Two-Stage Label Embedding via Neural Factorization Machine for Multi-Label Classification

Chen Chen,[1,3]* Haobo Wang,[1,3]* Weiwei Liu,[4] Xingyuan Zhao,[1,3] Tianlei Hu,[1,3]† Gang Chen[2,3]
[1]The Key Laboratory of Big Data Intelligent Computing of Zhejiang Province, China
[2]CAD & CG State Key Lab, Zhejiang University, China
[3]College of Computer Science and Technology, Zhejiang University, China
[4]School of Computer Science and Software Engineering, East China Normal University, China
{cc33, wanghaobo, xyzhao1, htl, cg}@zju.edu.cn, liuweiwei863@gmail.com

Abstract

Label embedding has been widely used as a method to exploit label dependency with dimension reduction in multi-label classification tasks. However, existing embedding methods intend to extract label correlations directly, and thus they might be easily trapped by complex label hierarchies. To tackle this issue, we propose a novel Two-Stage Label Embedding (TSLE) paradigm that involves Neural Factorization Machine (NFM) to jointly project features and labels into a latent space. In encoding phase, we introduce a Twin Encoding Network (TEN) that digs out pairwise feature and label interactions in the first stage and then efficiently learn higher-order correlations with deep neural networks (DNNs) in the second stage. After the codewords are obtained, a set of hidden layers is applied to recover the output labels in decoding phase. Moreover, we develop a novel learning model by leveraging a max margin encoding loss and a label-correlation aware decoding loss, and we adopt the mini-batch Adam to optimize our learning model. Lastly, we also provide a kernel insight to better understand our proposed TSLE. Extensive experiments on various real-world datasets demonstrate that our proposed model significantly outperforms other state-of-the-art approaches.

Introduction

Single-label classification (Gong et al. 2015; 2016; Deng et al. 2018b) is one of the most well-known machine learning problems, where each instance x is associated with a single label. However, in many real-world applications, an object can be associated with multiple labels simultaneously. For instance, a document may belong to a set of topics such as *finance* and *news* (Liu et al. 2018); a video can be annotated with *government* and *policy* (Liu and Tsang 2017); an image can be tagged with various keywords like *beach* and *trees* (Liu, Tsang, and Müller 2017).

Many techniques (Schapire and Singer 2000; Zhang and Zhou 2007) have been proposed to deal with Multi-Label Classification (MLC) problems. Binary Relevance (BR) (Tsoumakas, Katakis, and Vlahavas 2010) is one of the most popular methods which decomposes the multi-label task into several independent binary classification tasks. However, these methods fail to exploit label dependency which may lead to degenerated performance. The key challenging issue for multi-label learning is how to learn the correlations between labels, and some methods have been developed to address this issue. For example, Rank-SVM (Elisseeff and Weston 2001) learns pairwise label interactions and Classifier Chains (Read et al. 2011) exploit high-order label correlations.

Embedding method (Hsu et al. 2009; Cao et al. 2016; Yang et al. 2018; Deng et al. 2018a) is one of the most popular frameworks to learn label and feature correlations and has shown promising results. MMOC (Zhang and Schneider 2012) proposes a max margin formulation to produce discriminative and predictable codewords. LM-kNN (Liu and Tsang 2015) reduces the exponentially large number of constraints in MMOC to linear order by preserving pairwise distances between only the closest (rather than all) label vectors.

Several state-of-the-art techniques (Zhang and Zhou 2006; Wang et al. 2016; Yeh et al. 2017; Nam et al. 2017) exploit deep neural network (DNN) for MLC. For instance, BP-MLL (Zhang and Zhou 2006) is one of the earliest methods to use neural network architectures in MLC. Wang et al. (2016) propose a unified CNN-RNN framework that linearly embeds labels using recurrent neural network (RNN) to model label dependency. To discover higher-order label correlations, C2AE (Yeh et al. 2017) learns deep latent spaces by combining DNN and embedding framework. Although DNN brings about inspiring results, it is difficult to learn high-order correlations directly due to the sparsity of labels.

To ameliorate this problem, we propose a novel Two-Stage Label Embedding (TSLE) paradigm that involves Neural Factorization Machine (NFM) (He and Chua 2017) into embedding framework for MLC. In encoding phase, we introduce a Twin Encoding Network (TEN) to obtain the codewords. TEN is comprised of two parts: a feature network and a label network. The two networks share a similar Two-Stage Label Embedding architecture. As a variant of NFM, TEN involves Factorization Machines (FMs) (Rendle 2012), which demonstrates great promise in prediction tasks under sparse setting, to learn pairwise interactions in the first stage. Then it uses deep neural network to learn

*Equal contribution.
†Corresponding author.

higher-order correlations in the second stage. In decoding phase, output labels are recovered from the codewords with a set of hidden layers. Moreover, we develop a novel learning model by leveraging a max margin encoding loss and a label-correlation aware loss of decoding network.

The main contributions of this paper include:

1. We present a neural network based architecture dubbed Two-Stage Label Embedding to jointly embed features and labels into a lower dimensional space and recover output labels from the codewords of features.
2. To the best of our knowledge, we are the first to involve Neural Factorization Machine to exploit feature and label correlations in MLC. A factorization layer is introduced to dig out pairwise feature and label interactions in the first stage. In the second stage, we efficiently learn higher-order correlations with deep neural network.
3. We provide a kernel insight to better understand our proposed TSLE.
4. Extensive experiments on a number of real-world multi-label datasets indicate that TSLE outperforms state-of-the-art approaches.

