

Hierarchical Multilabel Classification with Optimal Path Prediction

Zhengya Sun¹ · Yangyang Zhao¹ ·
Dong Cao¹ · Hongwei Hao¹

Published online: 28 April 2016
© Springer Science+Business Media New York 2016

Abstract We consider multilabel classification problems where the labels are arranged hierarchically in a tree or directed acyclic graph (DAG). In this context, it is of much interest to select a well-connected subset of nodes which best preserve the label dependencies according to the learned models. Top-down or bottom-up procedures for labelling the nodes in the hierarchy have recently been proposed, but they rely largely on pairwise interactions, thus susceptible to get stuck in local optima. In this paper, we remedy this problem by directly finding a small number of label paths that can cover the desired subgraph in a tree/DAG. To estimate the high-dimensional label vector, we adopt the advantages of partial least squares techniques which perform simultaneous projections of the feature and label space, while constructing sound linear models between them. We then show that the optimal label prediction problem with hierarchy constraints can be reasonably transformed into the optimal path prediction problem with the structured sparsity penalties. The introduction of path selection models further allows us to leverage the efficient network flow solvers with polynomial time complexity. The experimental results validate the promising performance of the proposed algorithm in comparison to the state-of-the-art algorithms on both tree- and DAG-structured data sets.

Keywords Hierarchical classification · Multilabel classification · Network flow optimization

1 Introduction

The multilabel classification problem has aroused considerable interest in recent years. In many real-world applications, the labels often exhibit a hierarchical structure, typically in the form of a tree, or more generally, an arbitrarily directed acyclic graph (DAG) [14, 20, 22]. For example, in text classification [19], a document may belong to more than one

✉ Zhengya Sun
zhengya.sun@ia.ac.cn

¹ Institute of Automation, Chinese Academy of Sciences, Beijing, China

category that resides in the topic hierarchy. In functional genomic [2], a gene usually relates to multiple functions with a DAG structured hierarchy. In image annotation [13], an image that belongs to some class automatically belongs to at least one of its superclasses. In effect, the hierarchical structure among the labels can be problematic to many multilabel classification algorithms. This is partly due to the highly skewed data distribution, and some labels (such as those at the lower levels of the hierarchy) tend to have very few positive instances. Besides, the inconsistent predictions between child and parent make little sense for the underlying taxonomy. Thus, how to improve the prediction performance by effectively exploiting the hierarchical information becomes an important research issue [19,21].

Recent advances on hierarchical multilabel classification (HMC) can be tracked in two directions: algorithm adaptation and problem transformation. Algorithm adaptation approaches extend some specific algorithms to guarantee the hierarchical consistency. Typical examples include decision trees [1,6,21] and support vector machines [19]. Problem transformation approaches, on the other hand, convert a HMC problem into one or more well-understood problems that can benefit from many off-the-shelf algorithms, which is also our concern. One straightforward way for reducing the HMC is to train a binary classifier at each node with the structural dependencies properly encoded [4,10,25]. An alternative way is to train, for each parent node, a multiclass classifier to discriminate between its child nodes [15–17]. In many cases, the number of labels can reach thousands or even more, making the learning process computationally infeasible. Some researchers proposed to train one multiclass classifier to each level of the hierarchy [7–9]. However, there is probably not sufficient data for training the classifiers near the bottom of the hierarchy. To remedy both deficiencies, some researchers further followed the kernel dependency estimation (KDE) framework, which projects the labels to a lower-dimension space, followed by learning a mapping from input space to each projected dimension [3]. However, they ignore all information about the feature parts during compression, rendering the selected components possibly less well suited to the subsequent learning tasks. In other words, many of the features may be irrelevant to the reduced tasks, especially when there exist a huge number of features with a sharp increase of the computational burden.

Given a test instance, these algorithms then estimate its multilabel from the resulting classifiers. They can be categorized according to the hierarchy depths at which their predictions end [5]. We consider a more general scenario, where the prediction paths might stop at an internal node. To make the output labels consistent with the hierarchical structure, several studies are devoted to a top-down or bottom-up procedures [8–11,25]. In this manner, a particular node receives a label that is conditioned on the parent or child nodes. Such schemes does ensure consistency in principle, but they do not exploit long-range interactions, thus suffering from error propagation between adjacent nodes in the hierarchy. Existing solutions that address this issue include [2] and [3]. The former allows all levels of classifiers to be influenced by one another, whereas it requires training the Bayesian network and also requires high computational expense. Instead, the latter can be interpreted as a label selection problem within the whole connected graph, which is combinatorially hard and approximately addressed with greedy algorithms. These results further leads to the question of whether there exists certain structure that is both rich enough to involve long-range interactions and computationally feasible.

In this paper, we propose a novel solution which directly finds a small number of paths that can cover the desired label subgraph in a tree/DAG. To estimate the high-dimensional label vector, we adopt the advantages of partial least squares (PLS) techniques which address compression and learning in a unified framework. Specifically, they perform simultaneous projections of the feature and label space, while minimizing both the encoding error and

training error. We then show that the optimal label prediction problem with hierarchy constraints can be reasonably transformed into the optimal path prediction (OPP) problem with the structured sparsity penalties. The resulting formulation can be regarded as path selection problems, where each path starts from a root and terminates on a leaf or an internal node. Apparently, the labels in each path serve as context for one another, going beyond pairwise interactions. In this view, predicting a hierarchical multilabel for an instance amounts to selecting the union of one or more paths from an exponential number of candidates. The introduction of path selection models allows us to leverage the efficient network flow solvers with polynomial time complexity. The experimental results validate the promising performance of the proposed algorithm in comparison to the state-of-the-art algorithms on both tree- and DAG-structured data sets.

