Published in final edited form as:
Proc SIAM Int Conf Data Min. 2015 ; 2015: 712–720. doi:10.1137/1.9781611974010.80.

A Generalized Mixture Framework for Multi-label Classification

Charmgil Hong[*], **Iyad Batal**[†], and **Milos Hauskrecht**[*]
[*]Department of Computer Science, University of Pittsburgh
[†]GE Global Research

Abstract

We develop a novel probabilistic ensemble framework for multi-label classification that is based on the *mixtures-of-experts* architecture. In this framework, we combine multi-label classification models in the *classifier chains family* that decompose the class posterior distribution $P(Y_1, \ldots, Y_d|\mathbf{X})$ using a product of posterior distributions over components of the output space. Our approach captures different input–output and output–output relations that tend to change across data. As a result, we can recover a rich set of dependency relations among inputs and outputs that a single multi-label classification model cannot capture due to its modeling simplifications. We develop and present algorithms for learning the mixtures-of-experts models from data and for performing multi-label predictions on unseen data instances. Experiments on multiple benchmark datasets demonstrate that our approach achieves highly competitive results and outperforms the existing state-of-the-art multi-label classification methods.

Keywords

Multi-label classification; Mixtures-of-experts

1 Introduction

Multi-Label Classification (MLC) refers to a classification problem where the data instances are associated with multiple class variables that reflect different views, functions or components describing the data. MLC naturally arises in many real-world problems, such as *text categorization* [19, 36] where a document can be associated with multiple topics reflecting its content; *semantic image/video tagging* [5, 24] where each image/video can have multiple tags based on its subjects; and *genomics* where an individual gene may have multiple functions [6, 36]. MLC formulates such situations by assuming each data instance is associated with d class variables. Formally speaking, the problem is specified by learning a function $h : \mathbb{R}^m \to \mathbf{Y} = \{0, 1\}^d$ that maps each data instance, represented by a feature vector $\mathbf{x} = (x_1, \ldots, x_m)$, to class assignments, represented by a vector of d binary values $\mathbf{y} = (y_1, \ldots, y_d)$ indicate the absence or presence of the corresponding classes.

The problem of learning multi-label classifiers from data has been studied extensively by the machine learning community in recent years. A key challenge in solving the problem is how to efficiently model and learn the dependencies among class variables given the fact that the space of possible dependency relations is exponentially large. Early methods assumed that all class variables ($Y_1, \ldots, Y_d$) are conditionally independent of each other and learned d independent functions to predict each class [6, 5]. However, this ignores the conditional dependencies among class variables which often contain crucial modeling information. To overcome this limitation, more advanced machine learning methods that model class relations have been proposed, such as conditional tree-structured Bayesian networks [2], classifier chains [27, 7], multi-dimensional Bayesian network classifiers [30, 4, 1] and output coding methods [16, 29, 37].

However, the methods of learning multi-label classifiers are still rather limited especially when the relations among features and class variables become more complex. For example, in *semantic image tagging*, an object can be tagged as {*cat*, *pet*} or {*cat*, *wild animal*} according to its context; similarly, in *medical informatics*, patients who are suffering from the same disease may receive different sets of medications due to their medical history or allergic reactions. As in the examples, if the relations tend to change across a dataset, existing methods may fail to respond with correct classification since they are designed to capture only one kind of dependency structure from data. One approach to address this issue is to employ various ensemble methods that combine multiple MLC classifiers to obtain an improved model. Unfortunately, ensemble methods that were adopted to the MLC settings [27, 7, 1] are rather limited in that: (1) they rely on simple averaging of multiple MLC models, (2) the MLC models averaged were not specifically optimized but restricted to randomized MLC structures (by choosing a random permutation for ordering the classes in the chain). As a result, the improvements we could obtain from such ensembles were often not very significant.

In this paper, we propose a new ensemble approach that aims to remedy the limitations of the MLC models by employing the *mixtures-of-experts* (ME) framework [18, 35]. Our ensemble approach incorporates the MLC models that belong to the *classifier chains family* (CCF) [27, 7, 2]. Briefly, the CCF models define the multivariate class posterior probability $P(Y_1, \ldots, Y_d|\mathbf{X})$ where the dependencies among class variables for different inputs are modeled by a collection of univariate probabilistic classifiers, one classifier for each output, that are organized in a chain, where a specific output variable is conditioned on all input variables and on output variables that precede it in the chain. The univariate classifiers in CCF can be implemented in many different ways, for example, as logistic regression models.

One limitation of the MLC models in CCF is that when they are learned from data, the dependencies among class variables are typically approximated by a specific classification model used (e.g. logistic regression) and hence may not be perfect. Moreover, in some applications, the dependencies among output variables may vary depending on the input context. Our new ME architecture lets us remedy these limitations by learning and combining multiple MLC models, where each model covers a different region of the input space. The intuition is that while a single MLC model may represent well the relations for some part of the input space, it may not be sufficient to model the relations globally (full input space), and hence multiple models may be needed to assure a good and accurate coverage. We develop and present an EM algorithm for learning the ME model for multiple MLC models from data.

The rest of the paper is organized as follows. Section 2 formally defines the problem of MLC. Section 3 provides the fundamentals of ME and CCF, which are necessary to understand our approach. Section 4 describes our proposed MLC solution. Section 5 presents the experiment results and evaluations. Lastly, section 6 concludes the paper.

2 Problem Definition[1]

Multi-Label Classification (MLC) is a classification problem in which each data instance is associated with a subset of labels from a labelset L. Denoting $d = |L|$, we define d binary class variables $Y_1, \ldots, Y_d$, whose value indicates whether the corresponding label in L is associated with an instance $\mathbf{x}$. We are given labeled training data $D = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^{N}$, where $\mathbf{x}^{(n)} = (x_1^{(n)}, \ldots, x_m^{(n)})$ is the m-dimensional feature variable of the n-th instance (the input) and $\mathbf{y}^{(n)} = (y_1^{(n)}, \ldots, y_d^{(n)})$ is its d-dimensional class variable (the output). We want to learn a function h that fits D and assigns to each instance a class vector ($h : \mathbb{R}^m \to \{0, 1\}^d$).