The rest of this paper is organized as follows. We first review some related algorithms. Next, we provide a detailed description and a kernel view of TSLE. Then our experiments are reported, followed by the conclusion of our work.

Preliminaries

Neural Factorization Machine

Factorization Machines (FMs) (Rendle 2012) are a class of powerful algorithms for digging out pairwise interactions between features. Given a real-valued feature vector $\mathbf{x} \in \mathbb{R}^p$ where p denotes the number of features, FM models the target by the inner product of two embedding vectors:

$$\hat{y}_{FM}(\mathbf{x}) = \underbrace{w_0 + \sum_{j=1}^{p} w_j x_j}_{linear\ regression} + \underbrace{\sum_{j=1}^{p}\sum_{k=j+1}^{p} \mathbf{v}_j^T \mathbf{v}_k \cdot x_j x_k}_{pairwise\ factorization} \quad (1)$$

where x_j is a singular that represents the j-th entry of $\mathbf{x}$. $\mathbf{v}_j \in \mathbb{R}^t$ denotes the embedding parameter for feature j and t is the size of $\mathbf{v}_j$. Note that $\mathbf{v}_j^T \mathbf{v}_k$ models the factorized interaction between j-th and k-th features. Here $\mathcal{M}^T$ represents the transpose of matrix $\mathcal{M}$. w_j models the interaction of the j-th feature to the target, and w_0 is the global bias.

It is worth noting that FM can only model the second-order feature interactions, which leads to insufficient representation ability for modelling real-world data with complicated inherent structures and regularities. To learn high-order and non-linear feature interactions, He and Chua (2017) develop a novel NFM model to deepen FM under the neural network framework. NFM estimates the target as:

$$\hat{y}_{NFM}(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j x_j + f(\mathbf{x}) \quad (2)$$

where the first and second terms are the same as the regression part of FM and the third part $f(\mathbf{x})$ is a multi-layered neural network.

In this work, we employ a variant of NFM to sufficiently exploit feature and label correlations in TEN. It is worth noting that TSLE is the first label embedding technique that involves FM in MLC.

Max Margin Based Embedding Approaches

We denote the instance vectors by $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N] \in \mathbb{R}^{p\times N}$ and label vectors by $Y = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N] \in \{0,1\}^{q\times N}$. p, q and N are the number of features, labels and training samples respectively. For each sample i, MMOC (Zhang and Schneider 2012) jointly maps instance $\mathbf{x}_i$ and label $\mathbf{y}_i$ to a low-dimensional latent space to obtain their codewords $\mathbf{c}_{\mathbf{x}_i}$ and $\mathbf{c}_{\mathbf{y}_i}$. Then $\mathbf{x}_i$ and $\mathbf{y}_i$ can be compared in the latent space (d-dimension). Intuitionally, if the encoding scheme works well, the prediction distance to the correct codeword, denoted by $||\mathbf{c}_{\mathbf{x}_i} - \mathbf{c}_{\mathbf{y}_i}||_2^2$, should tend to zero and be smaller than the prediction distance to any other codeword $||\mathbf{c}_{\mathbf{x}_i} - \mathbf{c}_{\tilde{\mathbf{y}}}||_2^2$, where $\mathbf{c}_{\tilde{\mathbf{y}}}$ denotes the codeword of any label $\tilde{\mathbf{y}} \in \{0,1\}^q$. However, searching the entire label space involves an exponential huge number of constraints *w.r.t.* the number of labels, which makes it prohibitive to high-dimensional datasets. To tackle this problem, LM-kNN (Liu and Tsang 2015) proposes the following formulation to capture the label dependency by preserving pairwise distances between only the closest label vectors.

$$\begin{aligned} \underset{V,\{\xi_i\geq 0\}_{i=1}^N}{\operatorname{argmin}} \ & \frac{\lambda}{2}||V||_F^2 + \frac{1}{N}\sum_{i=1}^{N}\xi_i \\ s.t. \quad & ||\mathbf{c}_{\mathbf{x}_i} - \mathbf{c}_{\mathbf{y}_i}||_2^2 + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \\ & \leq ||\mathbf{c}_{\mathbf{x}_i} - \mathbf{c}_{\mathbf{y}}||_2^2, \forall \mathbf{y} \in Nei(i), \forall i \end{aligned} \quad (3)$$

where $||\cdot||_F$ is the Frobenius norm, $||\cdot||_2$ is the l_2 norm, λ is a regularization parameter. Note that LM-kNN maps instance $\mathbf{x}_i$ and label $\mathbf{y}_i$ as follow:

$$\begin{aligned} \mathbf{c}_{\mathbf{x}_i} &= V^T P^T \mathbf{x}_i \\ \mathbf{c}_{\mathbf{y}_i} &= V^T \mathbf{y}_i \end{aligned} \quad (4)$$

where $V \in \mathbb{R}^{q\times d}$ and $P \in \mathbb{R}^{p\times q}$ are projection matrices. To give Eq. (3) more robustness, LM-kNN subtracts slack variable ξ_i and adds $\Delta(\mathbf{y}_i, \mathbf{y}) = ||\mathbf{y} - \mathbf{y}_i||_1$ as margin, where $||\cdot||_1$ is the l_1 norm. $Nei(i)$ is the output set of k nearest neighbors of the input instance $\mathbf{x}_i$, whose elements can be encoded in the same way as $\mathbf{y}_i$. Here, the Euclidean distance in original feature space is used as the metric for searching $Nei(i)$.

We adapt the max margin based objective function in TEN. The detailed architecture will be discussed in the following section.

