

On large margin hierarchical classification with multiple paths

Junhui Wang, Xiaotong Shen and Wei Pan*

Abstract

Hierarchical classification is critical to knowledge management and exploration, as in gene function prediction and document categorization. In hierarchical classification, an input is classified according to a structured hierarchy. In a situation as such, the central issue is how to effectively utilize the inter-class relationship to improve the generalization performance of flat classification ignoring such dependency. In this article, we propose a novel large margin method through constraints characterizing a multi-path hierarchy, where class membership can be non-exclusive. The proposed method permits a treatment of various losses for hierarchical classification. For implementation, we focus on the symmetric difference loss and two large margin classifiers: support vector machines and ψ -learning. Finally, theoretical and numerical analyses are conducted, in addition to an application to gene function prediction. They suggest that the proposed method achieves the desired objective and outperforms strong competitors in the literature.

Key words: Directed acyclic graph, Functional genomics, Generalization, Structured learning, Tuning

1 Introduction

In knowledge management, categorizing documents into a predefined hierarchy, known as hierarchical classification, is critical to knowledge exploration. For instance, in gene function

*Junhui Wang is Assistant Professor, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, IL 60607 (Email: jwang@math.uic.edu). Xiaotong Shen is Professor, School of Statistics, University of Minnesota, Minneapolis, MN 55455 (Email: xshen@stat.umn.edu). Wei Pan is Professor, Division of Biostatistics, University of Minnesota, Minneapolis, MN 55455 (Email: weip@biostat.umn.edu). The authors thank the editor, the associate editor and two referees for their helpful comments and suggestions. This work is supported in part by NSF grant DMS-0604394, NIH grant 1R01GM081535-01 and the Supercomputing Institute at University of Minnesota.

prediction, functions of genes are often organized by a gene function annotation system such as MIPS (Mewes et al., 2002), which defines a hierarchy with lower level categories being more detailed while upper level categories being more general. Furthermore, a gene can be classified into one or more classes non-exclusively according to the prespecified hierarchy. The central issue to be addressed is how to effectively utilize the inter-class relationship to improve the generalization performance of flat classification ignoring such dependency.

In the literature, conventional approaches have been applied to hierarchical classification, including the nearest neighbor method (Yang and Liu, 1999), Naive Bayes (Lewis, 1998), boosting (Schapire, Singer and Singhal, 1998), and support vector machines (Joachims, 1998). As argued in Cai and Hoffman (2004), these methods have not taken into account the inter-class relationship. Therefore, recent effort has been centered at incorporating hierarchy into classification, c.f., Cai and Hoffman (2004), Rousu et al. (2006), Cesa-Bianchi, Gentile and Zaniboni (2006), Shahbaba and Neal (2007), among others. Despite the progress, problems remain with regard to how to utilize a hierarchical structure without loss of information. In this article, we shall develop a large margin classifier to achieve the objective of effectively incorporating the hierarchical structure for higher classification performance.

This article concerns multi-path hierarchical classification, where class membership can be non-exclusive, that is, one input can be assigned to more than one class. Here each class is represented by a node in a directed acyclic graph (DAG), and the inter-class relationship is represented by directed paths, where a directed path from node u to node v indicates that an input must be assigned to class u if it is assigned to class v . Specifically, we introduce a novel large margin method through constraints describing path connectivity as well as a hierarchy-induced decision rule for classification. The advantage of this proposed method is two-fold: first, it effectively captures the hierarchical structure through simple constraints; second, it can deal with DAG permitting each node to have multiple parents, which differs from most existing hierarchical classification methods designed for tree only.

The proposed method is implemented for the symmetric difference loss with support vector machine (SVM, Cortes and Vapnik, 1995) and ψ -learning (Shen, et al., 2003) through quadratic programming and difference convex programming. The operating characteristics of the proposed method are examined in simulations, and we show that it outperforms several strong competitors. Moreover, rates of convergence of the proposed method are quantified in terms of the generalization error, which provides an insight into hierarchical classification. Indeed, incorporating the hierarchy into classification improves a large margin classifier’s generalization performance, owing to the fact that the hierarchical structure described by the constraints reduces the size of the underlying parameter space. With regard to the hierarchy, a deep one tends to lead to high improvement, especially when its depth increases with the sample size, whereas a short one specifies an weak inter-class relationship, hence that its improvement is expected to be small.

