Neural Conditional Energy Models for Multi-Label Classification

How Jing
Dept. of Computer Science
National Taiwan University
Taipei, Taiwan
kublai.jing@gmail.com

Shou-De Lin
Dept. of Computer Science
National Taiwan University
Taipei, Taiwan
sdlin@csie.ntu.edu.tw

***Abstract*—Multi-label classification (MLC) is a type of structured output prediction problems where a given instance can be associated to more than one labels at a time. From the probabilistic point of view, a model predicts a set of labels $\mathbf{y}$ given an input vector $\mathbf{v}$ by learning a conditional distribution $p(\mathbf{y}|\mathbf{v})$. This paper presents a powerful model called a Neural Conditional Energy Model (NCEM) to solve MLC. The model can be viewed as a hybrid deterministic-stochastic network of which we use a deterministic neural network to transform the input data, before contributing to the energy landscape of $\mathbf{v}$, $\mathbf{y}$, and a single stochastic hidden layer $\mathbf{h}$. Non-linear transformation given by the neural network makes our model more expressive and more capable of capturing complex relations between input and output, and using deterministic neurons facilitates exact inference. We present an efficient learning algorithm that is simple to implement. We conduct extensive experiments on 15 real-world datasets from wide variety of domains with various evaluation metrics to confirm that NCEM is significantly superior to current state-of-the-art models most of the time based on pair-wise *t-test* at 5% significance level. The MATLAB source code to replicate our experiments are available at https://github.com/Kublai-Jing/NCEM.**

I. Introduction

The goal of multi-label classification (MLC) is to classify a given input vector $\mathbf{v}$ into a set of class labels $\mathbf{y}$ by learning a mapping function $g : \mathbf{v} \mapsto \mathbf{y}$ where $\mathbf{y} \in \{0, 1\}^M$ and M being the number of distinct labels. Such scenario may arise in applications such as text mining, vision, or bioinformatics. For instance, a document can be associated with more than one categories; a gene is very often multi-functional; a picture can contain multiple objects. MLC is attracting much more research attentions because there is a wide range of potential applications in various domains such as scene classification [3], music and text classification [13], [33], object recognition [7], ... etc.

In MLC, one important issue is to capture the correlations between labels. This is believed to be the key to a state-of-the-art MLC model because dependencies between labels could be a good feature when trying to learn the correct mapping [34]. For example, a document being tagged as *sports* should have a higher probability to also be tagged as *entertainment*, whereas it is not likely to be tagged as *politics* at the same time. Effective exploitation of this information is crucial for the success of a MLC system, while ignoring it and simply considering each label independently may significantly hamper system performance.

Several algorithms have been proposed for the purpose of encoding label correlations into their models [5], [8], [44], [23], [43]. We will briefly introduce some of them in Section 2 when talking about related work.

While a lot of previous works have focused on encoding label correlations into the model in order to incorporate label dependencies during training, there is another class of approaches based on undirected graphical models that does not explicitly "hand-engineer" label correlations like many of the previous approaches do. Conditional restricted Boltzmann machines (CRBMs) fall into this category. CRBMs are basically undirected models with hidden units, aiming at learning a conditional distribution between input $\mathbf{v}$ and output $\mathbf{y}$ [14], [15]. In CRBMs, label correlations can be automatically captured by the hidden units. Without having prior knowledge about how labels are structured (such as trees or chains), we simply maximize the conditional likelihood $\mathbf{p}(\mathbf{y}|\mathbf{v})$ to fit the training data. Previous results show that CRBMs work well on some multi-label tasks [15], [19] and various other problems [25], [31]. Our preliminary experiments also confirm that CRBMs outperform some popular MLC algorithms such as [5], [44], and is comparable to more recent methods like [42], [43]. This preliminary results are shown in Table 1.

Probably the most important reason why CRBMs are so widely applicable to various tasks is that, given the data, inferring the hidden states can be done exactly, leading to an efficient approximate maximum likelihood learning algorithm called the Contrastive Divergence (CD) that has been used extensively in training related undirected models [9].

Nonetheless, such models may be restricted because they have to learn the mapping from a low-level representation $\mathbf{v}$ to a high-level concept $\mathbf{y}$ using only a single stochastic hidden layer. Whereas in MLC the relationships between input and output could be highly complex and non-linear, requiring more modeling power to capture all the richness present in the data. Limiting ourselves with one hidden layer may prevent us from learning more complicated relationships [1].

One plausible solution is to extend such model by introducing multiple hidden layers to form a multi-layer graphical model, usually called a deep Boltzmann machine (DBM) [24]. Multiple hidden layers are used to capture the statistical patterns of the input to benefit later prediction. However, performing conditional maximum likelihood learning as well as posterior inference (given the data) are difficult in such

TABLE I: Results of pair-wise *t-test* applied to CRBM compared with other popular methods with different metrics. Each cell contains three numbers: (from left to right) the counts that CRBM is significantly better/tied/significantly worse of 15 datasets we use. Here the significance level of *t-test* is set to 5%. More details about experiments will come in Section 5.

Algorithm	*Ranking loss*	*Coverage*	*Average precision*	*Micro-f1*
ECC [23]	14 / 1 / 0	14 / 1 / 0	12 / 3 / 0	11 / 3 / 1
LEAD [43]	12 / 3 / 0	12 / 3 / 0	13 / 2 / 0	14 / 1 / 0
LIFT [42]	7 / 4 / 4	8 / 4 / 3	9 / 5 / 1	13 / 1 / 1

model because of the presence of multiple stochastic hidden layers. Due to this reason, DBMs were mostly trained in a generative manner to model the density of the input $p(\mathbf{v})$ using variational method and stochastic approximation procedure [24]. Then for prediction, the whole model was treated as a deterministic feed-forward network with labels placed at the top, and trained with back-propagation where the objective is cross-entropy or squared distance. Conditional likelihood $p(\mathbf{y}|\mathbf{v})$ was never used as the training objective. In such case, we throw away the benefit of probabilistic modeling, such as inferring $p(\mathbf{y}_S|\mathbf{v}, \mathbf{y}_{-S})$, where S is some subset of labels that are given a priori. Moreover, sometimes a given input could be associated of multiple set of labels. For example, an image could possibly be associated to more than one set of tags due to the variability of users who tag it. However, for an identical input, a neural network always produces the same result because the mapping from $\mathbf{v}$ to $\mathbf{y}$ is assumed to be uni-modal and deterministic in a neural net. While a rich probabilistic model could potentially capture highly multi-modal distribution, resulting to a possibly more powerful model that can produce multiple explanations for a single $\mathbf{v}$. See [29] for a good example of probabilistic models generating different tags for an image.