Two-Stage Label Embedding (TSLE)

We propose a novel deep neural network based Two-Stage Label Embedding paradigm to efficiently characterize the high-order correlations via Neural Factorization Machine for MLC. The illustration of TSLE is shown in Fig. 1. The

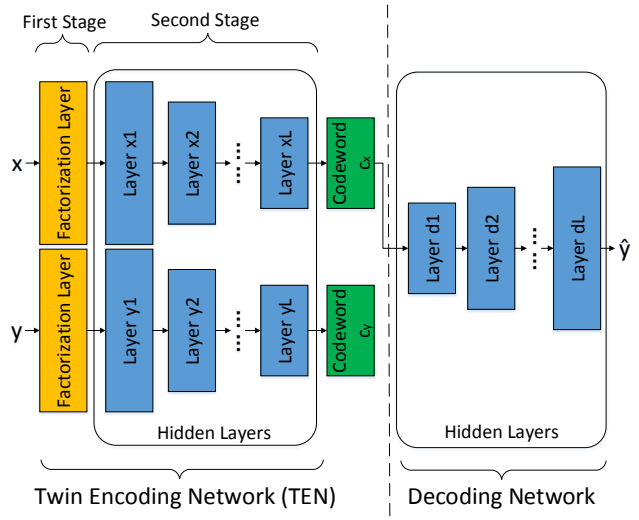


Figure 1: The architecture of the proposed TSLE paradigm. TEN is comprised of two networks, and each of which is a Two-Stage Label Embedding network: a factorization layer in the first stage; several hidden layers in the second stage. The decoding network includes a set of hidden layers.

proposed model contains two phases: a Twin Encoding Network projects instances and labels into a lower embedding space to get their codewords; a decoding network recovers label outputs from the codewords with a set of hidden layers.

Twin Encoding Network (TEN)

We present a novel neural network based encoding scheme. Since TSLE projects features and labels into a shared latent space, we introduce an architecture called Twin Encoding Network for encoding process. Note that instances and labels have their own distribution. Therefore TEN uses a feature network and a label network which have a similar two-stage encoding architecture to produce codewords respectively. In what follows, we elaborate the feature network of TEN while the label network is under the same schema. Concretely, the feature network is comprised of two parts: a factorization layer and a set of hidden layers.

Factorization Layer In the first stage, the factorization layer maps each feature to a dense vector representation. Formally, we feed an instance vector $\mathbf{x} = [x_1, x_2, ..., x_p]^T \in \mathbb{R}^p$ into the layer and then obtain a set of embedding vectors $\mathcal{V}_{emb} = \{\mathbf{v}_1 x_1, \mathbf{v}_2 x_2, ..., \mathbf{v}_p x_p\}$ where $\mathbf{v}_i \in \mathbb{R}^t$ is the embedding parameter corresponding to the i-th feature. Then, we conduct a pairwise product of the embedding vectors, which can be regarded as a pooling operation since it does not involve extra parameters. Now we get a single vector $g(\mathbf{x}; V)$:

$$g(\mathbf{x}; V) = \sum_{j=1}^{p} \sum_{k=j+1}^{p} x_j \mathbf{v}_j \odot x_k \mathbf{v}_k \tag{5}$$

where $V = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p] \in \mathbb{R}^{t \times p}$ is the feature embedding matrix, and $\odot$ denotes element-wise product. This is the core operation of factorization layer that extracts second-order interactions between features.

However, the time complexity of Eq. (5) is $O(tp^2)$, which makes our model impractical when the inputs have a huge number of features or labels. To address this problem, we can reformulate Eq. (5) as:

$$g(\mathbf{x}; V) = \frac{1}{2}[(\sum_{j=1}^{p} x_j \mathbf{v}_j)^2 - \sum_{j=1}^{p} (x_j \mathbf{v}_j)^2] \tag{6}$$

where $\mathbf{v}^2$ denotes $\mathbf{v} \odot \mathbf{v}$. Now the pairwise product operation can be linearly computed in $O(tp)$ time.

Moreover, owing to a sparse representation of inputs, which is common in real-world applications, we can only consider the embedding vectors for non-zero features, *i.e.*,$\mathcal{V}_{emb} = \{\mathbf{v}_i x_i | x_i \neq 0\}$. Consequently, the time complexity is reduced to $O(tp_x)$, where p_x may be a small positive integer that represents the average number of non-zero elements of each vector in the input matrix.

Notice that FM consists of two parts: a regression part and a pairwise factorization part. To generalize FM, the regression part is not fed into the neural network in vanilla NFM, which limits its expressiveness. Hence, we combine the regression operation and the factorization operation in the first layer to sufficiently utilize the expressive ability of neural network:

$$\text{Fac}(\mathbf{x}; V, A) = \mathbf{a}_0 + \sum_{j=1}^{p} x_j \mathbf{a}_j + g(\mathbf{x}; V) \tag{7}$$

where $A = [\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_p] \in \mathbb{R}^{t \times (p+1)}$ is the regression parameter. To output a vector rather than a singular, we perform multi-output regression which is different from the default setting in vanilla NFM.

In contrast to other neural network based models (Yeh et al. 2017; Zhao et al. 2015) which learn high-order correlations directly, TSLE uses the factorization layer to extract second-order interactions between features and labels in advance. Then we can efficiently learn higher-order correlations with deep neural network.