This article is organized in seven sections. Section 2 formulates the problem of hierarchical classification and Section 3 introduces the proposed large margin classification method. Section 4 presents some numerical examples, together with an application to gene function prediction. Section 5 develops a statistical learning theory. Section 6 contains a discussion, followed by technical proofs in the Appendices.

2 Hierarchical classification

In hierarchical classification, input $\mathbf{X} = (X_{(1)}, \dots, X_{(p)}) \in \mathcal{X} \subseteq \mathcal{R}^p$ is a vector of p variables, and class $\mathbf{Y} = (Y_{(1)}, \dots, Y_{(K)})$ is coded as $\{-1, 1\}^K$ with $Y_{(j)} = \pm 1$ indicating if \mathbf{X} belongs to class j . A hierarchy is defined by a DAG with nodes $0, 1, \dots, K$, where nodes $1, \dots, K$ correspond to classes $1, \dots, K$, and node 0 is the root corresponding to the union of classes $1, \dots, K$. The dependency among all classes is described by the paths in DAG, connecting the root to any non-root node. A decision function vector $\mathbf{f} = (f_1, \dots, f_K) \in \mathcal{F} = \prod_{j=1}^K \mathcal{F}_j$

is introduced with $f_j \in \mathcal{F}_j$ representing class j ; $j = 1, \dots, K$, in addition to a classification rule: $\text{Sgn}(\mathbf{f}(\mathbf{X})) \equiv (\text{sign}(f_1(\mathbf{X})), \dots, \text{sign}(f_K(\mathbf{X})))$.

Before proceeding, we introduce some notations to be used. Given node j , denoted by $par(j)$, $chi(j)$ $sib(j)$, $anc(j)$ its parent(s) (immediate ancestors), its children (immediate offsprings), its siblings (nodes sharing the same parent with node j), and its ancestors (immediate or remote). Note that $par(j)$, $chi(j)$ and $sib(j)$ are allowed to have multiple elements, or empty in absence of parents, children or siblings for node j . Moreover, when each $par(j)$ contains at most one node, the hierarchy becomes a tree.

2.1 Hierarchical structure for classification

One salient aspect of hierarchical classification is that the prespecified hierarchy imposes constraints on \mathbf{f} and thus needs to be built into classification. Through suitable constraints, the hierarchical structure can be fully taken into account.

A multi-path hierarchical structure is one kind of hierarchy requiring that input \mathbf{x} must be assigned to class $par(j)$ if it is assigned to class j . This assures that a path, possibly multiple paths, can be constructed from $anc(j)$ to j within the hierarchy. Mathematically, it is equivalent to enforcing constraints

$$\text{sign}(f_k(\mathbf{x})) - \text{sign}(f_j(\mathbf{x})) \geq 0, \quad \text{for all } k \in par(j) \text{ and } \mathbf{x} \in \mathcal{X}, \quad (1)$$

implying that $\text{sign}(f_k(\mathbf{x})) = 1$ for all $k \in anc(j)$ if $\text{sign}(f_j(\mathbf{x})) = 1$, whereas no constraint is imposed on $anc(j)$ if $\text{sign}(f_j(\mathbf{x})) = -1$. Note that (1) is enforced only on \mathcal{X} as opposed to the entire \mathcal{R}^p . Moreover, (1) permits \mathbf{x} to be classified to multiple classes, which differs from the conventional classification where class membership of \mathbf{x} is exclusive.

2.2 Generalization error and hierarchical structure

For hierarchical classification, the generalization error is used to measure a classifier’s generalization performance. The generalization error for a decision function vector \mathbf{f} is

$$GE(\mathbf{f}) = El(\mathbf{Y}, \mathbf{f}(\mathbf{X})), \quad (2)$$

where loss l measures accuracy of predicting outcome of Y by classifier $\text{Sgn}(\mathbf{f}(\mathbf{X}))$.