The remainder of this paper is organized as follows. In the next section, we begin by presenting some notations used throughout the paper and giving a brief review on PLS. Then we elaborate our proposed algorithm in Sect. 3, and report the experimental results in Sect. 4. We finally give some concluding remarks in the last section.

2 Preliminaries

In this section, we provide some preliminaries including notations used throughout the paper and the linear PLS techniques.

2.1 Notations

Let $\mathcal{X} \subset \mathcal{R}^M$ be the instance space, where M is the feature dimension of each instance. Let $\mathcal{Y} = \{1, \dots, K\}$ be the label space with K different labels. A multilabel for an instance is any subset of \mathcal{Y} , including the empty set. For any instance $\mathbf{x} \in \mathcal{X}$, we denote the associated multilabel by a vector $\mathbf{y} = (y_1, \dots, y_K) \in \{0, 1\}^K$, where $y_i = 1$ indicates i belongs to the multilabel of \mathbf{x} , and $y_i = 0$ otherwise. Note that each \mathbf{y} can have more than one non-zero entries. Given a dataset \mathcal{D} that contains n training instances of the form (\mathbf{x}, \mathbf{y}) , let \mathbf{X} be a $n \times M$ matrix consisting of n instances with M features, and let \mathbf{Y} be a $n \times K$ matrix consisting of n instances with K labels.

2.2 Partial Least Squares

PLS is a class of methods for modeling the relationship between sets of observed variables by means of latent vectors (also called score vectors or components). It is widely applicable to the regression and classification as well as dimension reduction tasks. The derivations underlying PLS are closely related to those of principal component analysis (PCA). Unlike PCA, PLS constructs low-rank approximations for both X and Y , which plus the linear projections are utilized to compute the final predictive model.

Consider the general setting of a linear PLS algorithm to model the relationship between two data sets (blocks of observed variables). After observing n training instances from each block of variables, PLS decomposes the $(n \times M)$ zero-mean matrix \mathbf{X} and $(n \times K)$ zero-mean matrix \mathbf{Y} into the form [24]

$$\begin{aligned}\mathbf{X} &= \mathbf{TP}^T + \mathbf{E} \\ \mathbf{Y} &= \mathbf{UQ}^T + \mathbf{F}\end{aligned}\tag{1}$$

where \mathbf{T} and \mathbf{U} are $(n \times p)$ matrices of p extracted latent vectors. The $(M \times p)$ matrix \mathbf{P} and the $(K \times p)$ matrix \mathbf{Q} represent matrices of loadings, and the $(n \times M)$ matrix \mathbf{E} , the $(n \times K)$ matrix \mathbf{F} are the residual matrices. The solution for PLS is based on the nonlinear iterative partial least squares (NIPALS) algorithm [23], which at each iteration seeks normalized basis vectors \mathbf{w} and \mathbf{c} for projecting \mathbf{X} and \mathbf{Y} into one-dimension space. In this way the covariance between the score vectors \mathbf{t} and \mathbf{u} (rows of \mathbf{T} and \mathbf{U}) is maximized as:

$$\begin{aligned} cov(\mathbf{t}, \mathbf{u})^2 &= [cov(\mathbf{X}\mathbf{w}, \mathbf{Y}\mathbf{c})]^2 \\ &= \max_{|r|=|s|=1} [cov(\mathbf{X}\mathbf{r}, \mathbf{Y}\mathbf{s})]^2 \end{aligned} \tag{2}$$

An equivalent form of Eq. (2) is

$$\begin{aligned} J_{\mathbf{w}, \mathbf{c}} &= \arg \max_{\mathbf{w}, \mathbf{c}} \mathbf{w}^T \mathbf{X}^T \mathbf{Y} \mathbf{c} \\ s.t. \quad &\mathbf{w}^T \mathbf{w} = 1; \quad \mathbf{c}^T \mathbf{c} = 1 \end{aligned} \tag{3}$$

Apparently, the desired \mathbf{w} can be obtained by calculating the eigenvector with the largest eigenvalue in the following equation.

$$\mathbf{S}^T \mathbf{S} \mathbf{w} = \lambda \mathbf{w} \tag{4}$$

where $\mathbf{S} = \mathbf{X}^T \mathbf{Y}$. The other score or weight vectors are updated iteratively according to the NIPALS algorithm. In most cases, PLS is built upon two assumptions: i) the score vectors extracted from \mathbf{X} are good predictors of \mathbf{Y} , and ii) a linear inner relation between the scores vectors \mathbf{t} and \mathbf{u} exists; that is [18],

$$\mathbf{U} = \mathbf{T}\mathbf{D} + \mathbf{H} \tag{5}$$

where \mathbf{D} is a $(p \times p)$ diagonal matrix and \mathbf{H} indicates a residual matrix. In turn, the decomposition of \mathbf{Y} matrix in Eq. (1) can be rewritten as

$$\mathbf{Y} = \mathbf{T}\mathbf{D}\mathbf{Q}^T + (\mathbf{H}\mathbf{Q}^T + \mathbf{F}) \tag{6}$$

and this defines the linear PLS regression model

$$\mathbf{Y} = \mathbf{T}\mathbf{C}^T + \mathbf{F}^* \tag{7}$$

where $\mathbf{C}^T = \mathbf{D}\mathbf{Q}^T$ denotes the $(p \times K)$ matrix of regression coefficients and $\mathbf{F}^* = \mathbf{H}\mathbf{Q}^T + \mathbf{F}$ is the residual matrix. Equation (7) can also be expressed using the originally observed data \mathbf{X} and written as [18]

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{F}^* \tag{8}$$

where the estimate of \mathbf{B} has the following form

$$\mathbf{B} = \mathbf{X}^T \mathbf{U} (\mathbf{T}^T \mathbf{X} \mathbf{X}^T \mathbf{U})^{-1} \mathbf{T}^T \mathbf{Y} \tag{9}$$

For a test instance \mathbf{x}_t , the estimated multilabel is $\hat{\mathbf{y}}_t = \mathbf{x}_t \mathbf{B}$. In fact, PLS requires far less computational time than traditional multilabel classification algorithms. This can be attributed to the fact that PLS integrates dimension reduction and parameter estimation in a simple and unified manner, as shown above. While the counterparts often need to perform three isolated processes including projection, training and prediction, with the training process comparatively time consuming when the number of features is large.

