log}$, $\gamma = \gamma_{\log}$, $\mathbf{f} = \widehat{\mathbf{f}}$, and $\mathbf{h} = \widehat{\mathbf{h}}$. For $\mathbf{u} \in \mathbb{R}^r$, let $\boldsymbol{\gamma}^{-1}(\mathbf{u}) \in [0,1]^r$ be such that the i-indexed entry of $\boldsymbol{\gamma}^{-1}(\mathbf{u})$ is simply $\gamma^{-1}(u_i)$.

$$
\begin{aligned}
&\text{regret}^{\mathbf{L}}_{D}[\mathbf{h}] \\
&= \mathbf{E}_x\left[\langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)}\rangle - \min_{\widehat{\mathbf{y}}\in\mathcal{Y}} \langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}} \langle \boldsymbol{\eta}(x), \boldsymbol{\ell}_{\mathbf{h}(x)} - \boldsymbol{\ell}_{\widehat{\mathbf{y}}}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}} \langle \boldsymbol{\eta}(x), \mathbf{A}^\top \mathbf{b}_{\mathbf{h}(x)} + t_{\mathbf{h}(x)}\mathbf{1} - \mathbf{A}^\top \mathbf{b}_{\widehat{\mathbf{y}}} - t_{\widehat{\mathbf{y}}}\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}} \langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}) + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\mathbf{1}\rangle\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}} \Big[\langle \boldsymbol{\eta}(x), \mathbf{A}^\top (\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}})\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\Big]\right] \\
&= \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}} \Big[\langle \mathbf{A}\boldsymbol{\eta}(x), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\Big]\right] \quad \text{(by property of adjoint)} \\
&= \mathbf{E}_x\Bigg[\max_{\widehat{\mathbf{y}}} \Big[\langle \mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + \\
&\qquad \langle (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}})\Big]\Bigg] \\
&\le \mathbf{E}_x\left[\max_{\widehat{\mathbf{y}}} \langle \mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle\right] \\
&\quad \Big(\text{Since by Eq. (24), } \langle (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1}), \mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\rangle + (t_{\mathbf{h}(x)} - t_{\widehat{\mathbf{y}}}) \le 0 \text{ for all } \widehat{\mathbf{y}}\ \Big) \\
&\le \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2 \cdot \max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\mathbf{h}(x)} - \mathbf{b}_{\widehat{\mathbf{y}}}\|_2\right] \\
&\quad \text{(by the Cauchy-Schwarz inequality)} \\
&\le 2\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right] \\
&= 2\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) + \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right] \\
&\le 2\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2 + \|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right] \\
&= 2\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2\right] + 2\max_{\widehat{\mathbf{y}}} \|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2\right].
\end{aligned} \tag{25}
$$

Note that,

$$
\begin{aligned}
&\mathbf{E}_x\left[\|\mathbf{A}\boldsymbol{\eta}(x) - \widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&= \mathbf{E}_x\left[\|\mathbf{A}(\mathbf{C}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x) - \mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&= \mathbf{E}_x\left[\|\mathbf{A}\big((\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\big)\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&\le \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 \cdot \mathbf{E}_x\left[\|\widetilde{\boldsymbol{\eta}}(x)\|_2\right] \\
&\le \|\mathbf{A}\|_2 \cdot \|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 \,.
\end{aligned} \tag{26}
$$

Moreover,

$$\begin{aligned}
&\mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2^2\right] \\
&= \mathbf{E}_x\left[\|(\widehat{\mathbf{A}} - \widehat{\beta}\mathbf{1}\mathbf{1}^\top)\widetilde{\boldsymbol{\eta}}(x) - \widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x))\|_2^2\right] \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2\|\frac{(\widehat{\mathbf{A}} - \widehat{\beta}\mathbf{1}\mathbf{1}^\top)}{\widehat{\alpha}}\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x))\|_2^2\right] \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2\|\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x) - \boldsymbol{\gamma}^{-1}(\mathbf{f}(x))\|_2^2\right] \quad \text{(because } \widehat{\mathbf{A}}' = \frac{\widehat{\mathbf{A}} - \widehat{a}_{\min}\mathbf{1}\mathbf{1}^\top}{\widehat{a}_{\max} - \widehat{a}_{\min}} = \frac{\widehat{\mathbf{A}} - \widehat{\beta}\mathbf{1}\mathbf{1}^\top}{\widehat{\alpha}}\text{)} \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2\sum_{j=1}^r\left[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j - \gamma^{-1}(f_j(x))\right]^2\right]. \qquad (27)
\end{aligned}$$

Note that $(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j \in [0, 1]$. By Lemma 5, we have

$$\left[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j - \gamma^{-1}(f_j(x))\right]^2 \le \frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\left((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\right)}\left[\phi\Big(y, f_j(x)\Big) - \phi\Big(y, \gamma\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\big)\Big)\right].$$