In this paper, we propose Neural Conditional Energy Model (NCEM) that 1.) improves the performance and expressive power of CRBMs, and 2.) overcomes the limitation in using DBMs for discrimination where one has to resort to approximate inference technique that is either inaccurate (such as mean-field variational inference) or inefficient (such as Markov Chain Monte Carlo) in order to learn the parameters. To be specific, our model is a hybrid deterministic, stochastic network with two types of hidden layers. First, the goal of deterministic hidden layers, consist of deterministic neurons, is to extract patterns in the data before contributing to the energy landscape. Second, the goal of stochastic hidden layers, consist of random variables, is to enforce the discriminative power for making predictions. The network as a whole, being a single discriminative model then enjoys a simple, efficient learning algorithm (Section 4).

The main contributions of this paper are summarized as follows:

- We empirically show that CRBMs are highly competitive to current state-of-the-art MLC methods. Although, as mentioned above, using CRBMs for multi-label problems (or more generally, structured output problems) seems to be natural, we have not yet found extensive experiments and comparison with other popular MLC algorithms on benchmark datasets. We believe this is the first ever work that performs this comparison, and we hope our results can attract more research attention on improving this class of models to further benefit the field of MLC.
- Built upon the success of CRBMs, we present our hybrid deterministic, stochastic network called the Neural Conditional Energy Model, along with its learning algorithm to capture the rich relations between $\mathbf{v}$ and $\mathbf{y}$.
- We conduct our experiments on 15 real-world datasets from a wide range of domains, and confirm that our model performs significantly better than CRBMs and other state-of-the-art models with statistical guarantees.

II. Related Work

There have been many multi-label classification algorithms proposed. Here we briefly introduce some of them. For more details, we refer to [17], [35], [45]. Existing approaches roughly fall into two categories: *Algorithm Adaptation* (AA) and *Problem Transformation* (PT). Methods in AA extend algorithms designed for multi-class problems to solve MLC. Some well-known examples include those that adapt *ada-boost* [27], *decision tree* [38], *k-nearest neighbors* [44], [5] and *neural networks* [16]. PT transforms the problem of multi-label classification to a multi-class problem, or possibly other type of problems, then solves the transformed tasks. For example, *Label Powerset* (LP) methods transform the problem by treating each label combination that appears in the training set as a distinct class label, then the problem becomes a multi-class one where traditional classification algorithms can be exploited [37]. On the other hand, *Binary Relevance* (BR) methods transform MLC into several binary classification problems and solve them accordingly [23], [6], [3].

Here, we highlight three methods that we will compare in the experiments section, as they are considered state-of-the-art with superior performance, namely *ensemble of classifier chains* (ECC), *multi-label Learning by Exploiting lAbel Dependency* (LEAD), and *multi-label learning with Label specIfic* FeaTures (LIFT).

In ECC [23], the task of learning label dependencies is divided into several binary classification problems. Each binary classifier would incorporate the predictions propagated from the previous ones to make its own prediction. More specifically, there are in total M binary classifiers to be learned in one *chain*, where M is the number of distinct classes. Assuming that we are currently training the lth classifier, the conditional likelihood $p(y_l|\mathbf{v}, y_1, y_2, ..., y_{l-1})$ would be optimized. Thus, M classifiers form a *chain* in which latter classifiers are trained using the results of former predictions as additional features; hence the label correlations are learned. Finally, many such

chains with different data subset and different ordering of permutations of the labels are ensembled to make the final prediction. *Classifier chain* (CC) model is shown to be a deterministic approximation of *probabilistic classifier chain* (PCC) [6], where the problem is formalized from Bayes point of view. However in their proposed version of PCC, the computational complexity is exponential (w.r.t. M), hence we will not compare PCC in the experiments section as many of the datasets we use have number of labels way beyond 20, making it impossible to train PCC within a reasonable time.

LEAD is a recently proposed model [43]. LEAD first constructs binary classifiers for each label independently. Then it uses the error produced by classifiers to learn the structure of a Bayesian network that encodes the label dependencies. Finally, it trains classifiers that takes label dependencies into account via the Bayesian network structure.

LIFT considers the problem from a different perspective [42]. While most methods aim at capturing the dependencies between labels, LIFT aims at finding label-dependent features for better discriminative power. It uses k-means to cluster data into several groups, and create a mapping function that maps from the original feature space $\mathbf{v}$ to a label-specific space. Then the classifiers are trained in this new space instead of the old one for prediction.

We note that LEAD and LIFT have been compared with ML-KNN [44], ECC [23], B-SVM [3], and BP-MLL [16], all popular algorithms for MLC, and both of these two models outperform previous methods significantly. Also, methods that aim to reduce the computation based on singular value decomposition will not be compared with our methods, because they have comparable performance of BR, which is inferior than LIFT and LEAD [20]. So, we will mainly focus on the comparison of LEAD and LIFT with our method. More details about experiments will come later in Section 5.

III. Preliminaries

A. Conditional Restricted Boltzmann Machines

A conditional restricted Boltzmann machine (CRBM, see Figure 1(a)) is an undirected graphical model with its energy function defined as

$$E(\mathbf{v},\mathbf{h},\mathbf{y}) = -\mathbf{v^T W h} - \mathbf{y^T U h} - \mathbf{v^T L y} - \mathbf{c^T h} - \mathbf{d^T y}.$$

where $\mathbf{W}$ is a matrix of weights between elements of $\mathbf{v}$ and $\mathbf{h}$, $\mathbf{U}$ is a matrix of weights between elements of $\mathbf{y}$ and $\mathbf{h}$, and $\mathbf{L}$ captures the interactions between $\mathbf{v}$ and $\mathbf{y}$. $\mathbf{c}, \mathbf{d}$ are biases for $\mathbf{h}$ and $\mathbf{y}$ respectively. The conditional probability distributions are defined by

$$p(\mathbf{y}|\mathbf{v}) = \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h},\mathbf{y}))}{\sum_{\mathbf{h},\mathbf{y}} \exp(-E(\mathbf{v},\mathbf{h},\mathbf{y}))},$$

$$p(\mathbf{h}|\mathbf{v},\mathbf{y}) = \prod_{\mathbf{j=1}}^{|\mathbf{h}|} \sigma(\mathbf{v^T W_{\bullet j}} + \mathbf{y^T U_{\bullet j}} + c_j),$$

$$p(\mathbf{y}|\mathbf{v},\mathbf{h}) = \prod_{\mathbf{l=1}}^{|\mathbf{y}|} \sigma(\mathbf{v^T L_{\bullet l}} + \mathbf{U_{l\bullet} h} + d_l).$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function and we use $\mathbf{X_{\bullet j}}$ to denote the jth column vector in the matrix $\mathbf{X}$, and $\mathbf{X_{i\bullet}}$ to denote the ith row vector in matrix $\mathbf{X}$. Unlike traditional restricted Boltzmann machines (RBMs) that are used for modeling the density of the data, CRBMs learn to capture the conditional distribution $\mathbf{p(y|v)}$ using a single stochastic hidden layer $\mathbf{h}$. Many variants of such model have been successfully applied to problems such as collaborative filtering [25], automatic music tagging [18], and motion style modeling [31].