Due to the high-dimensionality of many data sets, such as the Gene Ontology (thousands of labels, and tens of thousands of features sometimes), implementation simplicity becomes

a main reason that we adopt PLS as the learning model for multilabel prediction. Another reason lies in its joint linear projections in both the feature space and the label space, thus enjoying performance advantages because of the additional feature information.

3 Proposed Algorithm

In general, the label hierarchy can be modeled as a directed acyclic graph (DAG), in which each node can have multiple parents. Let $H = (V, E)$ denote the label hierarchy, where V is a node set indexed by $\{1, \dots, K\}$, and $E \subseteq V \times V$ is an edge set. Let \mathcal{H}_ρ denote the group structure containing all the paths in H , where each path ρ starts from a root and terminates on a leaf or an internal node. In the context of hierarchical classification, each multilabel actually takes the form of a connected subgraph that can be exactly covered by a small number of paths. As shown in the left side of Fig. 1, the i -th component of the ground truth multilabel \mathbf{y} can be formalized as

$$y_i = \min \left\{ 1, \sum_{j=1}^L \alpha_j G_{ij} \right\}, \tag{10}$$

where L denotes the number of paths involved in the label structure, $G_{ij} \in \{0, 1\}$ indicates whether the i -th label resides in the j -th path, and $\alpha_j \in \{0, 1\}$ indicates whether the j -th path is selected. In this regard, we are allowed to map the hierarchical multilabel prediction problem to the path selection problem. And the final prediction is simply the union of these resulting paths.

If the labels are flat, one can simply pick the labels with the largest entries in the estimated multilabel $\hat{\mathbf{y}}$, as shown in Fig. 1a. Here we refer to it as the pointwise selection. By combining multiple pairwise interactions, top-down or bottom-up procedures cater for the hierarchical structure, but they suffer from error propagation, as shown in Fig. 1b. The use of path descriptors, however, helps diminish the influences of mistakes made at adjacent levels, as shown in Fig. 1c. Note that a path corresponds to the co-occurrence of a set of nodes, and is able to model long-range interactions between labels. In the following section, we will

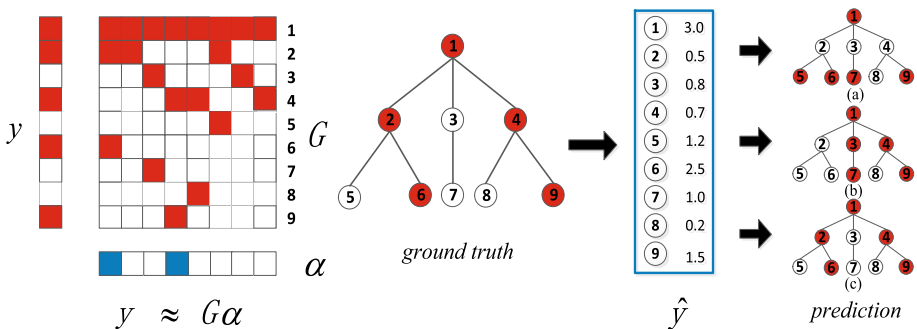


Fig. 1 An example with the ground truth multilabel $\mathbf{y} = \{1, 1, 0, 1, 0, 1, 0, 0, 1\}$ and the estimated multilabel $\hat{\mathbf{y}} = \{3.0, 0.5, 0.8, 0.7, 1.2, 2.5, 1.0, 0.2, 1.5\}$. Left the multilabel \mathbf{y} can be viewed as the union of the first and the fourth paths. Right comparison of different prediction results obtained by pointwise (a), pairwise (b) and pathwise (c) selection strategies

introduce the path-based objective function for consistent prediction, and demonstrate how to leverage the efficient network flow solvers to the OPP.

3.1 Path-Based Objective

It is of great interest to select a set of labels that preserve the priori label dependency and approximate the probability estimates as close as possible. Based on the logistic function, we obtain the probability estimate $P(\hat{y}_i = 1)$ according to the i -th entry of the multilabel $\hat{\mathbf{y}}$, that is,

$$P(\hat{y}_i = 1) = \frac{1}{1 + \exp(-\theta \hat{y}_i)}, \tag{11}$$

where \hat{y}_i is estimated by the PLS regressor, $\theta > 0$ is a tuning parameter and set to 1 for convenience. When no ambiguity arises, we abbreviate $P(\hat{y}_i = 1)$ as \hat{p}_i . One way to describe the hierarchical label prediction is as finding the parameters $\psi = [\psi_1, \dots, \psi_K]^T$ that solve the following optimization problem,

$$\begin{aligned} & \max_{\psi} \sum_{i=1}^K \hat{p}_i \psi_i, \\ & \text{s.t. } \psi_i \in \{0, 1\}, \quad \sum_{i=1}^K \psi_i \leq \gamma, \end{aligned} \tag{12}$$

$\text{supp}(\psi)$ satisfies the hierarchy constraints,

where $\text{supp}(\psi) = \{j \in \{1, \dots, K\} : \psi_j \neq 0\}$ is defined as the set of nonzero entries in a vector ψ , and $\gamma > 0$ is a tuning integer parameter.