Then Eq. (27) becomes

$$\begin{aligned}
&\mathbf{E}_x\left[\|\widehat{\mathbf{A}}\widetilde{\boldsymbol{\eta}}(x) - (\widehat{\alpha}\boldsymbol{\gamma}^{-1}(\mathbf{f}(x)) + \widehat{\beta}\mathbf{1})\|_2^2\right] \\
&= \mathbf{E}_x\left[\widehat{\alpha}^2\sum_{j=1}^r\left[(\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j - \gamma^{-1}(f_j(x))\right]^2\right] \\
&\le \mathbf{E}_x\left[\widehat{\alpha}^2\sum_{j=1}^r\frac{1}{2}\mathbf{E}_{y\sim\text{Bernoulli}\left((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\right)}\left[\phi\Big(y, f_j(x)\Big) - \phi\Big(y, \gamma\big((\widehat{\mathbf{A}}'\widetilde{\boldsymbol{\eta}}(x))_j\big)\Big)\right]\right] \\
&= \frac{\widehat{\alpha}^2}{2}\mathbf{E}_x\left[\mathbf{E}_{\mathbf{y}|x\sim\widetilde{\boldsymbol{\eta}}(x)}\left[\psi(\mathbf{y}, \mathbf{f}(x)) - \inf_{\mathbf{u}\in\mathbb{R}^r}\psi(\mathbf{y}, \mathbf{u})\right]\right] \\
&= \frac{\widehat{\alpha}^2}{2}\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]. \qquad (28)
\end{aligned}$$

Combining Eqs. (25), (26) and (28) and applying Jensen's inequality (to the convex function $x \mapsto x^2$), we have

$$\text{regret}_D^{\mathbf{L}}[\mathbf{h}] \le 2\max_{\widehat{\mathbf{y}}}\|\mathbf{b}_{\widehat{\mathbf{y}}}\|_2 \cdot \left(\|\mathbf{A}\|_2\|(\mathbf{C}^\top)^{-1} - (\widehat{\mathbf{C}}^\top)^{-1}\|_2 + \widehat{\alpha}\sqrt{\frac{1}{2}\text{regret}_{\widetilde{D}}^{\psi}[\mathbf{f}]}\right).$$

We can further bound $\widehat{\alpha}$ by

$$\begin{aligned}
\widehat{\alpha} &= \widehat{a}_{\max} - \widehat{a}_{\min} \\
&\le 2\|\widehat{\mathbf{A}}\|_1 \\
&= 2\|\mathbf{A}(\widehat{\mathbf{C}}^\top)^{-1}\|_1 \\
&\le 2\|\mathbf{A}\|_1\|(\widehat{\mathbf{C}}^\top)^{-1}\|_1.
\end{aligned}$$

□

C Supplement to Section 6

Under an STSN model where the set of s tags $\mathcal{T} = [s]$ is partitioned into K ($K \leq s$) groups of tags $G_1, ..., G_K$, the label space $\mathcal{Y} \subseteq \{0,1\}^s$ contains only $\mathbf{y} \in \{0,1\}^s$ with $\|\mathbf{y}_{G_k}\|_1 \leq 1$ for all groups G_k (since each group has at most 1 active tag). In this setting as well, the output mappings $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ that appear at the end of the NCEFP and NCOC algorithms, that require solving a combinatorial optimization problem over $\widehat{\mathbf{y}} \in \mathcal{Y}$, can be computed efficiently. We give details below.

Reduced low-rank decomposition for $\mathbf{L}^{F_1}$ for sparse label space $\mathcal{Y}$, and reduced vector function $\widehat{\mathbf{f}}$. Let $\mathcal{Y}_K = \{\mathbf{y} \in \{0,1\}^s : \|\mathbf{y}\|_1 \leq K\}$. For $\mathcal{Y} \subseteq \mathcal{Y}_K$ (as is the case under the STSN model), in the decomposition of $\mathbf{L}^{F_1}$ in Example 2, several entries of matrix $\mathbf{A}$ become 0; in particular, $a_{jk,\mathbf{y}} = 0$ for $k > K$. Therefore, the factorization in this case simplifies to

$$\ell^{F_1}_{\mathbf{y},\widehat{\mathbf{y}}} = 1 - \mathbf{1}(\|\mathbf{y}\|_1 = 0) \cdot \mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0) - \sum_{j=1}^{s} \sum_{k=1}^{K} \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \cdot \frac{2 \cdot \widehat{y}_j}{k + \|\widehat{\mathbf{y}}\|_1}. \tag{29}$$

In other words, we have $\mathbf{L}^{F_1} = (\mathbf{A}^{\text{red}})^\top (\mathbf{B}^{\text{red}}) + \mathbf{1}\mathbf{t}^\top$ where $\mathbf{A}^{\text{red}} \in [0,1]^{(sK+1)\times|\mathcal{Y}|}$ with $a^{\text{red}}_{0,\mathbf{y}} = a_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)$ and $a^{\text{red}}_{jk,\mathbf{y}} = a_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k) \cdot y_j \ \forall j \in [s], \forall k \in [K]$, $\mathbf{B}^{\text{red}} \in \mathbb{R}^{(sK+1)\times|\mathcal{Y}|}$ with $b^{\text{red}}_{0,\widehat{\mathbf{y}}} = b_{0,\widehat{\mathbf{y}}} = -\mathbf{1}(\|\widehat{\mathbf{y}}\|_1 = 0)$ and $b^{\text{red}}_{jk,\widehat{\mathbf{y}}} = b_{jk,\widehat{\mathbf{y}}} = -\frac{2\cdot\widehat{y}_j}{k+\|\widehat{\mathbf{y}}\|_1} \ \forall j \in [s], \forall k \in [K]$, and $\mathbf{t} \in \mathbb{R}^{|\mathcal{Y}|}$ with $t_{\widehat{\mathbf{y}}} = 1$. Accordingly, the NCEFP algorithm in this case reduces to solving one binary problem and s ($(K+1)$-class) multiclass problems. The NCOC algorithm requires solving $(sK+1)$ binary problems. Therefore, in both cases, one needs to learn only a $(sK+1)$-dimensional real-valued vector function $\widehat{\mathbf{f}} : \mathcal{X} \to \mathbb{R}^{sK+1}$; vector predictions $\widehat{\mathbf{f}}(x)$ then need to be mapped to the label space $\mathcal{Y}$ to obtain the final multi-label predictions $\widehat{\mathbf{h}}(x) \in \mathcal{Y}$. The output mappings for F_1-measure below therefore map vectors $\mathbf{u} \in \mathbb{R}^{sK+1}$ to multi-label predictions $\widehat{\mathbf{y}} \in \mathcal{Y}$.

