Can Infinitely Wide Deep Nets Help Small-data Multi-label Learning?

Guoqiang Wu GUOQIANGWU@SDU.EDU.CN
School of Software, Shandong University

Jun Zhu[*] DCSZJ@TSINGHUA.EDU.CN
Department of Computer Science and Technology, BNRist Center, THU-Bosch Joint ML Center
State Key Laboratory of Intelligent Technology and Systems
Tsinghua University
Beijing, 100084 China

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

In Multi-label Learning (MLL), kernel methods and deep neural networks (DNNs) are two typical families of approaches. Recent theory discovers an interesting connection between infinitely wide DNNs and neural tangent kernel (NTK) based methods. Further, recent work has shown the promising performance of NTK-based methods in *small-data single-labeled tasks*. Then, a natural question arises: can infinitely wide DNNs help small-data multi-label learning? To answer this question, in this paper, we present to utilize infinitely wide DNNs for the MLL task. Specifically, we propose an NTK-based kernel method for MLL, which aims to minimize Hamming and ranking loss simultaneously. Moreover, to efficiently train the model, we use the Nyström method, which has rarely been used in MLL. Further, we give rigorous theoretical analyses on learning guarantees of the proposed algorithm w.r.t. these two measures. Finally, empirical results on small-scale datasets illustrate its superior performance along with efficiency over several related baselines.

Keywords: Multi-label Learning; Infinitely Wide Neural Network; Neural Tangent Kernel.

1. Introduction

Multi-label Learning (MLL) (or Multi-label Classification, MLC) (Tsoumakas and Katakis, 2006) finds applications in various areas, such as computer vision (Boutell et al., 2004), natural language processing (Zhang and Zhou, 2006), and bioinformatics (Elisseeff et al., 2001). Different from typical (single-labeled) classification tasks, each example can be assigned with multiple labels simultaneously in MLL, making it more challenging. Besides, to evaluate its performance, many MLL-specific measures (Zhang and Zhou, 2014) have been developed from diverse aspects, such as Hamming Loss (HL), Ranking Loss (RL), etc. To solve the MLL task, various algorithms (Zhang and Zhou, 2014; Wu and Zhu, 2020) have been proposed to optimize one or few specific measures. Among them, there are two typical families of approaches based on the choice of the hypothesis space (i.e., the set of input-output mapping functions).

[*] Corresponding author

The first family of approaches in MLL is based on kernel methods (Schölkopf and Smola, 2002; Cai et al., 2022), where the hypothesis space is the associated reproducing kernel Hilbert space (RKHS). Among them, BR-SVM (Boutell et al., 2004) and Rank-SVM (Elisseeff et al., 2001) are two typical methods, which aim to optimize HL and RL, respectively. Despite kernel methods being well understood in theory, their empirical performance highly depends on the type of kernel functions. However, most of these kernel methods in MLL use prefixed Gaussian or linear kernel functions and utilize classical training algorithms, which involve the kernel matrix and might lack scalability, especially in large-scale settings.

The second family of approaches in MLL is based on neural networks (NNs), which can learn abstract representations from data and have shown promise in many large-scale applications. Among them, BP-MLL (Zhang and Zhou, 2006) is one classical NN-based method, which uses the backpropagation learning algorithm to optimize the RL. Recently, deep neural networks (DNNs) have achieved promising performance in various application fields of MLL, such as computer vision (Li et al., 2017; Wu et al., 2020b), and natural language processing (Liu et al., 2017; Chang et al., 2020). However, such methods are often treated as a black box and have not yet been understood deeply in theory.

Remarkably, recent theory (Jacot et al., 2018) discovers an interesting connection between kernel methods and neural networks. Specifically, for infinitely wide NNs with random initialization, gradient descent with infinitesimally small learning rates (i.e., gradient flow) equivalently optimizes the prediction function of kernel methods in the RKHS induced by the neural tangent kernel (NTK). For example, the prediction of infinitely wide NNs with ℓ_2 loss under the gradient flow and random initialization is equivalent to the kernel regression prediction. The NTK with the specific architecture of NNs can be viewed as a new powerful kernel function, which can be used in kernel methods. Further, recent work (Arora et al., 2020) has empirically shown its superior performance to Gaussian kernel methods for *small-data single-labeled tasks.* However, the existing work mainly focuses on single-labeled tasks, while leaving the MLL setting largely under-explored. Then, a natural question arises here:

Can infinitely wide deep NNs help small-data multi-label learning?

To answer the above question, in this paper, we aim to utilize the infinitely wide DNNs to solve small-data MLL tasks. This not only can help understand existing methods that connect kernel methods and NNs, but also might make the NTK-based method a good candidate in practice. Specifically, we propose a novel NTK-based kernel method called MLC-NTK, which aims to optimize HL and RL simultaneously. For MLC-NTK, if the Tikhonov regularization is discarded, its prediction is equivalent to the output function of the corresponding infinitely wide NN under gradient flow and random initialization. Moreover, to efficiently train the kernel model, we use the Nyström method, which has rarely been used in MLL to our knowledge (Mehrkanoon and Suykens, 2016). Further, we give formal theoretical analyses about the learning guarantees of the proposed method w.r.t. these two measures. Finally, experimental results on small-scale datasets demonstrate the superior performance and efficiency of MLC-NTK in comparison with the related baselines.

2. Related Work

2.1. Multi-label Learning

MLL has been widely studied in recent years, and many approaches have been proposed. According to the taxonomy (Zhang and Zhou, 2014), they can be mainly divided into two categories: problem transformation and algorithm adaptation approaches.

The core idea of problem transformation approaches is to transform MLL to current well-studied learning tasks, such as binary classification (e.g., Binary Relevance (Boutell et al., 2004)) and multi-class classification (e.g., RAKEL (Tsoumakas et al., 2011)).

The main idea of algorithm adaptation approaches lies in adjusting existing learning methods to adapt to MLL directly, such as K-nearest neighbor method (e.g., ML-KNN (Zhang and Zhou, 2007)), neural network-based method (e.g., BP-MLL (Zhang and Zhou, 2006), LLSL (Hsieh et al., 2018)), and kernel-based method (e.g., Rank-SVM (Elisseeff et al., 2001), CPNL (Wu et al., 2018), mlODM (Tan et al., 2020), RBRL (Wu et al., 2020a)), etc. Although various methods based on different base models might share similar loss functions, they construct different hypothesis spaces, which may result in different expressive power and generalization performance. Thus, it is essential to explore the powerful and effective hypothesis space to improve performance.

We mention that the combination of surrogate HL and RL terms with the least squared hinge base loss has been partially used in prior work (Wu et al., 2020a), which utilizes the RBF or linear kernel and classical training algorithms without formal generalization analyses. In comparison, in this paper, we provide a more general form of the base loss and consider the NTK kernel with rigorous theoretical analyses. Besides, we employ the Nyström method which is more efficient to train the kernel model.