One approach to this task is to model and learn the *conditional joint distribution* $P(\mathbf{Y}|\mathbf{X})$ from D. Assuming the 0–1 loss function, the optimal classifier h^* assigns to each instance $\mathbf{x}$ the *maximum a posteriori* (MAP) assignment of class variables:

$$h^*(\mathbf{x}) = \arg\max_{y_1, \ldots, y_d} P(Y_1 = y_1, \ldots, Y_d = y_d | \mathbf{X} = \mathbf{x}) \quad (2.1)$$

The key challenge in modeling, learning and MAP inferences is that the number of configurations defining $P(\mathbf{Y}|\mathbf{X})$ is exponential in d. Overcoming this bottleneck is critical for obtaining efficient MLC solutions.

3 Preliminary

The MLC solution we propose in this work combines multiple MLC classifiers using the *mixtures-of-experts* (ME) [18] architecture. While in general the ME architecture may combine many different types of probabilistic MLC models, this work focuses on the models that belong to the *classifier chains family* (CCF). In the following we briefly review the basics of ME and CCF.

The ME architecture is a mixture model that consists of a set of *experts* combined by a *gating function* (or *gate*). The model represents the conditional distribution $P(y|\mathbf{x})$ by the following decomposition:

[1]**Notation:** For notational convenience, we will omit the index superscript $^{(n)}$ when it is not necessary. We may also abbreviate the expressions by omitting variable names; e.g., $P(Y_1=y_1, \ldots, Y_d=y_d|\mathbf{X}=\mathbf{x}) = P(y_1, \ldots, y_d|\mathbf{x})$.

$$\begin{aligned}P(y|\mathbf{x}) &= \sum_{k=1}^{K} P(E_k|\mathbf{x})P(y|\mathbf{x},E_k),\\ &= \sum_{k=1}^{K} g_k(\mathbf{x})P(y|\mathbf{x},E_k),\end{aligned} \quad (3.2)$$

where $P(y|\mathbf{x}, E_k)$ is the output distribution defined by the k-th expert E_k; and $P(E_k|\mathbf{x})$ is the context-sensitive prior of the k-th expert, which is implemented by the gating function $g_k(\mathbf{x})$. Generally speaking, depending on the choice of the expert model, ME can be used for either regression or classification [35].

Note that the gating function in ME defines a soft-partitioning of the input space, on which the K experts represent different input-output relations. The ability to switch among the experts in different input regions allows to compensate for the limitation of individual experts and improve the overall model accuracy. As a result, ME is especially useful when individual expert models are good in representing local input-output relations but may fail to accurately capture the relations on the complete input space.

ME has been successfully adopted in a wide range of applications, including handwriting recognition [9], text classification [11] and bioinformatics [25]. In addition, ME has been used in time series analysis, such as speech recognition [23], financial forecasting [33] and dynamic control systems [17, 32]. Recently, ME was used in social network analysis, in which various social behavior patterns are modeled through a mixture [12].

In this work, we apply the ME architecture to solve the MLC problem. In particular, we explore how to combine ME with the MLC models that belong to the classifier chains family (CCF). The CCF models decompose the multivariate class posterior distribution $P(\mathbf{Y}|\mathbf{X})$ using a product of the posteriors over individual class variables as:

$$P(\boldsymbol{Y}|\mathbf{X},\mathbf{M}) = \prod_{i=1}^{d} P(Y_i|\mathbf{X},\boldsymbol{Y}_{\boldsymbol{\pi}(i,M)}), \quad (3.3)$$

where $\mathbf{Y}_{\pi(i,M)}$ denotes the parent classes of class variable Y_i defined by model M. An important advantage of the CCF models over other MLC approaches is that they give us a well-defined model of posterior class probabilities. That is, the models let us calculate $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$ for any $(\mathbf{x}, \mathbf{y})$ input-output pair. This is extremely useful not only for prediction, but also for decision making [26, 3], conditional outlier analysis [13, 14], or performing any inference over subsets of output class variables. In contrast, the majority of existing MLC methods aim to only identify the best output configuration for the given $\mathbf{x}$.

The original *classifier chains* (CC) model was introduced by Read et al. [27]. Due to the efficiency and effectiveness of the model, CC has quickly gained large popularity in the multi-label learning community. Briefly, it defines the class posterior distribution $P(\mathbf{Y}|\mathbf{X})$ using a collection of classifiers that are tied together in a chain structure. To capture the dependency relations among features and class variables, CC allows each class variable to have only classes that precede it along the chain as parents ($\mathbf{Y}_{\pi(i,M)}$ in (3.3)). Figure 1(a) shows an example CC, whose chain order is $Y_3 \to Y_2 \to Y_1 \to Y_4$. Hence, the example defines the conditional joint distribution of class assignment (y_1, y_2, y_3, y_4) given **x** as:

$$P(y_1, y_2, y_3, y_4|\mathbf{x}, M_{Fig.1(a)}) = P(y_3|\mathbf{x}) \cdot P(y_2|\mathbf{x}, y_3) \cdot P(y_1|\mathbf{x}, y_3, y_2) \cdot P(y_4|\mathbf{x}, y_3, y_2, y_1)$$

Likewise, CCF is defined by a collection of classifiers, $P(Y_i|\mathbf{X}, \mathbf{Y}_{\pi(i,M)}) : i = 1, \dots, d$, one classifier for each output variable Y_i in the chain (3.3). Theoretically, the CCF decomposition lets us accurately represent the complete conditional distribution $P(\mathbf{Y}|\mathbf{X})$ using a fully connected graph structure of **Y** (see Figure 1(a)). However, this property does not hold in practice [7]. First, the choice of the univariate classifier model in CC (such as logistic regression), or other structural restrictions placed on the model, limit the types of multivariate output relations one can accurately represent. Second, the model is learned from data, and the data we have available for learning may be limited, which in turn may influence the model quality in some parts of the input space. As a result, a specific CC model is best viewed as an approximation of $P(\mathbf{Y}|\mathbf{X})$. In such a case, a more accurate approximation of $P(\mathbf{Y}|\mathbf{X})$ may be obtained by combining multiple CCs, each optimized for a different input subspace.