NCEFP output mapping for F_1-measure under STSN. Algorithm 5 specifies the output mapping $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ in this case. The complexity for computing $\mathbf{T}$ is $O(sK^2)$. The complexity for the for loop is $O(sK^2)$. So the total complexity of the output mapping is $O(sK^2)$. (The procedure here is a modification of the procedure described in [6, 49] for the case when $\mathcal{Y} = \{0,1\}^s$. The complexity of the original output mapping in that case is $O(s^3)$; therefore, our complexity of $O(sK^2)$ is an improvement under the STSN setting.)

Algorithm 5 NCEFP output mapping for F_1-measure under STSN

```
1:  Input: Vector $\mathbf{u} = \big(u_0, (u_{jk})_{j=1,...,s,k=1,...,K}\big)^\top \in \mathbb{R}^{sK+1}$
2:  Define matrices $\mathbf{Q} \in [0,1]^{s\times K}$ and $\mathbf{V} \in \mathbb{R}^{K\times K}$ as follows:
        $Q_{j,k} = s(\widetilde{a}_{\max} - \widetilde{a}_{\min})(\gamma^{-1}_{\text{mlog}}(u_{j1}, ..., u_{js}))_k + \widetilde{a}_{\min}$
        $V_{k,l} = \frac{-2}{k+l}$
3:  Compute $\mathbf{T} = \mathbf{Q}\mathbf{V}$
4:  For $l = 1 \ldots K$: // $K$ is the number of groups
5:      For each group $G_k$, define $g^l_k = \text{argmin}_{j\in G_k}\{T_{j,l}\}$
6:      Find the $l$ smallest numbers among $\{T_{g^l_1,l}, ..., T_{g^l_K,l}\}$; call them $T_{j^l_1,l}, \ldots, T_{j^l_l,l}$
7:      Define $\widehat{\mathbf{y}}^{l,*} \in \mathcal{Y} \cap \{\mathbf{y} \in \{0,1\}^s : \|\mathbf{y}\|_1 = l\}$ as follows:
            $\widehat{y}^{l,*}_j = \begin{cases} 1 & \text{if } j \in \{j^l_1, \ldots, j^l_l\} \\ 0 & \text{otherwise.} \end{cases}$
8:      Set $z^*_l = \sum_{j=1}^{s} \widehat{y}^{l,*}_j T_{j,l}$
9:  End for
10: Pick $\widehat{\mathbf{y}}^*$ as follows:
        $\widehat{\mathbf{y}}^* \in \underset{\widehat{\mathbf{y}}\in\{\mathbf{0},\widehat{\mathbf{y}}^{1,*},...,\widehat{\mathbf{y}}^{K,*}\}}{\text{argmin}} -\mathbf{1}(\widehat{\mathbf{y}} = \mathbf{0}) \cdot ((\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma^{-1}_{\log}(u_0) + \widetilde{a}_{\min}) + \mathbf{1}(\widehat{\mathbf{y}} \neq \mathbf{0}) \cdot z^*_{\|\widehat{\mathbf{y}}\|_1}$
11: Output: $\widehat{\mathbf{y}}^* \in \mathcal{Y}$
```

NCOC output mapping for Hamming loss under STSN. Algorithm 6 specifies the output mapping $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ in this case. The total complexity is $O(s)$.

Algorithm 6 NCOC output mapping for Hamming loss under STSN

```
1: Input: Vector $\mathbf{u} \in \mathbb{R}^s$
2: Define $\widehat{\mathbf{y}} \in \{0,1\}^s$ as follows:
   $$\widehat{y}_j = \begin{cases} 1 & \text{if } (\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_j) + \widetilde{a}_{\min} > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$
3: For $k = 1 \ldots K$: // $K$ is the number of groups
4:     if $\|\widehat{\mathbf{y}}_{G_k}\|_1 > 1$:
5:         Set $\widehat{y}_j = 0$ for all $j \in G_k$
6:         Define $j_k = \operatorname{argmax}_{j \in G_k} (\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_j) + \widetilde{a}_{\min}$, and set $\widehat{y}_{j_k} = 1$
7:     End if
8: End for
9: Output: $\widehat{\mathbf{y}} \in \mathcal{Y}$
```

NCOC output mapping for F_1-measure under STSN. Algorithm 7 specifies the output mapping $\widehat{\mathbf{f}}(x) \mapsto \widehat{\mathbf{h}}(x)$ in this case (Algorithm 7 differs from Algorithm 5 in line 2.). The complexity for computing $\mathbf{T}$ is $O(sK^2)$. The complexity for the for loop is $O(sK^2)$. So the total complexity of the output mapping is $O(sK^2)$. (The procedure here is a modification of the procedure described in [6, 49] for the case when $\mathcal{Y} = \{0,1\}^s$. The complexity of the original output mapping in that case is $O(s^3)$; therefore, our complexity of $O(sK^2)$ is an improvement under the STSN setting.)

