Mandatory Leaf Node Prediction in Hierarchical Multilabel Classification

Wei Bi and James T. Kwok

***Abstract*— In hierarchical classification, the output labels reside on a tree- or directed acyclic graph (DAG)-structured hierarchy. On testing, the prediction paths of a given test example may be required to end at leaf nodes of the label hierarchy. This is called mandatory leaf node prediction (MLNP) and is particularly useful, when the leaf nodes have much stronger semantic meaning than the internal nodes. However, while there have been a lot of MLNP methods in hierarchical multiclass classification, performing MLNP in hierarchical multilabel classification is difficult. In this paper, we propose novel MLNP algorithms that consider the global label hierarchy structure. We show that the joint posterior probability over all the node labels can be efficiently maximized by dynamic programming for label trees, or greedy algorithm for label DAGs. In addition, both algorithms can be further extended for the minimization of the expected symmetric loss. Experiments are performed on real-world MLNP data sets with label trees and label DAGs. The proposed method consistently outperforms other hierarchical and flat multilabel classification methods.**

***Index Terms*— Bayesian decision, hierarchical classification, integer linear program (ILP), multilabel classification.**

I. Introduction

IN MANY real-world classification problems, the output labels are organized in a hierarchy, and an example can belong to a certain label only if it also belongs to that label's parent in the hierarchy. For example, in bioinformatics, a gene is assigned with its gene functions arranged in the form of a tree in the functional catalog (Funcat) or as a directed acyclic graph (DAG) in the gene ontology (GO) [1]; in music, the musical signals are organized in an audio taxonomy [2]; and in Wikipedia, its documents are classified into different topics from the category graph. Yet, many classification algorithms are flat, in the sense that they simply ignore the label structure and treat the labels separately. Hierarchical classification algorithms, on the other hand, utilize these hierarchical relationships between labels in making predictions. Not surprisingly, they can often achieve better prediction performance.

Hierarchical classification algorithms can be categorized along several dimensions. First, they can be classified as either: 1) hierarchical multiclass classification (HMC), in which the labels of each example are on a single path in the hierarchy or 2) HMC, in that the labels can fall on a union of multiple paths [3]. An everyday example of HMC is that a document/image/song/video may have multiple tags from different meta-categories [3]–[6]. Similarly, in bioinformatics, a gene may be associated with more than one biological functions on different parts of a functional ontology [7]. Because of this great practical significance, HMC has been studied extensively in recent years.

Hierarchical (multiclass or multilabel) classification algorithms can also be categorized based on the depths in the label hierarchy of which their predictions end. Some algorithms ensure that the label predictions always lie on a path from the root to the leaf(s) (full-path prediction); whereas some produce predictions that might stop at an internal node (partial-path prediction) [5]. Following the terminology in a recent survey in [8], the first prediction mode is called mandatory leaf node prediction (MLNP); whereas the second one is called nonmandatory leaf node prediction (NMLNP). Depending on the application domain and how the label hierarchy is generated, either one of these prediction modes may be more relevant. For example, in the taxonomies of musical signals [2] and genes [7], the leaf nodes have much stronger semantic/biological meanings than the internal nodes, and MLNP is more important. Besides, sometimes the label hierarchy is learned from the data, using methods, such as hierarchical clustering [9], Bayesian network structure learning [10], and label tree methods [11], [12]. In these cases, the internal nodes are only artificial and MLNP is again more relevant. In the recent second Pascal challenge on large-scale hierarchical text classification (LSHTC), the tasks also require MLNP.

Most of the existing HMC algorithms are designed for NMLNP [1], [3]–[6]. As far as we are aware of, the only HMC algorithm designed specifically for MLNP is the recent HMC-label powerset (HMC-LP) algorithm [13]. Its main idea is to reduce the hierarchical problem to a nonhierarchical one by running the (nonhierarchical) multilabel classification method of label powerset (LP) [14] at each level of the hierarchy. However, this relies on the powerset of all labels at the same level, making it unsuitable for large label hierarchies. In addition, as it processes the hierarchy level-by-level, this cannot be applied on DAGs, where levels are not well defined.

While there have been a lot of MLNP methods in HMC [8], they cannot be easily extended to the HMC setting. Recall that

Manuscript received March 26, 2013; revised February 19, 2014; accepted February 19, 2014. Date of publication May 6, 2014; date of current version November 17, 2014. This work was supported by the Research Grants Council of the Hong Kong Special Administrative Region under Grant 614012.

The authors are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: weibi@cse.ust.hk; jamesk@cs.ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2309437

in HMC, for each internal node labeled positive in an MLNP, exactly one of its children is positive [15], [16]. In contrast, for HMC, one has to consider at each node the more difficult question of how many and which child/children are to be labeled positive. Even after these decisions are made (e.g., by adjusting the classification threshold heuristically), it is hard to ensure that all the prediction paths will end at leaf nodes, and so a lot of partial paths may be resulted.

To address this problem, one possibility is to first predict the number of leaf labels (k) that the test example has, and then pick the k leaf labels whose posterior probabilities are the largest. However, learning k can be difficult and computationally expensive. In addition, the posterior probability computed at each leaf l corresponds only to a single prediction path from the root to l, while the target multilabel in HMC can have multiple paths. A better justified approach is to compute the posterior probabilities of all subtrees/subgraphs that have k leaf nodes; and pick the one with the highest probability. However, as there are $\binom{N}{k}$ such possible subsets (where N is the number of leafs), this can be expensive when N is large.

In this paper, we propose efficient algorithms for MLNP in both tree- and DAG-structured hierarchical multilabel classification. The target multilabel is obtained by maximizing the posterior probability over all the feasible multilabels. For label trees, we show that this maximization can be efficiently performed by dynamic programming. For label DAGs, by adopting a weak nested approximation assumption, the probability can be maximized in a greedy manner.

Besides, as in any classification problem, the loss function plays an important role [17]. In flat multilabel classification, Dembczynski *et al.* [18] showed how to minimize the expected risks corresponding to the hamming loss (H-loss), rank loss, subset zero-one loss, and F1 score. In HMC, Cesa-Bianchi *et al.* [6] proposed the B-SVM, which minimizes the expected risk corresponding to the so-called H-loss. However, there is yet no work on risk minimization for MLNP. In this paper, we also extend the proposed algorithms to MLNP that minimizes the risk of the weighted symmetric loss [19]. We show that the resultant optimization problem can be solved by essentially the same algorithms, but with different parameter settings.

The rest of this paper is organized as follows. Section II provides a brief review on hierarchical classification algorithms. Section III describes the proposed MLNP algorithm on label trees. The MLNP on label DAGs is discussed in Section IV. Risk-minimizing MLNP is discussed in Section V. Experimental results are presented in Section VI. Finally, conclusions are drawn in Section VII.

Preliminary results have been recently reported in [20]. Besides providing a more thorough literature review, this paper: 1) proposes a new MLNP algorithm for label trees, which is faster than the one in [20] and has its optimality guaranteed without requiring extra assumptions; 2) provides intuitions and technical proofs to the proposed algorithms; and 3) provides stronger experimental evidence to demonstrate the merits of the proposed algorithms. In particular, experiments on two new large data sets (Caltech256 and DMOZ from the second Pascal challenge on LSHTC) have been added.

TABLE I
REPRESENTATIONS OF SOME POPULAR HIERARCHICAL CLASSIFICATION ALGORITHMS, TOGETHER WITH THE PROPOSED ALGORITHMS, USING THE 4-TUPLE NOTATION IN [8]

algorithm	$\langle \Delta, \Xi, \Omega, \Theta \rangle$
HBR [5], [6], [21] greedy algorithm in [4]	$\langle$ multilabel, NMLNP, tree, local classifier per node $\rangle$
Rousu *et al.* [3]	$\langle$ multilabel, NMLNP, tree, global classifier $\rangle$
CLUS-HMC [1]	$\langle$ multilabel, NMLNP, tree/DAG, global classifier $\rangle$
HMC-LP [13]	$\langle$ multilabel, MLNP, tree, local classifier per level $\rangle$
the proposed MAT	$\langle$ multilabel, MLNP, tree, local classifier per node $\rangle$
the proposed MAS	$\langle$ multilabel, MLNP, tree/DAG, local classifier per node $\rangle$

II. RELATED WORK

Using the categorization in [8], hierarchical classification algorithms can be differentiated and represented by the 4-tuple $\langle \Delta, \Xi, \Omega, \Theta \rangle$. Here, Δ indicates whether the classification of each example can be associated with only one path (multiclass classification) or multiple paths (multilabel classification), Ξ indicates the prediction depth of the algorithm, either it must always end at the leaf node (MLNP) or can end at an internal node in the hierarchy (NMLNP), Ω specifies the type of label hierarchy, either a tree or a DAG, and Θ specifies how the classifier is constructed (e.g., the algorithm can train a local classifier at each node/level/parent, or use a global classifier for the whole hierarchy). Using this 4-tuple notation, a summary of the hierarchical classification algorithms discussed in this section is shown in Table I.

As mentioned in Section I, most existing hierarchical classification algorithms can only handle NMLNP. A popular approach is hierarchical binary relevance (HBR) [5], [6], [21], which learns a local binary classifier (such as a SVM with probabilistic output, for the H-SVM [5]) at each node of the label hierarchy. Prediction is then performed recursively in a top-down manner. If a node is predicted positive, HBR continues with the prediction on its children nodes; otherwise, all its descendants are labeled negative. HBR is originally designed for tree-structured label hierarchies, but can be easily extended for DAGs. Specifically, the training step remains the same (with one classifier per DAG node). For prediction, the recursion continues only if all the parents of a node are predicted positive. On the other hand, CLUS-HMC [1] trains a global classifier (which is a decision tree) and predicts the labels of all the nodes together. Rousu *et al.* [3] proposed a global kernel-based algorithm in which the label tree is viewed as a Markov tree and an exponential family is defined on the edges. Bi and Kwok [4] proposed efficient greedy algorithms to find the best subtree/subgraph in a label tree/DAG that is consistent with a given hierarchical property.

As far as we are aware of, HMC-LP [13] is the only HMC algorithm designed for MLNP. It is based on the idea of LP [8], which trains a multiclass classifier with the label set consisting of all the label combinations. HMC-LP trains a local LP classifier at each level of the hierarchy. The label set of this LP classifier contains label combinations of the nodes, of which parent nodes appearing in any label combination in the LPs label set from its upper level. However, such a label combination strategy considerably increases the number of classes when the number of nodes per level is large, making it not scalable

to large hierarchies. In addition, this method can only handle tree-structured hierarchies. Since a node can reside at different levels in the DAG, such a level-by-level label combination cannot be directly used for DAG-structured hierarchies.

In principle, structured prediction methods (such as the structured SVM [22]) can also be used for MLNP (in HMC) by restricting the feasible set of structured outputs to be the union of all possible MLNPs. However, the underlying cutting plane algorithm needs to find the most violated MLNP in each iteration. This may require an exhaustive search over the potentially exponential number of possible MLNPs, making it difficult to scale up to practical applications.

Alternatively, some algorithms not designed for MLNP in HMC can be adapted for this purpose. For example, consider the popular binary relevance (BR) algorithm [23], which trains an independent binary classifier for each label. It is originally designed for flat multilabel classification. A straightforward approach to adapt this for MLNP in HMC is to train the independent classifiers at only the leaf nodes of the label trees or label DAGs [2], [24]. On prediction, if a leaf node is labeled positive, all its ancestors are also labeled positive, thus turning it into an MLNP. However, the flat BR does not make use of any hierarchy information.

Some NMLNP algorithms can also be adapted for MLNP by combining with the MetaLabeler [25], which predicts the number of leaf labels (k) that the test example should be assigned. For instance, given a test example, the HMC-CLUS [1] provides a probability estimate for labeling each leaf node in the label hierarchy as positive. A similar probability estimate can also be obtained for HBR [5] with a probabilistic base learner, by simply multiplying the probabilities of the nodes along the path from the leaf to the root. With the k value predicted by MetaLabeler, we then pick the k leaf nodes with largest probabilities as the MLNP solution.