Conditional tree-structured Bayesian networks (CTBN) [2] is another model in CCF. The model is defined by an additional structural restriction: the number of parents is set to at most one (using the notation in (3.3), $\mathbf{Y}_{\pi(i,M)} := Y_{\pi(i,M)}$) and the dependency relations among classes form a tree:

$$P(\mathbf{y}|\mathbf{x}, M) = \prod_{i=1}^{d} P(y_i|\mathbf{x}, y_{\pi(i,M)}),$$

where $y_{\pi(i,M)}$ denotes the parent class of class Y_i in M. Figure 1(b) shows an example CTBN that defines:

$$P(\mathbf{y}|\mathbf{x}, M_{Fig.1(b)}) = P(y_3|\mathbf{x}) \cdot P(y_2|\mathbf{x}, y_3) \cdot P(y_1|\mathbf{x}, y_2) \cdot P(y_4|\mathbf{x}, y_2)$$

The advantage of the tree-structured restriction is that the model allows efficient structure learning and exact MAP inference [2].

The *binary relevance* (BR) [6, 5] model is a special case of CC that assumes all class variables are conditionally independent of each other ($\mathbf{Y}_{\pi(i,M)} = \{\} : i = 1, \dots, d$)[2]. Figure 1(c) illustrates BR when $d = 4$.

Finally, we would like to note that besides building simple ensembles for MLC in the literature [27, 7, 1], the mixture approach for a restricted chain model was studied recently by Hong et al. [15], which uses CTBNs [2] and extends the mixtures-of-trees framework

[2]By convention, $\mathbf{Y}_{\pi(i,M)} = \{\}$ if Y_i in M does not have a parent class.

[22, 31] for multi-label prediction tasks. In this work, we further generalize the approach using ME and CCF.

4 Proposed Solution

In this section, we develop a *Multi-Label Mixtures-of-Experts* (ML-ME) framework, that combines multiple MLC models that belong to *classifier chains family* (CCF). Our key motivation is to exploit the divide and conquer principle: a large, more complex problem can be decomposed and effectively solved using simpler sub-problems. That is, we want to accurately model the relations among inputs **X** and outputs **Y** by learning multiple CCF models better fitted to the different parts of the inout space and hence improve their predictive ability over the complete space. In section 4.1, we describe the mixture defined by the ML-ME framework. In section 4.2–4.4, we present the algorithms for its learning from data and for prediction of its outputs.

4.1 Representation

By following the definition of ME (3.2), ML-ME defines the multivariate posterior distribution of class vector **y** = $(y_1, \ldots, y_d)$ by employing K CCF models described in the previous section.

$$P(\mathbf{y}|\mathbf{x})=\sum_{k=1}^{K} g_k(\mathbf{x}) P(\mathbf{y}|\mathbf{x}, M_k) \quad (4.4)$$

$$=\sum_{k=1}^{K} g_k(\mathbf{x}) \prod_{i=1}^{d} P(y_i|\mathbf{x}, \mathbf{y}_{\boldsymbol{\pi}(i,M_k)}), \quad (4.5)$$

where $P(\mathbf{y}|\mathbf{x}, M_k)=\prod_{i=1}^{d} P(y_i|\mathbf{x}, \mathbf{y}_{\boldsymbol{\pi}(i,M_k)})$ is the joint conditional distribution defined by the k-th CCF model M_k and $g_k(\mathbf{x}) = P(M_k|\mathbf{x})$ is the gate reflecting how much M_k should contribute towards predicting classes for input **x**. We model the gate using the Softmax function, also known as normalized exponential:

$$g_k(\mathbf{x})=\frac{\exp(\boldsymbol{\theta}_{G_k}\mathbf{x})}{\sum_{k'=1}^{K}\exp(\boldsymbol{\theta}_{G_{k'}}\mathbf{x})}, \quad (4.6)$$

where $\boldsymbol{\Theta}_G=\{\boldsymbol{\theta}_{G_k}\}_{k=1}^{K}$ is the set of Softmax parameters. Figure 2 illustrates an example ML-ME model, which consists of K CCFs whose outputs are probabilistically combined by the gating function.

Algorithm 1

learn-mixture-parameters

Input: Training data D; base CCF experts $M_1, \ldots, M_K$

```
Output: Model parameters {$\Theta_G$, $\Theta_T$}
1:  repeat
2:    E-step:
3:    for $k$ = 1 to $K$, $n$ = 1 to $N$ do
4:      Compute $h_k^{(n)}$ using Equation (4.9)
5:    end for
6:    M-step:
7:    $\Theta_G = \arg\max_{\Theta_G} f_G(D; \Theta_G) - R(\Theta_G)$
8:    for $k$ = 1 to $K$ do
9:      $\boldsymbol{\theta}_{M_k} = \arg\max \sum_{n=1}^{N} h_k^{(n)} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M_k)$
10:   end for
11: until convergence
```

Parameters: Let $\Theta = \{\Theta_G, \Theta_M\}$ denote the set of parameters for an ML-ME model, where $\boldsymbol{\Theta}_G = \{\boldsymbol{\theta}_{G_k}\}_{k=1}^{K}$ are the gate parameters and $\boldsymbol{\Theta}_M = \{\boldsymbol{\theta}_{M_k}\}_{k=1}^{K}$ are the parameters of the CCF models defining individual experts. We define a gate output for each expert by a linear combination of inputs, which requires $|\theta_{G_k}| = (m + 1) = O(m)$ parameters. On the other hand, we parameterize each CCF expert by learning a set of classifiers. This in turn requires $|\theta_{M_k}| = d(m + O(d) + 1) = O(dm + d^2)$ parameters.

In summary, the total number of parameters for our ML-ME model is $|\Theta_G|+|\Theta_M| = O(Kmd + Kd^2)$. Table 1 summarizes the parameters and notations.