Hidden Layers Deep neural network has demonstrated its ability in learning representations from the raw data. The factorized vectors are then fed into a stack of fully connected layers to learn higher-order correlations in the second stage. Each layer can be customized to discover certain latent structures between features or labels. It is subjected to design and can be abstracted as $\mathbf{c}_\mathbf{x} = h_E(\text{Fac}(\mathbf{x}; V, A); \Theta_x)$ where h_E denotes the hidden layers with parameters Θ_x. $\mathbf{c}_\mathbf{x} \in \mathbb{R}^d$ is the output codeword. In general, the input of each layer is linearly transformed and then activated by a non-linear function, *e.g.*, sigmoid, hyperbolic tangent (tanh) or Rectifier (ReLU).

Eventually, the last hidden layer outputs the codeword vector $\mathbf{c}_\mathbf{x}$ without activation.

Label Network in TEN As for label network, the twin of feature network, the structure is similar but the parameters

are distinct. Formally, the model can be expressed as follow:

$$\begin{aligned}\mathbf{c_y} =& h_E(\text{Fac}(\mathbf{y}; U, B); \Theta_y) \\ =& h_E(\mathbf{b}_0 + \sum_{j=1}^{q} y_j \mathbf{b}_j + g(\mathbf{y}; U); \Theta_y)\end{aligned} \tag{8}$$

where $\mathbf{c_y} \in \mathbb{R}^d$ is the output codeword of labels. $U = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_q] \in \mathbb{R}^{t \times q}$ and $B = [\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_q] \in \mathbb{R}^{t \times (q+1)}$ are the parameters of the label factorization layer. Θ_y are the parameters of the hidden layers in the label network.

Through such a twin architecture, we jointly encode the features and labels to the latent space.

Decoding Network

The decoding procedure recovers label output $\hat{\mathbf{y}}$ from codeword $\mathbf{c_x}$ which is the feature representation in latent space. Traditionally, decoding is performed by maximizing a joint probability function (Zhang and Schneider 2011). Due to the requirement of solving a quadratic problem, such decoding technique is computationally expensive. To handle this problem, we adapt the idea of (Yeh et al. 2017) and introduce a multi-layer decoding network h_D:

$$\begin{aligned}\mathbf{s}_1 &= \sigma_1(W_{d1}\mathbf{c_x} + \mathbf{b}_{d1}) \\ \mathbf{s}_2 &= \sigma_2(W_{d2}\mathbf{s}_1 + \mathbf{b}_{d2}) \\ &..... \\ \mathbf{s}_L &= \sigma_L(W_{dL}\mathbf{s}_{L-1} + \mathbf{b}_{dL}) \\ \hat{\mathbf{y}} &= W_{out}\mathbf{s}_L + \mathbf{b}_{out}\end{aligned} \tag{9}$$

where $\mathbf{s}_i$, W_{di}, $\mathbf{b}_{di}$, σ_i denote the output vector, weight matrix, bias vector and activation function for the i-th layer respectively. $\mathbf{W}_{out}$ and $\mathbf{b}_{out}$ are the weight matrix and bias vector for the last layer. By exploiting the inherent nonlinearity of deep neural network, we can reconstruct our predicted labels from the lower-dimensional codeword vector.

Training

Since our framework is comprised of two separate networks (an encoding network and a decoding network), the final objective function can also be decomposed into two parts: a encoding loss and a decoding loss.

In encoding phase, inspired by (Zhang and Schneider 2012; Liu and Tsang 2015), a max margin formulation is involved such that the codeword is both discriminative and predictable. Define $\mu(i) = \max_{\mathbf{y} \in Nei(i)}\{||\mathbf{c}_{\mathbf{x}_i} - \mathbf{c}_{\mathbf{y}_i}||_2^2 - ||\mathbf{c}_{\mathbf{x}_i} - \mathbf{c}_{\mathbf{y}}||_2^2 + \Delta(\mathbf{y}_i, \mathbf{y})\}$, the encoding loss $\mathcal{L}_E$ is designed as follow:

$$\mathcal{L}_E = \frac{1}{N}\sum_{i=1}^{N} \max\{0, \mu(i)\} \tag{10}$$

For decoding, the goal is to reduce the prediction mistakes on unseen data. Among various choices of global error definitions, we choose to adapt a popular label-correlation aware error function which is proposed by (Zhang and Zhou 2006):

$$\mathcal{L}_D = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{|\mathbf{y}_i^1||\mathbf{y}_i^0|} \sum_{(e,g) \in \mathbf{y}_i^1 \times \mathbf{y}_i^0} \exp((\hat{\mathbf{y}}_i)_g - (\hat{\mathbf{y}}_i)_e) \tag{11}$$

Algorithm 1 Training procedure of TSLE

Input: Feature matrix X, label matrix Y, learning rate η, leverage parameters λ and α, dimension parameters d and t, size of nearest neighbor k
Output: The optimal trainable parameters of TSLE

```
Compute the label set Nei(i) of instance x_i for i = 1, 2, ..., N
Initialize all trainable parameters with random values from Gaussian distribution
repeat
    Randomly select a data sample x_j and y_j
    Compute the output of the factorization layer Fac(x_j; V, A), Fac(y_j; U, B) and Fac(y; U, B) where y ∈ Nei(j)
    Compute the codewords c_{x_j}, c_{y_j} and c_y
    Recover the output label ŷ_j from c_{x_j}
    Compute the encoding loss L_E by Eq. (10) and the decoding loss L_D by Eq. (11)
    Compute the final loss L by Eq. (12)
    Update all trainable parameters with Adam algorithm
until Converge
```

Here, $\mathcal{L}_D$ is the decoding loss. For the i-th instance $\mathbf{x}_i$, $\mathbf{y}_i^1$ is the set of the positive labels in $\mathbf{y}_i$ and $\mathbf{y}_i^0$ is that of the negative labels. $|\cdot|$ measures the cardinality of a set. $(\hat{\mathbf{y}}_i)_e$ denotes the e-th entry of the predicted label $\hat{\mathbf{y}}_i$.