Proposition 1 *Let ψ^* be the optimal solution of Eq. (12). Suppose $0 < \gamma \leq K$ and $\hat{p}_i > 0$, then for any given integer parameter γ , ψ^* must have γ nonzero entries, i.e., $|\text{supp}(\psi^*)| = \gamma$.*

Apparently, this requires solving the selection of isolated variables with the underlying dependency constraints. For interpretation purposes, we seek to handle the more general variables and directly select a few path groups with desired properties. Hence, the problem in (12) can be reformulated as follows,

$$\max_{\psi} \sum_{i=1}^K \hat{p}_i \psi_i, \quad \text{s.t. } \psi_i \in \{0, 1\}, \quad \text{supp}(\psi) = \bigcup_{h \in \mathcal{P}} h, \quad \sum_{h \in \mathcal{P}} \eta_h \leq \tau, \tag{13}$$

where \mathcal{P} is a subset of groups in \mathcal{H}_ρ whose union exactly covers the support of ψ , the term $\sum_{h \in \mathcal{P}} \eta_h$ is called the structured sparsity, $\tau > 0$ is a regularization parameter, and η_h is a non-negative weight associated with the path h . Equivalently, we have

$$\min_h \sum_{i \notin \bigcup_{h \in \mathcal{P}} h} \hat{p}_i, \quad \text{s.t. } \sum_{h \in \mathcal{P}} \eta_h \leq \tau, \tag{14}$$

Then, for any path $h = (v_1, v_2, \dots, v_k)$ in \mathcal{H}_ρ , we define the weight η_h as

$$\eta_h = \beta + \sum_{i=1}^{k-1} c_{v_i v_{i+1}} = \beta + \sum_{(u,v) \in h} c_{uv}, \tag{15}$$

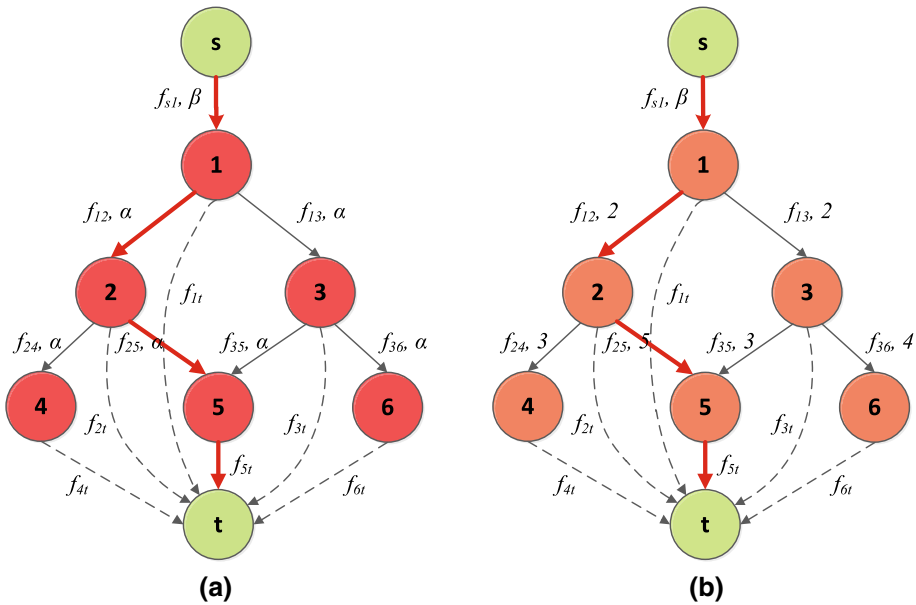


Fig. 2 Graph H' representation obtained by adding a source s and sink t to a DAG with six nodes and a path $(s, 1, 2, 5, t)$ in bold red. **a** All edges in H have cost α . For path $h = (1, 2, 5)$, the cost of a flow sending one unit from s to t along h is $\beta + 2\alpha$, which is exactly η_h . **b** Different edges in H have different costs. For path $h = (1, 2, 5)$, the cost of a flow sending one unit from s to t along h is $\beta + 7$, which is exactly η_h . (Color figure online)

where $\beta > 0$ is a tuning parameter, c_{uv} represents certain cost on edge $(u, v) \in E$. Note that designing c_{uv} may require some priori knowledge about the dependency importance or reliability. When no additional knowledge is available, a simple choice is to set $c_{uv} = \alpha$ for all edges. As can be seen, the parameters α and β provide a tradeoff between two factors including the number of selected paths and the length of each path. From the viewpoint of semantics, α encourages the abstraction of the solution whereas β encourages concreteness. When β is large and α is negligible, the term $\sum_{h \in \mathcal{P}} \eta_h$ simply “counts” how many paths are required to cover the support of ψ , thereby encouraging the choice of longer paths. On the contrary, the labels distributed across different paths are preferred.

In fact, the structured sparsity enriches the original constraints in (12) by introducing the structural information, while monotonically increases as more and more nodes are added. In this sense, the hierarchical label prediction problem can be reasonably translated into the OPP problem with each path appropriately weighted.

3.2 Optimal Path Selection

The optimization problem in (14) involves an exponential number of variables, one for every candidate path in the DAG. The explicit search for a few path groups is a combinatorial optimization problem. Instead, we exploit the network structure of flows to implicitly handle the exponential combinations, allowing the exact solution to be computed in polynomial time.