However, unless the NMLNP method provides a ranking on the candidate leaf nodes on prediction, it is not easy to be adapted for MLNP, even with the help of the MetaLabeler. For example, the greedy algorithms in [4] select a subtree/subgraph with k nodes by greedily selecting the node (or supernode, which is a union of multiple nodes) and merging with its parent. However, it is difficult to require these selected nodes to satisfy the MLNP requirement. The kernel-based method in [3] is another example algorithm that is difficult to extend for MLNP. Its joint kernel mapping is based on all the output subtrees in the label hierarchy. If one restricts the allowable outputs to the MLNPs, the joint kernel mapping needs to be significantly changed, so are the corresponding dual problem and the associated tailor-made optimization algorithm.

III. Maximum a Posteriori MLNP on Label Trees

In this section, we assume that the label hierarchy is a tree $\mathcal{T}$. The nodes in $\mathcal{T}$ are indexed from 0 (for the root), $1, 2, \ldots, |\mathcal{T}|$. With a slight abuse of notation, we will also use $\mathcal{T}$ to denote the set of all these nodes. For a subset $A \subseteq \mathcal{T}$, its complement is denoted by $A^c = \mathcal{T} \backslash A$. For a node i, we denote its parent by $\text{pa}(i)$ and its set of children by $\text{child}(i)$. In addition, given a vector $\mathbf{y}$, $\mathbf{y}_A$ is the subvector of $\mathbf{y}$ with indices from A. The transpose of a vector is denoted by the superscript $(')$.

In HMC, we are given a set of training examples $\{(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x}$ is the input and $\mathbf{y} = [y_1, \ldots, y_{|\mathcal{T}|}]' \in \{0, 1\}^{|\mathcal{T}|}$ is the multilabel denoting memberships of $\mathbf{x}$ to each of the nodes. Equivalently, $\mathbf{y}$ can be represented by a set $\Omega \subseteq \mathcal{T}$, such that $y_i = 1$ if $i \in \Omega$; 0 otherwise. For $\mathbf{y}$ (or Ω) to respect the tree structure, we require that if any node $i \in \mathcal{T} \backslash \{0\}$ is labeled positive, its parent $\text{pa}(i)$ must also be labeled positive, that is

$$y_i = 1 \Rightarrow y_{\text{pa}(i)} = 1. \tag{1}$$

In addition, for any group of siblings $\{i_1, i_2, \ldots, i_m\}$, we assume that their labels are conditionally independent given the label of their parent $\text{pa}(i_1)$ and $\mathbf{x}$, that is

$$p(y_{i_1}, y_{i_2}, \ldots y_{i_m} | y_{\text{pa}(i_1)}, \mathbf{x}) = \prod_{j=1}^{m} p(y_{i_j} | y_{\text{pa}(i_1)}, \mathbf{x}). \tag{2}$$

This simplification is standard in Bayesian networks and also commonly used in HMC [5], [26]. By repeated application of the probability product rule, we have

$$p(y_0, \ldots, y_{|\mathcal{T}|} | \mathbf{x}) = p(y_0 | \mathbf{x}) \prod_{i \in \mathcal{T} \backslash \{0\}} p(y_i \mid y_{\text{pa}(i)}, \mathbf{x}). \tag{3}$$

For MLNP, obviously the root is always labeled, and so

$$p(y_0 = 1 \mid \mathbf{x}) = 1. \tag{4}$$

A. Training

With the condition in (1) and the simplification in (3), we only need to train probability estimators $p(y_i = 1 \mid y_{\text{pa}(i)} = 1, \mathbf{x})$ for each $i \in \mathcal{T} \backslash \{0\}$. This is the same as for the H-SVM [6]. The algorithms to be proposed are independent of the way these probability estimators are learned. For example, one can train a binary SVM at each node i, using only those training examples that the parent of i is labeled positive, then convert the SVM output to a probability estimate (using schemes, such as sigmoid scaling [27]). Other methods, such as regularized logistic regression and multitask SVM may also be used.

B. Prediction

1) Optimization Problem: For a test example $\mathbf{x}$, the maximum *a posteriori* MLNP corresponds to the multilabel Ω^* that: 1) maximizes the posterior probability in (3) and 2) respects $\mathcal{T}$. This prediction task can be formulated as the following optimization problem:

$$\Omega^* = \arg\max_{\Omega} \quad p(\mathbf{y}_\Omega = \mathbf{1}, \mathbf{y}_{\Omega^c} = \mathbf{0} | \mathbf{x}) \tag{5}$$

$$\text{s.t.} \quad y_0 = 1,$$

$$\Omega \text{ contains no partial path,}$$

$$\text{all } y_i\text{'s respect the label hierarchy.} \tag{6}$$

Note that $p(\mathbf{y}_\Omega = \mathbf{1}, \mathbf{y}_{\Omega^c} = \mathbf{0} | \mathbf{x})$ considers all the node labels in the hierarchy simultaneously. In contrast, as discussed in Section I, existing MLNP methods in hierarchical multiclass/multilabel classification only consider the local hierarchy information at each node.

Associate an indicator function $\psi : \mathcal{T} \to \{0,1\}^{|\mathcal{T}|}$ with the multilabel Ω, such that $\psi_i \equiv \psi(i) = 1$ if $i \in \Omega$ and 0 otherwise. Let the set of leaf nodes in $\mathcal{T}$ be $\mathcal{L}$. It is easy to see that the constraints in (5) translates to

$$\begin{cases} \psi_0 = 1 \\ \forall i \in \mathcal{L}^c \text{ with } \psi_i = 1 : \sum\limits_{j \in \text{child}(i)} \psi_j \geq 1 \\ \forall i \in \mathcal{T} \backslash \{0\} : \psi_i \leq \psi_{\text{pa}(i)}. \end{cases} \tag{7}$$

In addition, the following Proposition shows that the posterior probability in (5) can be rewritten as a weighted combination of the unknown ψ_i's.

Proposition 1: For a label tree $\mathcal{T}$

$$\log p(\mathbf{y}_\Omega = \mathbf{1}, \mathbf{y}_{\Omega^c} = \mathbf{0} | \mathbf{x}) = \sum_{i \in \mathcal{T}} w_i \psi_i$$

where

$$w_i = \begin{cases} \sum\limits_{l \in \text{child}(0)} \log(1 - p_l), & i = 0 \\ \log \dfrac{p_i}{1 - p_i}, & i \in \mathcal{L} \\ \log \dfrac{p_i}{1 - p_i} + \sum\limits_{l \in \text{child}(i)} \log(1 - p_l), & i \in \mathcal{L}^c \backslash \{0\} \end{cases} \tag{8}$$

and $p_i \equiv p(y_i = 1 \mid y_{\text{pa}(i)} = 1, \mathbf{x})$.

Proof: From (3), we have

$$\begin{aligned} & p(\mathbf{y}_\Omega = \mathbf{1}, \mathbf{y}_{\Omega^c} = \mathbf{0} \mid \mathbf{x}) \\ & \quad = p(y_0 = 1 \mid \mathbf{x}) \cdot \prod_{i \in \Omega \backslash \{0\}} p(y_i = 1 \mid y_{\text{pa}(i)} = 1, \mathbf{x}) \\ & \qquad \times \prod_{i \in \Omega^c \wedge \text{pa}(i) \in \Omega} p(y_i = 0 \mid y_{\text{pa}(i)} = 1, \mathbf{x}) \\ & \qquad \times \prod_{i \in \Omega^c \wedge \text{pa}(i) \in \Omega^c} p(y_i = 0 \mid y_{\text{pa}(i)} = 0, \mathbf{x}). \end{aligned} \tag{9}$$

Using (1) and (4), and the definition of p_i, (9) can be simplified as

$$\begin{aligned} & \log p(\mathbf{y}_\Omega = \mathbf{1}, \mathbf{y}_{\Omega^c} = \mathbf{0} \mid \mathbf{x}) \\ & \quad = \sum_{i \in \Omega \backslash \{0\}} \log p_i + \sum_{i \in \Omega^c \wedge \text{pa}(i) \in \Omega} \log(1 - p_i). \end{aligned} \tag{10}$$

The combination $(\text{pa}(i) = 0, \psi_i = 1)$ violates the tree hierarchy and cannot occur. Hence, (10) can be equivalently rewritten as

$$\begin{aligned} & \log p(\mathbf{y}_\Omega = \mathbf{1}, \mathbf{y}_{\Omega^c} = \mathbf{0} \mid \mathbf{x}) \\ & = \sum_{i \in \mathcal{T} \backslash \{0\}} \psi_i \log p_i + \sum_{i \in \mathcal{T} \backslash \{0\}} (\psi_{\text{pa}(i)} - \psi_i) \log(1 - p_i) \\ & = \sum_{i \in \mathcal{T} \backslash \{0\}} \psi_i (\log p_i - \log(1 - p_i)) + \sum_{i \in \mathcal{T} \backslash \{0\}} \psi_{\text{pa}(i)} \log(1 - p_i) \\ & = \sum_{i \in \mathcal{T} \backslash \{0\}} \psi_i (\log p_i - \log(1 - p_i)) + \sum_{i \in \mathcal{L}^c} \psi_i \sum_{l \in \text{child}(i)} \log(1 - p_l) \\ & = \sum_{i \in \mathcal{L}} \psi_i (\log p_i - \log(1 - p_i)) \\ & \quad + \sum_{i \in \mathcal{L}^c \backslash \{0\}} \psi_i \left(\log p_i - \log(1 - p_i) + \sum_{l \in \text{child}(i)} \log(1 - p_l) \right) \\ & \quad + \sum_{l \in \text{child}(0)} \log(1 - p_l) \end{aligned}$$

which is equal to the objective in (11). ■

Algorithm 1 MAT (Mandatory Leaf Node Prediction on Trees)

Input: Tree hierarchy $\mathcal{T}$; weights $\{w_i\}_{i \in \mathcal{T}}$.
Output: $\{\psi_i\}_{i \in \mathcal{T}}$.

1: sort the nodes of $\mathcal{T}$ in reverse topological order and store them in the queue $\mathcal{Q}$;
2: **while** $\mathcal{Q}$ is not empty **do**
3: retrieve the first element i in $\mathcal{Q}$ and delete it from $\mathcal{Q}$;
4: **if** i is a leaf node **then**
5: $H_i^+(\mathbf{x}) = w_i$;
6: **else if** $H_j(\mathbf{x}) = 0$ for every $j \in \text{child}(i)$ **then**
7: $r = \arg\max_{j \in \text{child}(i)} H_j^+(\mathbf{x})$;
8: $H_i^+(\mathbf{x}) = w_i + H_r^+(\mathbf{x})$;
9: $H_r(\mathbf{x}) = H_r^+(\mathbf{x})$.
10: **else**
11: $H_i^+(\mathbf{x}) = w_i + \sum_{j \in \text{child}(i)} H_j(\mathbf{x})$;
12: **end if**
13: $H_i(\mathbf{x}) = \max\{H_i^+(\mathbf{x}), 0\}$;
14: **end while**
15: Traverse $\mathcal{T}$ using breadth-first-search and output ψ_i's as:

$$\psi_i = \begin{cases} 1 \ i = 0 \\ 1 \ H_i(\mathbf{x}) \neq 0 \text{ and } \psi_{\text{pa}(i)} = 1 \\ 0 \ H_i(\mathbf{x}) = 0 \text{ or } \psi_{\text{pa}(i)} = 0 \end{cases}.$$

Hence, we immediately obtain that (5) can be rewritten as an integer linear program (ILP).

Corollary 1: For a label tree $\mathcal{T}$, problem (5) can be rewritten as

$$\begin{aligned} \max_{\{\psi_i\}_{i \in \mathcal{T}}} \quad & \sum_{i \in \mathcal{T}} w_i \psi_i \\ \text{s.t.} \quad & \psi_i \in \{0, 1\} \quad \forall i \in \mathcal{T}, \\ & \text{the constraints in (7)} \end{aligned} \tag{11}$$

where w_i is as defined in (8).