4.2 Learning parameters of CCF

In this section, we describe how to learn the parameters of ML-ME when the structures of individual CCF models are known and fixed. We return to the structure learning problem in Section 4.3. Our objective here is to find the parameters $\Theta = \{\Theta_G, \Theta_M\}$ that optimize the log-likelihood of the training data:

$$l(D;\boldsymbol{\Theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{N} g_k(\mathbf{x}^{(n)}) P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M_k) \quad (4.7)$$

We refer to (4.7) as the *observed log-likelihood*. However, direct optimization of this function is very difficult because the summation inside the log results in a non-convex function. To avoid this, we instead optimize the *complete log-likelihood*, which is defined by associating each instance ($\mathbf{x}^{(n)}$, $\mathbf{y}^{(n)}$) with a hidden variable $z^{(n)} \in \{1, \ldots, K\}$ indicating to which expert it belongs:

$$\begin{aligned} l_c(D;\mathbf{\Theta}) &= \sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}, z^{(n)}|\mathbf{x}^{(n)}) \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \mathbb{1}[z^{(n)}=k] \log\left(g_k(\mathbf{x}^{(n)})P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M_k)\right), \end{aligned} \quad (4.8)$$

where $\mathbb{1}[z^{(n)} = k]$ is the indicator function that evaluates to one if the *n*-th instance belongs to the *k*-th expert and to zero otherwise. We use the EM framework that iteratively optimizes the *expected complete log-likelihood* ($E[l_c(D; \Theta)]$), which is always a lower bound of the observed log-likelihood [8]. In the following, we derive an EM algorithm for ML-ME.

Each EM iteration consists of E-step and M-step. In the *E-step*, we compute the expectation of the complete log-likelihood. This reduces to computing the expectation of the hidden variable $z^{(n)}$, which is equivalent to the posterior of the *k*-th expert given the observation and the current set of parameters.

$$\begin{aligned} E\left[\mathbb{1}[z^{(n)}=k]\right] &= P(z^{(n)}=k|\mathbf{y}^{(n)}, \mathbf{x}^{(n)}) \\ &= \frac{g_k(\mathbf{x}^{(n)})P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M_k)}{\sum_{k'=1}^{K} g_{k'}(\mathbf{x}^{(n)})P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M_{k'})} \end{aligned} \quad (4.9)$$

In the *M-step*, we learn the model parameters $\{\Theta_G, \Theta_M\}$ that maximize the expected complete log-likelihood. Let $h_k^{(n)}$ denote $E[\mathbb{1}[z^{(n)} = k]]$. Then we can rewrite the expectation of (4.8) using $h_k^{(n)}$ and by switching the order of summations:

$$\sum_{k=1}^{K}\sum_{n=1}^{N} h_k^{(n)} \log g_k(\mathbf{x}^{(n)}) + h_k^{(n)} \log P\left(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M_k\right)$$

As $h_k^{(n)}$ is fixed in the M-step, we can decompose this into two parts, which respectively involves the gate parameters Θ_G and the CCF model parameters Θ_M:

$$\begin{aligned} f_G(D;\mathbf{\Theta}_G) &= \sum_{k=1}^{K}\sum_{n=1}^{N} h_k^{(n)} \log g_k(\mathbf{x}^{(n)}) \\ f_M(D:\mathbf{\Theta}_M) &= \sum_{k=1}^{K}\sum_{n=1}^{N} h_k^{(n)} \log P\left(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, M_k\right) \end{aligned}$$

By taking advantage of this modular structure, we optimize $f_G(D; \Theta_G)$ and $f_M(D; \Theta_M)$ individually to learn Θ_G and Θ_M, respectively. We first optimize $f_G(D; \Theta_G)$, which we rewrite as (using (4.6)):

$$f_G(D;\mathbf{\Theta}_G) = \sum_{k=1}^{K}\sum_{n=1}^{N} h_k^{(n)} \boldsymbol{\theta}_{G_k}\mathbf{x}^{(n)} - h_k^{(n)} \log \sum_{k'=1}^{K} \exp(\boldsymbol{\theta}_{G_{k'}}\mathbf{x}^{(i)})$$

Since $f_G(D; \Theta_G)$ is concave in Θ_G, we can find the optimal solution using a gradient-based method. The derivative of the log-likelihood with respect to θ_{G_j} is:

$$\nabla_{\theta_j} f_G(D;\mathbf{\Theta}_G)=\sum_{n=1}^{N}\left\{h_j^{(n)}-g_j(\mathbf{x}^{(n)})\right\}\mathbf{x}^{(n)} \quad (4.10)$$

Note that this equation has an intuitive interpretation as the derivative becomes zero when $g_j(\mathbf{x}^{(n)}) = P(M_k|\mathbf{x}^{(n)})$ and $h_j^{(n)}=P(M_k|\mathbf{y}^{(n)},\mathbf{x}^{(n)})$ are equal.

In our experiments, we solve this optimization using the L-BFGS algorithm [21], which is a quasi-Newton method that uses a sparse approximation to the inverse Hessian matrix to achieve a faster convergence rate even with a large number of variables. To prevent overfitting in high-dimensional space, we regularize with the L_2-norm of the parameters $R(\mathbf{\Theta}_G)=\frac{\lambda}{2}\sum_{k=1}^{K}\|\boldsymbol{\theta}_{G_k}\|_2^2$.

Now we optimize $f_M(D; \Theta_M)$, which can be further broken down into learning K individual CCF models. Note that f_M forms the weighted log-likelihood where $h_k^{(n)}$ serves as the instance weight. In our experiments, we optimize this by applying L_2-regularized instance-weighted logistic regression models.

4.2.1 Complexity—Algorithm 1 summarizes our parameter learning algorithm. The E-step computes $h_k^{(n)}$ for each instance on each expert. This requires $O(md)$ multiplications. Hence, the complexity of a single E-step is $O(KNmd)$. The M-step optimizes the parameters Θ_G and Θ_M. Optimizing Θ_G computes the derivative (4.10) which requires $O(mN)$ multiplications. Denoting the number of L-BFGS steps by l, this requires $O(mNl)$ operations. Optimizing Θ_M learns K CCF models. We do this by learning $O(Kd)$ instance-weight logistic regression models.

4.3 Structure Learning

We previously described the parameter learning of ML-ME by assuming we have fixed the individual structures. In this section, we present how to obtain useful structures for learning a mixture from data. We first show how to obtain CCF structures from weighted data. Then, we present our sequential boosting-like heuristic that, on each iteration, learns a structure by focusing on "hard" instances that previous mixture tends to misclassify.

4.3.1 Learning a Single CCF Structure on Weighted Data—To learn the structure that best approximates weighted data, we find the structure that maximizes the weighted conditional log-likelihood (WCLL) on $\{D, \Omega\}$, where $\Omega=\{\omega^{(n)}\}_{n=1}^{N}$ is the instance weight. Note that we further split D into training data D_{tr} and hold-out data D_h for internal validation.