We note that existing label embedding approaches usually manipulate the encoding and decoding processes separately. To give our proposed model more robustness, the encoding and decoding losses are combined and leveraged by a positive constant α. A regularization term is also added to prevent overfitting. Then the final loss L can be formulated as:

$$\mathcal{L} = \mathcal{L}_E + \alpha\mathcal{L}_D + \lambda \sum_{\phi \in \Phi} ||\phi||^2 \tag{12}$$

where Φ denotes the set of all parameters and λ controls the regularization strength. $||\cdot||$ represents the l_2 norm of vectors or the Frobenius norm of matrices.

We adapt Adam (Kingma and Ba 2014) instead of vanilla Stochastic Gradient Descent to iteratively update the parameters with learning rate η. Adam has two main advantages: capability of dealing with sparse gradients and non-stationary objectives, and requiring little memory. Note that our proposed model is clearly defined, thus the computational graphs (Fig. 1) can be easily built and the model can be straightly implemented using Machine Learning Toolkits like TensorFlow (Abadi et al. 2016) or Caffe (Jia et al. 2014). The pseudo code of TSLE is summarized in Algorithm 1.

Once the parameters are learnt, we can predict the label of a test input $\hat{\mathbf{x}}$ by rounding $\hat{\mathbf{y}} = h_D(\mathbf{c}_{\hat{\mathbf{x}}})$.

Kernel View of TSLE

It is well-known that the theoretical properties of deep neural network are still not well understood. Therefore, we demonstrate a different view of TSLE for a better understanding.

In the first place, let us focus on the hidden layers in TEN. Technically speaking, each hidden layer can be designed as

any function that takes a matrix as input and outputs a vector. Consequently, we can remove all the activation functions and biases. Then the hidden layers linearly project the embedding vectors into a latent space. Moreover, we preserve only two hidden layers in the feature network and one in the label network:

$$\begin{aligned} \mathbf{c_x} &= W_x^2 W_x^1 \operatorname{Fac}(\mathbf{x}; V, A) \\ \mathbf{c_y} &= W_y^1 \operatorname{Fac}(\mathbf{y}; U, B) \end{aligned} \tag{13}$$

where W_x^1, W_x^2 and W_y^1 are weight parameters. Here, we set $W_x^2 = W_y^1$. Comparing Eq. (4) with Eq. (13), we can see that the specialised hidden layers project the embedding vectors in the same way as LM-kNN encodes the inputs.

Recall that a factorization layer is used to obtain the embedding vectors. While a recent work (Blondel et al. 2016) provides a kernel view of FM, our factorization layer actually maps the original inputs into a polynomial kernel space. It is worth pointing that TEN and LM-kNN have similar objective functions. Thus, if we regard the factorization layer as a preprocessing procedure of data, the specialised TSLE will have the same encoding phase as a kernelized LM-kNN, where the kernel function is exactly FM.

We have demonstrated the generalization ability of our proposed model. Furthermore, we show that TSLE has two main advantages. First, instead of linear projection, deep neural networks are used in encoding phase to better exploit high-order dependency. Second, TSLE is a typical margin-based algorithm and it is well-known that kernel trick can provide significant improvements for such algorithms. Since pairwise interactions commonly exist in multi-label datasets, FM will be a promising kernel function. Empirical study also proves that our proposed model outperforms LM-kNN.

Experiment

In this section, we evaluate the performance of our proposed TSLE and five state-of-the-art multi-label techniques on many real-world datasets over eight measurements. We conduct all experiments on a same workstation with an i7-5930K CPU, a TITAN Xp GPU and 64GB main memory running Linux platform.

Experimental Settings

Datasets We conduct experiments on seven real-world datasets from various domains.

- Cal500 (Turnbull et al. 2008): A music dataset containing human-generated musical annotations that describes 502 popular western musical tracks with 174 tags representing emotions, instruments, and other related concepts.
- Emotions (Trohidis et al. 2008): A music dataset consisting of hundreds of songs from 6 genres.
- Yeast (Elisseeff and Weston 2001): A biology dataset formed by micro-array expression data and phylogenetic profiles with 14 genes.
- Eurlex (Mencía and Fürnkranz 2008): A collection of documents about European Union law. Several EUROVOC descriptors, directory codes and subject matters are available, corresponding to three individual datasets: Eurlex_desc, Eurlex_dc and Eurlex_sm. Following the setting of (Zhang and Schneider 2012), we select the 10 most common labels to study.
- NUS-WIDE (Chua et al. 2009): A large-scale web image dataset with hundreds of thousands of instances that includes 500-dimensional bag of words based on SIFT descriptions for 81 concepts.

More detailed information about the datasets can be found on the website[1].