Remark 1: For a leaf node i, note that its w_i value in (8) is equal to its log-odds. Consider the special case where the label hierarchy is flat. Every label is then a leaf node connected directly to the root and the last two sets of constraints in (7) become vacuous. Hence, to maximize (11), we immediately have $\psi_i = 1$ if the log-odds $w_i > 0$ and is 0 otherwise. In other words, label i is predicted positive if $p_i > 0.5$, which agrees with the Bayes decision rule.

2) Dynamic Programming Solver: For a general label tree $\mathcal{T}$, we propose to solve (11) using dynamic programming. The procedure, called mandatory leaf node prediction on trees (MAT), is shown in Algorithm 1.

For any node i in $\mathcal{T}$, let $\text{sub}(i)$ be the subtree under i (including i itself). Consider replacing $\mathcal{T}$ in problem (11) by $\text{sub}(i)$. Let $H_i^+(\mathbf{x})$ be the maximum value of $\sum_{j \in \text{sub}(i)} w_i \psi_i$ when i is labeled positive. When i is a leaf node, $H_i^+(\mathbf{x})$ is simply w_i (step 5). When i is an internal node, the maximum value that $\sum_{j \in \text{sub}(i)} w_j \psi_j$ can take (denoted $H_i(\mathbf{x})$) is either $H_i^+(\mathbf{x})$, when i is labeled positive ($\psi_i = 1$); or 0, when i (and all its descendants) is labeled negative ($\psi_i = 0$). We compute $H_i^+(\mathbf{x})$ in a bottom-up manner from the $H_j(\mathbf{x})$'s ($j \in \text{child}(i)$).

a) $H_j(\mathbf{x}) > 0$ *for some child* j*:* We can simply combine the optimal MLNPs from i's children (step 11).
b) $H_j(\mathbf{x}) = 0$ *for all* $j \in child(i)$*:* In other words, all descendants of i prefer to be labeled negative. However, as we are considering $H_i^+(\mathbf{x})$, which corresponds to the case where node i is labeled positive, MLNP then requires $\psi_j = 1$ for at least one $j \in \text{child}(i)$. To maximize the objective $\sum_{j\in\text{sub}(i)} w_i \psi_i$, the best child to pick is the r in step 7.

Hence, if we visit the nodes in reverse topological order (i.e., bottom-up), we can compute the $H_i^+(\mathbf{x})$ and $H_i(\mathbf{x})$ values for all $i \in \mathcal{T}$. Using the fact that the root node ($i = 0$) is always labeled positive, we immediately obtain that $H_0^+(\mathbf{x})$ is the optimal objective value in (11).

3) Time Complexity: Let V be the number of nodes in $\mathcal{T}$. Performing the reverse topological sort in step 1 takes $O(V)$ time. If i is not a leaf node, $H_i^+(\mathbf{x})$ is calculated by either steps 6–9 or step 11. In both cases, we have to visit all its children. Recall that when we want to compute $H_i^+(\mathbf{x})$, all the $H_j^+(\mathbf{x})$'s for its children have already been obtained. Thus, computing $H_i^+(\mathbf{x})$ for node i takes $O(d_i)$ time, where d_i is the number of i's children. Hence, the while loop takes a total of $O(\sum_{i\in\mathcal{T}} d_i) = O(V)$ time. Step 15 takes $O(V)$ time. Thus, the total time complexity of Algorithm 1 is $O(V)$.

IV. Maximum a Posteriori MLNP on Label DAGs

In this section, we consider the case where the label hierarchy is a DAG $\mathcal{G}$. Analogous to (2), we use the following conditional independence assumption:

$$p(y_{i_1}, y_{i_2}, \ldots y_{i_m}|\mathbf{y}_{\text{Pa}(i_1)}, \mathbf{x}) = \prod_{j=1}^{m} p(y_{i_j}|\mathbf{y}_{\text{Pa}(i_1)}, \mathbf{x})$$

where $\text{Pa}(i)$ is the set of (possibly multiple) parents of node i. The joint posterior probability $p(y_1, \ldots, y_{|\mathcal{G}|}|\mathbf{x})$ can be simplified accordingly as

$$p(y_1, \ldots, y_{|\mathcal{G}|}|\mathbf{x}) = p(y_0|\mathbf{x}) \prod_{i\in\mathcal{G}\setminus\{0\}} p(y_i \mid \mathbf{y}_{\text{Pa}(i)}, \mathbf{x}). \quad (12)$$

The prediction task involves the same optimization problem as in (5), except that the label hierarchy is now a DAG. However, there are two interpretations on how the labels should respect the DAG in (6) [1], [4]. The first one requires that if a node is labeled positive, all its parents must also be labeled positive. In bioinformatics, this is called the true path rule which governs, for example, the GO taxonomy on gene functions [28]. The alternative is that a node can be labeled positive if at least one of its parents is positive. Here, we adopt the first interpretation which is more common.

A direct maximization of $p(y_1, \ldots, y_{|\mathcal{G}|}|\mathbf{x})$ by (12) is NP-hard [29]. In addition, the size of each probability table $p(y_i|\mathbf{y}_{\text{Pa}(i)}, \mathbf{x})$ in (12) grows exponentially with $|\text{Pa}(i)|$. Hence, it can be impractical and inaccurate when $\mathcal{G}$ is large and the sample size is limited.

Instead of using (12), we assume that $p(y_1, \ldots, y_{|\mathcal{G}|}|\mathbf{x})$ can be factored as follows:

$$p(y_1, \ldots, y_{|\mathcal{G}|}|\mathbf{x}) = \frac{1}{n(\mathbf{x})} p(y_0|\mathbf{x}) \prod_{i\in\mathcal{G}\setminus\{0\}} \prod_{j\in\text{Pa}(i)} p(y_i|y_j, \mathbf{x}) \quad (13)$$

where $n(\mathbf{x})$ is for normalization. Essentially, this replaces the complicated $p(y_i|\mathbf{y}_{\text{Pa}(i)}, \mathbf{x})$ in (12) by a product of $p(y_i|y_j, \mathbf{x})$'s. This follows the approach of composite likelihood (or pseudolikelihood) [30] which replaces a difficult probability density function by a set of marginal or conditional events that are easier to evaluate. In particular, (13) corresponds to the so-called pairwise conditional likelihood that has been commonly used in longitudinal studies and bioinformatics [31]. Composite likelihood has been successfully used in different applications, such as genetics, spatial statistics, and image analysis. The connection between composite likelihood and various (flat) multilabel classification models is also recently discussed in [31]. Using (13), the $2^{|\text{Pa}(i)|}$ numbers in the probability table of $p(y_i|\mathbf{y}_{\text{Pa}(i)}, \mathbf{x})$ are replaced by the $|\text{Pa}(i)|$ numbers of $\{p(y_i|y_j, \mathbf{x})\}_{j\in\text{Pa}(i)}$. Thus, the estimates obtained are much more reliable.

A. Known Number of Labels

In this section, we first consider the case where the prediction of $\mathbf{x}$ is known to have k leaf labels. The following Proposition shows that maximizing (13) can be reduced to a problem similar to (11).

Proposition 2: With assumption (13), problem (5) for a label DAG can be rewritten as

$$\max_{\{\psi_i\}_{i\in\mathcal{G}}} \sum_{i\in\mathcal{G}} w_i \psi_i \quad (14)$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{L}} \psi_i = k \quad \psi_0 = 1$$
$$\forall i \in \mathcal{G}\, \psi_i \in \{0, 1\}$$
$$\forall i \in \mathcal{L}^c \text{ with } \psi_i = 1, \sum_{j\in\text{child}(i)} \psi_j \geq 1$$
$$\forall j \in \text{Pa}(i) \quad \forall i \in \mathcal{G}\setminus\{0\} \psi_i \leq \psi_j \quad (15)$$

where

$$w_i = \begin{cases} \sum_{l\in\text{child}(0)} \log(1 - p_{l0}), & i = 0 \\ \sum_{j\in\text{Pa}(i)} (\log p_{ij} - \log(1 - p_{ij})), & i \in \mathcal{L} \\ \sum_{j\in\text{Pa}(i)} (\log p_{ij} - \log(1 - p_{ij})) & \\ \quad + \sum_{l\in\text{child}(i)} \log(1 - p_{li}), & i \in \mathcal{L}^c\setminus\{0\} \end{cases} \quad (16)$$

and $p_{ij} \equiv p(y_i = 1|y_j = 1, \mathbf{x})$ for $j \in \text{Pa}(i)$.

Proof: Using (13) and that the root is always labeled positive, we have

$$p(\mathbf{y}_\Omega = \mathbf{1}, \mathbf{y}_{\Omega^c} = \mathbf{0} \mid \mathbf{x}) = \frac{1}{n(\mathbf{x})} \prod_{i \in \mathcal{G} \setminus \mathbf{0}} \prod_{j \in \mathrm{Pa}(i)} p(y_i | y_j, \mathbf{x})$$

$$= \frac{1}{n(\mathbf{x})} \prod_{i \in \Omega \setminus \{0\}} \prod_{j \in \mathrm{Pa}(i)} p(y_i = 1 \mid y_j = 1, \mathbf{x}) \times \prod_{i \in \Omega^c : \mathrm{Pa}(i) \subseteq \Omega} \prod_{j \in \mathrm{Pa}(i)} p(y_i = 0 \mid y_j = 1, \mathbf{x}) \times \prod_{i \in \Omega^c \,:\, \mathrm{Pa}(i) \setminus \Omega \neq \emptyset} \left(\prod_{j \in \mathrm{Pa}(i) \cap \Omega} p(y_i = 0 \mid y_j = 1, \mathbf{x}) \prod_{j \in \mathrm{Pa}(i) \cap \Omega^c} p(y_i = 0 \mid y_j = 0, \mathbf{x}) \right)$$

$$= \frac{1}{n(\mathbf{x})} \left(\prod_{i \in \Omega \setminus \{0\}} \prod_{j \in \mathrm{Pa}(i)} p(y_i = 1 \mid y_j = 1, \mathbf{x}) \right) \times \left(\prod_{i \in \Omega^c} \prod_{j \in \mathrm{Pa}(i) \cap \Omega} p(y_i = 0 \mid y_j = 1, \mathbf{x}) \right) \times \left(\prod_{i \in \Omega^c} \prod_{j \in \mathrm{Pa}(i) \cap \Omega^c} p(y_i = 0 \mid y_j = 0, \mathbf{x}) \right). \quad (17)$$

Since $p(y_i = 0 \mid y_j = 0, \mathbf{x}) = 1$, maximizing (17) reduces to maximizing

$$\sum_{i \in \Omega \setminus \{0\}} \sum_{j \in \mathrm{Pa}(i)} \log p(y_i = 1 \mid y_j = 1, \mathbf{x}) + \sum_{i \in \Omega^c} \sum_{j \in \mathrm{Pa}(i) \cap \Omega} \log(1 - p(y_i = 1 \mid y_j = 1, \mathbf{x})).$$

Similar to the proof of Proposition 1, this can be rewritten as

$$\sum_{i \in \mathcal{G} \setminus \{0\}} \sum_{j \in \mathrm{Pa}(i)} \psi_i \log p_{ij} + \sum_{i \in \mathcal{G} \setminus \{0\}} \sum_{j \in \mathrm{Pa}(i)} (\psi_j - \psi_i) \log(1 - p_{ij})$$

$$= \sum_{i \in \mathcal{L}} \psi_i \sum_{j \in \mathrm{Pa}(i)} (\log p_{ij} - \log(1 - p_{ij}) + \sum_{i \in \mathcal{L}^c \setminus \{0\}} \psi_i \left(\sum_{j \in \mathrm{Pa}(i)} (\log p_{ij} - \log(1 - p_{ij})) + \sum_{l \in \mathrm{child}(i)} \log(1 - p_{li}) \right) + \sum_{l \in \mathrm{child}(0)} \log(1 - p_{l0}).$$

■

However, even when k is known, (14) is an ILP with $\binom{|\mathcal{L}|}{k}$ candidate solutions, and can be expensive to solve when $\mathcal{G}$ is large. In the following, we extend the nested approximation property (NAP), first introduced for model-based compressed sensing [32], to constrain the optimal solution.