Algorithm 7 NCOC output mapping for F_1-measure under STSN

```
1: Input: Vector $\mathbf{u} = \big(u_0, (u_{jk})_{j=1,\ldots,s,k=1,\ldots,K}\big)^\top \in \mathbb{R}^{sK+1}$
2: Define matrices $\mathbf{Q} \in [0,1]^{s \times K}$ and $\mathbf{V} \in \mathbb{R}^{K \times K}$ as follows:
   $$Q_{j,k} = (\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_{jk}) + \widetilde{a}_{\min}$$
   $$V_{k,l} = \frac{-2}{k+l}$$
3: Compute $\mathbf{T} = \mathbf{QV}$
4: For $l = 1 \ldots K$: // $K$ is the number of groups
5:     For each group $G_k$, define $g_k^l = \operatorname{argmin}_{j \in G_k} \{T_{j,l}\}$
6:     Find the $l$ smallest numbers among $\{T_{g_1^l,l}, ..., T_{g_K^l,l}\}$; call them $T_{j_1^l,l}, \ldots, T_{j_l^l,l}$
7:     Define $\widehat{\mathbf{y}}^{l,*} \in \mathcal{Y} \cap \{\mathbf{y} \in \{0,1\}^s : \|\mathbf{y}\|_1 = l\}$ as follows:
       $$\widehat{y}_j^{l,*} = \begin{cases} 1 & \text{if } j \in \{j_1^l, \ldots, j_l^l\} \\ 0 & \text{otherwise.} \end{cases}$$
8:     Set $z_l^* = \sum_{j=1}^s \widehat{y}_j^{l,*} T_{j,l}$
9: End for
10: Pick $\widehat{\mathbf{y}}^*$ as follows:
   $$\widehat{\mathbf{y}}^* \in \operatorname*{argmin}_{\widehat{\mathbf{y}} \in \{\mathbf{0}, \widehat{\mathbf{y}}^{1,*}, \ldots, \widehat{\mathbf{y}}^{K,*}\}} -\mathbf{1}(\widehat{\mathbf{y}} = \mathbf{0}) \cdot ((\widetilde{a}_{\max} - \widetilde{a}_{\min})\gamma_{\log}^{-1}(u_0) + \widetilde{a}_{\min}) + \mathbf{1}(\widehat{\mathbf{y}} \neq \mathbf{0}) \cdot z^*_{\|\widehat{\mathbf{y}}\|_1}$$
11: Output: $\widehat{\mathbf{y}}^* \in \mathcal{Y}$
```

Estimation of STSN parameters. We aim to estimate the K noise parameters in the STSN model, $\sigma_k \in [0,1]$ for $k \in [K]$, by exploiting anchor points. In multiclass classification, an instance is an anchor point if it belongs to a class with probability 1 under the clean distribution D. Under the assumption that anchor points exist, several algorithms have been proposed to estimate noise parameters in the multiclass CCN model [19, 27, 45, 48]. Here, we show how to estimate noise parameters in the STSN model by exploiting anchor points. We start with some notations that will facilitate the description of our method. Let $\mathbf{e}^1, ..., \mathbf{e}^s \in \{0,1\}^s$ be the standard basis vectors in

$\mathbb{R}^s$, and $\mathbf{0}$ be the zero vector. For group G_k in the STSN model, let $\mathbf{e}^j_{G_k} \in \{0,1\}^{|G_k|}$ denote the sub-vector of $\mathbf{e}^j$ restricted to tags in G_k. Then, we define anchor points in the STSN model as follows.

Definition 9 (Anchor points)**.** *Let $D = (\mu, \boldsymbol{\eta})$ be the clean distribution, where $\mu(x)$ is the marginal density of x and $\boldsymbol{\eta}(x)$ is the multi-label class probabilities of x. Suppose tag j belongs to group G_k in the STSN model. Then, an instance $\bar{x}^j \in \mathcal{X}$ is called an* anchor point *of tag j if $\mu(\bar{x}^j) > 0$ and*

$$\mathbf{P}_{\mathbf{y}\sim\boldsymbol{\eta}(\bar{x}^j)}(\mathbf{y}_{G_k} = \mathbf{e}^j_{G_k}) = 1\,. \tag{30}$$

Under the assumption that there exists at least one anchor point for group G_k, we can estimate σ_k as follows. Suppose tag j belongs to group G_k in the STSN model and $\bar{x}^j \in \mathcal{X}$ is an anchor point of tag j.

When $|G_k| = 1$, by the description of STSN, we have

$$\begin{aligned}\mathbf{P}_{\widetilde{\mathbf{y}}\sim\widetilde{\boldsymbol{\eta}}(\bar{x}^j)}(\widetilde{y}_j = 1) &= \mathbf{P}_{\mathbf{y}\sim\boldsymbol{\eta}(\bar{x}^j)}(y_j = 1)\cdot(1-\sigma_k) + (1 - \mathbf{P}_{\mathbf{y}\sim\boldsymbol{\eta}(\bar{x}^j)}(y_j = 1))\cdot\sigma_k \\ &= 1 - \sigma_k\,. \end{aligned} \tag{31}$$

Therefore, $\sigma_k = 1 - \mathbf{P}_{\widetilde{\mathbf{y}}\sim\widetilde{\boldsymbol{\eta}}(\bar{x}^j)}(\widetilde{y}_j = 1)$, where $\mathbf{P}_{\widetilde{\mathbf{y}}\sim\widetilde{\boldsymbol{\eta}}(\bar{x}^j)}(\widetilde{y}_j = 1)$ can be estimated from the noisy training sample.