Learning in a CRBM is usually done using gradient ascent in conditional log-likelihood, $\log p(\mathbf{y}|\mathbf{v})$, with training set $\{\mathbf{V},\mathbf{Y}\}$. We can write the conditional log-likelihood term for a single training case as

$$\log p(\mathbf{y}|\mathbf{v}) = \log \sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h},\mathbf{y})) - \log \sum_{\mathbf{h},\mathbf{y}^*} \exp(-E(\mathbf{v},\mathbf{h},\mathbf{y}^*)).$$

and differentiating with respect to a parameter θ to get the gradient

$$\frac{\partial \log p(\mathbf{y}|\mathbf{v})}{\partial \theta} = \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v},\mathbf{y}) \frac{\partial - E(\mathbf{v},\mathbf{h},\mathbf{y})}{\partial \theta} - \sum_{\mathbf{h},\mathbf{y}^*} p(\mathbf{h},\mathbf{y}^*|\mathbf{v}) \frac{\partial - E(\mathbf{v},\mathbf{h},\mathbf{y}^*)}{\partial \theta}. \quad (1)$$

The first term in Equation 1, also known as the positive phase during learning, can be computed exactly because hidden units are conditionally independent given the visible and label vectors, hence the distribution $p(\mathbf{h}|\mathbf{v},\mathbf{y})$ factorizes. The second term, also known as the negative phase during learning, however is intractable to compute except in the special case where the size of the label space is small [14]. To see this, we can rewrite the gradient rule in Equation 1 in the 'free energy' form

$$\frac{\partial \log p(\mathbf{y}|\mathbf{v})}{\partial \theta} = \frac{\partial - F(\mathbf{v},\mathbf{y})}{\partial \theta} - \sum_{\mathbf{y}^*} p(\mathbf{y}^*|\mathbf{v}) \frac{\partial - F(\mathbf{v},\mathbf{y}^*)}{\partial \theta}, \quad (2)$$

where $F(\mathbf{v},\mathbf{y})$ is called the free energy

$$\begin{aligned} F(\mathbf{v},\mathbf{y}) &= -\log \sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h},\mathbf{y})) \\ &= -\sum_j \log\left(1 + \exp(c_j + \mathbf{v^T W_{\bullet j}} + \mathbf{y^T U_{\bullet j}})\right) \\ &\quad - \mathbf{d^T y} - \mathbf{v^T L y}. \end{aligned}$$

For any given pair of $(\mathbf{v},\mathbf{y})$, we can compute its free energy with time linear to the number of hidden units. However, to compute the second term in Equation 2, we need to visit all possible label combinations in the output space. In multi-class problems, this number is linear w.r.t. the number of labels, while in more general problems such as MLC, the number is exponential; hence it is not feasible to exhaustively enumerate all of them.

It is possible to estimate the negative phase gradient by drawing samples from the model using Markov Chain Monte Carlo (MCMC). Since both $p(\mathbf{y}|\mathbf{v},\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v},\mathbf{y})$ are factorial distributions, we can define an efficient block Gibbs sampling procedure by alternately updating $\mathbf{h}$ and $\mathbf{y}$. However,

in practice it usually takes a long time to wait for the Markov Chain to reach the stationary distribution.

There is, however, a fast approximation algorithm for estimating the negative phase gradient for CRBMs, called Contrastive Divergence (CD) [4], [9]. CD works by starting the chain at the training vector, and running for only a few steps to obtain the negative phase statistics. This method clearly does not follow the likelihood gradient, yet it does a reasonable job and has been used extensively in learning restricted Boltzmann machines family [10], [25].

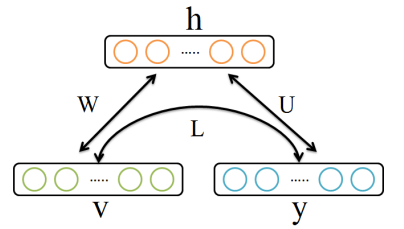
B. Deep Boltzmann Machines

While CD is applicable to train CRBMs without paying much computation, a single hidden layer may not be powerful enough to model all the statistical correlations observed in the data. Usually, the complexity of the model can be improved by adding extra hidden layers to form a hierarchical model [1], [11], [24]. A deep Boltzmann Machine (see Figure 1(b)) is a type of deep graphical model that is popular in the last few years for many tasks [29], [30]. For example we can define the following energy function for a DBM that has three hidden layers $\mathbf{h^1}, \mathbf{h^2}, \mathbf{h^3}$ and one label layer $\mathbf{y}$,

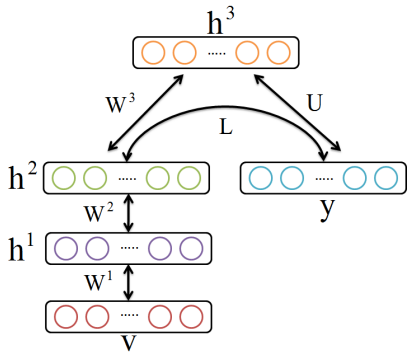
$$\begin{aligned} E(\mathbf{v}, \mathbf{h^1}, \mathbf{h^2}, \mathbf{h^3}, \mathbf{y}) = &-\mathbf{v^T W^1 h^1} - \mathbf{h^{1^T} W^2 h^2} - \mathbf{h^{2^T} W^3 h^3} \\ &-\mathbf{h^{2^T} L y} - -\mathbf{y^T U h^3} \\ &-\mathbf{c^{1^T} h^1} - \mathbf{c^{2^T} h^2} - \mathbf{c^{3^T} h^3} - \mathbf{d^T y}. \end{aligned}$$

Conditional maximum likelihood learning has the same expression as in CRBM. However, since now hidden units are *not* conditionally independent given visible and label vectors, we cannot perform exact inference to estimate the positive phase gradient which makes learning difficult. Moreover, since now it is required to sample from the space consisting of $\{\mathbf{h^1}, \mathbf{h^2}, \mathbf{h^3}, \mathbf{y}\}$, we would expect the Markov Chain to take even longer to mix when performing Gibbs sampling to estimate the negative phase gradient. Also, stochastic approximation procedure (SAP) cannot be used in learning conditional distribution because each given instance $\mathbf{v^s}$ leads to a unique distribution $p(\mathbf{y}|\mathbf{v^s})$, preventing us from using 'persistent' contrastive divergence (PCD) [32].

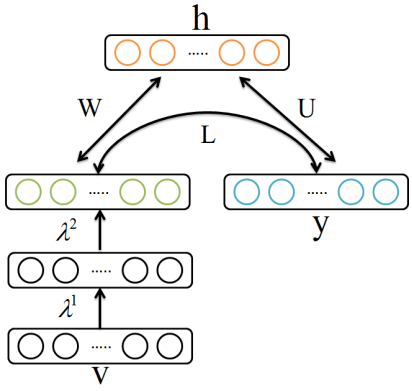
Due to these issues in discriminative training of DBMs, they are mostly trained in a generative manners using mean-field inference and PCD to learn the distribution of the input data $p(\mathbf{v})$ first, then use the weights learned in DBMs to initialize the parameters in a feed-forward neural network. Finally a different objective function, such as cross-entropy error is used for back-propagation fine-tuning instead of conditional likelihood. See [24] for details.