Baselines We compare TSLE with several state-of-the-art multi-label classification approaches:

- BR (Tsoumakas, Katakis, and Vlahavas 2010): Binary Relevance predicts each label independently with a binary classifier. In this paper, we use neural network as the binary classifier.
- ML-kNN (Zhang and Zhou 2007): Derived from the traditional k nearest neighbor algorithm, ML-kNN utilizes maximum a posteriori principle to determine the label set for the unseen instance.
- CPLST (Chen and Lin 2012): Based on minimizing an upper bound of the Hamming loss, CPLST combines the concepts of Principal Component Analysis (PCA) and Canonical Correlation Analysis (CCA) to improve PLST (Tai and Lin 2012) through the addition of feature information.
- LM-kNN (Liu and Tsang 2015): By linearly embedding features and labels into a low dimensional space, LM-kNN uses the Euclidean distance of instances in the latent space as a metric to find k nearest neighbors and takes the weighted average of their labels as the predicted labels.
- C2AE (Yeh et al. 2017): As the first deep neural network based label embedding approach for multi-label classification, C2AE integrates Deep Canonical Correlation Analysis (DCCA) (Reichart, Vulic, and Rotman 2018) and autoencoder to exploit label dependency.

Parameter Settings We implement our proposed TSLE based on TensorFlow. We randomly select 80% of the data for training and predict the labels with the rest 20%. The feature and label embedding matrices V and U are initialized with random values sampled from a standard Gaussian distribution. The dimension of embedding vector t is set to 256. Both h_E and h_D are composed of three fully connected layers with ReLU as activation functions. The hidden dimensions of h_E and h_D are $[64, 64, d]$ and $[d, 64, q]$ respectively, where $d = \min(32, q - 1)$ is the dimension of the latent space. Each base classifier of BR is a neural network with dimensions of $[64, 64, 1]$. We set $k = 3$ for ML-kNN, LM-kNN and our method. The default learning rates η of all baselines (expect CPLST) and TSLE are 10^{-3}. The default leverage parameter α and regularization parameter λ are set to 1 and 0.01 respectively. To test the stability of the proposed model, we also experiment TSLE on various learning rates ranging from 10^{-4} to 5×10^{-3} and different leverage

[1] http://mulan.sourceforge.net/datasets-mlc.html

Table 1: Results of Micro-F1 on all datasets (mean ± standard deviation), the best ones are in bold.

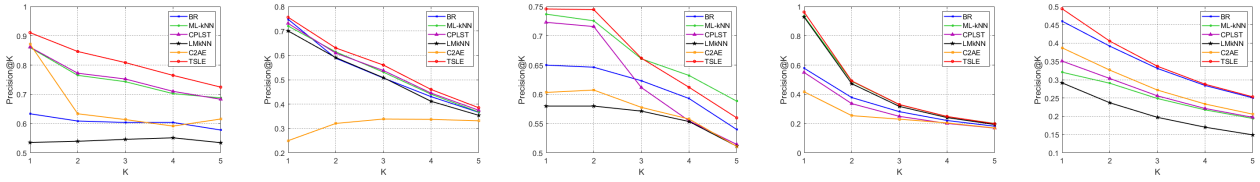
Datasets	BR	ML-kNN	CPLST	LM-kNN	C2AE	TSLE
Cal500	0.3643±0.0153	0.3162±0.0146	0.3329±0.0064	0.3551±0.0092	0.4586±0.0011	**0.4648**±0.0088
Emotions	0.6047±0.0178	0.6585±0.0110	0.6430±0.0051	0.6152±0.0113	0.4563±0.0113	**0.6933**±0.0040
Yeast	0.4037±0.0042	**0.6262**±0.0142	0.4670±0.0051	0.5365±0.0098	0.5459±0.0099	0.5908±0.0068
Eurlex_desc	0.3605±0.0093	0.6584±0.0061	0.3173±0.0106	0.7050±0.0122	0.3671±0.0146	**0.7057**±0.0032
Eurlex_dc	0.5455±0.0117	0.9193±0.0063	0.4928±0.0193	0.9206±0.0059	0.4091±0.0146	**0.9492**±0.0098
Eurlex_sm	0.4155±0.0070	0.8177±0.0059	0.5897±0.0194	0.8046±0.0070	0.3583±0.0070	**0.8230**±0.0060
NUS-WIDE	0.2356±0.0053	0.1781±0.0102	0.2236±0.0036	0.2751±0.0029	0.3685±0.0044	**0.3815**±0.0098

Table 2: Results of Macro-F1 on all datasets (mean ± standard deviation), the best ones are in bold.

Datasets	BR	ML-kNN	CPLST	LM-kNN	C2AE	TSLE
Cal500	0.1065±0.0077	0.0482±0.0049	0.0642±0.0077	0.1356±0.0092	0.1703±0.0012	**0.1994**±0.0037
Emotions	0.5782±0.0115	0.6362±0.0140	0.5741±0.0131	0.6044±0.0143	0.3353±0.0085	**0.6899**±0.0036
Yeast	0.2618±0.0039	0.3741±0.0206	0.1403±0.0159	0.3603±0.0094	0.3888±0.0028	**0.4044**±0.0215
Eurlex_desc	0.3320±0.0074	0.6022±0.0048	0.3024±0.0105	0.6800±0.0111	0.3657±0.0199	**0.6903**±0.0048
Eurlex_dc	0.4610±0.0149	0.8910±0.0097	0.3202±0.0087	0.8934±0.0066	0.3467±0.0068	**0.9286**±0.0198
Eurlex_sm	0.3448±0.0141	0.7980±0.0081	0.4729±0.0148	0.7819±0.0076	0.2758±0.0133	**0.8011**±0.0085
NUS-WIDE	0.0180±0.0023	0.0216±0.0035	0.0172±0.0007	0.0501±0.0076	**0.0694**±0.0104	0.0522±0.0021

Table 3: Results of Example-F1 on all datasets (mean ± standard deviation), the best ones are in bold.