2.2. Neural Tangent Kernel

Previous work (Lee et al., 2018) has found the connection between the kernel methods and infinitely wide neural networks (NNs). However, these kernels correspond to NNs where only the last layer is trained. Recently, Jacot et al. (2018) remarkably proposed the neural tangent kernel (NTK), which is different from previous kernels. Specifically, NTK corresponds to NNs where all the layers are trained, which can have more expressive power. Since NTK is induced from a specific NN architecture, different architectures of NNs can induce different neural tangent kernels, such as convolutional neural tangent kernel (CNTK) (Arora et al., 2019), and graph neural tangent kernel (GNTK) (Du et al., 2019), which have shown good performance in respective fields. Moreover, Arora et al. (2020) utilized the NTK for binary or multi-class classification, which has shown surprising superiority empirically.

3. Preliminaries

Notations. Let bold-face letters denote vectors, matrices, or tensors. For a matrix $\mathbf{A}$, $\mathbf{a}_i$, $\mathbf{a}^j$ and a_{ij} denote its i-th row, j-th column, and (i, j)-th element respectively. For a function $g(\cdot) : \mathbb{R} \to \mathbb{R}$ and $\mathbf{A} \in \mathbb{R}^{m\times n}$, define $g(\mathbf{A}) : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$, where $g(\mathbf{A})_{ij} = g(a_{ij})$. $\mathrm{Tr}(\cdot)$ denotes the trace operator for a square matrix. Let $\circ$ and $\otimes$ denote the Hadamard (element-wise) product and tensor product respectively. $\mathbf{1}$ denotes the vector with ones. $[\![\pi]\!]$ returns 1 when the proposition π holds, or 0 otherwise. $[n]$ denotes the set $\{1, 2, ..., n\}$.

Problem Setting. Given a multi-label dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ which is i.i.d. sampled from a distribution P, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input, d is the feature dimension, $\mathbf{y}_i \in \{-1, +1\}^c$ is the corresponding label vector, c is the number of potential labels, and n is the number of data points. Besides, $y_{ij} = 1$(or -1) indicates the j-th label is relevant (or irrelevant) with $\mathbf{x}_i$. The task of MLL is to learn a multi-label classifier $H : \mathbb{R}^d \to \{-1, +1\}^c$. A typical approach is often to first learn a score function $f = [f_1, \ldots, f_c] : \mathbb{R}^d \to \mathbb{R}^c$, and then get the classifier via a thresholding function, where we set it as $sign(\cdot)$ in this paper.

Evaluation Measure. Here we mainly consider two evaluation measures, i.e., Hamming Loss (HL) and Ranking Loss (RL), which can be defined for each example $(\mathbf{x}_i, \mathbf{y}_i)$, as follows:

$$\text{HL}: \quad L_h^{0/1}(f(\mathbf{x}_i), \mathbf{y}_i) = \frac{1}{c}\sum_{j=1}^{c} [\![sgn(f_j(\mathbf{x}_i)) \neq y_{ij}]\!], \tag{1}$$

$$\text{RL}: \quad L_r^{0/1}(f(\mathbf{x}_i), \mathbf{y}_i) = \frac{1}{|Y_i^+||Y_i^-|}\sum_{(p,q)\in Y_i^+\times Y_i^-} [\![f_p(\mathbf{x}_i) \leq f_q(\mathbf{x}_i)]\!], \tag{2}$$

where Y_i^+ (or Y_i^-) denotes the index set of relevant (or irrelevant) labels for $\mathbf{x}_i$.

4. Methodology

In this section, we first introduce the infinitely wide NN model for MLL and derive its corresponding kernel model with NTK. Then, to improve its generalization performance, we add the Tikhonov regularization into the learning objective. Finally, we propose an efficient optimization algorithm to solve the kernel model.

4.1. Infinitely Wide Neural Network

Formally, we define a multilayer fully-connected neural network. For convenience, let $g^{(0)}(\mathbf{x}) = \mathbf{x}$ and $d_0 = d$. It can be defined recursively as follows

$$f^{(h)}(\mathbf{x}) = \mathbf{W}^{(h)} g^{(h-1)}(\mathbf{x}), \quad g^{(h)}(\mathbf{x}) = \sqrt{\frac{c_\sigma}{d_h}}\sigma(f^{(h)}(\mathbf{x})), \quad \forall h \in [L], \tag{3}$$

where $f^{(h)}(\mathbf{x}) = \left[f_1^{(h)}(\mathbf{x}), \ldots, f_{d_h}^{(h)}(\mathbf{x})\right]$, $g^{(h)}(\mathbf{x}) = \left[g_1^{(h)}(\mathbf{x}), \ldots, g_{d_h}^{(h)}(\mathbf{x})\right]$, $\mathbf{W}^{(h)} \in \mathbb{R}^{d_h \times d_{h-1}}$, $\sigma : \mathbb{R} \to \mathbb{R}$ is the pointwise activation function, and $c_\sigma = (\mathbb{E}_{x\sim\mathcal{N}(0,1)}[\sigma(x)^2])^{-1}$ is the scaling factor. Here we use the ReLU activation function, i.e., $\sigma(x) = \max(0, x)$. The last layer of the neural network is

$$f(\mathbf{x}, \boldsymbol{\theta}) = f^{(L+1)}(\mathbf{x}) = \mathbf{W}^{(L+1)} g^{(L)}(\mathbf{x}), \tag{4}$$

where $f(\mathbf{x}, \boldsymbol{\theta}) = [f_1(\mathbf{x}, \boldsymbol{\theta}), \ldots, f_c(\mathbf{x}, \boldsymbol{\theta})]$, $\mathbf{W}^{(L+1)} \in \mathbb{R}^{c\times d_L}$, and $\boldsymbol{\theta} \equiv \text{vec}\ (\mathbf{W}^{(1)}, ..., \mathbf{W}^{(L+1)})$ denotes all the parameters of the network. Here, all the parameters are i.i.d. initialized as $\mathcal{N}(0, 1)$.

For MLL, we minimize the following objective function, which aims to optimize the HL and RL simultaneously:

$$C(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i, \boldsymbol{\theta}), \mathbf{y}_i) = \frac{1}{n}\sum_{i=1}^{n}\Big(L_h(f(\mathbf{x}_i, \boldsymbol{\theta}), \mathbf{y}_i) + \lambda L_r(f(\mathbf{x}_i, \boldsymbol{\theta}), \mathbf{y}_i)\Big), \tag{5}$$

where λ is a tradeoff hyper-parameter and

$$L_h(f(\mathbf{x}_i, \boldsymbol{\theta}), \mathbf{y}_i) = \frac{1}{c}\sum_{j=1}^{c} \ell_{base}(y_{ij} f_j(\mathbf{x}_i, \boldsymbol{\theta})), \tag{6}$$

$$L_r(f(\mathbf{x}_i, \boldsymbol{\theta}), \mathbf{y}_i) = \frac{1}{|Y_i^+||Y_i^-|}\sum_{(p,q)\in Y_i^+\times Y_i^-} \ell_{base}(f_p(\mathbf{x}_i, \boldsymbol{\theta}) - f_q(\mathbf{x}_i, \boldsymbol{\theta})). \tag{7}$$

whereℓ_{base} denotes the base (convex) surrogate loss (e.g., the hinge and logistic loss). In experiments, we adopt the logistic(-like) loss, i.e., $\ell_{base}(yf(\mathbf{x})) = \log(1+\exp(-yf(\mathbf{x})))$ and $\ell_{base}(f_p(\mathbf{x}) - f_q(\mathbf{x})) = \log(1+\exp(f_q(\mathbf{x}) - f_p(\mathbf{x})))$, due to its widespread use in neural networks.