Definition 1 (k-Leaf-Sparse): A multilabel $\mathbf{y}$ is k-leaf-sparse if k of the leaf nodes are labeled one.

Definition 2 (NAP): For an example $\mathbf{x}$, let its optimal k-leaf-sparse multilabel be Ω_k. The NAP is satisfied if $\{i : i \in \Omega_k\} \subset \{i : i \in \Omega_{k'}\}$ for all $k < k'$.

Algorithm 2 MAS (Mandatory Leaf Node Prediction on Structures)

Input: Hierarchy $\mathcal{G}$; $\{w_i\}_{i \in \mathcal{G}}$; number of leaf nodes to be predicted (k).
Output: $\{\psi_i\}_{i \in \mathcal{G}}$.

```
Initialize every node (except the root) with ψ_i ← 0; ψ_0 ← 1; Ω ← {0}; Create a supernode from each leaf with its ancestors.
for iteration=1 to k do
    select the unassigned supernode S* with the largest SNV;
    assign all unselected nodes in S* with ψ_i ← 1;
    Ω ← Ω ∪ S*;
    for each unassigned supernode S do
        update the SNV of S with Ω using Algorithm 3;
    end for
end for
Output ψ_i = 1 if i ∈ Ω; 0 otherwise.
```

Algorithm 3 Updating the SNV of an Unassigned DAG Supernode S, Containing the Leaf l, With the Set Ω

```
Initialize T as an empty BST and insert l to T;
SNV(S) ← SNV(Ω);
repeat
    node ← find-max(T);
    delete node from T;
    SNV(S) ← SNV(S) + w_node;
    insert nodes in Pa(node)\(Ω ∪ T) to T;
until T = ∅.
```

Note that NAP is often implicitly assumed in many HMC algorithms. For example, consider the common approach that trains a binary classifier at each node and recursively predicts from the root to the subtrees. When the classification threshold at each node is high, prediction stops early; whereas when the threshold is lowered, prediction can go further down the hierarchy. Hence, nodes that are labeled positive at a high threshold will always be labeled at a lower threshold, implying NAP. Another example is the CSSA algorithm in [4]. Since it is greedy, a larger solution (with more labels predicted positive) always includes the smaller solutions. The validity of NAP will be further verified empirically in Section VI-E.

Algorithm 2 shows the proposed greedy algorithm, which will be called mandatory leaf node prediction on structures (MAS). Each node $i \in \mathcal{G}$ is associated with the weight w_i in (16). Initially, only the root is selected ($\psi_i = 1$). For each leaf l in $\mathcal{L}$, we create a supernode, which is a subset in $\mathcal{G}$ containing l and all its ancestors along all paths to the root.[1] Given $|\mathcal{L}|$ leaves in $\mathcal{G}$, there are initially $|\mathcal{L}|$ supernodes. In addition, all of them are unassigned (i.e., each contains an unselected leaf node). Each supernode S has a supernode value (SNV) which is defined as $\mathrm{SNV}(S) = \sum_{i \in S} w_i$ (Fig. 1).

[1]We borrow this terminology from [4]. Note however that our definition of a supernode and its updating are different from those in [4].

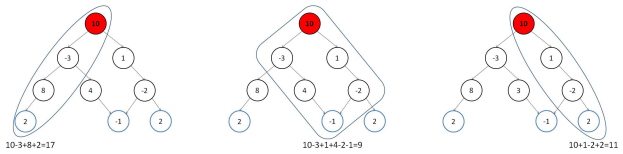

Fig. 1. DAG example showing the initial supernodes and the associated SNVs. The number inside each node i is its w_i. The leftmost supernode, which has the largest SNV, will be selected in the first iteration.

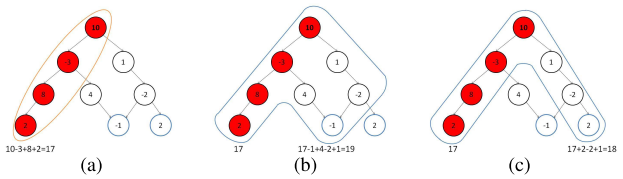

Fig. 2. Result after selecting the first leaf node, and candidate supernodes at the next iteration. (a) Result after selecting the first leaf node. (b) First candidate supernode with two leaf nodes selected. (c) Second supernode with two leaf nodes selected.

In Algorithm 2, the set Ω is used to keep track of the current multilabel solution. In each iteration, supernode S^* with the largest SNV is selected among all the unassigned supernodes. S^* is then assigned, i.e., with the ψ_i's of all its constituent nodes set to 1 [Fig. 2(a)], and Ω is updated accordingly to include nodes in S^*. Because of the NAP assumption, candidate supernodes in future iterations should contain S^*. For each remaining unassigned supernode S, Algorithm 3 then updates its SNV to the value it will take if S is merged with Ω [Fig. 2(a) and (b)], i.e., $\text{SNV}(S) \leftarrow \text{SNV}(S) + \sum_{i \in \Omega \setminus S} w_i$. The T in Algorithm 3 is a self-balancing binary search tree (BST) that keeps track of the nodes in $S \setminus \Omega$ in their topological order. We number the sorted order such that nodes nearer to the root are assigned smaller identifiers. Note that this topological sort only needs to be performed once as part of preprocessing. To facilitate the checking of whether a node is in Ω (step 7 of Algorithm 3), Ω also stores its nodes in a self-balancing BST.

The following Proposition shows that MAS finds the best k-leaf-sparse prediction.

Proposition 3: Under the NAP assumption, Algorithm 2 obtains an optimal ψ solution of (14).

Proof: We will prove this by induction. After initialization, each supernode contains exactly one leaf node. Hence, the solution at iteration 1, with $k = 1$, is clearly optimal.

Assume Algorithm 2 obtains an optimal k-leaf-sparse prediction at iteration k. Recall each unassigned supernode S contains exactly one leaf node. After step 7, its SNV is updated as if it were merged with Ω (thus contains $k+1$ leaf nodes). Because of NAP, the optimal $(k+1)$-leaf-sparse solution must be one of these merged supernodes. At step 3 of iteration $k+1$, the supernode with the largest SNV will be selected, which is thus an optimal $(k+1)$-leaf-sparse prediction. ■

B. Unknown Number of Labels

In practice, the value of k may not be known. A straightforward approach is to run Algorithm 2 with $k = 1, \ldots, |\mathcal{L}|$ and find the $\Omega_k \in \{\Omega_1, \ldots, \Omega_{|\mathcal{L}|}\}$ that maximizes the posterior probability (13) for DAG. However, recall that $\Omega_k \subset \Omega_{k+1}$ under the NAP assumption. Hence, we can simply set $k = |\mathcal{L}|$ and Ω_i is immediately obtained as the Ω in iteration i.

C. Time Complexity

In Algorithm 2, step 3 takes $O(|\mathcal{L}|)$ time; steps 4 and 5 take $O(h)$ time, where h is the longest path from any leaf to the root. In Algorithm 3, both T and Ω are self-balancing BSTs, whose insert/delete/find-max operations and also the finding of an element all take $O(\log \tilde{V})$ time, where $\tilde{V} \leq |\mathcal{G}|$ is the number of nodes in the BST. Hence, updating the SNV of one supernode by Algorithm 3 takes $O(|\mathcal{G}| \log |\mathcal{G}|)$ time. As $O(|\mathcal{L}|)$ supernodes need to be updated in each iteration of Algorithm 2, this step (which is the most expensive step) takes $O(|\mathcal{L}| \cdot |\mathcal{G}| \log |\mathcal{G}|)$ time. The total time for Algorithm 2 is thus $O(k \cdot |\mathcal{L}| \cdot |\mathcal{G}| \log |\mathcal{G}|)$. When k is unknown, we set $k = |\mathcal{L}|$, and the total time of Algorithm 2 becomes $O(|\mathcal{L}|^2 \cdot |\mathcal{G}| \log |\mathcal{G}|)$.

D. Special Case When $\mathcal{G}$ is a Tree

In the special case where the DAG is just a tree, Algorithm 2 can still be used. Since each supernode then contains exactly one leaf and we have a tree structure, updating of the SNV in step 7 can be simplified. More details on this special case can be found in [20].

However, on comparing with the MAT algorithm in Section III, using MAS on tree-structured label hierarchies has two disadvantages. First, the optimality of MAS relies on the NAP assumption, while MAT does not. Second, MAS requires as input the number of leaf nodes to be predicted. In particular, if k leaf nodes are to be selected, the time complexity of the tree-version of MAS is $O(h^2 k|\mathcal{L}|)$, where h is the height of the tree [20]. When the number of labels is unknown, we can set $k = |\mathcal{L}|$ as in Section IV-B and the complexity becomes $O(h^2|\mathcal{L}|^2)$. If the tree is a full d-ary tree (i.e., every internal node has exactly d children), $|\mathcal{L}| = O(V)$, $h = O(\log V)$, and the complexities above become $O(kV \log^2 V)$ and $O(V^2 \log^2 V)$, respectively. In contrast, MAS can automatically determine the number of labels, and its complexity is only $O(V)$. Hence, when the DAG is simply a tree, it is better to use the MAT algorithm instead of the tree-version of MAS in [20].

V. MLNP That Minimizes the Risk

For a given example $\mathbf{x}$, let $\mathring{\Omega}$ be the true multilabel and Ω the prediction. It is well known that maximizing the posterior probability minimizes the zero-one loss $I(\Omega \neq \mathring{\Omega})$, where $I(\cdot)$ is the indicator function that returns 1 when the argument holds and 0 otherwise. Another loss function popularly used in hierarchical classification is the H-loss, which penalizes only the first classification mistake closest to the root along each prediction path [6]. However, in MLNP, we are more interested in the leaf nodes. Thus, we adopt the symmetric loss $\ell(\Omega, \mathring{\Omega}) = |\Omega \setminus \mathring{\Omega}| + |\mathring{\Omega} \setminus \Omega|$, which counts the number of nodes in the symmetric difference of Ω and $\mathring{\Omega}$. Note that

this weights mistakes in any part of the hierarchy equally. However, in HMC, a mistake that occurs at the higher level of the hierarchy is often considered more crucial [3], [6]. We can incorporate this hierarchy information by extending $\ell(\Omega, \mathring{\Omega})$ as

$$\sum_i c_i I(i \in \Omega \backslash \mathring{\Omega}) + c_i I(i \in \mathring{\Omega} \backslash \Omega). \tag{18}$$

Here, c_i can be set, for example, as in [5]

$$c_i = \begin{cases} 1, & i = 0 \\ c_{\mathrm{pa}(i)}/n_{\mathrm{sibl}(i)}, & i > 0 \end{cases} \tag{19}$$

where $n_{\mathrm{sibl}(i)}$ is the number of siblings of i (including i itself), such that smaller i's (higher level nodes) are assigned larger c_i values. One can also assign different relative importance for the false positives and negatives, and generalize $\ell(\Omega, \mathring{\Omega})$ as

$$\ell(\Omega, \mathring{\Omega}) = \sum_i c_i^+ I(i \in \Omega \backslash \mathring{\Omega}) + c_i^- I(i \in \mathring{\Omega} \backslash \Omega) \tag{20}$$

where for some $\gamma \geq 0$

$$c_i^+ = \frac{2c_i}{1+\gamma}, \quad c_i^- = \frac{2\gamma\, c_i}{1+\gamma}. \tag{21}$$

This setting of c_i^+, c_i^- ensures that the total weight $c_i^+ + c_i^-$ for each i remains the same (equals $2c_i$) as in (18).