Unlike in multi-class classification, loss l in (2) may take various forms, depending on the cost of misclassification with respect to the hierarchy. For a tree, there have been proposed three types of losses. They are the 0-1 loss $l_{0-1}(\mathbf{Y}, \mathbf{f}(\mathbf{X})) = I(\mathbf{Y} \neq \text{Sgn}(\mathbf{f}(\mathbf{X})))$, the symmetric difference loss (Tsochantaridis et al., 2005) $l_{\Delta}(\mathbf{Y}, \mathbf{f}(\mathbf{X})) = K^{-1} \sum_{j=1}^K I(Y_{(j)} \neq \text{sign}(f_j(\mathbf{X})))$, and the H-loss (Cesa-Bianchi et al., 2004) $l_H(\mathbf{Y}, \mathbf{f}(\mathbf{X})) = \sum_{j=1}^K c_j I(Y_{(j)} \neq \text{sign}(f_j(\mathbf{X})) \ \& \ Y_{(k)} = \text{sign}(f_k(\mathbf{X})), \ \forall k \in \text{anc}(j))$. For the H-loss, two common choices of c_j ’s have been suggested:

$$c_j = K^{-1} |\text{subtree}(j)|; \ j = 1, \dots, K, \quad (3)$$

$$\text{or } c_0 = 1 \ \text{and} \ c_j = c_{\text{par}(j)} / |\text{sib}(j)|; \ j = 1, \dots, K, \quad (4)$$

where $\text{subtree}(j)$ is the subtree rooted by j and $|\cdot|$ is the size of a tree.

By comparison, l_{0-1} penalizes path disagreements, whereas l_{Δ} penalizes disagreements at each node and l_H penalizes disagreements of nodes while tolerating subsequent errors of offsprings. In other words, l_{0-1} focuses on complete correctness of predicting the entire path(s), whereas l_{Δ} and L_H discriminate partial correctness from complete correctness. In many real applications such as gene function prediction, a partially correct prediction is usually preferable than a completely wrong one, and so are l_{Δ} and L_H . In addition, l_H modifies l_{Δ} by incorporating the hierarchical structure into the loss function, as claimed

in Cesa-Bianchi et al. (2004). However, this modification becomes unnecessary when the hierarchical structure has already been fully captured through constraints (1). Moreover, the definition of l_Δ can be straightforwardly extended to a DAG, but such an extension for l_H remains unclear. Based on the forgoing discussion, we will focus our attention on l_Δ in implementation subsequently.

2.3 Decision rule with respect to hierarchy

In multi-path hierarchical classification, the decision rule is important as well. Here we adopt a top-down decision rule (Cesa-Bianchi, Conconi and Gentile, 2004) induced by the hierarchy, that is, \mathbf{X} is assigned to class j if $\text{sign}(\hat{f}_j) = 1$ and $\text{sign}(\hat{f}_k) = 1$ for all $k \in \text{anc}(j)$; and it is not otherwise. This is performed by classifying each node from the top to bottom of the hierarchy sequentially, while the order of siblings does not matter. Note that classifying \mathbf{X} to none of the classes in the hierarchy is permitted, which occurs when all $\text{sign}(\hat{f}_j) = -1$. This is sensible in applications such as gene function prediction where some genes are unannotated by any of known function classes. Furthermore, it is worthy of pointing out that simply applying the top-down decision rule is inadequate. However, when (1) is enforced through appropriate constraints, the top-down rule offers an effective way of incorporating the hierarchical structure into classification, and hence that better classification performance can be realized.

3 Proposed method

3.1 Large margin classification

In general $l(\mathbf{Y}, \mathbf{f}(\mathbf{X}))$ is nonconvex, and hence that minimizing (2) becomes difficult if not intractable. For this reason, it is often replaced by a surrogate loss $L(\mathbf{Y}, \mathbf{f}(\mathbf{X}))$, especially so in multi-class classification. In hierarchical classification, we say that loss L is a margin loss

if $L(\mathbf{Y}, \mathbf{f}(\mathbf{X}))$ can be written as $L(\mathbf{Y} \cdot \mathbf{f}(\mathbf{X}))$ with $\mathbf{Y} \cdot \mathbf{f}(\mathbf{X}) = (Y_{(1)}f_1(\mathbf{X}), \dots, Y_{(K)}f_K(\mathbf{X}))$ the componentwise product, and is large margin if $L(\mathbf{z})$ is non-increasing with respect to each component of $\mathbf{z} \equiv (z_1, \dots, z_K)$.