Calculation of NTK. When the hidden widths go into infinity (i.e., $d_h \to \infty$, $\forall h \in [L]$), each coordinate of the h-layer output (i.e., $f_j^{(h)}(\mathbf{x}), \forall j \in [d_h]$) tends into i.i.d centered Gaussian Processes with the covariance $\Sigma^{(h)}$, where $\Sigma^{(h)}$ is defined recursively as follows. For arbitrary two data points $\mathbf{x}$ and $\mathbf{x}'$, we have

$$\Sigma^{(0)}(\mathbf{x},\mathbf{x}') = \mathbf{x}^\top\mathbf{x}', \quad \Lambda^{(h)}(\mathbf{x},\mathbf{x}') = \begin{pmatrix} \Sigma^{(h-1)}(\mathbf{x},\mathbf{x}) & \Sigma^{(h-1)}(\mathbf{x},\mathbf{x}') \\ \Sigma^{(h-1)}(\mathbf{x}',\mathbf{x}) & \Sigma^{(h-1)}(\mathbf{x}',\mathbf{x}') \end{pmatrix} \in \mathbb{R}^{2\times 2},$$
$$\Sigma^{(h)}(\mathbf{x},\mathbf{x}') = c_\sigma \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda^{(h)})}[\sigma(u)\sigma(v)].$$

Besides, we define a derivative covariance

$$\dot{\Sigma}^{(h)}(\mathbf{x},\mathbf{x}') = c_\sigma \mathbb{E}_{(u,v)\sim\mathcal{N}(\mathbf{0},\Lambda^{(h)})}[\dot\sigma(u)\dot\sigma(v)], \tag{8}$$

where $\dot\sigma(\cdot)$ denotes the derivative function of $\sigma(\cdot)$.

Finally, the *neural tangent kernel (NTK)* is as follows:

$$\begin{aligned}\kappa(\mathbf{x},\mathbf{x}') \triangleq \left\langle \frac{\partial f_j(\mathbf{x},\boldsymbol\theta)}{\partial\boldsymbol\theta}, \frac{\partial f_j(\mathbf{x}',\boldsymbol\theta)}{\partial\boldsymbol\theta} \right\rangle &= \sum_{h=1}^{L+1} \left\langle \frac{\partial f_j(\mathbf{x},\boldsymbol\theta)}{\partial\mathbf{W}^{(h)}}, \frac{\partial f_j(\mathbf{x}',\boldsymbol\theta)}{\partial\mathbf{W}^{(h)}} \right\rangle \\ &= \sum_{h=1}^{L+1} \left(\Sigma^{(h-1)}(\mathbf{x},\mathbf{x}') \cdot \prod_{h'=h}^{L+1} \dot\Sigma^{(h')}(\mathbf{x},\mathbf{x}') \right),\end{aligned} \tag{9}$$

where $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the scalar-based kernel and we set $\dot\Sigma^{(L+1)}(\mathbf{x},\mathbf{x}') = 1$ and kernel matrix $\mathbf{K} = [\kappa(\mathbf{x}_i,\mathbf{x}_j)] \in \mathbb{R}^{n\times n}$ for convenience. We refer readers to Arora et al. (2019) for the derivation details. Besides, if we fix the first L' layers and only train the remaining $(L+1-L')$ layers, it is easy to verify that the resulting NTK is $\kappa(\mathbf{x},\mathbf{x}') = \sum_{h=L'+1}^{L+1}\left(\Sigma^{(h-1)}(\mathbf{x},\mathbf{x}')\cdot\prod_{h'=h}^{L+1}\dot\Sigma^{(h')}(\mathbf{x},\mathbf{x}')\right)$. Similarly to Arora et al. (2020), we also view L' as a hyper-parameter of the NTK classifier, which is tuned in our experiments.

Equivalence between infinitely wide NN and kernel method with NTK.[1]

Proposition 1 *Consider minimizing $C(\boldsymbol\theta)$ in Eq.(5) by gradient descent with infinitesimally small learning rate (i.e., gradient flow): $\frac{\mathrm{d}\boldsymbol\theta(t)}{\mathrm{d}t} = -\nabla C(\boldsymbol\theta(t))$. Let $\mathbf{F}(t) = [f(\mathbf{x}_1,\boldsymbol\theta(t)); \dots; f(\mathbf{x}_n,\boldsymbol\theta(t))] \in \mathbb{R}^{n\times c}$ be network outputs on all $\mathbf{x}_i$'s at time t. When the hidden widths $d_1,\dots,d_L \longrightarrow \infty$, then $\mathbf{F}(t)$ follows the following evolution:*

$$\frac{\mathrm{d}\mathbf{F}(t)}{\mathrm{d}t} = -\frac{1}{n}\mathbf{K}\cdot\frac{\partial\ell(t)}{\partial\mathbf{F}(t)}, \tag{10}$$

where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the NTK kernel matrix and

$$\frac{\partial\ell(t)}{\partial\mathbf{F}(t)} \triangleq \left[\frac{\partial\ell(f(\mathbf{x}_1,\boldsymbol\theta(t)),\mathbf{y}_1)}{\partial f(\mathbf{x}_1,\boldsymbol\theta(t))}; \dots; \frac{\partial\ell(f(\mathbf{x}_n,\boldsymbol\theta(t)),\mathbf{y}_n)}{\partial f(\mathbf{x}_n,\boldsymbol\theta(t))}\right] \in \mathbb{R}^{n\times c}.$$

1. Note that, the subsequent proposition follows the asymptotic analysis w.r.t. the widths in Jacot et al. (2018) rather than the non-asymptotic one in Arora et al. (2019).

Remark 2 *This proposition can be viewed as a corollary of prior theoretical results (i.e., Theorem 1 and 2 in (Jacot et al., 2018)), where the loss function is assumed to be convex with another mild condition, and the general matrix-based kernel is considered. In contrast, it can be easily checked that our loss satisfies these conditions, and here we consider the separable matrix-based kernel that can be equivalently transformed into the corresponding scalar-based kernel (Alvarez et al., 2012). For completeness, we add the proof in Appendix A.*

Note that the above dynamics in Eq.(10) is identical to the one of the following kernel methods w.r.t. NTK under gradient flow:

$$\min_{f^\kappa} \frac{1}{n}\sum_{i=1}^{n} \ell(f^\kappa(\mathbf{x}_i), \mathbf{y}_i), \tag{11}$$

where $f^\kappa(\cdot) = [f_1^\kappa(\cdot), f_2^\kappa(\cdot), ..., f_c^\kappa(\cdot)]$, and $\forall j \in [c], f_j^\kappa(\cdot) \in \mathbb{H}$, in which $\mathbb{H}$ denotes the RKHS induced by the NTK kernel function κ. In other words, for infinitely wide NNs under gradient flow and random initialization, the output function of NNs is equivalent to the prediction function of kernel methods w.r.t. NTK under gradient flow. Thus, we can transform the optimization of NNs into that of kernel methods w.r.t. NTK. Besides, we can also get the output of NNs in prediction by the prediction of kernel methods.