From Bayesian decision theory, the optimal multilabel Ω^* is the one that minimizes the expected loss: $\Omega^* = \arg\min_\Omega \sum_{\mathring{\Omega}} \ell(\Omega, \mathring{\Omega})\ p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x})$. The following Proposition shows that it leads to a problem very similar to (11). Note that this Proposition is independent of the manner in which c_i^+, c_i^- are defined. Hence, other settings besides the one in (21) can also be used.

Proposition 4: With a label tree and loss (20) is used, the optimal Ω^* can be obtained by solving problem (11), but with

$$w_i = (c_i^+ + c_i^-)p(y_i = 1|\mathbf{x}) - c_i^+. \tag{22}$$

Proof: Let $\mathcal{M}$ be the set of all possible predictions. Using (20), the expected risk can be rewritten as

$$\begin{aligned} R(\Omega) &= \sum_{\mathring{\Omega} \in \mathcal{M}} \ell(\Omega, \mathring{\Omega}) p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x}) \\ &= \sum_{\mathring{\Omega} \in \mathcal{M}} \sum_{i \in \mathcal{T}} \left(c_i^+ I(i \in \Omega \backslash \mathring{\Omega}) + c_i^- I(i \in \mathring{\Omega} \backslash \Omega) \right) \\ &\quad \times p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x}) \\ &= \sum_{i \in \mathcal{T}} \left(\sum_{\mathring{\Omega} \in \mathcal{M}: i \notin \mathring{\Omega}} c_i^+ I(i \in \Omega) + \sum_{\mathring{\Omega} \in \mathcal{M}: i \in \mathring{\Omega}} c_i^- I(i \notin \Omega) \right) \\ &\quad \times p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x}) \\ &= \sum_{i \in \Omega} c_i^+ \sum_{\mathring{\Omega} \in \mathcal{M}: i \notin \mathring{\Omega}} p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x}) \\ &\quad + \sum_{i \notin \Omega} c_i^- \sum_{\mathring{\Omega} \in \mathcal{M}: i \in \mathring{\Omega}} p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x}). \end{aligned}$$

Note that $\sum_{\mathring{\Omega} \in \mathcal{M}: i \in \mathring{\Omega}} p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x})$ is simply the marginal probability that i is predicted positive, i.e., $p(y_i = 1|\mathbf{x})$. Using the conditional independence assumption, $p(y_i = 1|\mathbf{x}) = \prod_{j \in \mathrm{anc}(i) \cup \{i\} \backslash \{0\}} p(y_j = 1|y_{\mathrm{pa}(j)} = 1, \mathbf{x})$. Similarly, $\sum_{\mathring{\Omega} \in \mathcal{M}: i \notin \mathring{\Omega}} p(\mathbf{y}_{\mathring{\Omega}} = \mathbf{1}, \mathbf{y}_{\mathring{\Omega}^c} = \mathbf{0}|\mathbf{x})$ is the marginal probability that i is predicted negative, i.e., $p(y_i = 0|\mathbf{x}) = 1 - p(y_i = 1|\mathbf{x})$. Thus, $R(\Omega)$ can be rewritten as $R(\Omega) = \sum_{i \in \Omega} c_i^+\ p(y_i = 0|\mathbf{x}) + \sum_{i \notin \Omega} c_i^-\ p(y_i = 1|\mathbf{x})$. Introduce an indicator variable ψ_i for each node. Then

$$\begin{aligned} R(\Omega) &= \sum_{i \in \mathcal{T}} c_i^+ \psi_i\ (1 - p(y_i = 1|\mathbf{x})) \\ &\quad + \sum_{i \in \mathcal{T}} c_i^- (1 - \psi_i)\ p(y_i = 1|\mathbf{x}) \\ &= \sum_{i \in \mathcal{T}} \psi_i\ (c_i^+ - (c_i^+ + c_i^-) p(y_i = 1|\mathbf{x})) \\ &\quad + c_i^- p(y_i = 1|\mathbf{x}) \end{aligned}$$

$$\begin{aligned} &\arg\min_\Omega R(\Omega) \\ &\quad = \arg \min_{\{\psi_i\}_{i \in \mathcal{T}}} \sum_{i \in \mathcal{T}} \psi_i\ (c_i^+ - (c_i^+ + c_i^-) p(y_i = 1|\mathbf{x})) \\ &\qquad + c_i^- p(y_i = 1|\mathbf{x}) \\ &\quad = \arg \max_{\{\psi_i\}_{i \in \mathcal{T}}} \sum_{i \in \mathcal{T}} ((c_i^+ + c_i^-) p(y_i = 1|\mathbf{x}) - c_i^+) \psi_i. \end{aligned}$$

The other constraints on Ω in Proposition 1 are for enforcing that Ω is an MLNP, and thus are still needed. ■

Extension to a label DAG is analogous. For example, c_i can be defined as 1 when $i = 0$ and $\sum_{j \in \mathrm{Pa}(i)} c_j / n_{\mathrm{child}(j)}$ otherwise. On using (13), the proof in Proposition 4 still holds for label DAGs. Thus, we can use (14), with w_i in (22), to obtain the MLNP that minimizes the given risk.

VI. Experiments

In this section, we evaluate the performance of the proposed MAT and MAS algorithms, together with their risk minimizing versions discussed in Section V.

A. Setup

Experiments are performed on a number of benchmark multilabel data sets with tree- and DAG-structured label hierarchies (Tables II and III).[2]

1) *Five Subsets of Rcv1 [33]:* These contain REUTERS documents, with the label hierarchy coming from the LYRL2004 distribution.
2) *Delicious [34]:* It contains the textual data of web pages from the del.icio.us social bookmarking site, along with their tags. The label (tag) hierarchy is from [35].

[2] rcv1 is from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html, delicious from http://mulan.sourceforge.net/datasets.html, enron from http://bailando.sims.berkeley.edu/enron_email.html, wipo from http://www.wipo.int/classifications/ipc/en/ITsupport/Categorization/dataset/index.html, caltech101 from http://www.vision.caltech.edu/Image_Datasets/Caltech101/, caltech256 from http://www.vision.caltech.edu/Image_Datasets/Caltech256/, and dmoz from http://lib.iit.demokritos.gr/LSHTC2_datasets.

TABLE II
DATA SETS WITH TREE-STRUCTURED LABEL HIERARCHIES

data set	#examples	#features	total #nodes	#leafs	avg #leafs / example
rcv1 (subset1)	4,422	47,236	57	42	1.3
rcv1 (subset2)	4,485	47,236	59	43	1.3
rcv1 (subset3)	4,513	47,236	63	46	1.3
rcv1 (subset4)	4,569	47,236	59	44	1.3
rcv1 (subset5)	4,452	47,236	60	45	1.4
delicious	768	500	72	49	5.4
enron	1,607	1,001	29	24	2.6
wipo	569	74,435	38	21	1.0
caltech-101	9,144	21,504	143	102	1.0
caltech-256	29,780	21,504	319	256	1.0
dmoz	394,765	829,208	35,448	27,875	1.0
seq (funcat)	1,115	489	68	36	1.8
pheno (funcat)	330	170	27	14	1.6
church (funcat)	1,104	26	67	35	1.8
eisen (funcat)	768	79	51	29	1.8
struc (funcat)	1,065	19,629	65	33	1.8
hom (funcat)	1,124	47,035	64	35	1.8
cellcycle (funcat)	1,080	77	43	33	1.9
derisi (funcat)	995	63	69	33	1.8
gasch1 (funcat)	1,038	173	66	32	1.8
gasch2 (funcat)	1,076	52	66	33	1.8
spo (funcat)	1,053	83	60	32	1.8
expr (funcat)	1,109	551	62	32	1.8

TABLE III
DATA SETS WITH DAG-STRUCTURED LABEL HIERARCHIES

data set	#examples	#features	total #nodes	#leafs	avg #leafs / example
seq	518	489	82	32	3.6
pheno	227	170	47	19	3.5
church	511	26	85	28	3.2
eisen	404	79	72	28	3.4
struc	505	19,629	91	33	3.5
hom	507	47,035	82	29	3.2
cellcycle	484	77	84	29	3.1
derisi	492	63	84	31	3.4
gasch1	512	173	87	32	3.4
gasch2	508	52	86	32	3.3
spo	494	83	90	32	3.3
expr	504	551	93	35	3.5

3) *Enron [36]:* This is a text data set for email analysis and the label hierarchy was developed in UC Berkeley.
4) *Wipo:* this data set consists of text documents from the D section of WIPO-alpha patent hierarchy.
5) *Caltech101 and Caltech256 [37]:* These are image data sets for object annotation. The label hierarchy for caltech256 is from [38], of which caltech101 is a subset.
6) *Dmoz:* It is the training data set used in the second LSHTC, which contains web pages from the open directory project. The label hierarchy is provided by the challenge.
7) *Twelve Genomics Data Sets [1]:* These contain different aspects of genes in the yeast genome, with tree-structured annotations from the MIPSs Funcat and DAG-structured from the GO.[3]

All these data sets, except those on genomics, are for MLNP and have been commonly used for evaluating flat [39], [40] and hierarchical multilabel learning algorithms [13], [16]. For the genomics data sets, we follow the paper on HMC-LP [13] (which is the only existing MLNP algorithm for HMC) and turn them into MLNP data sets by removing examples that contain partial label paths.

We will compare the following algorithms.

1) *Algorithms That Are Designed for MLNP:* 1) the proposed MAT (Algorithm 1); 2) the proposed MAS (Algorithm 2); and 3) HMC-LP [13]. At each parent node, we train a multitask lasso model with logistic loss using the MALSAR package [41] and obtain the probability estimates for MAT and MAS. In addition, recall that MAT and HMC-LP can only be used on trees, while MAS can be used for both trees and DAGs.
2) A*lgorithms That Are Originally Designed for NMLNP, but Are Extended for MNLP Using MetaLabeler [25]:*[4] 1) HBR: this is modified from the hierarchical classifier H-SVM [5], by replacing its base learner from SVM to multitask lasso (as for MAT and MAS) and 2) CLUS-HMC [1], a decision-tree based classifier.
3) *Flat BR [23]:* as discussed in Section II, BR is originally for flat multilabel classification. Here, at each leaf node, we train an independent lasso model with the logistic loss to obtain the probability estimates, which is then thresholded to obtain the binary labels. We avoid the threshold selection problem by instead using the MetaLabeler to determine the number of leaf nodes that should be assigned.

All these are implemented in MATLAB2009 (except for CLUS-HMC, which is from the authors of [1][5] and in JAVA).

As in other hierarchical classification algorithms [13], [42], we try to obtain more accurate probability estimators by removing label nodes with fewer than ten positive examples. Alternatively, this data-sparsity problem might be mitigated with more powerful probability estimators or smoothing techniques. In addition, this is not performed on the dmoz data, as 72% of its leaf nodes have fewer than four positive examples.

For performance evaluation, we use the hierarchical F-measure (HF), which has been commonly used in hierarchical classification [8]. Let $\boldsymbol{\Omega}(\mathbf{x})$ be the predicted multilabel for example $\mathbf{x}$, $\mathring{\boldsymbol{\Omega}}(\mathbf{x})$ the ground truth, leaf(A) contains the leaf nodes in the set A, and anc$^+(i)$ the set containing i and i's ancestors. HF is defined as $\sum_{\mathbf{x}} 2\mathrm{HP}(\mathbf{x}) \cdot \mathrm{HR}(\mathbf{x})/\mathrm{HP}(\mathbf{x}) + \mathrm{HR}(\mathbf{x})$, where the sum is over all the test examples, and

$$\mathrm{HP}(\mathbf{x}) = \frac{1}{|\mathrm{leaf}(\boldsymbol{\Omega}(\mathbf{x}))|} \sum_{i \in \mathrm{leaf}(\boldsymbol{\Omega}(\mathbf{x}))} \frac{|\mathring{\boldsymbol{\Omega}}(\mathbf{x}) \cap \mathrm{anc}^+(i)|}{|\mathrm{anc}^+(i)|}$$

$$\mathrm{HR}(\mathbf{x}) = \frac{1}{|\mathrm{leaf}(\mathring{\boldsymbol{\Omega}}(\mathbf{x}))|} \sum_{i \in \mathrm{leaf}(\mathring{\boldsymbol{\Omega}}(\mathbf{x}))} \frac{|\boldsymbol{\Omega}(\mathbf{x}) \cap \mathrm{anc}^+(i)|}{|\mathrm{anc}^+(i)|}$$

[3]Available at: http://dtai.cs.kuleuven.be/clus/hmcdatasets/

[4]As discussed in Section II, for each test example, the MetaLabeler predicts the number (k) of leaf nodes that should be assigned. The NMLNP algorithm (such as HBR and CLUS-HMC) is run, and the k leaf nodes with the largest probabilities of being positive are output in the final MLNP prediction.