When $|G_k| \geq 2$, we similarly have

$$\begin{aligned}\mathbf{P}_{\widetilde{\mathbf{y}}\sim\widetilde{\boldsymbol{\eta}}(\bar{x}^j)}(\widetilde{\mathbf{y}}_{G_k} = \mathbf{e}^j_{G_k}) &= \mathbf{P}_{\mathbf{y}\sim\boldsymbol{\eta}(\bar{x}^j)}(\mathbf{y}_{G_k} = \mathbf{e}^j_{G_k})\cdot(1-\sigma_k) + \\ &\qquad \sum_{i\in G_k, i\neq j} \mathbf{P}_{\mathbf{y}\sim\boldsymbol{\eta}(\bar{x}^j)}(\mathbf{y}_{G_k} = \mathbf{e}^i_{G_k})\cdot\frac{\sigma_k}{|G_k|-1} \\ &= 1 - \sigma_k\,. \end{aligned} \tag{32}$$

Again, $\sigma_k = 1 - \mathbf{P}_{\widetilde{\mathbf{y}}\sim\widetilde{\boldsymbol{\eta}}(\bar{x}^j)}(\widetilde{\mathbf{y}}_{G_k} = \mathbf{e}^j_{G_k})$, where $\mathbf{P}_{\widetilde{\mathbf{y}}\sim\widetilde{\boldsymbol{\eta}}(\bar{x}^j)}(\widetilde{\mathbf{y}}_{G_k} = \mathbf{e}^j_{G_k})$ can be estimated from the noisy training sample.

Finally, we note that if anchor points are not available, they can be learned from the noisy training sample as well [19, 27, 41, 45, 48].

D Supplement to Section 7

D.1 Synthetic data: data generating process for F_1-measure under STSN

We generated a multi-label dataset with instances x in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 10$ tags with $K = 5$ groups $\mathcal{G} = \{\{1,2,3\},\{4,5,6\},\{7,8\},\{9\},\{10\}\}$, so $|\mathcal{Y}| = 192$ and $\|\mathbf{y}\|_1 \leq K = 5$ for all $\mathbf{y} \in \mathcal{Y}$. By the reduced factorization of $\mathbf{L}^{F_1}$ in Eq. (29) in Appendix C, in this case, $\mathbf{A}^{\text{red}} \in [0,1]^{(sK+1)\times|\mathcal{Y}|} = [0,1]^{51\times 192}$ consists of:

$$a^{\text{red}}_{0,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = 0)\,; \qquad a^{\text{red}}_{jk,\mathbf{y}} = \mathbf{1}(\|\mathbf{y}\|_1 = k)\cdot y_j \quad \forall j \in [s],\, \forall k \in [K]\,.$$

We first fixed matrix $\mathbf{W} \in [0.1,1]^{51\times 100}$ with entries drawn uniformly; we checked that $\mathbf{W}$ has full row rank. We also fixed a vector $\boldsymbol{\alpha} \in [0.01, 0.02]^{192}$ with entries drawn uniformly. To generate a data point $(x,\mathbf{y})$, we then did the following: we first sampled $\boldsymbol{\eta}(x) \in \Delta_{192} \equiv \Delta_{|\mathcal{Y}|}$ from Dirichlet$(\boldsymbol{\alpha})$. We set $\mathbf{q}(x) = \mathbf{A}^{\text{red}}\boldsymbol{\eta}(x) \in [0,1]^{51}$. We then took $x = \mathbf{W}^{\dagger}\mathbf{q}(x)$, and drew $\mathbf{y} \sim \boldsymbol{\eta}(x)$, where $\mathbf{W}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{W}$.