4.2. Kernel Method w.r.t. NTK

Therefore, we now focus on the corresponding kernel method in Eq.(11) w.r.t. NTK. Substituting ℓ with L_h and L_r based on Eq.(5), we can get

$$\min_{f^\kappa} \frac{1}{n}\sum_{i=1}^{n} \Big(L_h(f^\kappa(\mathbf{x}_i), \mathbf{y}_i) + \lambda L_r(f^\kappa(\mathbf{x}_i), \mathbf{y}_i)\Big). \tag{12}$$

For kernel methods, their generalization performance can be improved by endowing regularization constraints, which we will give formal analyses in the following section. Thus, we add the Tikhonov regularization as follows

$$\min_{f^\kappa} \frac{1}{n}\sum_{i=1}^{n} \Big(L_h(f^\kappa(\mathbf{x}_i), \mathbf{y}_i) + \lambda L_r(f^\kappa(\mathbf{x}_i), \mathbf{y}_i)\Big) + \frac{\tau}{2}\sum_{j=1}^{c} \|f_j^\kappa\|_{\mathbb{H}}^2, \tag{13}$$

where τ is a tradeoff hyper-parameter and $\|f\|_{\mathbb{H}}$ denotes its norm in the Hilbert space $\mathbb{H}$.

The above convex optimization problem is hard to handle since it is infinite-dimensional. Consequently, we give a representer theorem, which is shown in Theorem 3, to equivalently transform the original problem to a finite-dimensional convex minimization problem.

Theorem 3 (The Representer Theorem) *Assume that $f_j^\kappa(\cdot) \in \mathbb{H}, \forall j \in [c]$, where $\mathbb{H}$ is the RKHS induced by the kernel function κ. If $\bar{f}^\kappa$ is an optimal solution of Eq.(13), then it admits the following linear representer form*

$$\bar{f}_j^\kappa(\cdot) = \sum_{i=1}^{n} \alpha_{ij}\kappa(\cdot, \mathbf{x}_i),\ \forall j \in [c], \tag{14}$$

where $\bar{f}^\kappa(\cdot) = \big[\bar{f}_1^\kappa(\cdot), ..., \bar{f}_c^\kappa(\cdot)\big]$ and $\alpha_{ij} \in \mathbb{R}, \forall i \in [n], j \in [c]$.

Remark 4 *The proof of the theorem shares the same spirit of the classical representer theorem (Schölkopf and Smola, 2002).*[2] *Here we omit it for brevity.*

Based on Theorem 3 and the property of $\mathbb{H}$, i.e., $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \kappa(\cdot, \mathbf{x}_i), \kappa(\cdot, \mathbf{x}_j)\rangle_{\mathbb{H}}$, where $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ is the inner product in $\mathbb{H}$, we can equivalently transform the original optimization problem Eq.(13) to the following optimization problem (by substituting the base loss for L_h and L_r), denoted by the learning algorithm $\mathcal{A}$ for convenience:

$$\begin{aligned} \mathcal{A}: \ \min_{\mathbf{A}} \ & \frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c} \ell_{base}(y_{ij}\langle \mathbf{K}_i, \boldsymbol{\alpha}^j\rangle) + \frac{\tau}{2}\mathrm{Tr}(\mathbf{A}^\top \mathbf{K}\mathbf{A}) \\ & + \frac{\lambda}{n}\sum_{i=1}^{n} \frac{1}{|Y_i^+||Y_i^-|} \sum_{(p,q)\in Y_i^+\times Y_i^-} \ell_{base}(\langle \mathbf{K}_i, \boldsymbol{\alpha}^p - \boldsymbol{\alpha}^q\rangle)), \end{aligned} \tag{15}$$

where $\mathbf{A} = [\alpha_{ij}] \in \mathbb{R}^{n\times c}$, and $\boldsymbol{\alpha}^j \in \mathbb{R}^n$ for $j \in [c]$.

4.3. Optimization

For kernel methods, one potential limitation is their high computational costs, which is at least quadratic w.r.t. the number of training instances, due to the involvement of the kernel matrix during training. To this end, one common approach is to approximate the kernel model by a linear model, which explicitly constructs a feature space to approximate the kernel function. *Random Fourier Features* (Rahimi and Recht, 2008) and the *Nyström method* (Williams and Seeger, 2001) are two popular methods for such approximation. While recent work Han et al. (2021) focuses on the random feature method to approximate the NTK, Nyström method usually has better performance theoretically and empirically (Yang et al., 2012) than Random Fourier Features. Hence, to efficiently train the NTK model in Eq.(15), we use the Nyström method, which is rarely used in MLL to our knowledge, and then learn the linear model by a recent fast stochastic variance reduction algorithm SVRG-BB (Tan et al., 2016).

Specifically, the Nyström method approximates the full kernel matrix by a low-rank matrix. First, it samples m instances, denoted by $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_m$, and gets the best rank-r approximation $\mathbf{C}_r$ of $\mathbf{C} = [\kappa(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)] \in \mathbb{R}^{m\times m}$, i.e., $\mathbf{C}_r = \sum_{i=1}^{r} \sigma_i \mathbf{v}_i \mathbf{v}_i^\top$, where $\mathbf{C}$ has the Singular Value Decomposition (SVD) $\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$, $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_m]$, and $\mathbf{D} = diag(\sigma_1, ..., \sigma_m)$. Then, the approximate rank-r kernel matrix is constructed by $\hat{\mathbf{K}} = \mathbf{B}\mathbf{C}_r^\dagger \mathbf{B}^\top \approx \mathbf{K}$, where $\mathbf{B} = [\kappa(\mathbf{x}_i, \hat{\mathbf{x}}_j)] = [\mathbf{b}_1; ...; \mathbf{b}_n] \in \mathbb{R}^{n\times m}$, and $\mathbf{C}_r^\dagger$ is the pseudo inverse of $\mathbf{C}_r$. Thus, the approximate kernel function of two data points becomes $\hat{\kappa}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{b}_i \mathbf{V}_r \mathbf{D}_r^{-1/2}(\mathbf{b}_j \mathbf{V}_r \mathbf{D}_r^{-1/2})^\top = [\kappa(\mathbf{x}_i, \hat{\mathbf{x}}_1), ..., \kappa(\mathbf{x}_i, \hat{\mathbf{x}}_m)]\mathbf{V}_r\mathbf{D}_r^{-1/2}([\kappa(\mathbf{x}_j, \hat{\mathbf{x}}_1), ..., \kappa(\mathbf{x}_j, \hat{\mathbf{x}}_m)]\mathbf{V}_r\mathbf{D}_r^{-1/2})^\top$, where $\mathbf{V}_r = [\mathbf{v}_1, ..., \mathbf{v}_r]$, and $\mathbf{D}_r = diag(\sigma_1, ..., \sigma_r)$. Therefore, a data point $\mathbf{x}$ can be represented by $z(\mathbf{x}) = \mathbf{D}_r^{-1/2}\mathbf{V}_r^\top [\kappa(\mathbf{x}, \hat{\mathbf{x}}_1), ..., \kappa(\mathbf{x}, \hat{\mathbf{x}}_m)]^\top$. Hence, the kernel model has been transformed into the