[5]Available at: http://dtai.cs.kuleuven.be/clus/

TABLE IV
HF VALUES ON DATA SETS WITH TREE HIERARCHIES. HMC-LP AND CLUS-HMC CANNOT BE RUN ON THE CALTECH101, CALTECH256, DMOZ DATA, WHICH ARE LARGE AND DENSE

data set	MAT	MAS(tree)	HMC-LP	HBR (w/ MetaLabeler)	CLUS-HMC (w/ MetaLabeler)	flat BR (w/ MetaLabeler)
rcv1v2 subset1	**0.849** ± 0.008	**0.849** ± 0.008	0.215 ± 0.008	0.833 ± 0.010	0.631 ± 0.015	0.828 ± 0.006
rcv1v2 subset2	**0.853** ± 0.008	**0.853** ± 0.008	0.211 ± 0.014	0.835 ± 0.008	0.637 ± 0.030	0.839 ± 0.005
rcv1v2 subset3	0.846 ± 0.006	**0.847** ± 0.006	0.196 ± 0.012	0.832 ± 0.002	0.626 ± 0.010	0.832 ± 0.006
rcv1v2 subset4	**0.855** ± 0.009	**0.855** ± 0.009	0.211 ± 0.016	0.836 ± 0.009	0.635 ± 0.007	0.842 ± 0.013
rcv1v2 subset5	**0.843** ± 0.009	**0.843** ± 0.008	0.206 ± 0.011	0.825 ± 0.009	0.630 ± 0.011	0.828 ± 0.009
delicious	0.525 ± 0.013	0.525 ± 0.021	0.230 ± 0.032	0.281 ± 0.024	**0.571** ± 0.020	0.542 ± 0.018
enron	**0.749** ± 0.008	**0.749** ± 0.008	0.715 ± 0.018	0.736 ± 0.008	0.683 ± 0.014	0.735 ± 0.009
wipo	**0.831** ± 0.019	**0.831** ± 0.019	0.418 ± 0.013	0.830 ± 0.019	0.708 ± 0.029	**0.831** ± 0.015
caltech-101	0.815 ± 0.012	0.815 ± 0.011	-	**0.824** ± 0.011	-	0.695 ± 0.009
caltech-256	0.387 ± 0.012	0.387 ± 0.011	-	**0.394** ± 0.011	-	0.356 ± 0.009
dmoz	0.280 ± 0.008	0.280 ± 0.008	-	**0.288** ± 0.004	-	0.204 ± 0.002
seq (funcat)	**0.263** ± 0.020	0.258 ± 0.017	0.151 ± 0.076	0.253 ± 0.028	0.262 ± 0.008	0.228 ± 0.020
pheno (funcat)	**0.256** ± 0.040	0.251 ± 0.039	0.121 ± 0.048	0.253 ± 0.021	0.197 ± 0.046	0.234 ± 0.041
struc (funcat)	0.230 ± 0.022	0.230 ± 0.023	0.030 ± 0.013	**0.247** ± 0.020	0.214 ± 0.015	0.238 ± 0.020
hom (funcat)	0.345 ± 0.022	0.345 ± 0.022	0.210 ± 0.055	**0.362** ± 0.023	0.274 ± 0.045	0.360 ± 0.055
cellcycle (funcat)	**0.224** ± 0.022	0.203 ± 0.024	0.122 ± 0.027	0.211 ± 0.023	0.186 ± 0.023	**0.194** ± 0.032
church (funcat)	0.172 ± 0.014	0.177 ± 0.015	0.045 ± 0.056	0.177 ± 0.008	**0.198** ± 0.023	0.170 ± 0.012
derisi (funcat)	0.181 ± 0.019	0.181 ± 0.015	0.077 ± 0.040	0.181 ± 0.020	**0.207** ± 0.012	0.181 ± 0.007
eisen (funcat)	**0.300** ± 0.023	0.281 ± 0.020	0.104 ± 0.007	0.285 ± 0.024	0.279 ± 0.025	0.274 ± 0.029
gasch1 (funcat)	0.261 ± 0.020	0.251 ± 0.022	0.110 ± 0.058	0.229 ± 0.030	**0.287** ± 0.018	0.224 ± 0.018
gasch2 (funcat)	0.227 ± 0.016	0.241 ± 0.010	0.052 ± 0.008	0.221 ± 0.023	**0.251** ± 0.037	0.207 ± 0.024
spo (funcat)	0.180 ± 0.014	0.180 ± 0.018	0.102 ± 0.010	0.178 ± 0.024	**0.228** ± 0.035	0.176 ± 0.010
expr (funcat)	**0.283** ± 0.020	0.282 ± 0.017	0.123 ± 0.009	0.253 ± 0.028	0.247 ± 0.019	0.270 ± 0.010

are hierarchical extensions of the standard precision and recall measures. Besides, timing comparison will also be performed. As the training procedures of MAT, MAS, and HBR are the same, we will focus on comparing their prediction time. Results are based on fivefold cross-validation on all data sets except dmoz, which is large. Instead, we perform random splitting of the dmoz data three times, and report the average. All experiments are run on a PC with 2.80-GHz Intel i7-860 CPU and 16-GB RAM.

B. Maximum a Posteriori MLNP on Tree Hierarchies

The HF results for data sets with tree-structured hierarchies are shown in Table IV. As can be seen, MAT and MAS(tree), which solve the same optimization problem, yield similar HF values. On 12 of the 23 data sets, they are the best or comparable with the best. However, as discussed in Section IV-D and will be seen in the next paragraph, MAT is much faster than MAS(tree). On the other hand, HMC-LP performs poorly. This is because HMC-LP has to train a classifier for every label combination at each level of the hierarchy. As each label combination contains very few positive training examples, the resultant classifiers are not accurate.

For statistical significance testing of the various methods (over the multiple data sets), we use the Friedman test with post hoc Shaffer's static procedure [43]. Results are visualized using the graphical representation in [44] [Fig. 3(a)]. The top horizontal line indicates the average rank. Methods are placed so that those with lower (better) ranks are placed on the right, and methods that are not significantly different are connected together by a bold line. As can be observed from Fig. 3(a), the proposed method is significantly better than HMC-LP, CLUS-HMC, and BR at a significance level of 0.05.

Table V shows the prediction time results. As can be seen, MAT is consistently the most efficient. However, though the Friedman test (with post hoc Shaffer's static procedure) has been advocated for comparing multiple methods over multiple data sets, it can be too conservative unless the number of algorithms to be compared is very small[6]. Hence, in computing the significance tests, we remove the two worst algorithms, HMC-LP and CLUS-HMC. As can be observed from Fig. 3, the speed improvement of MAT is statistically significant.

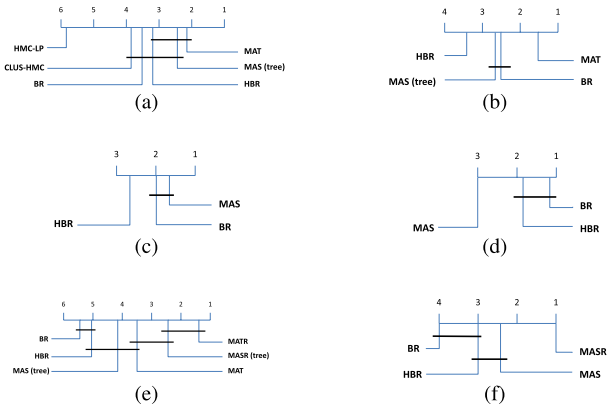

Fig. 3. Visualization of statistical significance for results from Tables IV–IX. (a) Results for Table IV. (b) Results for Table V (with HMC-LP and CLUS-HMC removed). (c) Results for Table VI (with HMC-CLUS removed). (d) Results for Table VII (with HMC-CLUS removed). (e) Results for Table VIII (with HMC-LP and CLUS-HMC removed). (f) Results for Table IX (with CLUS-HMC removed).

C. Maximum a Posteriori MLNP on DAG Hierarchies

For data sets with DAG-structured hierarchies, Tables VI and VII show the comparisons with respect to the HF and prediction time, respectively. Recall that HMC-LP cannot

[6]For example, suppose that we have 5 algorithms with consistent performance ranks over all data sets, the Friedman test will still require 32 data sets to accept the hypothesis that the performance difference between the best and second-best algorithms is statistically significant.

TABLE V
PREDICTION TIME (IN SECONDS) ON DATA SETS WITH TREE HIERARCHIES

data set	MAT	MAS(tree)	HMC-LP	HBR (w/ MetaLabeler)	CLUS-HMC (w/ MetaLabeler)	flat BR (w/ MetaLabeler)
rcv1v2 subset1	**0.065** ± 0.012	0.362 ± 0.037	3.628 ± 0.079	1.299 ± 0.026	319.613 ± 35.080	1.299 ± 0.031
rcv1v2 subset2	**0.078** ± 0.012	0.375 ± 0.028	3.914 ± 0.303	1.382 ± 0.022	310.893 ± 54.575	1.331 ± 0.026
rcv1v2 subset3	**0.081** ± 0.014	0.367 ± 0.027	3.564 ± 0.070	1.661 ± 0.042	356.664 ± 39.118	1.655 ± 0.039
rcv1v2 subset4	**0.073** ± 0.015	0.339 ± 0.018	3.800 ± 0.208	1.620 ± 0.192	339.550 ± 17.317	1.621 ± 0.192
rcv1v2 subset5	**0.068** ± 0.008	0.359 ± 0.040	3.709 ± 0.410	1.411 ± 0.029	307.695 ± 8.563	1.382 ± 0.034
delicious	**0.026** ± 0.002	0.065 ± 0.090	0.355 ± 0.042	0.167 ± 0.010	0.265 ± 0.058	0.168 ± 0.022
enron	**0.105** ± 0.002	0.114 ± 0.020	2.751 ± 0.139	0.947 ± 0.050	1.497 ± 0.176	0.943 ± 0.052
wipo	**0.018** ± 0.004	0.097 ± 0.009	2.975 ± 0.109	0.068 ± 0.006	60.200 ± 4.368	0.043 ± 0.006
caltech-101	0.452 ± 0.018	**0.430** ± 0.011	-	0.888 ± 0.039	-	0.849 ± 0.030
caltech-256	**0.670** ± 0.010	**0.670** ± 0.010	-	1.212 ± 0.029	-	1.210 ± 0.029
dmoz	0.452 ± 0.018	**0.430** ± 0.011	-	0.888 ± 0.039	-	0.849 ± 0.030
seq (funcat)	**0.076** ± 0.008	0.131 ± 0.014	0.667 ± 0.051	0.382 ± 0.021	1.498 ± 0.154	0.381 ± 0.020
pheno (funcat)	0.003 ± 0.000	0.030 ± 0.005	0.033 ± 0.002	0.025 ± 0.004	**0.001** ± 0.000	0.024 ± 0.003
struc (funcat)	**0.117** ± 0.013	0.119 ± 0.008	16.728 ± 0.952	14.577 ± 0.151	16.583 ± 1.081	14.563 ± 0.155
hom (funcat)	**0.129** ± 0.003	0.171 ± 0.047	37.596 ± 1.852	19.620 ± 1.949	52.089 ± 5.372	19.610 ± 1.094
cellcycle (funcat)	**0.061** ± 0.016	0.130 ± 0.012	0.206 ± 0.015	0.079 ± 0.004	0.530 ± 0.040	0.077 ± 0.008
church (funcat)	0.074 ± 0.006	0.123 ± 0.014	0.167 ± 0.020	0.064 ± 0.014	**0.016** ± 0.004	0.053 ± 0.007
derisi (funcat)	0.067 ± 0.008	0.120 ± 0.009	0.163 ± 0.008	0.077 ± 0.009	0.251 ± 0010	**0.066** ± 0.009
eisen (funcat)	**0.031** ± 0.007	0.075 ± 0.010	0.083 ± 0.015	0.071 ± 0.062	0.078 ± 0.005	0.037 ± 0.001
gasch1 (funcat)	**0.055** ± 0.006	0.129 ± 0.006	0.296 ± 0.044	0.132 ± 0.006	0.281 ± 0.044	0.116 ± 0.008
gasch2 (funcat)	0.127 ± 0.009	0.132 ± 0.013	0.188 ± 0.015	0.073 ± 0.006	0.234 ± 0.105	**0.065** ± 0.007
spo (funcat)	0.093 ± 0.011	0.125 ± 0.020	0.185 ± 0.028	0.091 ± 0.015	0.109 ± 0.031	**0.083** ± 0.017
expr (funcat)	**0.062** ± 0.010	0.123 ± 0.015	0.706 ± 0.055	0.287 ± 0.015	2.434 ± 0.772	0.279 ± 0.016