Datasets	BR	ML-kNN	CPLST	LM-kNN	C2AE	TSLE
Cal500	0.3635±0.0163	0.3200±0.0135	0.3325±0.0056	0.3510±0.0082	0.4554±0.0011	**0.4611**±0.0073
Emotions	0.5629±0.0131	0.6175±0.0137	0.5011±0.0064	0.5855±0.0136	0.4498±0.0164	**0.6714**±0.0080
Yeast	0.3608±0.0058	0.5726±0.0155	0.4380±0.0060	0.5094±0.0098	0.5159±0.0098	**0.5748**±0.0076
Eurlex_desc	0.2898±0.0087	0.5718±0.0064	0.3320±0.0083	**0.7101**±0.0156	0.3489±0.0129	0.7006±0.0050
Eurlex_dc	0.4720±0.0121	0.8898±0.0090	0.5383±0.0066	0.9270±0.0053	0.4736±0.0062	**0.9566**±0.0112
Eurlex_sm	0.3534±0.0103	0.7777±0.0087	0.5840±0.0013	0.7600±0.0083	0.3460±0.0037	**0.8123**±0.0077
NUS-WIDE	0.1643±0.0088	0.0813±0.0020	0.1116±0.0015	0.1663±0.0070	0.2864±0.0056	**0.3334**±0.0027



(a) Cal500 (b) Emotions (c) Yeast (d) Eurlex_dc (e) NUS-WIDE

Figure 2: Precision@K of all methods on various datasets.

parameters ranging from 0.3 to 3. Other parameters in the baselines are set to their default values.

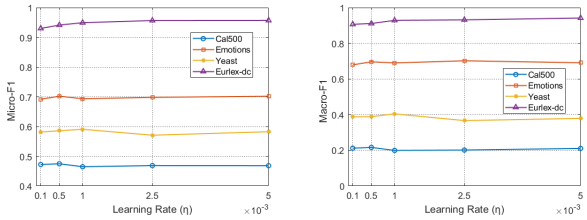
Measurements To better measure the performance, we adapt several widely-used metrics:

- Micro-F1: It calculates true positives/negatives and false positives/negatives over labels, and then computes an overall F-measure.
- Macro-F1: It is the unweighted mean of label F-measure.
- Example-F1: It is the unweighted mean of instance F-measure.
- Precision@K: It is the proportion of labels in the Top-K set that are correctly predicted.

Experimental Results

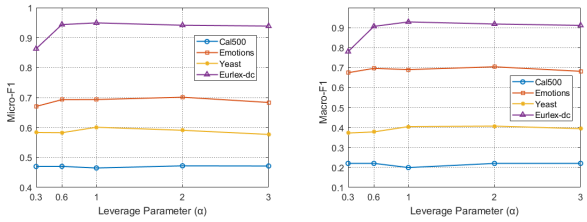
Prediction Performance Table 1, 2 and 3 list the Micro-F1, Macro-F1 and Example-F1 results of baselines and our model in respect of different datasets. Fig. 2 shows the Precision@K results on different datasets where the ranking position K ranges from 1 to 5. From the results, we can tell that:

- The proposed TSLE significantly outperforms all other approaches on both medium-sized and large-scale datasets. For example, on Emotions dataset, in term of Micro-F1, Macro-F1 and Example-F1, TSLE improves the best results of the baselines by 7.82%, 14.15% and 14.67%. From the precision perspective, TSLE shows consistent improvements over other methods across po-



(a) Micro-F1 (b) Macro-F1

Figure 3: Micro-F1 and Macro-F1 of TSLE *w.r.t.* different learning rates (η)



(a) Micro-F1 (b) Macro-F1

Figure 4: Micro-F1 and Macro-F1 of TSLE *w.r.t.* different leverage parameters (α)

sitions. These results demonstrate that TSLE achieves superior performance than those baselines.

- BR and ML-kNN underperform LM-kNN and TSLE over many measurements. Therefore, exploiting label correlations can notably improve the prediction performance.
- CPLST is much inferior to C2AE, LM-kNN and TSLE. With simple linear projection, CPLST cannot effectively explore the latent label spaces.
- It is worth pointing that C2AE works well on many datasets, which proofs that deep neural network can efficiently extract high-order label correlations. However, since its criterion does not optimize the discriminability of the codewords, it is unstable and sensitive to noisy data. Moreover, C2AE ignores the difficulty of learning high-order label correlations directly due to the sparsity of labels, and thus underperforms TSLE.
- LM-kNN and TSLE are the most successful methods on all datasets. However, TSLE is better for two reasons. First, a factorization layer, which works well under sparse setting, is used as a kernel function to learn pairwise correlations in advance. Second, DNN can efficiently exploit higher-order label dependency and effectively recover the output labels from codewords.

Parameter Sensitivity In this section, we explore the hyperparameter sensitivity of our model to find out the strength and relevance of the inputs in determining the variation in the output. Fig. 3 reports Micro-F1 and Macro-F1 results under different learning rates η on four datasets. Fig. 4 shows the same measurements under different loss leverage parameters α. The experimental results fluctuate lightly according to different orders of magnitude, but they are substantially robust within an acceptable range. In conclusion, the above results assure the quality and stability of TSLE.