Given a CCF structure M, we train its parameters using D_{tr}, which corresponds to learning instance-weighted logistic regression using D_{tr} and their weights. On the other hand, we use WCLL of D_h to define the score that measures the quality of M.

$$score(M)=\sum_{n\in D_h}\omega^{(n)}\sum_{i=1}^{d}\log P(y_i^{(n)}|\mathbf{x}^{(n)},\mathbf{y}_{\boldsymbol{\pi}(i,M)}^{(n)}) \quad (4.11)$$

The original CC [27] generates the underlying dependency structure (chain order) by a random permutation. In theory, this would not affect the model accuracy as CC still considers the complete relations among class variables. However, in practice, using a randomly generated structure may degrade the model performance due to the modeling and algorithmic simplifications (see section 3). In order to alleviate the issue, Read et al. [27] suggested to use ensembles of CC (ECC) that averages the predictions of multiple randomly ordered CCs trained on random subsets of the data. However, this is not a viable option because simply averaging the multidimensional output predictions may result in inconsistent estimates (does not correctly solve (2.1)).

Instead, we use a structure learning algorithm that learns a chain order greedily by maximizing WCLL. That is, starting from an empty ordered set ρ, we iteratively add a class index j to ρ by optimizing:

$$score_j(\rho)=\sum_{n\in D_h}\omega^{(n)}\log P(y_j^{(n)}|\mathbf{x}^{(n)},\mathbf{y}_{i\in\rho}^{(n)}), \quad (4.12)$$

where $\mathbf{y}_{i\in\rho}^{(n)}$ denotes the classes previously selected in ρ. We formalize our method in Algorithm 2. Note that this algorithm can be seen as a special case of [20] that optimizes the chain order using the beam search.

We would like to note that by incorporating additional restriction on the CC model, the optimal (restricted) CC structure may be efficiently computable. An example of such a model is the Conditional Tree-structured Bayesian Network (CTBN) [2]. Briefly, the optimal CTBN structure may be found using the maximum branch (weighted maximum spanning tree) [10] out of a weighted complete digraph, whose vertices represent class variables and the edges between them represent pairwise dependencies between classes.

4.3.2 Learning Multiple CCF Structures—To obtain multiple, effective CCF structures for ML-ME, we apply the above described algorithms multiple times with different sets of instance weights. This section explains how we assign the weights such that poorly predicted instances have higher weights; and well-predicted instances have lower weights.

Algorithm 2

learn-chain-structure

```
Input: Training data $D$
Output: Chain order $\rho$
1:  Split $D$ into $D_{tr}$ and $D_h$
2:  Initialize an ordered set $\rho = \{\}$
3:  for $i = 1$ to $d$ and $j \notin \rho$ do
```

```
4:      for $j = 1$ to $d$ do
5:          $\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta}_j} \omega^{(n)} \log P(y_j^{(n)} | \mathbf{x}^{(n)}, \mathbf{y}_\rho^{(n)}) : n \in D_{tr}$
6:      end for
7:      $\rho = \rho \cup \arg\max_j \omega^{(n)} \log P(y_j^{(n)} | \mathbf{x}^{(n)}, \mathbf{y}_\rho^{(n)}; \boldsymbol{\theta}_j) : n \in D_h$
8:  end for
```

To start with, we assign all instances uniform weights ($\omega^{(n)} = 1/N : n = 1, \ldots, N$; i.e., all instances are equally important a priori). Using this initial set of weights, we first obtain a CCF structure ρ_1 (i.e., either a CC or CTBN structure) and train a model M_1 that follows ρ_1. Then, by setting the current mixture $\mathcal{M}$ to be M_1, we compute the new instance weights to be the normalized prediction error:

$$\omega^{(n)} \propto 1 - P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, \mathcal{M}), \;\; s.t. \; \sum_{n=1}^{N} \omega^{(n)} = 1$$

With the updated weights $\{\omega^{(n)}\}$, we obtain another structure ρ_2, and train $\mathcal{M}$ with M_1 and M_2 that follow ρ_1 and ρ_2, respectively (Algorithm 1).

We incrementally inject new models to the mixture by repeating this process. To stop the process, we use internal validation approach. Specifically, the data used for learning are split to internal train and test sets. The structure of the trees and parameters are always learned on the internal train set. The quality of the current mixture is evaluated on the internal test set. The mixture growth stops when the log-likelihood on the internal test set for the new mixture does not improve any more. The structures included in the previous mixture are then fixed, and the parameters of the mixture are re-learned on the full training data.

4.3.3 Complexity—Learning a single CCF structure requires to estimate $P(Y_i|\mathbf{X}, Y_j)$ for $O(d^2)$ pairs of classes. Since we learn K CCF structures for a mixture, the overall complexity is $O(Kd^2)$ times the complexity of learning logistic regression.

4.4 Prediction

In order to make a prediction for a new instance $\mathbf{x}$, we want to find the MAP assignment of the class variables (see (2.1)). Our ML-ME model consists of multiple CCF models and the MAP solution may, at the end, require enumeration of exponentially many class assignments. To address this problem, we rely on approximate MAP inference. The two commonly applied MAP approximation approaches in the literature are: convex programming relaxation via dual decomposition [28], and simulated annealing using a Markov chain [34]. In this work, we use the latter approach. Briefly, we search the space of all assignments by defining a Markov chain that is induced by local changes to individual class labels. The annealed version of the exploration procedure [34] is then used to speed up the search. We initialize our MAP algorithm using the following heuristic: first, we identify the MAP assignments for each CCF model in the mixture individually [7, 2, 5]. After that, we pick the best assignment among these candidates. We have found this (efficient) heuristic to work very well and often results in the true MAP assignment.

5 Experiments

5.1 Data

We use seven publicly available MLC datasets obtained from different domains. Table 2 summarizes the characteristics of the datasets, including dataset size, label cardinality (the average number of labels per instance), distinct label set (the number of distinct class configurations that appear in the data) and data domain.

5.2 Methods

To demonstrate the benefits of our mixture framework, we compare the performance of the following eight methods: *binary relevance (BR)* [6, 5], *conditional tree-structured Bayesian networks (CTBN)* [2], *classifier chains (CC) and their ensembles (ECC)* [27], *probabilistic classifier chains (PCC) and their ensembles (EPCC)* [7], *ML-ME with CTBN (MCTBN)* and *ML-ME with CC (MCC)*.