TABLE VI
HF VALUES ON DATA SETS WITH DAG HIERARCHIES

data set	MAS	HBR (w/ MetaLabeler)	CLUS-HMC (w/ MetaLabeler)	flat BR (w/ MetaLabeler)
seq	**0.736** ± 0.016	0.575 ± 0.019	0.591 ± 0.050	0.611 ± 0.022
pheno	**0.567** ± 0.014	0.527 ± 0.031	0.486 ± 0.058	0.546 ± 0.039
struc	0.505 ± 0.015	0.475 ± 0.025	**0.550** ± 0.070	0.528 ± 0.037
hom	**0.651** ± 0.019	0.596 ± 0.040	0.591 ± 0.014	0.634 ± 0.073
cellcycle	0.494 ± 0.025	0.492 ± 0.011	**0.508** ± 0.057	0.506 ± 0.010
church	**0.572** ± 0.027	0.497 ± 0.007	0.530 ± 0.020	0.540 ± 0.011
derisi	**0.560** ± 0.030	0.485 ± 0.036	0.527 ± 0.016	0.540 ± 0.036
eisen	0.482 ± 0.022	0.544 ± 0.046	**0.567** ± 0.021	**0.567** ± 0.032
gasch1	**0.635** ± 0.013	0.563 ± 0.024	0.568 ± 0.022	0.583 ± 0.019
gasch2	**0.549** ± 0.014	0.503 ± 0.023	0.513 ± 0.016	0.532 ± 0.013
spo	0.504 ± 0.016	0.471 ± 0.019	0.490 ± 0.036	**0.511** ± 0.022
expr	**0.719** ± 0.029	0.566 ± 0.048	0.550 ± 0.028	0.601 ± 0.055

TABLE VII
PREDICTION TIME (IN SECONDS) ON DATA SETS WITH DAG HIERARCHIES

data set	MAS	HBR (w/ MetaLabeler)	CLUS-HMC (w/ MetaLabeler)	flat BR (w/ MetaLabeler)
seq	0.403±0.093	0.120±0.027	1.217±0.129	**0.107**±0.023
pheno	0.618±0.027	**0.023**±0.006	0.055±0.006	0.024±0.010
struc	0.887±0.268	0.067±0.703	10.608±0.167	**0.058**±0.712
hom	0.662±0.070	**0.573**±1.262	29.883±3.166	**0.573**±1.265
cellcycle	0.561±0.023	0.090±0.012	0.299±0.020	**0.083**±0.015
church	0.590±0.035	0.066±0.009	0.257±0.001	**0.055**±0.001
derisi	0.623±0.012	0.077±0.029	0.272±0.048	**0.072**±0.031
eisen	0.500±0.039	0.046±0.010	0.120±0.006	**0.032**±0.008
gasch1	0.694±0.100	0.094±0.014	0.422±0.110	**0.084**±0.014
gasch2	0.418±0.054	0.043±0.018	0.204±0.028	**0.032**±0.023
spo	0.654±0.026	0.073±0.016	0.183±0.004	**0.069**±0.015
expr	0.408±0.021	0.129±0.019	1.151±0.054	**0.117**±0.019

be used on DAG hierarchies and so is not compared here. In addition, since there are only 12 data sets, we remove the worst algorithm CLUS-HMC in the statistical significance tests [Fig. 3(c) and (d)]. Similar to the results in Section VI-B, MAS is more accurate than the other methods overall. It is also slower, though still fast enough for practical applications. It is also much faster than the brute-force approach of solving problem (5), which again takes more than an hour to predict one example, and its results are not reported.

D. MLNP That Minimizes Risk

In this section, we study the performance of the risk-minimizing versions in Section V, which are denoted MATR and MASR, respectively, We set γ, the relative importance of false positives versus false negatives in (21), to be the ratio of the numbers of negative and positive training labels. Results on the loss values for data sets with tree-structured and DAG-structured label hierarchies are shown in Tables VIII and IX, respectively. As for the statistical significance tests, we again remove the two worst algorithms HMC-LP and CLUS-HMC [Fig. 3(e) and (f)]. In addition, note that HMC-LP, HBR, CLUS-HMC, and BR cannot perform risk-minimizing predictions. Thus, predictions for these methods are the same as those in Tables IV and VI and only that different performance measures are used (symmetric loss instead of HF). As for the prediction time results, they are very similar to those reported in Section VI-B and C and so are not repeated here.

Fig. 4 illustrates example query images and their misclassifications by MAT, MATR, and BR on the caltech101 data set. As can be seen, even when MAT/MATR misclassifies the image, the hierarchy often helps to keep the prediction close to the true label.

E. Validating the NAP Assumption

In this section, we verify the validity of the NAP assumption used in MAS. For each test example, we use brute-force search to find its best k-leaf-sparse prediction, and check if it includes the best $(k-1)$-leaf-sparse prediction. As brute-force search is very expensive, experiments are only performed on four smaller data sets. Fig. 5 shows the percentage of test examples satisfying the NAP assumption at different values of k. As can be seen, the NAP holds almost 100% of the time.

TABLE VIII

HIERARCHICALLY WEIGHTED SYMMETRIC LOSS VALUES (20) ON DATA SETS WITH TREE-STRUCTURED LABEL HIERARCHIES

data set	MATR	MASR(tree)	MAT	MAS (tree)	HMC-LP	HBR (w/ MetaLabeler)	CLUS-HMC (w/ MetaLabeler)	flat BR (w/ MetaLabeler)
rcv1v2 subset1	**0.045** ± 0.003	0.046 ± 0.002	0.099 ± 0.012	0.098 ± 0.012	0.460 ± 0.007	0.118 ± 0.010	0.204 ± 0.007	0.125 ± 0.006
rcv1v2 subset2	**0.041** ± 0.003	**0.041** ± 0.004	0.092 ± 0.011	0.092 ± 0.011	0.454 ± 0.021	0.108 ± 0.012	0.195 ± 0.016	0.117 ± 0.010
rcv1v2 subset3	0.044 ± 0.003	**0.042** ± 0.003	0.097 ± 0.002	0.092 ± 0.001	0.452 ± 0.009	0.114 ± 0.003	0.200 ± 0.006	0.117 ± 0.003
rcv1v2 subset4	0.045 ± 0.003	**0.044** ± 0.003	0.090 ± 0.006	0.096 ± 0.006	0.444 ± 0.014	0.106 ± 0.006	0.193 ± 0.004	0.113 ± 0.007
rcv1v2 subset5	**0.044** ± 0.004	0.046 ± 0.004	0.097 ± 0.006	0.097 ± 0.006	0.455 ± 0.011	0.113 ± 0.007	0.199 ± 0.007	0.121 ± 0.006
delicious	**0.122** ± 0.010	0.231 ± 0.016	0.137 ± 0.010	0.189 ± 0.018	0.226 ± 0.026	0.137 ± 0.012	0.125 ± 0.010	0.135 ± 0.013
enron	0.273 ± 0.012	0.305 ± 0.012	0.315 ± 0.005	0.357 ± 0.017	**0.251** ± 0.010	0.353 ± 0.005	0.409 ± 0.013	0.346 ± 0.010
wipo	**0.073** ± 0.008	**0.073** ± 0.008	0.089 ± 0.008	0.089 ± 0.008	0.343 ± 0.009	0.091 ± 0.008	0.164 ± 0.022	0.092 ± 0.010
caltech101	**0.002** ± 0.000	**0.003** ± 0.000	0.008 ± 0.000	0.008 ± 0.000	-	0.008 ± 0.000	-	0.008 ± 0.000
caltech256	**0.013** ± 0.000	**0.013** ± 0.000	0.042 ± 0.000	0.042 ± 0.000	-	0.043 ± 0.000	-	0.045 ± 0.000
dmoz	**0.000** ± 0.000	**0.000** ± 0.000	**0.000** ± 0.000	**0.000** ± 0.000	-	0.001 ± 0.000	-	0.003 ± 0.000
seq	**0.221** ± 0.008	0.239 ± 0.006	0.249 ± 0.008	0.256 ± 0.006	0.409 ± 0.008	0.380 ± 0.027	0.375 ± 0.002	0.405 ± 0.026
pheno	0.391 ± 0.016	0.387 ± 0.020	0.381 ± 0.023	0.382 ± 0.024	0.612 ± 0.030	**0.380** ± 0.026	0.551 ± 0.026	0.406 ± 0.015
struc	**0.294** ± 0.015	**0.294** ± 0.011	0.387 ± 0.018	0.387 ± 0.017	0.419 ± 0.016	0.889 ± 0.016	0.407 ± 0.020	0.400 ± 0.011
hom	**0.316** ± 0.021	**0.316** ± 0.021	0.355 ± 0.019	0.355 ± 0.019	0.367 ± 0.010	0.363 ± 0.019	0.342 ± 0.015	0.318 ± 0.010
cellcycle	**0.169** ± 0.016	0.236 ± 0.020	0.281 ± 0.017	0.290 ± 0.018	0.405 ± 0.015	0.289 ± 0.019	0.384 ± 0.018	0.298 ± 0.015
church	**0.170** ± 0.010	0.255 ± 0.010	0.302 ± 0.008	0.303 ± 0.007	0.421 ± 0.007	0.298 ± 0.008	0.410 ± 0.013	0.305 ± 0.010
derisi	**0.168** ± 0.006	0.260 ± 0.012	0.295 ± 0.012	0.300 ± 0.011	0.448 ± 0.013	0.297 ± 0.013	0.433 ± 0.018	0.304 ± 0.015
eisen	**0.233** ± 0.020	0.302 ± 0.025	0.351 ± 0.029	0.364 ± 0.027	0.385 ± 0.040	0.375 ± 0.029	0.362 ± 0.033	0.384 ± 0.041
gasch1	**0.192** ± 0.013	0.236 ± 0.011	0.263 ± 0.012	0.270 ± 0.012	0.430 ± 0.011	0.289 ± 0.011	0.394 ± 0.017	0.294 ± 0.010
gasch2	**0.202** ± 0.006	0.303 ± 0.008	0.239 ± 0.015	0.274 ± 0.013	0.422 ± 0.008	0.293 ± 0.014	0.391 ± 0.023	0.289 ± 0.013
spo	**0.211** ± 0.022	0.311 ± 0.023	0.275 ± 0.017	0.294 ± 0.008	0.415 ± 0.010	0.301 ± 0.011	0.400 ± 0.021	0.300 ± 0.010
expr	**0.221** ± 0.007	0.238 ± 0.007	0.251 ± 0.007	0.255 ± 0.008	0.407 ± 0.008	0.282 ± 0.010	0.386 ± 0.016	0.279 ± 0.010

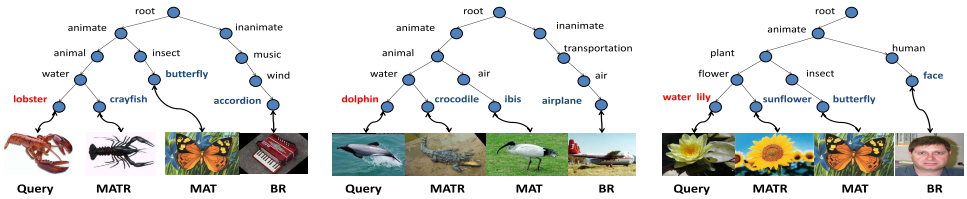

Fig. 4. Example misclassifications on the Caltech101 data set for the MATR, MAT, and BR methods.