D.2 Synthetic data: Hamming loss under IFN

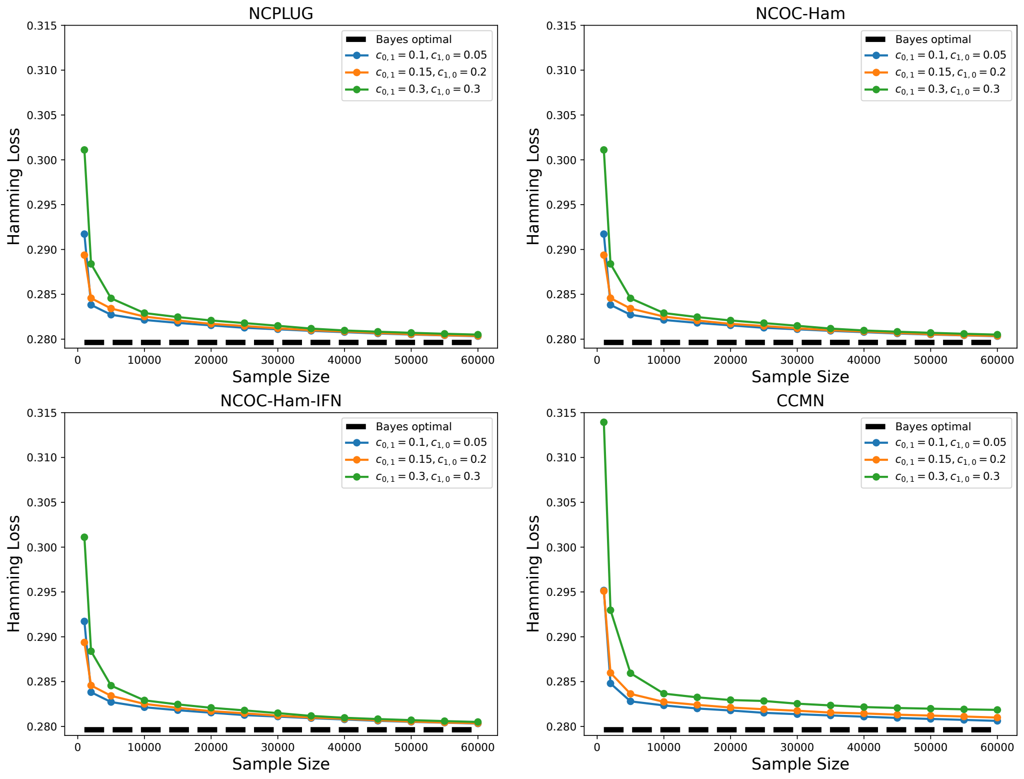


Figure 4: Sample complexity behavior of NCPLUG, NCOC-Ham, NCOC-Ham-IFN, and CCMN on synthetic multi-label data for 3 pairs of noise parameters under the IFN model. Performance measure is Hamming loss. For NCPLUG, NCOC-Ham and NCOC-Ham-IFN algorithms, as the overall noise increases, the sample size needed to reach a given level of performance generally increases.

We tested the sample complexity behavior of the NCPLUG algorithm, the NCOC-Ham algorithm, the NCOC-Ham-IFN algorithm, and the CCMN algorithm [44].

Specifically, we generated a multi-label dataset with instances x in $\mathcal{X} = \mathbb{R}^{100}$ and $s = 8$. Here, $\mathcal{Y} = \{0,1\}^8$. By the factorization of $\mathbf{L}^{\text{Ham}}$ in Eq. (3), in this case, $\mathbf{A} \in [0,1]^{s\times 2^s} = [0,1]^{8\times 256}$. We first fixed matrix $\mathbf{W} \in [0.1,1]^{8\times 100}$ with entries drawn uniformly; we checked that $\mathbf{W}$ has full row rank. Since labels $\mathbf{y}$ in real data tend to be very sparse (have few active tags), we simulated this observation by considering a subset of $\mathcal{Y}$, denoted by $\mathcal{Y}_4$, that contains labels $\mathbf{y}$ with $\|\mathbf{y}\|_1 \leq 4$, so $\mathcal{Y}_4 = \{\mathbf{y} \in \mathcal{Y} : \|\mathbf{y}\|_1 \leq 4\}$. Then we fixed a vector $\boldsymbol{\alpha} \in [0.01, 0.02]^{|\mathcal{Y}_4|}$ with entries drawn uniformly. To generate a data point $(x, \mathbf{y})$, we first sampled $\bar{\boldsymbol{\eta}}(x) \in \Delta_{|\mathcal{Y}_4|}$ from Dirichlet$(\boldsymbol{\alpha})$. Then we set $\boldsymbol{\eta}(x) \in \Delta_{2^s} \equiv \Delta_{|\mathcal{Y}|}$ as follows: for $\mathbf{y} \in \mathcal{Y}$, if $\mathbf{y} \in \mathcal{Y}_4$, set the $\mathbf{y}$-indexed entry of $\boldsymbol{\eta}(x)$ to be the value in the $\mathbf{y}$-indexed entry of $\bar{\boldsymbol{\eta}}(x)$; if $\mathbf{y} \notin \mathcal{Y}_4$, then set the $\mathbf{y}$-indexed entry of $\boldsymbol{\eta}(x)$ to be a small value 10^{-3}. Afterwards, we re-normalized $\boldsymbol{\eta}(x)$. We set $\mathbf{q}(x) = \mathbf{A}\boldsymbol{\eta}(x) \in [0,1]^8$. We then took $x = \mathbf{W}^\dagger \mathbf{q}(x)$, and drew $\mathbf{y} \sim \boldsymbol{\eta}(x)$, where $\mathbf{W}^\dagger$ denotes the pseudo-inverse of $\mathbf{W}$.

We then added label noise using 3 sets of IFN noise rates: specifically, we let $c_{1,0}^{(j)} = c_{1,0}$ and $c_{0,1}^{(j)} = c_{0,1}$ for all j, and chose 3 pairs of values of $(c_{0,1}, c_{1,0})$: $(0.1, 0.05)$, $(0.15, 0.2)$, $(0.3, 0.3)$. We ran all algorithms (with a linear function class) to learn multi-label classifiers from increasingly large noisy training samples generated in this way, and measured the Hamming loss on a test set of $10,000$ clean data points. The results are shown in Figure 4. We see that, as suggested by our regret bounds, as the overall noise increases, the sample size needed to reach a given level of performance generally increases.

D.3 Synthetic data: additional implementation details

For all synthetic data experiments (and for all algorithms in the experiments), we used the Adam optimizer [15] provided by PyTorch [25] with batch size 100 and no weight decay. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was halved at the end of every 5 epochs.

D.4 Synthetic data: computer resources and computation time

We ran all experiments on a desktop with one AMD Threadripper 3960X CPU, two NVIDIA RTX 3090 GPUs, 64GB RAM, and 1TB SSD.

Computation times for F_1-measure experiments under STSN.

Table 4: Approximate computation times for F_1-measure experiments under STSN.

	NCOC-F_1	**NCEFP**
Training time on 20,000 examples	11 s	67 s
Total time to run all experiments to generate the plot in Figure 2	30 min	140 min

Computation times for Hamming loss experiments under IFN.

Table 5: Approximate computation times for Hamming loss experiments under IFN.

	NCPLUG	**NCOC-Ham**	**NCOC-Ham-IFN**	**CCMN**
Training time on 10,000 examples	5 s	5 s	5 s	30 s
Total time to run all experiments to generate the plot in Figure 4	15 min	15 min	15 min	40 min

D.5 Real data: additional implementation details for experiments on Mediamill data

Regularization parameters were chosen by cross-validation from $\{0, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. We used the Adam optimizer provided by PyTorch with batch size 100. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was halved at the end of every 5 epochs.