BR is the simplest method that learns each class independently. CTBN and CC are our base method that fall in the classifier chains family. By testing them individually, we want to demonstrate the benefits of our method. PCC is an algorithmic extension of CC that exhaustively searches over its entire label space to perform exact MAP inference. ECC and EPCC are simple ensemble methods that rely on randomization to obtain multiple dependency relations (by choosing a random permutation for the class order in the chain) and use simple averaging to make ensemble predictions. MCTBN and MCC are our proposed methods that properly optimize the log-likelihood and produce context-sensitive mixture outputs.

For a fair comparison of the methods, we fix the following parameters throughout all experiments:

- We use L_2-penalized logistic regression for all of the methods and choose their regularization parameters by cross validation.
- We set the maximum number of experts to 10 for MCTBN/MCC. We use our heuristic (section 4.3.2) to stop early if possible; ECC/EPCC use 10 fixed number of base models in an ensemble.
- We use our structure learning algorithm (Algorithm 2) for CC/PCC; we use random chain orders for ECC/EPCC.
- For predictions on MCTBN/MCC, we use 150 iterations of simulated annealing.

5.3 Evaluation Metrics

To compare different MLC methods, we use the following two evaluation metrics.

- *Exact match accuracy* (EMA): EMA computes the percentage of instances whose predicted output vectors are exactly the same as their true class vectors (i.e., all classes are predicted correctly). EMA is proper for MLC as it evaluates the success of the method in finding the mode of $P(\mathbf{X}|\mathbf{Y})$. However, it could be too harsh especially when the output dimensionality is high.

- *Conditional log-likelihood loss* (CLL-loss): CLL-loss computes the negative conditional log-likelihood of the test instances.

$$CLL-loss=\sum_{n=1}^{N}-\log\left(P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})\right) \quad (5.13)$$

It measures the model fitness by evaluating how much probability mass is given to the true label vectors (the higher the probability, the smaller the loss). Note that CLL-loss is only defined for probabilistic methods.

5.4 Results

Tables 3 and 4 show the performance of all methods in terms of EMA and CLL-loss, respectively. All results are obtained using *ten-fold cross validation*. In parentheses, we indicate the relative ranking of the methods on each dataset. We do not report the results of PCC/EPCC on Medical and Enron because evaluating all $O(2^d)$ class assignments is clearly infeasible. Also, we do not report CLL-loss for ECC and EPCC because they do not produce probabilistic output.

Based on the results, our ML-ME framework clearly improves the performance of the base models. In terms of EMA (Table 3), the prediction accuracy of MCC is not only the highest but also the most stable. Although not as good as MCC, MCTBN also shows a large improvement compared with CTBN. These demonstrate that ML-ME compensates for the restrictions that the base MLC models have using their combinations. In addition, this is in contrast to simple averaging, which often leads to inconsistent estimation (ECC and EPCC). The model fitness of MCC measured by CLL-loss (Table 4) also indicates that MCC is competitive, followed by MCTBN, CTBN, BR and CC. Although PCC is recording the highest average ranking, it is computationally very expensive and does not scale up to large data.

In summary, the experimental results show that our ML-ME method with the CCF experts is able to outperform or match the existing state-of-the-art methods across a broad range of benchmark MLC datasets. We attribute this improvement to the ability of the CCF mixture that simultaneously compensates for the restricted dependencies modeled by an individual CCF, and to its ability that better fits the different regions of the input space with new expert models.

6 Conclusion

We presented a novel probabilistic ensemble framework for multi-label classification. Our approach attempts to capture different input-output and output-output relations that tend to change across data. We integrated the mixtures-of-experts architecture and the multi-label classification models in the classifier chains family, that decompose the class posterior distribution $P(Y_1, \ldots, Y_d|\mathbf{X})$ using a product of posterior distributions over components of the output space. We developed the learning and prediction algorithms for our mixture framework, and showed that our approach recovers a rich set of dependency relations among inputs and outputs that a single multi-label classification model cannot capture due to its modeling simplifications. Through the experiments on multiple benchmark datasets, we demonstrated that our approach achieves highly competitive results and outperforms the existing state-of-the-art multi-label classification methods.

Acknowledgments

This work was supported by grants R01LM010019 and R01GM088224 from the NIH.

References

1. Antonucci A, Corani G, Mauá DD, Gabaglio S. An ensemble of bayesian networks for multilabel classification. IJCAI. 2013:1220–1225.
2. Batal, I.; Hong, C.; Hauskrecht, M. An efficient probabilistic framework for multi-dimensional classification. Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM ‘13; ACM; 2013. p. 2417-2422.
3. Berger, J. Springer series in statistics. 2. Springer; New York, NY: 1985. Statistical decision theory and Bayesian analysis.
4. Bielza C, Li G, Larrañaga P. Multi-dimensional classification with bayesian networks. Int’l Journal of Approximate Reasoning. 2011; 52(6):705–727.
5. Boutell MR, Luo J, Shen X, Brown CM. Learning multi-label scene classification. Pattern Recognition. 2004; 37(9):1757–1771.
6. Clare, A.; King, RD. Lecture Notes in Computer Science. Springer; 2001. Knowledge discovery in multi-label phenotype data; p. 42-53.
7. Dembczynski, K.; Cheng, W.; Hüllermeier, E. Bayes optimal multilabel classification via probabilistic classifier chains. Proceedings of the 27th International Conference on Machine Learning (ICML-10); Omnipress; 2010. p. 279-286.
8. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B. 1977; 39:1–38.
9. Ebrahimpour R, Moradian MR, Esmkhani A, Jafarlou FM. Recognition of persian handwritten digits using characterization loci and mixture of experts. JDCTA. 2009; 3(3):42–46.
10. Edmonds J. Optimum branchings. Research of the National Bureau of Standards. 1967; 71B:233–240.
11. Estabrooks, A.; Japkowicz, N. A mixture-of-experts framework for text classification. Proceedings of the 2001 Workshop on Computational Natural Language Learning; Stroudsburg, PA, USA. Association for Computational Linguistics; 2001. p. 9:1-9:8.
12. Gormley, IC.; Murphy, TB. Mixture of Experts Modelling with Social Science Applications. John Wiley & Sons, Ltd; 2011. p. 101-121.
13. Hauskrecht M, Batal I, Valko M, Visweswaran S, Cooper GF, Clermont G. Outlier detection for patient monitoring and alerting. Journal of Biomedical Informatics. Feb; 2013 46(1):47–55. [PubMed: 22944172]
14. Hauskrecht, M.; Valko, M.; Batal, I.; Clermont, G.; Visweswaram, S.; Cooper, G. Conditional outlier detection for clinical alerting. Annual American Medical Informatics Association Symposium; 2010.
15. Hong, C.; Batal, I.; Hauskrecht, M. A mixtures-of-trees framework for multi-label classification. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management; ACM; 2014. p. 211-220.
16. Hsu D, Kakade S, Langford J, Zhang T. Multi-label prediction via compressed sensing. NIPS. 2009:772–780.