2. Note that, the classical representer theorem (Schölkopf and Smola, 2002) (i.e., Theorem 4.2 in Sec. 4.2, Page 90) handles the scalar output case while our theorem handles the vector output case which allows coupling between the output (i.e. the ranking loss).

linear model by solving the following optimization problem:

$$\begin{aligned}\min_{\mathbf{W}\in\mathbb{R}^{m\times c}} \frac{1}{nc}\sum_{i=1}^{n}\sum_{j=1}^{c}\ell_{base}(-y_{ij}\langle \mathbf{w}^j, z(\mathbf{x}_i)\rangle) \ + \frac{\tau}{2}\|\mathbf{W}\|_F^2 + \\ \frac{\lambda}{n}\sum_{i=1}^{n}\frac{1}{|Y_i^+||Y_i^-|}\sum_{(p,q)\in Y_i^+\times Y_i^-}\{\ell_{base}(\langle \mathbf{w}^q - \mathbf{w}^p, z(\mathbf{x}_i)\rangle)\}\,.\end{aligned} \tag{16}$$

Then, we aim to solve the above problem by stochastic gradient algorithms. While stochastic gradient descent (SGD) methods have been widely used in machine learning, they have a low convergence rate using diminishing step size, which results in a large variance for the gradient estimation. To this end, many methods, such as SVRG (Johnson and Zhang, 2013) adopt the variance reduction technique to accelerate the convergence. More specifically, SVRG keeps a *snapshot* that is updated after the internal loop, where the constant step size can be used and the linear convergence rate is guaranteed. However, in practice, the step size is hard to estimate, which usually needs time-consuming fine-tuning. Thus, more recently, SVRG-BB (Tan et al., 2016) incorporates the Barzilai-Borwein (BB) method (Barzilai and Borwein, 1988) into the SVRG to automatically compute step sizes. Hence, we use the SVRG-BB to train the linear model in Eq.(16), which is summarized in Algorithm 1. Specifically, for convenience, we denote Eq.(16) by $\min_{\mathbf{W}} \frac{1}{n}\sum_{i=1}^{n} g_i(\mathbf{W})$, where $\nabla g_i(\mathbf{W})$ denotes the gradient of $g_i(\mathbf{W})$ w.r.t. $\mathbf{W}$. (See Appendix B for detailed proofs).

Algorithm 1 SVRG-BB to train the linear model in Eq.(16)

```
Input: update frequency $\bar{m}$, initial point $\widetilde{\mathbf{W}}_0$, initial step size $\eta_0$
Output: $\mathbf{W}^* \in \mathbb{R}^{m\times c}$
 1: for $s = 0, 1, ...$ do
 2:   $\mathbf{G}_s = \frac{1}{n}\sum_{i=1}^{n}\nabla g_i(\widetilde{\mathbf{W}}_s)$
 3:   if $s > 0$ then
 4:     $\eta_s = \frac{1}{m}\|\widetilde{\mathbf{W}}_s - \widetilde{\mathbf{W}}_{s-1}\|_F^2 / \mathrm{Tr}((\widetilde{\mathbf{W}}_{\mathrm{s}} - \widetilde{\mathbf{W}}_{\mathrm{s-1}})^\top(\mathbf{G}_{\mathrm{s}} - \mathbf{G}_{\mathrm{s-1}}))$
 5:   end if
 6:   $\mathbf{W}_0 = \widetilde{\mathbf{W}}_s$
 7:   for $t = 0, 1, ..., \bar{m} - 1$ do
 8:     Randomly pick $i_t \in [n]$
 9:     $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_s(\nabla g_{i_t}(\mathbf{W}_t) - \nabla g_{i_t}(\widetilde{\mathbf{W}}_s) + \mathbf{G}_s)$
10:   end for
11:   $\widetilde{\mathbf{W}}_{s+1} = \mathbf{W}_{\bar{m}}$
12: end for
13: return $\mathbf{W}^* = \widetilde{\mathbf{W}}_{s+1}$
```

5. Theoretical Analyses

Here we give theoretical analyses of the algorithm $\mathcal{A}$ and the effect of Nyström method.

5.1. Generalization Analysis

Here we analyze the learning guarantees of the learning algorithm $\mathcal{A}$ (i.e., Eq.(15)) w.r.t. the measures of HL and RL, following recent work (Wu and Zhu, 2020), which is technically based on Rademacher complexity (Bartlett and Mendelson, 2002) and the vector-contraction inequality (Maurer, 2016).

Let $\Phi : \mathcal{X} \to \mathbb{H}$ be a feature mapping associated with the kernel function κ and define the hypothesis space as

$$\mathcal{F} = \left\{ \mathbf{x} \mapsto [\langle f_1^\kappa, \Phi(\mathbf{x})\rangle, \dots, \langle f_c^\kappa, \Phi(\mathbf{x})\rangle] : \sum_{j=1}^{c} \left\| f_j^\kappa \right\|_{\mathbb{H}}^2 \leq \Lambda \right\}. \tag{17}$$

For any $f \in \mathcal{F}$, the true (0/1) expected and surrogate empirical risk w.r.t. the HL measure can be defined as follows, respectively:

$$R_{0/1}^h(f) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim P} \left[L_h^{0/1}(f(\mathbf{x}_i), \mathbf{y}_i) \right], \quad \hat{R}_S^h(f) = \frac{1}{n} \sum_{i=1}^{n} L_h(f(\mathbf{x}_i), \mathbf{y}_i).$$

Similarly, we can define its counterparts (i.e., $R_{0/1}^r(f)$ and $\hat{R}_S^r(f)$) for the RL measure.

First, we give the common mild assumption for the subsequent analysis.

Assumption 1 *The base loss $\ell_{base}(\cdot)$ is ρ-Lipschitz continuous and bounded by B.*[3]

Then, we analyze the learning guarantee of $\mathcal{A}$ w.r.t. the Hamming Loss (HL) as follows.