D.6 Real data: experiments on Multi-MNIST data

Here we report results for the Multi-MNIST dataset (which was also used in [44]).[3]

Multi-MNIST dataset. We started with the TripleMNIST dataset (each image contains 3 digits; see Figure 5) in the Multi-MNIST repository and applied the following data processing steps.

[3]`https://github.com/shaohua0116/MultiDigitMNIST`, MIT License.

• Triple MNIST Datasets (1000 classes: `000` to `999`)

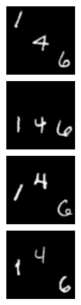
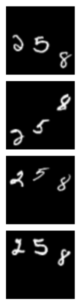
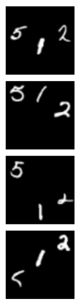
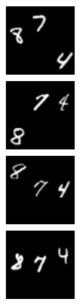
Class	039	146	258	512	874
Image					

Figure 5: Examples in TripleMNIST. Pictures were taken from `https://github.com/shaohua0116/MultiDigitMNIST`.

The original TripleMNIST has 1,000 images for each of 1,000 classes (000, 001, ..., 999), with feature dimension $84 \times 84 = 7,056$. We sampled 100 images for each class. For each instance x, we created a label vector $\mathbf{y} \in \{0,1\}^{10}$ in which $y_j = 1$ if digit $j-1$ is present in x, and 0 otherwise. Motivated by the noise model in [27], we divided the 10 tags into 5 groups: { {`2`, `7`, `1`}, {`5`, `6`}, {`3`, `8`}, {`0`, `9`}, {`4`} }. For consistency with the STSN model assumption, we removed instances that have more than one active tag in any group. We did not change the features in this process. Our modified Multi-MNIST dataset has 68,800 examples with 10 tags. Since the original data did not come with prescribed train/test splits, we split the data into training and test sets with ratio $8:2$. So we ended up with 55,040 training examples and 13,760 test examples.

Hamming loss under IFN. For IFN, we let $c^{(j)}_{1,0} = c_{1,0}$ and $c^{(j)}_{0,1} = c_{0,1}$ for all j, and chose noise parameters of the form $(c_{0,1}, c_{1,0})$. We compared our NCPLUG and NCOC-Ham-IFN algorithms with CCMN [44] and basic OC-Ham/BR [46]. We chose the following neural network architecture for all algorithms:[4]

```
model = torch.nn.Sequential(
    torch.nn.Conv2d(1, 32, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(32, 64, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(64, 128, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Flatten(),
    torch.nn.LazyLinear(128),
    torch.nn.ReLU(),
    torch.nn.LazyLinear(output_dim)
)
```

The results are shown in Table 6. As seen, our NCPLUG and NCOC-Ham-IFN algorithms often outperform other baselines.

Hamming loss and F_1-measure under STSN. For STSN model, we used single-parameter noise matrices respecting the partition into $K = 5$ groups described above, with $\sigma_1, ..., \sigma_5 = \sigma$. We compared our NCOC-Ham algorithm with basic OC-Ham/BR [46], as well as our NCOC-F_1 and NCEFP algorithms with basic OC-F_1 [49] and EFP [5].

[4]We used the Adagrad optimizer [8] provided by PyTorch with batch size 128 and weight decay 0.001. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was tuned automatically by the optimizer.

Table 6: Hamming loss on (modified) Multi-MNIST data with IFN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter $(c_{0,1}, c_{1,0})$	**Noise level (%)**	**NCPLUG**	**NCOC-Ham-IFN**	**CCMN**	**OC-Ham/BR**
(0.1,0.2)	74.18	**6.74±0.08**	6.8±0.09	9.9±0.31	6.87±0.14
(0.15,0.4)	91.89	12.82±0.29	**12.72±0.44**	16.04±0.65	14.17±0.26
(0.25,0.45)	97.43	20.01±0.24	20.36±0.17	21.92±0.7	**18.01±0.18**

For Hamming loss under STSN, we chose the following neural network architecture for all algorithms:[5]

```
model = torch.nn.Sequential(
    torch.nn.Conv2d(1, 32, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(32, 64, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(64, 128, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Flatten(),
    torch.nn.LazyLinear(128),
    torch.nn.ReLU(),
    torch.nn.LazyLinear(output_dim)
)
```

The results are shown in Table 7. As seen, our noise-corrected algorithm often outperforms the other baseline.

Table 7: Hamming loss on (modified) Multi-MNIST data with STSN model (*lower* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCOC-Ham**	**OC-Ham/BR**
0.1	28.96	**4.83±0.13**	6.13±0.22
0.2	51.11	**9.11±0.75**	10.28±0.54
0.3	68.14	16.9±1.21	**16.21±0.35**
0.4	80.23	28.06±1.45	**22.5±0.21**
0.6	94.29	**27.38±1.62**	34.19±0.05

For F_1-measure under STSN, we chose the following neural network architecture for all algorithms:[6]

```
model = torch.nn.Sequential(
    torch.nn.Conv2d(1, 32, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Conv2d(32, 64, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
```

[5] We used the Adagrad optimizer provided by PyTorch with batch size 128 and weight decay 0. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was tuned automatically by the optimizer.

[6] We used the Adagrad optimizer provided by PyTorch with batch size 128 and weight decay 0. The optimizer was run for 50 epochs over the training sample; the learning rate parameter was initially set to 0.01 and was tuned automatically by the optimizer.

```
    torch.nn.Conv2d(64, 128, 3, padding=1),
    torch.nn.ReLU(),
    torch.nn.MaxPool2d(2, 2),
    torch.nn.Flatten(),
    torch.nn.LazyLinear(512),
    torch.nn.ReLU(),
    torch.nn.LazyLinear(output_dim)
)
```

The results are shown in Table 8. Again, our noise-corrected algorithms often outperform other baselines.