Let us augment the label hierarchy H by introducing a source node s and a sink node t . Formally, we introduce a new graph $H' = (V', E')$ with

$$V' = V \cup \{s, t\},$$

$$E' = E \cup \{(s, v) : v \in V \text{ is a root node}\} \cup \{(u, t) : u \in V\}$$

Obviously, the node s are linked to the root nodes in H , while t are linked to every node in H . The construction of the graph H' is illustrated in Fig. 2a, b for two cost configurations. Let f_{uv} be the flow on edge $(u, v) \in E'$, and \mathcal{F} be the set of integer flows on H' . The cost of a flow $f \in \mathcal{F}$ sending one unit from s to t along a path h in H is defined as

$$\sum_{(u,v) \in E'} f_{uv} c_{uv} = \sum_{(u,v) \in (s,h,t)} c_{uv}, \tag{16}$$

which is exactly η_h when we set $c_{sh} = \beta$ and $c_{ut} = 0$ for every node u in V , as can be seen in Fig. 2. Typically, a node may appear in more than one path. A useful expression is therefore the amount of flow going through a node j in $V = \{1, \dots, K\}$, which is defined as

$$s_j(f) = \sum_{u \in V': (u,j) \in E'} f_{uj}, \tag{17}$$

Let $\text{supp}(f) = \{j \in \{1, \dots, K\} : s_j(f) > 0\}$ denote the set of nodes with positive flow values. Then, the optimization problem (14) over the paths in H can be formulated as optimization problems over (s, t) -path flows in H' ,

$$\begin{aligned} \min_{f \in \mathcal{F}} \quad & \sum_{i \notin \text{supp}(f)} \hat{p}_i + \lambda \sum_{(u,v) \in E'} f_{uv} c_{uv} \\ \text{s.t.} \quad & s_i(f) > 0, \quad \forall i \in \text{supp}(f), \end{aligned} \tag{18}$$

where $\lambda > 0$ is a regularization parameter. Note that

$$\sum_{i \notin \text{supp}(f)} \hat{p}_i = \sum_{i=1}^K \max(\hat{p}_i(1 - s_i(f)), 0).$$

Thus, the above optimization problem (18) can be equivalently rewritten as

$$\min_{f \in \mathcal{F}} \left\{ \sum_{i=1}^K \max(\hat{p}_i(1 - s_i(f)), 0) + \lambda \sum_{(u,v) \in E'} f_{uv} c_{uv} \right\}. \tag{19}$$

The resulting minimum cost flow problem with piecewise linear costs in turn is equivalent to a classical minimum cost flow problem with integral constraints, and can be solved in polynomial time.

4 Experiments

In this section, we carry out experiments to investigate whether our algorithm performs better than the state-of-the-art algorithms on both tree- and DAG-structured data sets, and to evaluate the contributions of its components.

4.1 Datasets and Baselines

We use 12 yeast data sets downloaded from Clare [12], where each data set records a specific aspect of the genes in the yeast genome. There are two versions of each data set. The

Table 1 The yeast data sets used in the experiments

Data sets	# Training instances	# Validation instances	# Test instances	# Features	# Labels	
					FunCat	GO
seq	1692	876	1332	478	500	4134
pheno	653	352	581	69	456	3128
struc	1659	859	1306	19,629	500	4133
hom	1661	876	1309	47,035	500	4127
cellcycle	1625	848	1278	77	500	4126
church	1627	844	1278	27	500	4126
derisi	1605	842	1272	63	500	4120
eisen	1055	528	835	79	462	3758
gasch1	1631	846	1281	173	500	4126
gasch2	1635	849	1288	52	500	4132
spo	1597	837	1263	80	500	4120
expr	1636	849	1288	551	500	4132

input features are identical in both versions, but the labels are annotated according to different classification schemes. In the first version, the labels are organized as a category tree from MIPS's FunCat (<http://mips.gsf.de/projects/funcat>), which has 6 levels and assigns an average of 8.8 labels for each instance. In the second version, the labels are organized as a DAG from the Gene Ontology (GO) (<http://www.geneontology.org>), which has 14 levels and assigns an average of 35 labels for each instance.

As in [21], we use two-thirds of each data set for training and use the remaining one-third for testing. Out of the training set, two-thirds are used for running algorithms and one-third for validating parameters. Besides, the final model is trained on the complete training set. As in [3], we manage missing features by replacing the missing values with the mean of the non-missing values. Table 1 summarizes the information about the 12 data sets.

We compare our proposed method which is abbreviated as OPP with the following state-of-the-arts for multilabel classification on tree- and DAG-structured hierarchies.

- OPP: In the learning step, we use the PLS technique for feature-aware label space dimension reduction and multilabel estimation, in which the number of latent variables is chosen via cross-validation. In the prediction step, we leverage the classical minimum cost flow solvers to deal with the transformed path selection problem.
- CSSA/CSSAG [3]: In the learning step, this method extends the KDE framework by encoding the label dependencies using the kernel trick. After projecting the labels to a 50-dimensional space, it uses ridge regression to estimate the multilabel with the ridge parameter chosen via cross-validation. In the prediction step, this method uses greedy strategies to produce consistent predictions.
- CLUS-HMC [21]: This method improves the predictive clustering trees by the data distance metric that assigns larger weights to the labels higher up in the hierarchy. During data partition, the variance reduction is estimated according to the weighted Euclidean distances, and its significance level parameter is chosen via cross-validation.
- HMC-PC [1]: This method adopts an expectation-maximization scheme to cluster the training instances, which are then used to define the per-cluster average class vector.

The definition is based on the probabilities of cluster membership and certain threshold selection strategy.