Theorem 5 (Learning guarantee w.r.t. HL measure; full proof in Appendix C.1) *Consider the hypothesis space $\mathcal{F}_h = \{f \in \mathcal{F} : \hat{R}_S^r(f) \leq \Lambda_1\}$ and the loss function L_h defined in Eq.(6). Besides,* Assumption 1 *holds. Then, for any $\delta > 0$, the following generalization bound holds for any $f \in \mathcal{F}_h$ with probability at least $1 - \delta$:*

$$R_{0/1}^h(f) \leq \hat{R}_S^h(f) + 2\rho\sqrt{\frac{2}{c}}\hat{\mathfrak{R}}_S(\mathcal{F}_h) + 3B\sqrt{\frac{\log \frac{2}{\delta}}{2n}}, \tag{18}$$

where the empirical Rademacher complexity satisfies

$$\hat{\mathfrak{R}}_S(\mathcal{F}_h) \leq \hat{\mathfrak{R}}_S(\mathcal{F}) \leq \sqrt{\frac{c \max\{\mathbf{K}_{ii}\}\Lambda^2}{n}}. \tag{19}$$

From the above theorem, we can observe that $\mathcal{A}$ has an error bound of $O\left(\frac{1}{\sqrt{n}}\right)$ w.r.t. the HL measure. To minimize the right side of InEq.(18) for a tight bound, we can add the Tikhonov regularization (i.e., $\sum_{j=1}^{c} \|f_j^\kappa\|_{\mathbb{H}}^2$) in the objective function to improve the generalization. Further, we can see that it might improve the generalization w.r.t. the HL measure by adding the minimization of the ranking risk term in the objective function, where this term can be viewed as a data-dependent regularizer to constrain the model complexity.

Similarly, we analyze the learning guarantee of $\mathcal{A}$ w.r.t. the RL measure in the following.

Theorem 6 (Learning guarantee w.r.t. RL measure; full proof in Appendix C.2) *Consider the hypothesis space $\mathcal{F}_r = \{f \in \mathcal{F} : \hat{R}_S^h(f) \leq \Lambda_2\}$ and the loss function L_r defined*

3. Note that the widely-used hinge and logistic loss are both 1-Lipschitz continuous.

in Eq.(7)*. Besides,* Assumption 1 *holds. Then, for any $\delta > 0$, the following generalization bound holds for any $f \in \mathcal{F}_r$ with probability at least $1-\delta$:*

$$R^r_{0/1}(f) \leq \hat{R}^r_S(f) + 2\sqrt{2}\rho\hat{\mathfrak{R}}_S(\mathcal{F}_r) + 3B\sqrt{\frac{\log\frac{2}{\delta}}{2n}}, \tag{20}$$

where the empirical Rademacher complexity satisfies

$$\hat{\mathfrak{R}}_S(\mathcal{F}_r) \leq \hat{\mathfrak{R}}_S(\mathcal{F}) \leq \sqrt{\frac{c\max\{\mathbf{K}_{ii}\}\Lambda^2}{n}}. \tag{21}$$

From the above theorem, we can observe that $\mathcal{A}$ has an error bound of $O(\sqrt{\frac{c}{n}})$ w.r.t. the RL measure. Similarly, it also justifies that the Tikhonov regularization can improve its generalization, and it might improve the performance w.r.t. the RL by adding the Hamming risk term in the learning objective function, where it can also be viewed as a data-dependent regularizer to constrain the model complexity.

In summary, our above analyses explain why we design such a learning objective function in Eq.(13). Moreover, we want to highlight these bounds can enjoy good theoretical guarantees from kernel methods while existing neural network-based methods cannot have such guarantees to our knowledge.

Comparison with prior work (Wu and Zhu, 2020). Although our theoretical analyses mainly follow the prior work (Wu and Zhu, 2020), there is a major difference in the considered hypothesis space. While they consider the hypothesis space $\mathcal{F}$, we consider the hypothesis space $\mathcal{F}_h$ (or $\mathcal{F}_r$) in Theorem 5 (or 6), which can be smaller than $\mathcal{F}$.

Limitations. Here we want to discuss the limitations of our analyses. First, our above generalization analyses hold for any kernel function, including NTK and RBF kernel. Thus, we cannot theoretically tell the performance superiority between them. To the best of our knowledge, this problem (a.k.a., the kernel choice problem) is largely open for kernel methods (Muandet et al., 2017). Second, we do not characterize the detailed relationships (or factors) between the empirical Rademacher complexity $\hat{\mathfrak{R}}_S(\mathcal{F}_h)$ (or $\hat{\mathfrak{R}}_S(\mathcal{F}_r)$) and $\hat{\mathfrak{R}}_S(\mathcal{F})$ because it is highly non-trivial to our knowledge.

5.2. Effect of Nyström method

For the Nyström method, there are many theoretical works (Drineas et al., 2005; Cortes et al., 2010; Jin et al., 2013; Derezinski et al., 2020) to analyze its performance mainly from two perspectives: the approximation error w.r.t. the kernel matrix and the generalization w.r.t. the approximate hypothesis.

For the approximation error, classical results (Drineas et al., 2005) show that the following bound holds generally (for uniform sampling used in this paper) with high probability:[4]

$$\|\mathbf{K} - \hat{\mathbf{K}}\|_2 \leq \sigma_{r+1} + O\left(\frac{\mathrm{Tr}(\mathbf{K})}{\sqrt{m}}\right). \tag{22}$$

For the generalization analysis of Nyström methods, we find that it is usually problem-specific w.r.t. learning algorithms. Further, it is highly non-trivial to analyze our algorithm,

4. Note that, $\mathrm{Tr}(\mathbf{K})$ can be empirically large for NTK and we can use its normalized kernel version almost without performance degradation. Besides, this bound can be improved w.r.t. m under additional assumptions on the kernel matrix (Jin et al., 2013).

and we leave it as future work. Notably, when setting $r = m = n$, we can precisely get $\hat{\mathbf{K}} = \mathbf{K}$ without any loss on both approximation and generalization (Mohri et al., 2018).[5]

6. Experiments

For experiments, the main purpose is to test whether the infinitely wide DNNs can help small-data multi-label learning, rather than to illustrate the superiority of our methods over the state-of-the-art approaches (Bogatinovski et al., 2022). Thus, in the following, we choose the widely-used small-scale MLL datasets and related baselines to our method.

6.1. Experimental Setups

Datasets. We use 8 widely-used small-scale multi-label datasets.[6] They cover varieties of domains and sizes, which are summarized in Table 1. For each dataset, we randomly split 60% for training, the remaining 40% for testing, and do ten repeats.

Table 1: Statistics of the benchmark multi-label datasets. "Card" represents the average relevant labels per instance, and "Den" normalizes the "Card" by the label number.

Dataset	#Instance	#Feature	#Label	Card	Den	Domain
flags	194	19	7	3.392	0.485	image
birds	351	260	19	1.863	0.098	audio
emotions	593	72	6	1.869	0.311	music
image	2000	294	5	1.240	0.248	image
scene	2407	294	6	1.074	0.179	image
yeast	2417	103	14	4.237	0.303	biology
enron	1702	1001	53	3.378	0.064	text
business	5000	438	30	1.588	0.053	text

Compared Methods. Here we compare our method MLC-NTK with the following baseline methods, which includes a NN-based method BP-MLL (Zhang and Zhou, 2006), three kernel-based methods (i.e., Rank-SVM (Elisseeff et al., 2001), BR-SVM (Boutell et al., 2004) and CPNL (Wu et al., 2018)) and a recent method MLFE (Zhang et al., 2018). Besides, we involve a method MLC-RBF which replaces the NTK kernel in Eq.(13) with the Gaussian (i.e., RBF) kernel. For the last two datasets (i.e., *enron* and *business*), we use the linear kernel for the related baselines because the linear kernel has better performance than the RBF kernel as shown in Wu et al. (2018). For the other datasets, all the kernel-based baselines employ the RBF kernel, where the hyper-parameter γ is searched in $\{10^{-4}, 10^{-3}..., 10^{4}\} * \frac{1}{d}$. For other hyper-parameters of all the baselines, the setting and search space are used as suggestions of the original literature. For MLC-NTK, the hyper-parameter L and L' are searched in $\{1, 2, ..., 5\}$ and $\{0, 1, ..., L-1\}$ respectively. Besides, τ and λ are tuned in $\{10^{-3}, 10^{-2}..., 10^{3}\}$. Furthermore, all the hyper-parameters are selected by 5-fold cross-validation on the training data.