17. Jacobs RA, Jordan MI. Learning piecewise control strategies in a modular neural network architecture. IEEE Transactions on Systems, Man, and Cybernetics. 1993; 23(2):337–345.
18. Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE. Adaptive mixtures of local experts. Neural Comput. Mar; 1991 3(1):79–87.
19. Kazawa, H.; Izumitani, T.; Taira, H.; Maeda, E. Advances in Neural Information Processing Systems. Vol. 17. MIT Press; 2005. Maximal margin labeling for multi-topic text categorization; p. 649-656.
20. Kumar, A.; Vembu, S.; Menon, AK.; Elkan, C. Learning and inference in probabilistic classifier chains with beam search. Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases; Springer-Verlag; 2012.
21. Liu DC, Nocedal J. On the limited memory bfgs method for large scale optimization. Math Program. Dec; 1989 45(3):503–528.
22. Meilă M, Jordan MI. Learning with mixtures of trees. Journal of Machine Learning Research. 2000; 1:1–48.
23. Mossavat, SI.; Amft, O.; De Vries, B.; Petkov, P.; Kleijn, WB. A bayesian hierarchical mixture of experts approach to estimate speech quality. 2010 2nd International Workshop on Quality of Multimedia Experience; 2010. p. 200-205.
24. Qi, G-J.; Hua, X-S.; Rui, Y.; Tang, J.; Mei, T.; Zhang, H-J. Correlative multi-label video annotation. Proceedings of the 15th international conference on Multimedia; ACM; 2007. p. 17-26.
25. Qi Y, Klein-Seetharaman J, Bar-Joseph Z. A mixture of feature experts approach for protein-protein interaction prediction. BMC bioinformatics. 2007; 8(Suppl 10):S6. [PubMed: 18269700]
26. Raiffia, H. Decision Analysis: Introductory Lectures on Choices Under Uncertainty. Mcgraw-Hill; Jan. 1997
27. Read, J.; Pfahringer, B.; Holmes, G.; Frank, E. Classifier chains for multi-label classification. Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD '09; Springer-Verlag; 2009.
28. Sontag, D. PhD thesis. Massachusetts Institute of Technology; 2010. Approximate Inference in Graphical Models using LP Relaxations.
29. Tai, F.; Lin, H-T. Multi-label classification with principle label space transformation. the 2nd International Workshop on Multi-Label Learning; 2010.
30. van der Gaag LC, de Waal PR. Multidimensional bayesian network classifiers. Probabilistic Graphical Models. 2006:107–114.
31. Šingliar, T.; Hauskrecht, M. Modeling highway traffic volumes. Proceedings of the 18th European Conference on Machine Learning, ECML '07; Springer-Verlag; 2007. p. 732-739.
32. Weigend AS, Mangeas M, Srivastava AN. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. International Journal of Neural Systems. 1995; 6:373–399. [PubMed: 8963468]
33. Weigend AS, Shi S. Predicting daily probability distributions of S&P500 returns. Journal of Forecasting. Jul.2000 19(4)
34. Yuan, C.; Lu, T-C.; Druzdzel, MJ. UAI. AUAI Press; 2004. Annealed map; p. 628-635.
35. Yuksel SE, Wilson JN, Gader PD. Twenty years of mixture of experts. IEEE Trans Neural Netw Learning Syst. 2012; 23(8):1177–1193.
36. Zhang ML, Zhou ZH. Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering. 2006; 18(10):1338–1351.
37. Zhang, Y.; Schneider, J. Maximum margin output coding. Proceedings of the 29th International Conference on Machine Learning; 2012. p. 1575-1582.

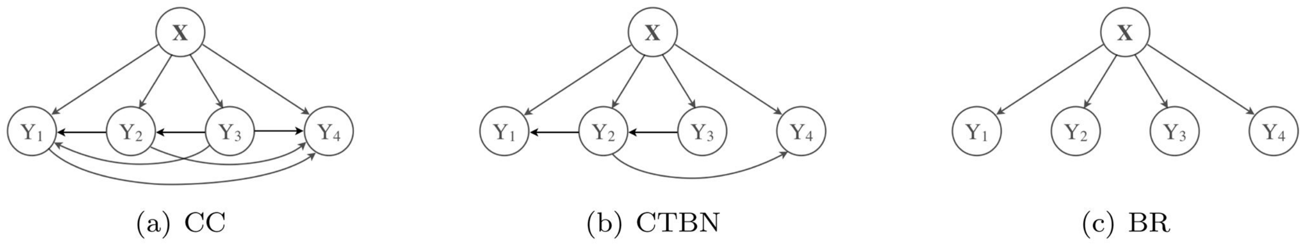

Figure 1.

Example models of the *classifier chains family*.