Conclusion

To handle complicated label hierarchies, this paper proposes a novel Two-Stage Label Embedding (TSLE) paradigm for MLC. Based on NFM (He and Chua 2017), a Twin Encoding Network is introduced to jointly embed features and labels into a latent space. In the first stage, a factorization layer extracts pairwise feature and label interactions. Then a set of hidden layers is applied to learn higher-order correlations in the second stage. Inspired by (Yeh et al. 2017), we use deep neural network to recover the output labels from the codewords of instances in decoding phase. To give TSLE more robustness, the final objective function is leveraged between two parts: a max margin formulated encoding loss for discriminative and predictable codewords, and a label-correlation aware decoding loss. Furthermore, a regularization term is also added to alleviate overfitting. For a better understanding, we provide a kernel insight to show the generalization ability of TSLE. In the experiments, we demonstrate that our TSLE notably outperforms other state-of-the-art methods with quality assurance.

Acknowledgments

This research is supported by National Key Research and Development Program (2017YFB1201001).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P. A.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; and Zheng, X. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, 265–283.

Blondel, M.; Ishihata, M.; Fujino, A.; and Ueda, N. 2016. Polynomial networks and factorization machines: New insights and efficient training algorithms. In *ICML*, 850–858.

Cao, B.; Zhou, H.; Li, G.; and Yu, P. S. 2016. Multi-view machines. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 427–436. ACM.

Chen, Y.-N., and Lin, H.-T. 2012. Feature-aware label space dimension reduction for multi-label classification. In *NIPS*, 1538–1546.

Chua, T.-S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y.-T. 2009. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*.

Deng, C.; Chen, Z.; Liu, X.; Gao, X.; and Tao, D. 2018a. Triplet-based deep hashing network for cross-modal retrieval. *IEEE Transactions on Image Processing* 27(8):3893–3903.

Deng, C.; Liu, X.; Li, C.; and Tao, D. 2018b. Active multi-kernel domain adaptation for hyperspectral image classification. *Pattern Recognition* 77:306–315.

Elisseeff, A., and Weston, J. 2001. A kernel method for multi-labelled classification. In *NIPS*, 681–687.

Gong, C.; Liu, T.; Tao, D.; Fu, K.; Tu, E.; and Yang, J. 2015. Deformed graph laplacian for semisupervised learning. *IEEE transactions on neural networks and learning systems* 26(10):2261–2274.

Gong, C.; Tao, D.; Maybank, S. J.; Liu, W.; Kang, G.; and Yang, J. 2016. Multi-modal curriculum learning for semi-supervised image classification. *IEEE Transactions on Image Processing* 25(7):3249–3260.

He, X., and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 355–364.

Hsu, D.; Kakade, S.; Langford, J.; and Zhang, T. 2009. Multi-label prediction via compressed sensing. In *NIPS*, 772–780.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R. B.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 675–678.

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

Liu, W., and Tsang, I. W. 2015. Large margin metric learning for multi-label prediction. In *AAAI*, 2800–2806.

Liu, W., and Tsang, I. W. 2017. Making decision trees feasible in ultrahigh feature and label dimensions. *The Journal of Machine Learning Research* 18(1):2814–2849.

Liu, W.; Xu, D.; Tsang, I.; and Zhang, W. 2018. Metric learning for multi-output tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liu, W.; Tsang, I. W.; and Müller, K.-R. 2017. An easy-to-hard learning paradigm for multiple classes and multiple labels. *The Journal of Machine Learning Research* 18(1):3300–3337.

Mencía, E. L., and Fürnkranz, J. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML/PKDD*, 50–65.

Nam, J.; Loza Mencía, E.; Kim, H. J.; and Fürnkranz, J. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *NIPS*, 5419–5429.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2011. Classifier chains for multi-label classification. *Machine Learning* 85(3):333–359.

Reichart, R.; Vulic, I.; and Rotman, G. 2018. Bridging languages through images with deep partial canonical correlation analysis. In *ACL*, 910–921.

Rendle, S. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3(3):57.

Schapire, R. E., and Singer, Y. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning* 39(2/3):135–168.

Tai, F., and Lin, H.-T. 2012. Multilabel classification with principal label space transformation. *Neural Computation* 24(9):2508–2542.

Trohidis, K.; Tsoumakas, G.; Kalliris, G.; and Vlahavas, I. P. 2008. Multi-label classification of music into emotions. In *ISMIR*, volume 8, 325–330.

Tsoumakas, G.; Katakis, I.; and Vlahavas, I. 2010. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*. 667–685.

Turnbull, D.; Barrington, L.; Torres, D. A.; and Lanckriet, G. R. G. 2008. Semantic annotation and retrieval of music and sound effects. *IEEE Trans. Audio, Speech & Language Processing* 16(2):467–476.

Wang, J.; Yang, Y.; Mao, J.; Huang, Z.; Huang, C.; and Xu, W. 2016. CNN-RNN: A unified framework for multi-label image classification. In *CVPR*, 2285–2294.

Yang, E.; Deng, C.; Li, C.; Liu, W.; Li, J.; and Tao, D. 2018. Shared predictive cross-modal deep quantization. *IEEE Transactions on Neural Networks and Learning Systems* (99):1–12.

Yeh, C.-K.; Wu, W.-C.; Ko, W.-J.; and Wang, Y.-C. F. 2017. Learning deep latent space for multi-label classification. In *AAAI*, 2838–2844.

Zhang, Y., and Schneider, J. G. 2011. Multi-label output codes using canonical correlation analysis. In *AISTATS*, 873–882.

Zhang, Y., and Schneider, J. G. 2012. Maximum margin output coding. In *ICML*.

Zhang, M., and Zhou, Z. 2006. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Trans. Knowl. Data Eng.* 18(10):1338–1351.

Zhang, M.-L., and Zhou, Z.-H. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition* 40(7):2038–2048.

Zhao, F.; Huang, Y.; Wang, L.; and Tan, T. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 1556–1564.