Evaluation Measures. For an overall and fair evaluation, we employ six widely-used measures, including three classification-based metrics (i.e., *Hamming Loss*, *Subset Accuracy*, and *instance-F1*), and three ranking-based measures (i.e., *Ranking Loss*, *Coverage* and *Average Precision*). Please refer to Zhang and Zhou (2014) for their detailed definitions.

5. Indeed, in our experiments, we adopt this setting for high performance while still enjoy efficiency over other baselines.
6. The datasets can be downloaded from `http://mulan.sourceforge.net/datasets-MLL.html` and `http://palm.seu.edu.cn/zhangml/`. Note that here we choose these small-scale datasets because not only they are widely used in MLL (Zhang and Zhou, 2014), but also prior work (Arora et al., 2020) has shown the effectiveness of the NTK for single-labeled tasks in low-data settings.

Table 2: Experimental results of approaches (mean_{std}) w.r.t. HL (↓) on all datasets. ↓ (↑) indicates the smaller (larger) the better. Best results are highlighted in bold.

Dataset	Rank-SVM	BP-MLL	BR-SVM	CPNL	MLFE	MLC-RBF	MLC-NTK
flags	$0.286_{0.017}$	$0.284_{0.006}$	$0.270_{0.009}$	$0.267_{0.027}$	$0.270_{0.010}$	$0.263_{0.016}$	$\mathbf{0.255_{0.018}}$
birds	$0.084_{0.003}$	$0.107_{0.003}$	$0.085_{0.003}$	$0.085_{0.007}$	$\mathbf{0.081_{0.005}}$	$0.091_{0.005}$	$0.083_{0.003}$
emotions	$0.189_{0.008}$	$0.218_{0.015}$	$\mathbf{0.183_{0.009}}$	$\mathbf{0.183_{0.009}}$	$0.186_{0.009}$	$0.185_{0.011}$	$\mathbf{0.183_{0.012}}$
image	$0.161_{0.005}$	$0.245_{0.008}$	$0.156_{0.006}$	$0.150_{0.006}$	$0.156_{0.007}$	$0.152_{0.005}$	$\mathbf{0.143_{0.006}}$
scene	$0.092_{0.005}$	$0.102_{0.007}$	$0.077_{0.002}$	$0.077_{0.003}$	$0.083_{0.003}$	$0.080_{0.002}$	$\mathbf{0.074_{0.002}}$
yeast	$0.203_{0.004}$	$0.233_{0.011}$	$0.188_{0.003}$	$0.192_{0.004}$	$0.194_{0.004}$	$0.192_{0.004}$	$\mathbf{0.187_{0.003}}$
enron	$0.051_{0.003}$	$0.057_{0.002}$	$0.052_{0.001}$	$0.049_{0.001}$	$\mathbf{0.046_{0.001}}$	$0.049_{0.001}$	$\mathbf{0.046_{0.001}}$
business	$0.030_{0.004}$	$0.036_{0.000}$	$0.026_{0.001}$	$\mathbf{0.025_{0.001}}$	$\mathbf{0.025_{0.001}}$	$0.028_{0.001}$	$\mathbf{0.025_{0.001}}$

Table 3: Experimental results of approaches (mean_{std}) w.r.t. RL (↓) on all datasets.

Dataset	Rank-SVM	BP-MLL	BR-SVM	CPNL	MLFE	MLC-RBF	MLC-NTK
flags	$0.200_{0.023}$	$0.213_{0.003}$	$0.225_{0.016}$	$0.232_{0.033}$	$0.237_{0.014}$	$0.199_{0.021}$	$\mathbf{0.198_{0.017}}$
birds	$0.164_{0.012}$	$0.188_{0.016}$	$0.167_{0.011}$	$0.157_{0.012}$	$0.173_{0.022}$	$0.166_{0.009}$	$\mathbf{0.144_{0.018}}$
emotions	$0.155_{0.009}$	$0.175_{0.015}$	$0.246_{0.015}$	$0.139_{0.010}$	$0.142_{0.011}$	$0.139_{0.008}$	$\mathbf{0.136_{0.014}}$
image	$0.143_{0.008}$	$0.195_{0.005}$	$0.220_{0.012}$	$0.132_{0.006}$	$0.142_{0.007}$	$0.134_{0.006}$	$\mathbf{0.128_{0.006}}$
scene	$0.065_{0.005}$	$0.084_{0.002}$	$0.128_{0.006}$	$0.059_{0.003}$	$0.063_{0.003}$	$0.060_{0.003}$	$\mathbf{0.057_{0.004}}$
yeast	$0.170_{0.005}$	$0.196_{0.013}$	$0.308_{0.008}$	$0.158_{0.006}$	$0.166_{0.005}$	$0.161_{0.005}$	$\mathbf{0.154_{0.004}}$
enron	$0.081_{0.008}$	$0.086_{0.002}$	$0.298_{0.007}$	$0.078_{0.003}$	$0.076_{0.004}$	$0.089_{0.002}$	$\mathbf{0.069_{0.003}}$
business	$0.034_{0.005}$	$0.048_{0.092}$	$0.205_{0.005}$	$0.030_{0.002}$	$0.041_{0.002}$	$0.035_{0.001}$	$\mathbf{0.029_{0.001}}$

6.2. Results

The experimental results w.r.t. the measures of Hamming and Ranking Loss are summarized in Table 2 and 3, respectively (see Appendix D.1 for other measures). Besides, Table 4 summarizes the average rank of compared methods over each measure.

Table 4: Average ranks of the compared approaches on all datasets in terms of each metric and all the metrics.