- HMC-LMLP [8]: This method treats a neural network as a multi-label classifier, and only designed for tree-structured hierarchies. It incrementally trains a multi-layer perceptron for each classification hierarchical level. Predictions made by a neural network in a given level are used as inputs to the neural network responsible for the prediction in the next level.
- Rprop-NoComp [9]: This method adapts HMC-LMLP to the DAG-structured hierarchies. It trains an individual multi-layer perceptron for each classification hierarchical level without employing the class labels to augment the feature vectors. The testing phase is performed by feeding all instances into all multi-layer perceptrons at every level.

We adopt the precision-recall curves (PR curves) for analyzing the multilabel classification models. Precision and recall are originally developed for binary classification with positive and negative labels. In particular, the precision measures the percentage of positive predictions that are correct, and the recall measures the percentage of positive instances that are correctly predicted. They are defined as

$$Prec = \frac{TP}{TP + FP}, \quad Rec = \frac{TP}{TP + FN},$$

where TP is the number of true positives (correctly predicted positive instances), FP is the number of false positives (incorrectly predicted negative instances), and FN is the number of false negatives (incorrectly predicted positive instances). They are further adapted for the multilabel cases [21], i.e.,

$$Prec = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i}, \quad Rec = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i},$$

where $i \in \{1, \dots, K\}$ ranges over all label indices. Note that the overall PR curve can be obtained by varying the threshold values. For the proposed method, we vary the regularization parameter λ within a particular range. As in [21], we use the performance score called Area Under the average PR curve (AUPRC) for model evaluation. The closer the AUPRC approaches 1.0, the better the model should be.

4.2 Results

We first examine the individual contribution of learning and prediction components in OPP. For clarity, we express the methods to be compared in explicit combining forms, including PLS+OPP, PLS+CSSA/CSSAG, and KDE+CSSA/CSSAG, where KDE represents the processes of KPCA projection in the label space and ridge regression. PLS differs from KDE in that it performs simultaneous projections of the feature and label space, and minimizes both the encoding error and training error, while KDE focuses on the label space reduction without considering the features, and minimizes the training error in a subsequent step.

In Table 2, we lay out performance comparisons of the different algorithms for the FunCat data sets in terms of AUPRC values. We can make several observations. First, we observe that for most cases, the performance of CSSA algorithm is improved by implementing the PLS regressors, indicating that predicting hierarchical labels can benefit from the estimates learned from it. Second, compared to PLS+CSSA algorithm, the proposed PLS+OPP algorithm yields better results on all the data sets except one. The most impressive case is for the largest data set “hom”, in which the improvement is up to about 2.2%. Figure 3 shows the PR curves for the two algorithms. As can be seen, our algorithm exhibits a promising

Table 2 AUCPR values of the different algorithms for FunCat

Data sets	PLS+OPP	PLS+CSSA	KDE+CSSA
seq	0.271	0.262	0.241
pheno	0.191	0.197	0.195
struc	0.234	0.225	0.203
hom	0.295	0.273	0.256
celcycle	0.237	0.233	0.222
church	0.211	0.211	0.203
derisi	0.216	0.216	0.209
eisen	0.282	0.269	0.242
gasch1	0.274	0.263	0.239
gasch2	0.248	0.239	0.219
spo	0.223	0.221	0.213
expr	0.232	0.227	0.251
Average	0.243	0.235	0.224

Table 3 AUCPR values of the different algorithms for GO

Data sets	PLS+OPP	PLS+CSSAG	KDE+CSSAG
seq	0.425	0.405	0.374
pheno	0.376	0.361	0.358
struc	0.388	0.365	0.359
hom	0.417	0.389	0.379
celcycle	0.392	0.370	0.355
church	0.409	0.376	0.374
derisi	0.410	0.395	0.373
eisen	0.449	0.422	0.384
gasch1	0.419	0.395	0.366
gasch2	0.408	0.386	0.365
spo	0.396	0.381	0.364
expr	0.398	0.382	0.370
Average	0.407	0.386	0.368

performance especially in the low-recall regions. Last, compared to the two baselines, our method yields the highest AUPRC on average. The differences between PLS+OPP and PLS+CSSA/KDE+CSSA are also statistically significant with a p-value of less than 0.01 in *t*-test, where the significance level is often set to 0.05.

Table 3 compares the different algorithms for the GO data sets. We can draw similar conclusions as above. Specifically, PLS+CSSAG algorithm consistently outperforms KDE+CSSAG algorithm by 1.8% on average. Compared to PLS+CSSAG algorithm, the proposed PLS+OPP algorithm further shows apparently superior performance, rising about 2.1% on average. We also depict the PR curves for the two best algorithms in Fig. 4. One can see that our algorithm appears superior performance across the entire recall regions. In addition, the differences between PLS+OPP and PLS+CSSAG/KDE+CSSAG are also statistically significant with a p-value of less than 0.01 in *t*-test, where the significance level is often set to 0.05.