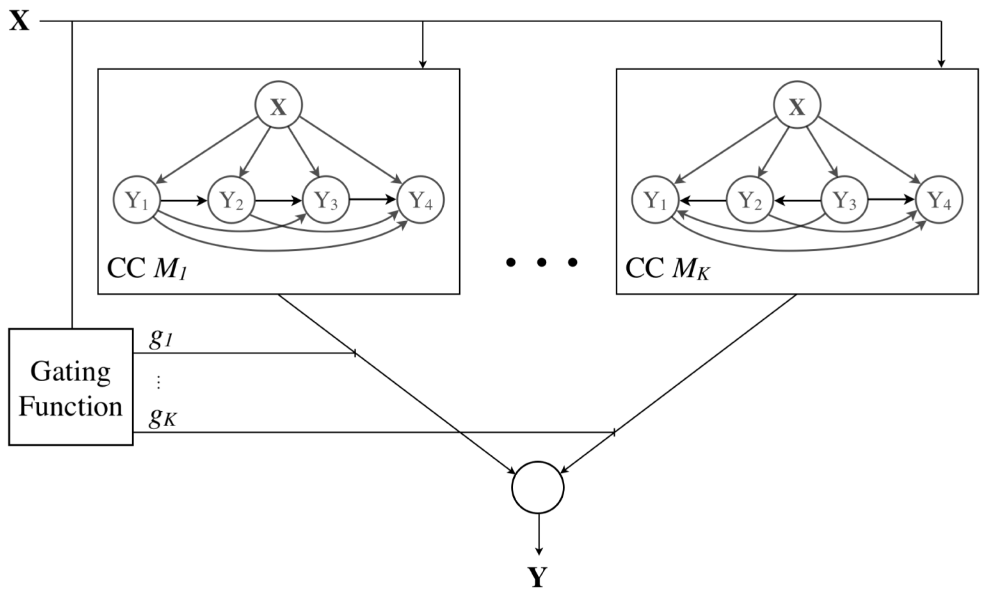

Figure 2.

An example of ML-ME.

Table 1

Notations

NOTATION	DESCRIPTION
m	Input (feature) dimensionality
d	Output (class) dimensionality
N	Number of data instances
K	Number of experts in a mixture
M_k	An MLC expert with index k
$\Theta_M = \{\theta_{M_1}, \ldots, \theta_{M_K}\}$	The parameters for MLC experts
$\Theta_G = \{\theta_{G_1}, \ldots, \theta_{G_K}\}$	The parameters for a gate

Table 2

Datasets characteristics

DATASET	*N*	*m*	*d*	LC	DLS	DM
Image	2,000	135	5	1.24	20	image
Scene	2,407	294	6	1.07	15	image
Emotions	593	72	6	1.87	27	music
Flags	194	19	7	3.39	54	image
Yeast	2,417	103	14	4.24	198	biology
Medical	978	1,449	45	1.25	94	text
Enron	1,702	1,001	53	3.38	753	text

[*] *N*: number of instances, *m*: number of features, *d*: number of classes, LC: label cardinality, DLS: distinct label set, DM: domain

[**] All data are taken from http://mulan.sourceforge.net and http://cse.seu.edu.cn/people/zhangml/Resources.htm

Table 3

Performance of each method on the benchmark datasets in terms of exact match accuracy.

EMA	BR	CTBN	CC	PCC	ECC	EPCC	MCTBN	MCC
Image	0.279±0.036 (8)	0.407±0.036 (6)	0.445±0.038 (2)	0.452±0.032 (2)	0.413±0.028 (6)	0.442±0.019 (2)	0.444±0.035 (2)	**0.486±0.038** (1)
Scene	0.542±0.028 (8)	0.624±0.035 (7)	0.694±0.023 (3)	**0.701±0.028** (1)	0.658±0.027 (5)	0.681±0.030 (3)	0.645±0.028 (5)	**0.708±0.028** (1)
Emotions	0.265±0.056 (8)	0.334±0.065 (4)	0.341±0.061 (4)	0.343±0.073 (4)	0.288±0.086 (4)	**0.344±0.072** (1)	**0.370±0.063** (1)	**0.356±0.062** (1)
Flags	0.139±0.042 (7)	0.155±0.069 (7)	**0.196±0.067** (1)	0.191±0.075 (6)	**0.212±0.089** (1)	**0.222±0.062** (1)	**0.216±0.063** (1)	**0.227±0.071** (1)
Yeast	0.151±0.024 (8)	0.195±0.026 (7)	0.220±0.027 (3)	0.242±0.023 (2)	0.204±0.024 (3)	0.219±0.015 (3)	0.218±0.025 (3)	**0.259±0.026** (1)
Medical	0.641±0.075 (6)	0.667±0.079 (4)	0.688±0.056 (4)	- (−)	**0.701±0.035** (1)	- (−)	**0.712±0.065** (1)	**0.711±0.055** (1)
Enron	0.173±0.024 (4)	0.184±0.016 (4)	**0.197±0.032** (1)	- (−)	0.181±0.030 (4)	- (−)	**0.196±0.023** (1)	**0.192±0.026** (1)
Avg.Rank	7.0	5.6	2.6	3.0	3.4	2.0	2.0	1.0

Numbers in parentheses show the relative ranking of the method on each dataset.

The best methods (by paired t-test at $\alpha = 0.05$) are shown in **bold**. The last row shows the average ranking of the methods.

Table 4

Performance of each method on the benchmark datasets in terms of conditional log-likelihood loss.

CLL-loss	BR	CTBN	CC	PCC	MCTBN	MCC
Image	432.6±20.8 (5)	391.0±22.2 (4)	475.9±34.9 (6)	**347.0±24.4** (1)	378.3±20.7 (3)	**342.6±30.7** (1)
Scene	343.6±22.5 (5)	287.2±16.4 (4)	371.8±32.3 (6)	**230.1±15.6** (1)	277.8±12.0 (3)	**234.2±18.4** (1)
Emotions	153.9±10.7 (5)	135.8±8.6 (4)	155.2±10.1 (5)	**130.1±10.1** (1)	134.2±9.9 (3)	**132.0±10.0** (1)
Flags	68.6±11.5 (2)	66.5±11.2 (2)	80.9±19.7 (6)	**57.6±11.4** (1)	66.6±12.1 (2)	66.3±9.1 (2)
Yeast	1502.4±45.1 (5)	1075.3±46.5 (3)	2233.4±126.4 (6)	932.1±72.6 (2)	1077.5±52.7 (3)	**915.7±38.6** (1)
Medical	155.9±25.2 (4)	145.4±23.8 (2)	152.7±35.6 (4)	- (−)	**133.3±34.8** (1)	140.6±36.6 (2)
Enron	1441.3±85.7 (5)	1316.4±78.6 (4)	1230.9±72.4 (3)	- (−)	**1127.4±63.8** (1)	1156.2±71.3 (2)
Avg.Rank	4.4	3.3	5.1	1.2	2.3	1.4

Numbers in parentheses show the relative ranking of the method on each dataset.

The best methods (by paired t-test at $\alpha = 0.05$) are shown in **bold**. The last row shows the average ranking of the methods.