Metric	Rank-SVM	BP-MLL	BR-SVM	CPNL	MLFE	MLC-RBF	MLC-NTK
Hamming Loss	5.63	6.88	3.38	2.38	3.25	3.75	**1.13**
Subset Accuracy	6.00	6.75	3.00	2.25	4.63	3.38	**1.50**
instance-F1	4.63	4.25	4.75	**1.88**	5.75	4.13	2.00
Ranking Loss	4.13	5.75	6.50	2.63	4.50	3.38	**1.00**
Coverage	3.75	5.88	6.38	2.63	4.63	3.50	**1.00**
Average Precision	4.75	5.63	6.13	2.50	3.88	3.63	**1.00**
Overall	4.81	5.85	5.02	2.38	4.44	3.63	**1.27**

Comparison with other kernel methods. Comparing MLC-NTK and MLC-RBF, we can conclude that the NTK kernel can be more effective than the RBF kernel for low-data settings. As mentioned in our theoretical parts, it's largely open to explain why this phenomenon happens in theory. We hypothesize that this is because the NTK kernel might have more flexible expressive power while keeping suitable model complexity than the RBF kernel. This also indicates the choice importance of the hypothesis space. Besides, comparing MLC-RBF and BR-SVM (or Rank-SVM), we can find that MLC-RBF performs better w.r.t. Hamming (or Ranking) loss, which supports our theoretical analyses that these two losses can be viewed as the data-dependent regularizer for each other.[7]

Comparison with NN-based methods. Comparing Rank-SVM and BP-MLL w.r.t. Ranking loss (since both aim to optimize this measure), we can find that Rank-SVM per-

7. Note that, although BR-SVM and Rank-SVM share the hinge base loss while MLC-RBF empirically adopts the logistic base loss, these two base losses have little effect on performance empirically.

forms better than BP-MLL in all datasets. This arises the generalization question about neural networks for low-data settings, which needs more to explore to explain why. Besides, MLC-NTK performs better than BP-MLL.

We also notice that CPNL performs better than MLC-NTK w.r.t. instance-F1 (see Table 2 in Appendix D.1), which is probably because CPNL involves a cost-sensitive loss that might be suitable for the F-score measure. We also find that MLC-NTK achieved promising performance w.r.t. Subset Accuracy (SA) (see Table 1 in Appendix D.1). This is because recent theoretical work (Wu and Zhu, 2020) has shown when optimizing HL on datasets with small label space, it can achieve promising performance w.r.t. SA.

In summary, MLC-NTK has achieved promising performance over related baselines for low-data settings, which indicates the NTK kernel might be a good candidate for kernel methods in MLL. Furthermore, we also empirically find our method trains faster than other kernel methods (see Appendix D.2 for details). Besides, we also provide the sensitivity analysis w.r.t. the hyper-parameter τ and λ (see Appendix D.3 for details).

7. Conclusion and Discussion

To answer the question of whether infinitely wide DNNs can help small-data multi-label learning, here we propose to utilize the NTK kernel to solve the MLL task. Specifically, we aim to minimize the Hamming and Ranking Loss based on the NTK kernel, which derives from the infinitely wide NN. Then, the Nyström method and a fast stochastic algorithm are used for efficiently training the model. Further, we give theoretical analyses about the learning guarantees of the proposed algorithm w.r.t. these two measures. Experimental results on small-scale datasets illustrate the effectiveness and efficiency of our method, which indicates that NTK-based methods can be a good choice in small-data multi-label learning.

Theoretically, it is interesting to explore why the NTK kernel usually performs better than the RBF kernel in low-data settings. Experimentally, we focus on low-data settings, while leaving large-scale settings for future work. Besides, the NTK and our analyses can be extended to other tasks, such as multi-dimensional classification (Jia and Zhang, 2022).

Acknowledgments

This work was supported by the NSFC (Nos. 62206159, 61620106010, 62061136001, 61621136008, U1811461, U19B2034, U19A2081), the Natural Science Foundation of Beijing (No. JQ19016), the Natural Science Foundation of Shandong Province (Nos. ZR2022QF117), a grant from Tsinghua Institute for Guo Qiang, and the Fundamental Research Funds of Shandong University. G. Wu was also sponsored by the TaiShan Scholars Program.

References

Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

Jasmin Bogatinovski, Ljupčo Todorovski, Sašo Džeroski, and Dragi Kocev. Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications*, 203:117215, 2022.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

Qian Cai, Guo-Chong Cui, and Hai-Xian Wang. Eeg-based emotion recognition using multiple kernel learning. *Machine Intelligence Research*, 19(5):472–484, 2022.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3163–3171, 2020.

Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 113–120. JMLR Workshop and Conference Proceedings, 2010.

Michal Derezinski, Rajiv Khanna, and Michael W Mahoney. Improved guarantees and a multiple-descent curve for column subset selection and the nystrom method. *Advances in Neural Information Processing Systems*, 33, 2020.

Petros Drineas, Michael W Mahoney, and Nello Cristianini. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(12), 2005.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

André Elisseeff, Jason Weston, et al. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14*, pages 681–687, 2001.

Insu Han, Haim Avron, Neta Shoham, Chaewon Kim, and Jinwoo Shin. Random features for the neural tangent kernel. *arXiv preprint arXiv:2104.01351*, 2021.

Cheng-Yu Hsieh, Yi-An Lin, and Hsuan-Tien Lin. A deep model with local surrogate loss for general cost-sensitive multi-label learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Bin-Bin Jia and Min-Ling Zhang. Multi-dimensional classification via selective feature augmentation. *Machine Intelligence Research*, 19(1):38–51, 2022.

Rong Jin, Tianbao Yang, Mehrdad Mahdavi, Yu-Feng Li, and Zhi-Hua Zhou. Improved bounds for the nyström method with application to kernel classification. *IEEE Transactions on Information Theory*, 59(10):6939–6949, 2013.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

Yuncheng Li, Yale Song, and Jiebo Luo. Improving pairwise ranking for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3617–3625, 2017.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 115–124, 2017.

Andreas Maurer. A vector-contraction inequality for rademacher complexities. In *International Conference on Algorithmic Learning Theory*, pages 3–17. Springer, 2016.

Siamak Mehrkanoon and Johan AK Suykens. Multi-label semi-supervised learning using regularized kernel spectral clustering. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4009–4016. IEEE, 2016.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

K Muandet, K Fukumizu, B Sriperumbudur, and B Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10 (1-2):1–144, 2017.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2002.

Conghui Tan, Shiqian Ma, Yu-Hong Dai, and Yuqiu Qian. Barzilai-borwein step size for stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 685–693, 2016.

Zhi-Hao Tan, Peng Tan, Yuan Jiang, and Zhi-Hua Zhou. Multi-label optimal margin distribution machine. *Machine Learning*, 109(3):623–642, 2020.

Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2006.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7): 1079–1089, 2011.

Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

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

Guoqiang Wu, Yingjie Tian, and Dalian Liu. Cost-sensitive multi-label learning with positive and negative label pairwise correlations. *Neural Networks*, 108:411–423, 2018.

Guoqiang Wu, Ruobing Zheng, Yingjie Tian, and Dalian Liu. Joint ranking svm and binary relevance with robust low-rank learning for multi-label classification. *Neural Networks*, 122:24–39, 2020a.

Tong Wu, Qingqiu Huang, Ziwei Liu, Yu Wang, and Dahua Lin. Distribution-balanced loss for multi-label classification in long-tailed datasets. In *European Conference on Computer Vision*, pages 162–178. Springer, 2020b.

Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484, 2012.

Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

Qian-Wen Zhang, Yun Zhong, and Min-Ling Zhang. Feature-induced labeling information enrichment for multi-label learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4446–4453. AAAI Press, 2018.