TABLE IX

HIERARCHICALLY WEIGHTED SYMMETRIC LOSS VALUES (20) ON DATA SETS WITH DAG-STRUCTURED LABEL HIERARCHIES

data set	MASR	MAS	HBR (w/ MetaLabeler)	CLUS-HMC (w/ MetaLabeler)	flat BR (w/ MetaLabeler)
seq	**0.651**±0.078	0.937±0.193	0.966±0.106	2.316±0.371	1.045±0.075
pheno	**1.735**±0.133	1.964±0.159	2.288±0.132	3.537±0.339	2.391±0.074
struc	**1.892**±0.067	2.111±0.094	2.525±0.090	3.004±0.393	3.149±0.232
hom	**1.627**±0.164	1.716±0.143	2.360±0.149	2.453±0.192	2.307±0.296
cellcycle	**2.146**±0.067	2.283±0.113	2.514±0.065	3.245±0.595	2.554±0.165
church	**1.591**±0.134	1.802±0.074	2.263±0.103	2.802±0.324	2.307±0.063
derisi	**1.531**±0.106	1.532±0.048	2.056±0.089	2.899±0.104	3.107±0.057
eisen	**1.760**±0.034	2.208±0.078	2.228±0.096	3.096±0.275	2.251±0.088
gasch1	**1.563**±0.147	1.669±0.062	2.300±0.089	2.437±0.187	2.455±0.211
gasch2	**0.772**±0.155	1.437±0.145	1.180±0.149	3.135±0.124	1.111±0.187
spo	**1.588**±0.146	1.982±0.144	2.760±0.119	3.404±0.279	3.449±0.132
expr	**0.632**±0.125	1.051±0.219	0.821±0.207	2.687±0.206	0.961±0.313

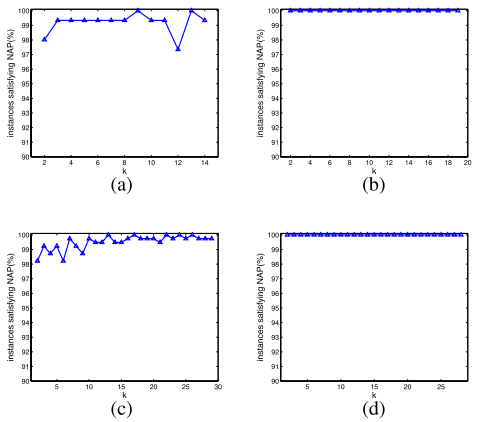

Fig. 5. Percentage of examples satisfying the NAP assumption at different values of k. (a) Pheno (funcat). (b) Pheno (GO). (c) Eisen (funcat). (d) Eisen (GO).

VII. CONCLUSION

In this paper, we proposed a novel HMC algorithm for MLNP. Unlike many hierarchical multilabel/multiclass classification algorithms, it utilizes the global hierarchy information by finding the multilabel with the largest posterior probability over all the node labels. For label trees, we showed that this can be efficiently optimized by a simple dynamic programming algorithm (MAT). For label DAGs, by adopting a weak nested approximation assumption which is already implicitly assumed in many HMC algorithms, the optimization problem can again be efficiently solved using a simple greedy algorithm (MAS). In addition, it can be extended to minimize the risk associated with the (hierarchically weighted) symmetric loss. Experiments performed on a number of real-world data sets demonstrate that the proposed algorithms are computationally simple and more accurate than existing HMC and flat multilabel classification methods.

In this paper, we assumed that the probability estimators $p(y_i = 1 \mid y_{\mathrm{pa}(i)} = 1, \mathbf{x})$ are accurate. This may be problematic when training data are limited. One future direction is to analyze how this estimation error affects the classification performance. In addition, here we have only considered minimizing the expected risk with respect to the weighted symmetric loss. Extension to some other losses will be explored in the future.

REFERENCES

[1] C. Vens, J. Struyf, L. Schietgat, S. Dvzeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Mach. Learn.*, vol. 73, no. 2, pp. 185–214, 2008.

[2] J. Burred and A. Lerch, "A hierarchical approach to automatic musical genre classification," in *Proc. 6th Int. Conf. Digit. Audio Effects*, London, U.K., Sep. 2003.

[3] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *J. Mach. Learn. Res.*, vol. 7, pp. 1601–1626, Dec. 2006.

[4] W. Bi and J.-T. Kwok, "Multi-label classification on tree- and DAG-structured hierarchies," in *Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, WA, USA, 2011, pp. 17–24.

[5] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Incremental algorithms for hierarchical classification," *J. Mach. Learn. Res.*, vol. 7, pp. 31–54, Dec. 2006.
[6] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical classification: Combining Bayes with SVM," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, 2006, pp. 177–184.
[7] Z. Barutcuoglu and O. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.
[8] C. Silla and A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining Knowl. Discovery*, vol. 22, no. 1, pp. 31–72, 2011.
[9] K. Punera, S. Rajan, and J. Ghosh, "Automatically learning document taxonomies for hierarchical classification," in *Proc. 14th Int. Conf. World Wide Web*, Chiba, Japan, 2005, pp. 1010–1011.
[10] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. 16th Int. Conf. Knowl. Discovery Data Mining*, Washington, DC, USA, 2010, pp. 999–1008.
[11] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Proc. Adv. NIPS*, vol. 23. 2010, pp. 163–171.
[12] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *Proc. Adv. NIPS*, vol. 24. 2011, pp. 567–575.
[13] R. Cerri, A. C. P. L. F. de Carvalho, and A. A. Freitas, "Adapting non-hierarchical multilabel classification methods for hierarchical multilabel classification," *Intell. Data Anal.*, vol. 15, no. 6, pp. 861–887, 2011.
[14] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Proc. 18th Eur. Conf. Mach. Learn.*, Warsaw, Poland, 2007, pp. 406–417.
[15] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *Proc. 14th Int. Conf. Mach. Learn.*, Nashville, TN, USA, 1997, pp. 170–178.
[16] D. Zhou, L. Xiao, and M. Wu, "Hierarchical classification via orthogonal transfer," in *Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, WA, USA, 2011, pp. 801–808.
[17] W.-L. Zhong, W. Pan, J.-Y. Kwok, and I.-H. Tsang, "Incorporating the loss function into discriminative clustering of structured outputs," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1564–1575, Oct. 2010.
[18] K. Dembczynski, W. Cheng, and E. Hüllermeier, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 279–286.
[19] L. Cai and T. Hofmann, "Exploiting known taxonomies in learning overlapping concepts," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, Pasadena, CA, USA, 2007, pp. 714–719.
[20] W. Bi and J.-T. Kwok, "Mandatory leaf node prediction in hierarchical multi-label classification," in *Proc. Adv. NIPS*, vol. 25. 2012, pp. 153–161.
[21] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int. J. Data Warehousing Mining*, vol. 3, no. 3, pp. 1–13, 2007.
[22] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, no. 2, p. 1453, 2005.
[23] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds., 2nd ed. New York, NY, USA: Springer-Verlag, 2010, pp. 667–685.
[24] J. Barbedo and A. Lopes, "Automatic genre classification of musical signals," *EURASIP J. Appl. Signal Process.*, vol. 2007, no. 1, p. 157, 2007.
[25] L. Tang, S. Rajan, and V. Narayanan, "Large scale multi-label classification via metalabeler," in *Proc. 18th Int. Conf. World Wide Web*, Madrid, Spain, 2009, pp. 211–220.
[26] J. Zaragoza, L. Sucar, and E. Morales, "Bayesian chain classifiers for multidimensional classification," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, Barcelona, Spain, 2011, pp. 2192–2197.
[27] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schölkopf, and D. Schurmans, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 61–74.
[28] G. Valentini, "True path rule hierarchical ensembles for genome-wide gene function prediction," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 8, no. 3, pp. 832–847, Jun. 2011.
[29] S. Shimony, "Finding MAPs for belief networks is NP-hard," *Artif. Intell.*, vol. 68, no. 2, pp. 399–410, 1994.
[30] C. Varin, N. Reid, and D. Firth, "An overview of composite likelihood methods," *Statist. Sinica*, vol. 21, no. 1, pp. 5–42, 2011.
[31] Y. Zhang and J. Schneider, "A composite likelihood view for multi-label classification," in *Proc. 15th Int. Conf. Artif. Intell. Statist.*, Ft. Lauderdale, FL, USA, 2012, pp. 1407–1415.
[32] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1982–2001, Apr. 2010.
[33] D. Lewis, Y. Yang, T. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, Dec. 2004.
[34] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. ECML/PKDD Workshop MMD*, Antwerp, Belgium, 2008, pp. 30–44.
[35] P. Heymann and H. Garcia-Molina, "Collaborative creation of communal hierarchical taxonomies in social tagging systems," Stanford InfoLab, Stanford, CA, USA, Tech. Rep. 2006-10, Apr. 2006.
[36] B. Klimt and Y. Yang, "The Enron corpus: A new dataset for email classification research," in *Proc. 18th Eur. Conf. Mach. Learn.*, Pisa, Italy, 2004, pp. 217–226.
[37] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.
[38] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2007-001, 2007.
[39] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Proc. Eur. Conf. Mach. Learn.*, 2009, pp. 254–269.
[40] J. Petterson and T. Caetano, "Submodular multi-label learning," in *Proc. Adv. NIPS*, 2011, pp. 1512–1520.
[41] J. Zhou, J. Chen, and J. Ye, *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Phoenix, AZ, USA: Arizona State Univ., 2011.
[42] G. Cesa-Bianchi and N. Valentini, "Hierarchical cost-sensitive algorithms for genome-wide gene function prediction," *J. Mach. Learn. Res.*, vol. 8, pp. 14–29, Mar. 2010.
[43] S. Garcıa and F. Herrera, "An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *J. Mach. Learn. Res.*, vol. 9, no. 12, pp. 2677–2694, 2008.
[44] J. Demvsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

Wei Bi received the bachelor's degree in computer science from the Sun Yat-sen University, Guangzhou, China, in 2010. She is currently pursuing the Ph.D. degree in computer science with the Hong Kong University of Science and Technology, Hong Kong.

Her current research interests include machine learning, data mining, application problems on computer vision, and other problems in artificial intelligence.

Dr. Bi received the Google Ph.D. Fellowship in Machine Learning in 2013.

James T. Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 1996.

He was with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, as an Assistant Professor. He is currently a Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. His current research interests include kernel methods, machine learning, example recognition, and artificial neural networks.

Dr. Kwok received the IEEE Outstanding 2004 Paper Award, and the Second Class Award in Natural Sciences by the Ministry of Education, China, in 2008. He has been a Program Co-Chair for a number of International conferences, and served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2006 to 2012. He is currently an Associate Editor for the *Neurocomputing* journal.