(a) CRBM



(b) DBM



(c) NCEM

Fig. 1: (a): A conditional restricted Boltzmann machine (CRBM). Given visible and label vector, hidden units are independent. (b): A three-layer deep Boltzmann machine (DBM). Multiple stochastic hidden layers makes $\mathbf{h^1}, \mathbf{h^2}, \mathbf{h^3}$ *dependent* on each other given $\mathbf{v}$ and $\mathbf{y}$. (c): A three-layer neural conditional energy model (NCEM). Replacing low-level hidden layers by deterministic neurons makes inference easy. Colored nodes refer to random variables, black nodes refer to deterministic neurons.

IV. Neural Conditional Energy Models

A. The Model

A Neural Conditional Energy Model (NCEM) (see Figure 1(c)) is defined according to the following energy function,

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}, \mathbf{y}) = &-\mathbf{f}(\mathbf{v}; \lambda)^\mathbf{T} \mathbf{W h} - \mathbf{y^T U h} - \mathbf{f}(\mathbf{v}; \lambda)^\mathbf{T} \mathbf{L y} \\ &- \mathbf{c^T h} - \mathbf{d^T y}. \end{aligned} \tag{3}$$

where $\mathbf{f}(\mathbf{v}; \lambda)$ denotes the output given by a deterministic, feed-forward neural network parameterized by λ. Note that $\mathbf{f}$ here can be any differentiable function. We use neural network because the optimization in neural network is well-studied, and it is a powerful model that can learn non-linear structure.

There is only one stochastic hidden layer in the model, and we could have one or many deterministic hidden layers, as shown in Figure 1(c). Since stochasticity only takes place at the top layer, low-level representations can be efficiently transformed into useful features via a single feed-forward pass without performing any kind of inference procedure. This is the most advantageous feature in NCEM where no approximate

posterior inference is involved in the model. Note that our model has similar motivation to [22], yet *Conditional Neural Field* (CNF) is specifically designed for sequence labeling in NLP, whereas our model is more general and can deal with arbitrary output distribution.

B. Learning

We train our model by maximizing the conditional log-likelihood of a training set using mini-batch stochastic gradient ascent. Specifically, given a training set $\{\mathbf{V}, \mathbf{Y}\}$, we learn the parameters $\{\mathbf{W}, \mathbf{U}, \mathbf{L}, \mathbf{c}, \mathbf{d}, \lambda\}$ to maximize $\log p(\mathbf{Y}|\mathbf{V}) = \sum_i \log p(\mathbf{y}^i|\mathbf{v}^i)$. The conditional log-likelihood for a single training case takes the same form as in a CRBM,

$$\frac{\partial \log p(\mathbf{y}|\mathbf{v})}{\partial \theta} = \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v},\mathbf{y}) \frac{\partial - E(\mathbf{v},\mathbf{h},\mathbf{y})}{\partial \theta} - \sum_{\mathbf{h},\mathbf{y}^*} p(\mathbf{h},\mathbf{y}^*|\mathbf{v}) \frac{\partial - E(\mathbf{v},\mathbf{h},\mathbf{y}^*)}{\partial \theta}.$$

with its energy function defined as Equation 3. We use $\mathbf{h}$ to denote the stochastic hidden layer at the very top of the model. We will not explicitly denote deterministic hidden layers hereafter, as it is already encoded in λ. Again, we need to compute two expectations under the distributions $p(\mathbf{h}|\mathbf{v},\mathbf{y})$ and $p(\mathbf{h},\mathbf{y}|\mathbf{v})$. The first term now can be computed exactly since $p(\mathbf{h}|\mathbf{v},\mathbf{y})$ is a factorial distribution. The second term can be approximated by contrastive divergence, as the sample space is now $\{\mathbf{h},\mathbf{y}\}$, and and efficient block Gibbs sampling procedure can be designed.

For clarity, we work out the gradient update rule for parameters in a NCEM here. First, we turn to the free energy form of the gradient,

$$\frac{\partial \log \mathbf{p}(\mathbf{y}|\mathbf{v})}{\partial \theta} = \frac{\partial - F(\mathbf{v},\mathbf{y})}{\partial \theta} - \sum_{\mathbf{y}^*} p(\mathbf{y}^*|\mathbf{v}) \frac{\partial - F(\mathbf{v},\mathbf{y}^*)}{\partial \theta},$$

where $F(\mathbf{v},\mathbf{y})$ is the corresponding free energy,

$$\begin{aligned} F(\mathbf{v},\mathbf{y}) &= -\log \sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h},\mathbf{y})) \\ &= -\sum_j \log\Big(1+\exp(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}})\Big) \\ &\quad - \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{L}\mathbf{y} - \mathbf{d}^{\mathbf{T}}\mathbf{y}. \end{aligned}$$

The derivatives of the free energy w.r.t $\{\mathbf{W}, \mathbf{U}, \mathbf{L}, \mathbf{c}, \mathbf{d}\}$ are

$$\begin{aligned} \frac{\partial F(\mathbf{v},\mathbf{y})}{\partial W_{ij}} &= -\frac{\exp(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}})}{1+\exp(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}})} v_i, \\ \frac{\partial F(\mathbf{v},\mathbf{y})}{\partial U_{lj}} &= -\frac{\exp(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}})}{1+\exp(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}})} y_l, \\ \frac{\partial F(\mathbf{v},\mathbf{y})}{\partial L_{il}} &= -y_l \mathbf{f}(\mathbf{v};\lambda)_i, \\ \frac{\partial F(\mathbf{v},\mathbf{y})}{\partial c_j} &= -\frac{\exp(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}})}{1+\exp(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}})}, \\ \frac{\partial F(\mathbf{v},\mathbf{y})}{\partial d_l} &= -y_l. \end{aligned}$$

We see that $\frac{\exp(x)}{1+\exp(x)} = 1 - \sigma(-x)$. By making use of the equivalence: $\sigma(-x) = 1 - \sigma(x)$, we obtain the following terser form of the derivatives.

$$\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial W_{ij}} = -\sigma\Big(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}}\Big) v_i, \quad (4)$$

$$\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial U_{lj}} = -\sigma\Big(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}}\Big) y_l, \quad (5)$$

$$\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial L_{il}} = -y_l \mathbf{f}(\mathbf{v};\lambda)_i, \quad (6)$$

$$\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial c_j} = -\sigma\Big(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}}\Big), \quad (7)$$

$$\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial d_l} = -y_l. \quad (8)$$

Next, we work out the derivative of the free energy w.r.t. the output given by the neural network.

$$\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial \mathbf{f}(\mathbf{v};\lambda)_i} = -\sum_j \sigma\Big(c_j + \mathbf{f}(\mathbf{v};\lambda)^{\mathbf{T}}\mathbf{W}_{\bullet\mathbf{j}} + \mathbf{y}^{\mathbf{T}}\mathbf{U}_{\bullet\mathbf{j}}\Big)\mathbf{W}_{\mathbf{ij}} - (\mathbf{L}\mathbf{y})_i. \quad (9)$$

Afterwards, the derivatives of the parameters in the neural network $\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial \lambda}$ can then be derived by standard back-propagation using the chain rule [2].