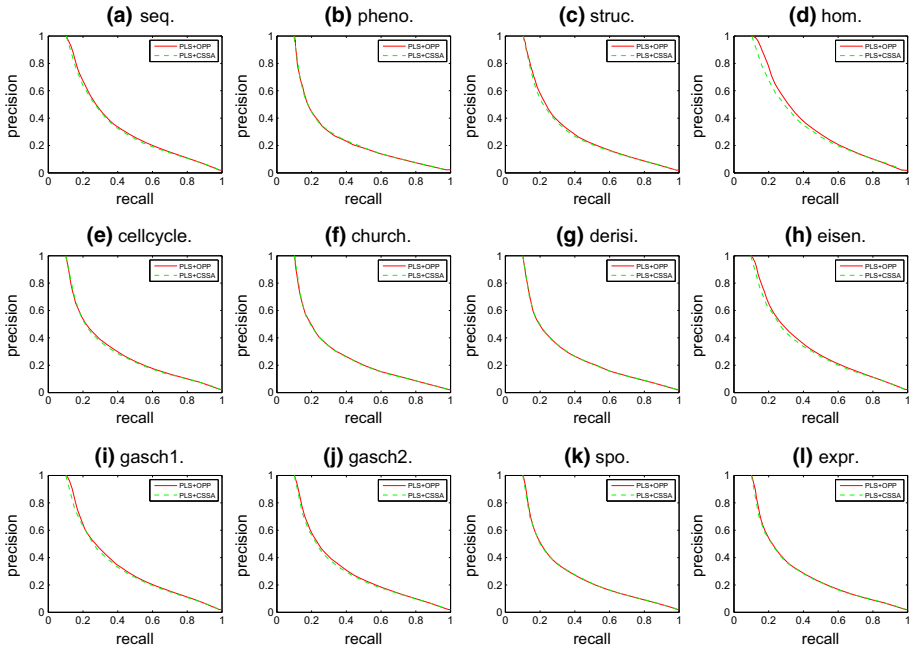


Fig. 3 PR curves for the FunCat data sets

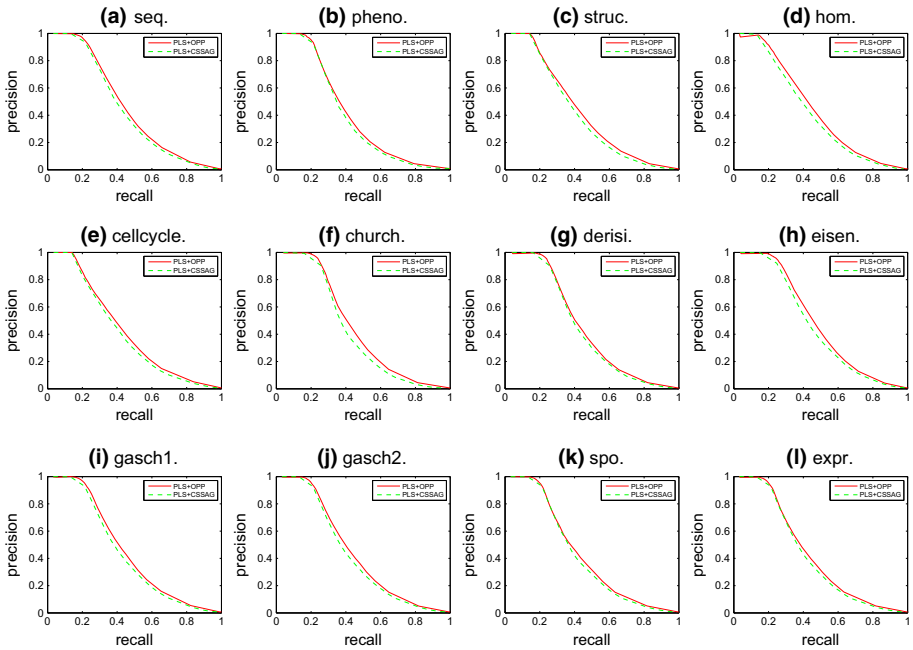


Fig. 4 PR curves for the GO data sets

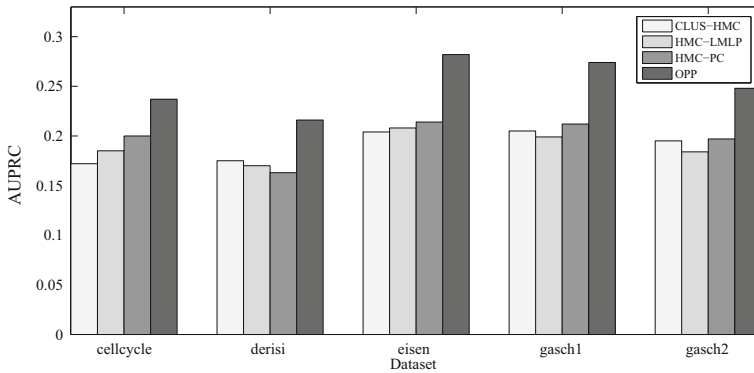


Fig. 5 AUPRC values for five FunCat datasets

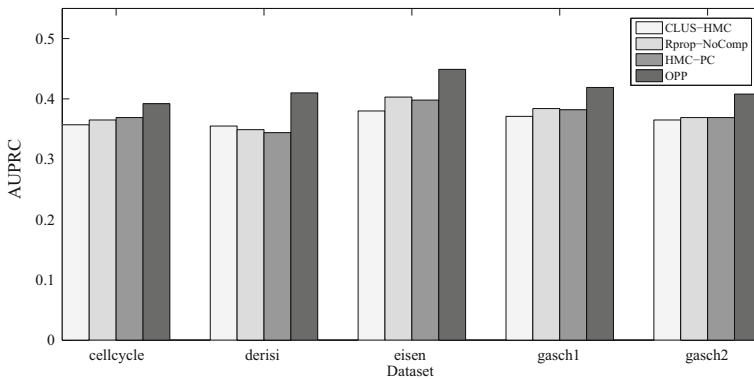


Fig. 6 AUPRC values for five GO datasets

Next, we compare our proposed method with other HMC methods that are developed more recently. Experimental results on five protein function datasets show that OPP consistently outperforms the three baselines in terms of AUPRC values on both tree- and DAG-structured hierarchies, as shown in Figs. 5 and 6. In particular, the improvement of OPP over the state-of-the-arts is highly statistically significant, further demonstrating the validity of our method.