Table 8: F_1-measure on (modified) Multi-MNIST data with STSN model (*higher* values are better). Performance values are in %, and are reported as Mean±SEM over five random trials. Noise level is $\frac{1}{m}\sum_{i=1}^{m}\mathbf{1}(\mathbf{y}_i \neq \widetilde{\mathbf{y}}_i)$.

Noise parameter (σ)	**Noise level (%)**	**NCEFP**	**NCOC-F_1**	**EFP**	**OC-F_1**
0.1	28.96	**91.14±0.16**	90.13±0.18	83.77±0.24	78.33±0.37
0.2	51.11	**84.98±1.24**	84.95±0.52	74.16±0.23	69.61±0.35
0.3	68.14	73.55±1.47	**75.74±0.35**	64.61±0.77	60.71±0.6
0.4	80.23	53.64±0.81	51.48±1.83	**53.87±0.6**	50.97±0.79
0.6	94.29	**43.82±1.72**	41.93±2.02	31.82±0.24	32.57±0.47

D.7 Real data: computer resources and computation time

We ran all experiments on a desktop with one AMD Threadripper 3960X CPU, two NVIDIA RTX 3090 GPUs, 64GB RAM, and 1TB SSD.

Computation times for experiments on Mediamill dataset. See Table 9, Table 10, and Table 11 for computation time of experiments for Hamming loss with IFN model, Hamming loss with STSN model, and F_1-measure with STSN model, respectively.

Computation times for experiments on Multi-MNIST dataset. See Table 12, Table 13, and Table 14 for computation time of experiments for Hamming loss with IFN model, Hamming loss with STSN model, and F_1-measure with STSN model, respectively.

Table 9: Approximate computation time of experiments for Hamming loss on (modified) Mediamill data with IFN model. (CV = cross validation)

	NCPLUG	NCOC-Ham-IFN	CCMN	OC-Ham/BR
Training time over full training set (w/o CV)	10 s	10 s	106 s	10 s
Training time over full training set (w/ CV)	59 s	58 s	602 s	58 s
Inference time over full test set	< 0.1 s	< 0.1 s	< 0.1 s	< 0.1 s

Table 10: Approximate computation time of experiments for Hamming loss on (modified) Mediamill data with STSN model. (CV = cross validation)

	NCOC-Ham	OC-Ham/BR
Training time over full training set (w/o CV)	10 s	10 s
Training time over full training set (w/ CV)	60 s	57 s
Inference time over full test set	0.4 s	0.4 s

Table 11: Approximate computation time of experiments for F_1-measure on (modified) Mediamill data with STSN model. (CV = cross validation)

	NCEFP	NCOC-F_1	EFP	OC-F_1
Training time over full training set (w/o CV)	73 s	10 s	96 s	10 s
Training time over full training set (w/ CV)	451 s	63 s	533 s	61 s
Inference time over full test set	5.3 s	2.6 s	5.3 s	2.1 s

Table 12: Approximate computation time of experiments for Hamming loss on (modified) Multi-MNIST data with IFN model.

	NCPLUG	NCOC-Ham-IFN	CCMN	OC-Ham/BR
Training time over full training set	171 s	171 s	225 s	171 s
Inference time over full test set	0.5 s	0.5 s	0.5 s	0.5 s

Table 13: Approximate computation time of experiments for Hamming loss on (modified) Multi-MNIST data with STSN model.

	NCOC-Ham	OC-Ham/BR
Training time over full training set	170 s	167 s
Inference time over full test set	0.9 s	0.9 s

Table 14: Approximate computation time of experiments for F_1-measure on (modified) Multi-MNIST data with STSN model.

	NCEFP	NCOC-F_1	EFP	OC-F_1
Training time over full training set	254 s	181 s	250 s	177 s
Inference time over full test set	5.8 s	2.7 s	5.7 s	2.6 s

E Broader impacts

Our efforts are centered around fundamental research in the broad area of learning from noisy labels. More specifically, we have developed a number of consistent noise-corrected multi-label learning algorithms, encompassing a variety of multi-label performance measures and general class-conditional noise (CCN) models, together with supporting theory and experimental validation. This work can have the positive broader impact of enabling the learning of more accurate multi-label classification models from noisy data. We do not foresee any negative broader impact of the work described here.

F Limitations

Our work proposes three consistent noise-corrected multi-label learning algorithms, encompassing a variety of multi-label performance measures and general class-conditional noise (CCN) models. The algorithms do not apply to noise models outside the broad family of CCN models. Of course, in the multi-label setting, general CCN models are computationally too expensive to work with, and therefore in practice, one needs to work with suitably structured subclasses of CCN models. Within the family of CCN models, while our algorithms are consistent for all noise models in the family, they are computationally efficient when the noise matrix $\mathbf{C}$ can be inverted efficiently, as is the case with the similar-tag switching noise (STSN) models that we have proposed (we have also given computationally efficient algorithms for the special case of Hamming loss under the independent flipping noise (IFN) model). One limitation of the STSN model is that it only allows at most one active tag within each group of similar/related tags. It remains to identify other natural classes of CCN models for which the noise matrix $\mathbf{C}$ can be inverted efficiently, and also to possibly explore the design of alternative consistent algorithms that could be computationally efficient for other classes of CCN models.