After deriving the derivatives of the free energy for all parameters in our model, we summarize the learning algorithm as follows (for a single training case $(\mathbf{v},\mathbf{y})$)

1) Given a visible vector $\mathbf{v}$, feed $\mathbf{v}$ through the neural network and obtain output $\mathbf{f}(\mathbf{v};\lambda)$.
2) Given an output $\mathbf{f}(\mathbf{v};\lambda)$ and a label vector $\mathbf{y}$, run Gibbs sampling for k steps by alternately updating $\mathbf{h}$ and $\mathbf{y}$.
3) Collect the sample produced by the previous step, call it $\mathbf{y}^{\mathbf{k}}$.
4) Update the parameter using the following rule,
$$\theta^{new} = \theta^{old} + \alpha(-\frac{\partial F(\mathbf{v},\mathbf{y})}{\partial \theta} + \frac{\partial F(\mathbf{v},\mathbf{y}^{\mathbf{k}})}{\partial \theta})$$
where α is the learning rate, $\theta \in \{\mathbf{W}, \mathbf{U}, \mathbf{c}, \mathbf{d}, \lambda\}$, and the derivatives of the free energy function for all parameters as specified in Equations 4-9.
5) Repeat steps 1-4.

We call this algorithm conditional stochastic back-propagation (CSBP).

C. Unsupervised Pre-Training

It is relatively hard to jointly learn the whole NCEM at once if we start with randomly initialized weights. We can alleviate this problem by first initializing the parameters in the model to sensible values, before performing CSBP.

Fortunately, there is a lot of work on initializing the parameters in a neural network using unsupervised learning. Training a stack of restricted Boltzmann machines or a stack of autoencoders are two of the many strategies [40], [11]. We

will learn a stack of RBMs in this paper because it works better in our experiments. In the experiments section we will show that unsupervised pre-training indeed helps the model find a better solution.

D. Prediction

At test time, we need to perform a Maximum a Posterior (MAP) inference to find the label set that has the highest posterior probability $p(\mathbf{y^{test}}|\mathbf{v^{test}})$. This inference is intractable, hence we again resort to Gibbs sampling to find such label set. To be specific, for a given test case $\mathbf{v^{test}}$, we first randomly initialize stochastic hidden units and label units, then feed $\mathbf{v^{test}}$ through the network to get $\mathbf{f}(\mathbf{v^{test}}; \lambda)$. Then we start to run alternating Gibbs sampling by updating $\mathbf{h}$ and $\mathbf{y}$ in turns, and collect label sample every 10 Gibbs steps and aggregate the results. This gives us a confidence score for each label that can be used to evaluate the quality of the prediction given by the model.

E. Summary

NCEM uses a deterministic neural network to transform the data, and a single stochastic hidden layer is used to capture the correlations between the transformed input and the labels. Comparing to CRBMs, we can potentially gain more modeling power because of the presence of multiple hidden layers. Comparing to DBMs, we can jointly train the whole model more efficiently to learn $p(\mathbf{y}|\mathbf{v})$ because we have deterministic layers at lower-level. Learning is done using conditional stochastic back-propagation that aims at maximizing the conditional log-likelihood of the training data. Our pre-training algorithm is necessary for initializing parameters to sensible values.

V. Experiments

A. Datasets

We use 15 real-world datasets to show the effectiveness of our model across a broad range of domains with diversified characteristics. All the datasets are available at http://mulan.sourceforge.net. Statistics about these data are shown in Table 2. Dimensionality reduction is performed on *rcv1-subset1* to *rcv1-subset5* by selecting top 2% features with the highest document frequency [41]. For numerical features, we normalize them to have zero-mean and unit variance.

B. Compared Algorithms and Evaluation Metrics

We compare our algorithm with CRBM, neural network (NNet), ECC, LEAD and LIFT. As mentioned in Section 2, LEAD and LIFT are our two main competitors since both of them have been proven to outperform many previous state-of-the-art models, but for reference, we also report the results of ECC. For CRBM, NNet and NCEM, we implement them in MATLAB; for ECC, we use the implementation from MULAN, a Java library for multi-label learning [36]. For LEAD and LIFT, we use the code released by the authors at http://cse.seu.edu.cn/people/zhangml/Publication.htm. It does not contain the structure learning part in the package, which is required for LEAD, hence we use DAGLearn, a MATLAB package for structure learning in Bayesian network [28].

We use 4 standard metrics to evaluate all multi-label learning algorithms. Among those, there are three ranking-based measures, namely *ranking loss, coverage, average precision*, and one label-based measure, *micro-f1*. Detailed definitions of these metrics can be found in [45].

C. Implementation Details

For ECC, the ensemble size is set to 10 [23]. The parameters for LEAD and LIFT are chosen as reported in the original papers. In the implementation of CRBM and NCEM, we use mini-batch stochastic gradient ascent with 100 total training epochs. NNets are first pre-trained with stack of RBMs and then fine-tuned using conjugate gradient descent with 3 line searches with cross-entropy error as the objective function [11]. A fixed learning rate and weight-decay are used in our experiments, selected from the set $\{0.05, 0.005, 0.0005\}$ for CRBM and NCEM. We also use momentum during training [10]. The size of each hidden layer is selected from the choices in $\{100, 200, 300, ..., 800\}$. For NCEM, we try 2 or 3 hidden layers (including top-level stochastic layer), and find out that in general using 3 layers gives roughly the same performance as using 2 layers; hence we only report the result of using two layers throughout the experiments. The best combination of learning rate, weight-decay, and model architecture is selected via cross-validation. The same architecture tried in NCEM is also tried for NNet. Finally, the number of Gibbs sampling performed to get the negative phase statistics is set to 1 and gradually increase to 5 during learning, as described in [26]. Our implementation is based on "DeepLearnToolBox" which is a MATLAB package [21].

D. Results

We now provide detailed results for algorithms compared across 15 datasets and 4 evaluation metrics. Ten-fold cross-validation is performed on each experimental dataset. The results are shown from Table 3 through Table 6, where the mean and standard deviation of each algorithm is reported. We **bold** the algorithm that performs the best for a given metric and a given dataset. Note that for *ranking loss, coverage*, lower values are preferred, and for *average precision* and *micro-f1*, higher values are desired.