5 Conclusion

In this paper, we propose a novel solution which directly finds a small number of paths that can cover the desired label subgraph in a tree/DAG. We adopt the advantages of PLS techniques for multilabel estimation, which address compression and learning in a unified framework. We then formulate the HMC problem into the constrained path selection problem. Apparently, a path can model long-range interactions, where the labels serve as context for one another, thus less affected by the mistakes of local classifiers. The introduction of path selection models allows us to leverage the efficient network flow solvers with polynomial time complexity. The encouraging empirical results show that our proposed method is more effective than the state-of-the-arts on both tree- and DAG-structured data sets. As future

work, we would like to preserve the hierarchical structure during the PLS pre-processing, and extend our approach to more larger, richer domains.

Acknowledgements The authors thank the anonymous reviewers for their valuable comments. This research work is funded by the National Natural Science Foundation of China under Grant No. 61303179.

References

1. Barros RC, Cerri R, Freitas AA, de Carvalho ACPLF (2013) Probabilistic clustering for hierarchical multi-label classification of protein functions. In: Machine learning and knowledge discovery in databases, proceedings, part II, pp 385–400
2. Barutcuoglu Z, Schapire RE, Troyanskaya OG (2006) Hierarchical multi-label prediction of gene function. *Bioinformatics* 22(7):830–836
3. Bi W, Kwok JT (2011) Multi-label classification on tree- and dag-structured hierarchies. In: Proceedings of the 28th international conference on machine learning, pp 17–24
4. Bi W, Kwok JT (2012) Hierarchical multilabel classification with minimum bayes risk. In: Proceedings of the 12th IEEE international conference on data mining, pp 101–110
5. Bi W, Kwok JT (2014) Mandatory leaf node prediction in hierarchical multilabel classification. *IEEE Trans Neural Netw Learn Syst* 25(12):2275–2287
6. Blockeel H, Schietgat L, Struyf J, Džeroski S, Clare A (2006) Decision trees for hierarchical multilabel classification: a case study in functional genomics. In: Proceedings of the 10th European conference on principles of data mining and knowledge discovery, pp 18–29
7. Cerri R, Barros RC, de Carvalho ACPLF (2011) Hierarchical multi-label classification for protein function prediction: a local approach based on neural networks. In: Intelligent systems design and applications, pp 337–343
8. Cerri R, Barros RC, de Carvalho ACPLF (2014) Hierarchical multi-label classification using local neural networks. *J Comput Syst Sci* 80:39–56
9. Cerri R, Barros RC, de Carvalho ACPLF (2015) Hierarchical classification of gene ontology-based protein functions with neural networks. In Proceedings of the 2015 international joint conference on neural networks, pp 1–8
10. Cesa-bianchi N, Zaniboni L, Collins M (2004) Incremental algorithms for hierarchical classification. *J Mach Learn Res* 7:31–54
11. Cesa-bianchi N, Gentile C, Zaniboni L (2006) Hierarchical classification: combining bayes with SVM. In: Proceedings of the 23rd international conference on machine learning, pp 177–184
12. Clare A (2003) Machine learning and data mining for yeast functional genomics. Ph.D. Thesis, University of Wales, Aberystwyth
13. Grauman K, Sha F, Hwang SJ (2011) Learning a tree of metrics with disjoint visual features. In: Advances in neural information processing systems 24, pp 621–629
14. Hariharan B, Zelnik-Manor L, Vishwanathan SVN, Varma M (2010) Large scale max-margin multi-label classification with priors. In: Proceedings of the 27th international conference on machine learning, pp 423–430
15. Hernandez J, Sucar LE, Morales EF (2013) A hybrid global-local approach for hierarchical classification. In: Proceedings of the twenty-sixth international Florida artificial intelligence research society conference, pp 432–437
16. Kiritchenko S, Matwin S, Famili AF (2004) Hierarchical text categorization as a tool of associating genes with gene ontology codes. In: European workshop on data mining and text mining in bioinformatics, pp 30–34
17. Ramírez-Corona M, Sucar LE, Morales EF (2014) Chained path evaluation for hierarchical multi-label classification. In Proceedings of the twenty-seventh international Florida artificial intelligence research society conference, pp 502–507
18. Rosipal R, Krämer N (2006) Overview and recent advances in partial least squares. In: Subspace, latent structure and feature selection techniques, pp 34–51
19. Rousu J, Saunders C, Szedlmák S, Shawe-Taylor J (2006) Kernel-based learning of hierarchical multilabel classification models. *J Mach Learn Res* 7:1601–1626
20. Silla CN Jr, Freitas AA (2011) A survey of hierarchical classification across different application domains. *Data Min Knowl Disc* 22(1–2):31–72
21. Vens C, Struyf J, Schietgat L, Džeroski S, Blockeel H (2008) Decision trees for hierarchical multi-label classification. *Mach Learn* 73(2):185–214

22. Wang P, Zhang P, Guo L (2012) Mining multi-label data streams using ensemble-based active learning. In: Proceedings of the 12th SIAM international conference on data mining, pp 1131–1140
23. Wold H (1975) Path models with latent variables: the nipals approach. In: Quantitative sociology: international perspectives on mathematical and statistical model building, pp 307–357
24. Wold S, Martens H, Wold H (1983) The multivariate calibration problem in chemistry solved by the pls method. In: Matrix pencils, pp 286–293
25. Zhou D, Xiao L, Wu M (2011) Hierarchical classification via orthogonal transfer. In: Proceedings of the 28th international conference on machine learning, pp 801–808