Maybe surprisingly, CRBM already outperforms ECC in almost every metrics on all datasets, and is also highly competitive comparing to LEAD and LIFT. We note that using CRBM for MLC is not new. However, to our best knowledge this paper is the first one that conducts extensive experiments on a wide variety of MLC tasks. The results show that CRBM as a general probabilistic model can perform pretty well on MLC. Also, in terms of time complexity and coding complexity, CRBM is efficient to train using contrastive divergence and is also simple to implement. The core training algorithm takes merely about sixty lines of (unoptimized) MATLAB code in our implementation with no dependencies on external libraries, and stochastic gradient method allows us to scale up to large datasets.

Clearly, NCEM outperforms CRBM significantly in most cases, showing that adding one or many extra hidden layers helps in terms of capturing the complex conditional distribution. NNet, on the other hand, does not perform as well as our model does.

TABLE II: Statistics of 15 datasets used in this work. N is the number of instances, D is the feature dimension, M is the number of labels, $cardinality$ denotes the average number of labels per example, and $nominal$ indicates whether the feature is nominal or numerical.

dataset	N	D	M	$cardinality$	$nominal?$	$domain$
scene	2407	294	6	1.074	no	image
emotions	593	72	6	1.869	no	music
yeast	2417	103	14	4.237	no	biology
medical	978	1449	45	1.245	yes	text
enron	1702	1001	53	3.378	yes	text
rcv-subset1	6000	944	101	2.880	yes	text
rcv-subset2	6000	944	101	2.634	yes	text
rcv-subset3	6000	944	101	2.614	yes	text
rcv-subset4	6000	944	101	2.484	yes	text
rcv-subset5	6000	944	101	2.642	yes	text
bibtex	7395	1836	159	2.402	yes	text
corel5k	5000	499	374	3.522	yes	image
corel16k-sample1	13766	500	153	2.859	yes	image
corel16k-sample2	13761	500	164	2.882	yes	image
corel16k-sample3	13760	500	154	2.829	yes	image

TABLE III: *Ranking loss* of each algorithm (mean±std), where **bold** number indicates the best performance. Avg Rank indicates the average ranking across all datasets.

	Algorithms					
dataset	ECC	LEAD	LIFT	CRBM	NNet	NCEM
scene	0.093±0.015	0.088±0.011	0.062±0.010	0.074±0.009	0.068±0.010	**0.059±0.007**
emotions	0.172±0.045	0.156±0.024	**0.142±0.020**	0.167±0.028	0.159±0.015	0.142±0.021
yeast	0.227±0.016	0.173±0.011	0.166±0.013	0.168±0.011	0.193±0.017	**0.164±0.009**
medical	0.048±0.019	0.022±0.012	0.027±0.013	0.022±0.010	0.024±0.010	**0.021±0.008**
enron	0.160±0.008	0.076±0.006	0.070±0.005	0.069±0.008	0.073±0.011	**0.065±0.009**
rcv1subset1	0.124±0.009	0.064±0.003	0.047±0.003	0.040±0.004	0.068±0.006	**0.037±0.003**
rcv1subset2	0.130±0.005	0.063±0.004	0.050±0.003	0.042±0.004	0.076±0.006	**0.040±0.002**
rcv1subset3	0.129±0.004	0.065±0.003	0.046±0.002	0.042±0.003	0.075±0.004	**0.040±0.003**
rcv1subset4	0.106±0.003	0.043±0.003	0.034±0.002	0.032±0.002	0.060±0.006	**0.031±0.001**
rcv1subset5	0.122±0.008	0.058±0.002	0.045±0.002	0.040±0.002	0.071±0.006	**0.037±0.001**
bibtex	0.209±0.005	0.085±0.006	0.067±0.002	0.059±0.004	0.091±0.007	**0.052±0.003**
corel5k	0.413±0.012	0.127±0.008	0.121±0.007	0.114±0.006	0.123±0.005	**0.099±0.005**
corel16k001	0.547±0.014	0.152±0.006	0.135±0.003	0.139±0.004	0.154±0.010	**0.130±0.005**
corel16k002	0.552±0.008	0.147±0.004	0.129±0.003	0.134±0.002	0.144±0.013	**0.122±0.003**
corel16k003	0.552±0.011	0.149±0.004	0.142±0.008	0.134±0.004	0.144±0.008	**0.123±0.003**
corel16k003	0.552±0.011	0.149±0.004	0.142±0.008	0.134±0.004	0.144±0.008	**0.123±0.003**
Avg Rank	6	4.1333	2.7333	2.6000	4.4667	**1.0667**

To further justify the improvements our model achieves, we apply pair-wise *t-test* at 5% significance level to compare NCEM with other models. In Table 8, we show three numbers: out of 15 datasets, how many times that NCEM is significantly superior/tied/significantly inferior to the compared method for a given metric. For the four metrics, NCEM almost always significantly outperforms all compared methods. Only rarely does NCEM perform significantly worse than other models. This result confirms that the proposed model is superior across different datasets and different metrics in a statistical sense.

E. The Effect of Unsupervised Pre-Training

Next, we show whether unsupervised initialization of the parameters helps in terms of finding a better solution. Fig. 3 shows the performance of *ranking-loss* without and with unsupervised pre-training. We observe that pre-training helps almost surely. Many times this gap is significant, which justifies that it is important to initialize the parameters to sensible values before learning the model as a whole.

VI. Conclusion

MLC is attracting more attention in the research field because of its wide range of potential application usage. We

TABLE IV: *Coverage* of each algorithm (mean±std), where **bold** number indicates the best performance. Avg Rank indicates the average ranking across all datasets.

	Algorithms					
dataset	ECC	LEAD	LIFT	CRBM	NNet	NCEM
scene	0.093±0.008	0.088±0.008	0.066±0.009	0.076±0.007	0.071±0.008	**0.063±0.006**
emotions	0.313±0.048	0.292±0.025	0.282±0.022	0.299±0.025	0.296±0.018	**0.279±0.021**
yeast	0.511±0.013	0.461±0.014	0.455±0.014	0.454±0.013	0.477±0.021	**0.443±0.015**
medical	0.074±0.012	**0.031±0.015**	0.041±0.016	0.036±0.014	0.038±0.013	0.033±0.012
enron	0.396±0.017	0.215±0.013	0.213±0.015	0.205±0.024	0.215±0.028	**0.196±0.022**
rcv1subset1	0.262±0.021	0.149±0.007	0.119±0.008	0.099±0.007	0.154±0.010	**0.093±0.005**
rcv1subset2	0.249±0.019	0.140±0.007	0.122±0.008	0.100±0.009	0.160±0.009	**0.096±0.005**
rcv1subset3	0.254±0.013	0.144±0.006	0.110±0.005	0.100±0.007	0.156±0.007	**0.094±0.006**
rcv1subset4	0.198±0.010	0.098±0.006	0.082±0.003	0.077±0.004	0.124±0.011	**0.073±0.004**
rcv1subset5	0.244±0.009	0.132±0.004	0.110±0.004	0.098±0.004	0.153±0.010	**0.090±0.003**
bibtex	0.355±0.020	0.157±0.010	0.119±0.004	0.107±0.007	0.165±0.010	**0.097±0.005**
corel5k	0.733±0.009	0.297±0.017	0.287±0.015	0.262±0.014	0.271±0.011	**0.225±0.012**
corel16k001	0.551±0.012	0.297±0.011	0.265±0.005	0.275±0.008	0.294±0.018	**0.259±0.008**
corel16k002	0.556±0.007	0.289±0.008	0.257±0.005	0.265±0.004	0.276±0.021	**0.242±0.005**
corel16k003	0.557±0.012	0.292±0.006	0.294±0.024	0.267±0.007	0.276±0.014	**0.241±0.006**
Avg Rank	6	4.0667	3.0667	2.5333	4.2667	**1.0667**

TABLE V: *Average precision* of each algorithm (mean±std), where **bold** number indicates the best performance. Avg Rank indicates the average ranking across all datasets.

	Algorithms					
dataset	ECC	LEAD	LIFT	CRBM	NNet	NCEM
scene	0.861±0.022	0.845±0.020	0.888±0.018	0.864±0.015	0.875±0.015	**0.891±0.013**
emotions	0.782±0.024	0.808±0.027	0.821±0.027	0.799±0.032	0.806±0.026	**0.821±0.029**
yeast	0.715±0.028	0.755±0.017	0.767±0.018	0.763±0.017	0.740±0.021	**0.769±0.013**
medical	0.874±0.022	**0.889±0.027**	0.876±0.029	0.885±0.026	0.879±0.022	0.886±0.018
enron	0.623±0.028	0.672±0.018	0.707±0.012	0.707±0.019	0.703±0.022	**0.722±0.020**
rcv1subset1	0.558±0.009	0.544±0.012	0.601±0.010	0.618±0.014	0.581±0.010	**0.628±0.016**
rcv1subset2	0.590±0.009	0.576±0.013	0.614±0.010	**0.634±0.009**	0.602±0.010	0.628±0.009
rcv1subset3	0.587±0.005	0.573±0.015	0.588±0.006	**0.631±0.012**	0.597±0.014	0.620±0.012
rcv1subset4	0.660±0.013	0.662±0.016	0.684±0.017	**0.698±0.014**	0.681±0.015	0.685±0.008
rcv1subset5	0.597±0.011	0.586±0.016	0.611±0.005	**0.637±0.014**	0.608±0.018	0.635±0.012
bibtex	0.518±0.022	0.542±0.012	0.551±0.010	0.576±0.014	0.557±0.015	**0.593±0.008**
corel5k	0.235±0.016	0.277±0.014	0.290±0.011	0.292±0.012	0.295±0.015	**0.317±0.011**
corel16k001	0.277±0.011	0.336±0.009	0.329±0.007	0.341±0.008	0.309±0.019	**0.346±0.010**
corel16k002	0.274±0.006	0.331±0.005	0.326±0.008	0.335±0.004	0.307±0.023	**0.346±0.004**
corel16k003	0.270±0.007	0.331±0.007	0.329±0.004	0.337±0.008	0.312±0.012	**0.348±0.008**
Avg Rank	5.6667	4.4667	3.2667	2.3333	3.9333	**1.3333**

show that a type of undirected model, called the conditional restricted Boltzmann machines, can achieve good performance, even though they are considered to be generic probabilistic models, and not sp1ecifically designed for MLC. We also introduce the Neural Conditional Energy Model, a type of hybrid network, and conduct extensive experiments to justify that NCEM performs significantly better than current state-of-the-art.

There are two future extensions of this work. First, it is needless to say that NCEMs are not restricted to making discrete predictions such as MLC only. We can generalize it to produce arbitrary output distribution such as Gaussian if we are interested in multiple regression problems, for example, to model expressions of human faces. It will be interesting to see how our probabilistic model can capture continuous outputs, where a lot of complex dependencies among them need to be learned. Second, we use a simple optimization algorithm that involves Gibbs sampling and back-propagation

TABLE VI: *Micro-f1* of each algorithm (mean±std), where **bold** number indicates the best performance. Avg Rank indicates the average ranking across all datasets.

	Algorithms					
dataset	ECC	LEAD	LIFT	CRBM	NNet	NCEM
scene	0.727±0.020	0.676±0.021	0.765±0.023	0.722±0.017	0.767±0.023	**0.787±0.022**
emotions	0.687±0.027	0.629±0.050	0.684±0.027	0.684±0.034	0.670±0.025	**0.711±0.035**
yeast	0.640±0.018	0.629±0.014	0.653±0.013	0.663±0.011	0.630±0.015	**0.675±0.009**
medical	**0.831±0.022**	0.794±0.025	0.765±0.023	0.793±0.029	0.775±0.028	0.795±0.020
enron	0.546±0.033	0.485±0.024	0.572±0.013	0.607±0.022	0.597±0.021	**0.625±0.018**
rcv1subset1	0.433±0.020	0.312±0.018	0.342±0.017	**0.518±0.013**	0.440±0.017	0.518±0.008
rcv1subset2	0.441±0.017	0.347±0.017	0.354±0.012	**0.499±0.010**	0.447±0.015	0.490±0.006
rcv1subset3	0.451±0.008	0.349±0.014	0.271±0.014	**0.501±0.011**	0.445±0.015	0.491±0.010
rcv1subset4	0.517±0.015	0.433±0.021	0.406±0.026	**0.541±0.012**	0.514±0.014	0.528±0.009
rcv1subset5	0.459±0.013	0.356±0.014	0.364±0.020	0.511±0.011	0.461±0.015	**0.514±0.010**
bibtex	0.448±0.015	0.382±0.011	0.319±0.019	0.457±0.016	0.453±0.014	**0.476±0.011**
corel5k	0.207±0.017	0.047±0.008	0.076±0.010	0.266±0.010	0.263±0.014	**0.287±0.010**
corel16k001	0.217±0.008	0.111±0.006	0.030±0.006	0.282±0.008	0.258±0.014	**0.286±0.007**
corel16k002	0.220±0.005	0.111±0.005	0.025±0.008	0.285±0.004	0.257±0.020	**0.292±0.006**
corel16k003	0.220±0.006	0.108±0.007	0.045±0.013	0.286±0.008	0.262±0.014	**0.291±0.008**
Avg Rank	3.6000	5.4000	5.0000	22.2000	3.4667	**1.3333**

TABLE VII: Results of pair-wise *t-test* applied to NCEM compared with other methods. Each cell contains three numbers: (from left to right) the counts that NCEM is significantly better/tied/significantly worse out of 15 datasets. Here the significance level of *t-test* is set to 5%.

Algorithm	*Ranking loss*	*Coverage*	*Average precision*	*Micro-f1*
ECC	14 / 1 / 0	14 / 1 / 0	14 / 1 / 0	12 / 2 / 1
LEAD	13 / 2 / 0	14 / 1 / 0	13 / 2 / 0	14 / 1 / 0
LIFT	10 / 5 / 0	10 / 5 / 0	9 / 6 / 0	15 / 0 / 0
CRBM	14 / 1 / 0	13 / 2 / 0	8 / 4 / 3	9 / 3 / 3
NNet	15 / 0 / 0	15 / 0 / 0	12 / 3 / 0	14 / 1 / 0

to train our model in this paper; it will be possible to consider more advanced methods like [39], or to use other specific loss function that favors different metric in different applications we care about.

VII. Acknowledgement

This work was sponsored by AOARD grant number No. FA2386-13-1-4045, Ministry of Science and Technology, National Taiwan University and Intel Corporation under Grants MOST102-2911-I-002-001 and NTU103R7501, and grant 102-2923-E-002-007-MY2, 102-2221-E-002-170, 103-2221-E-002-104-MY2, and Chunghwa Telecom grant 103-S-C3-C30.

References

[1] Y. Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[2] C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown. Learning multi-label scene classification, 2004.

[4] M.A. Carreira-Perpinan and G.E. Hinton. On contrastive divergence learning, 2005.

[5] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Mach. Learn.*, 76(2-3):211–225, September 2009.

[6] K. Dembczyski, W. Cheng, and E. Hllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, 2010.

[7] P. Duygulu, K. Barnard, J.F.G. Freitas, and D.A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV-Part IV*, 2002.

[8] Y. Guo and S. Gu. Multi-label classification using conditional dependency networks. In *IJCAI*, pages 1300–1305, 2011.

[9] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002.

[10] G.E. Hinton. A practical guide to training restricted boltzmann machines. Technical report, 2010.

[11] G.E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

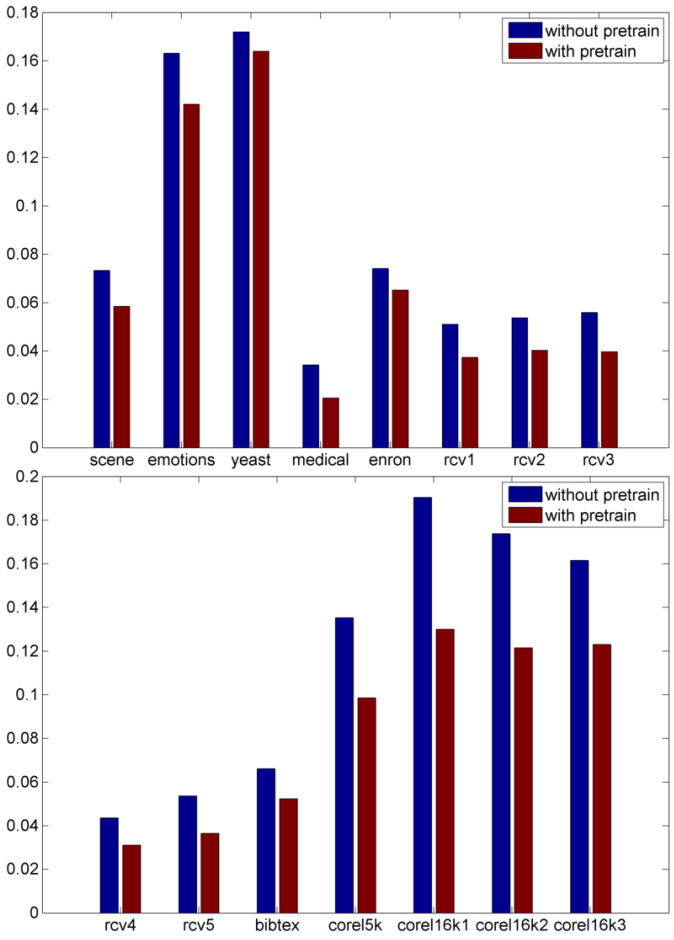
[12] G.E. Hinton, S. Osindero, M. Welling, and Y.-W. Teh. Unsupervised discovery of non-linear structure using contrastive backpropagation, 2004.

[13] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *ECML/PKDD-08 Workshop on Discovery Challenge*, 2008.

[14] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *ICML*, pages 536–543, New York, NY, USA, 2008. ACM.

[15] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio. Learning

Fig. 2: Ranking Loss with and without unsupervised pre-training



algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, 13(1):643–669, March 2012.

[16] Z.-H. Zhou M.-L. Zhang. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18:1338–1351, 2006.

[17] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Deroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 2012.

[18] M. Mandel, R. Pascanu, H. Larochelle, and Y. Bengio. Autotagging music with conditional restricted boltzmann machines. *CoRR*, abs/1103.2832, 2011.

[19] V. Mnih, H. Larochelle, and G.E. Hinton. Conditional restricted boltzmann machines for structured output prediction. *CoRR*, abs/1202.3748, 2012.

[20] Y. n. Chen and H. t. Lin. Feature-aware label space dimension reduction for multi-label classification. In *Advances in Neural Information Processing Systems 25*, pages 1529–1537. 2012.

[21] R. B. Palm. Prediction as a candidate for learning deep hierarchical models of data, 2012.

[22] J. Peng, L. Bo, and J. Xu. Conditional Neural Fields. In *Advances in Neural Information Processing Systems*, December 2009.

[23] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 254–269, 2009.

[24] R. Salakhutdinov and G.E. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

[25] R. Salakhutdinov, A. Mnih, and G.E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, New York, NY, USA, 2007. ACM.

[26] Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of Deep Belief Networks. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 872–879, 2008.

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

[28] M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using l1-regularization paths. In *AAAI - Volume 2*, pages 1278–1283. AAAI Press, 2007.

[29] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *NIPS*, pages 2231–2239. 2012.

[30] N. Srivastava, R. Salakhutdinov, and G.E. Hinton. Modeling documents with deep boltzmann machines. *CoRR*, abs/1309.6865, 2013.

[31] G.W. Taylor and G.E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *ICML*, pages 1025–1032, New York, NY, USA, 2009. ACM.

[32] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *ICML*, pages 1064–1071, 2008.

[33] K. Trohidis, G. Tsoumakas, G. Kalliris, and I.P. Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, 2008.

[34] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

[35] G. Tsoumakas, I. Katakis, and I. Vlahavas. A review of Multi-Label classification methods. In *Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD 2006)*, 2006.

[36] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[37] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *ECML*, pages 406–417, Berlin, Heidelberg, 2007. Springer-Verlag.

[38] Celine V., Jan S., Leander S., Sašo D., and Hendrik B. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, pages 185–214, 2008.

[39] D. Vickrey, C.-Y. Lin, and D. Koller. Non-local contrastive objectives. In *ICML*, pages 1103–1110, 2010.

[40] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11, December 2010.

[41] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.

[42] M.-L. Zhang. Lift: Multi-label learning with label-specific features. In *IJCAI*, 2011.

[43] M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *KDD*, pages 999–1008. ACM, 2010.

[44] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7), July 2007.

[45] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 99, 2013.