In implementation, we only consider $l_\Delta(\mathbf{Y}, \mathbf{f}(\mathbf{X})) = K^{-1} \sum_{j=1}^K I(Y_{(j)} \neq \text{sign}(f_j(\mathbf{X})))$ for our target gene function discovery where partial correctness of prediction is of interest, although the formulation is also applicable to other losses as well. For l_Δ , it is natural to employ a margin loss $L_\Delta(\mathbf{Y}, \mathbf{f}(\mathbf{X})) = K^{-1} \sum_{j=1}^K V(Y_{(j)}f_j(\mathbf{X}))$ defined by a binary margin loss $V(\cdot)$. Binary margin losses include, among others, the hinge loss $V(z) = (1 - z)_+$ for SVM with its variants $V(z) = (1 - z)_+^q$ for $q > 1$; c.f., Lin (2002); the ψ -losses $V(z) = \psi(z)$, with $\psi(z) = 1 - \text{sign}(z)$ if $z \geq 1$ or $z < 0$, and $2(1 - z)$ otherwise, c.f., Shen, et al. (2003), the logistic loss $V(z) = \log(1 + e^{-z})$, c.f., Zhu and Hastie (2005); the ρ -hinge loss $V(z) = (\rho - z)_+$ for nu-SVM (Schölkopf et al., 2000) with $\rho > 0$ being optimized; the sigmoid loss $V(z) = 1 - \tanh(cz)$; c.f., Mason, et al. (2000).

As discussed in Section 2.1, \mathbf{f} must satisfy (1) so that the hierarchical structure can be incorporated. However, (1) defines nonlinear constraints that are difficult to treat numerically. We therefore invoke linear constraints:

$$f_{par(j)}(\mathbf{x}) - f_j(\mathbf{x}) \geq 0, \quad \forall \mathbf{x} \in \mathcal{X}, \quad (5)$$

which imply (1) and are easier to work with in constrained optimization. Note that the linear constraints enforced at all values of \mathbf{x} may be infinite. We therefore approximate them through finite constraints for computation. Toward this end, we sample $\tilde{\mathbf{x}}_t$; $t = 1, \dots, m$, from \mathcal{X} and impose (5) over these points. Here the specific form of the sampling distribution is irrelevant as long as it covers \mathcal{X} .

In light of the aforementioned discussion, we propose a cost function for large margin

hierarchical classification:

$$\min_f C \sum_{i=1}^n L_{\Delta}(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i)) + J(\mathbf{f}), \quad (6)$$

$$\text{subject to } f_{par(j)}(\tilde{\mathbf{x}}_t) - f_j(\tilde{\mathbf{x}}_t) \geq 0; t = 1, \dots, m; j = 1, \dots, K, \quad (7)$$

where $\{\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ are the training sample and $C > 0$ is a regularization parameter. Subject to (7), minimizing (6) with respect to $\mathbf{f} \in \mathcal{F}$, the candidate function space, yields our estimated $\hat{\mathbf{f}}$, and thus the classification rule.

In linear classification, \mathbf{f} is linearized in that each $f_j(\mathbf{x}) = \mathbf{x}^T \mathbf{w}_j + b_j$ represents a hyperplane with $\mathbf{w}_j \in \mathcal{R}^p$ and $b_j \in \mathcal{R}^1$. In nonlinear classification, a kernel $\mathcal{K}(\cdot, \cdot)$, mapping from $\mathcal{R}^p \times \mathcal{R}^p$ to \mathcal{R}^1 , is used for a flexible representation: $f_j(x) = (\mathcal{K}(\mathbf{x}, \mathbf{x}_1), \dots, \mathcal{K}(\mathbf{x}, \mathbf{x}_n))^T \mathbf{a}_j + b_j$ with $\mathbf{a}_j \in \mathcal{R}^n$, where the reproducing kernel Hilbert spaces (RKHS) plays an important role; c.f., Gu (2000). For this reason, this is also referred as to kernel learning.

The penalty $J(\mathbf{f})$ is often related to the so called geometric separation margin, which can be defined when a specific form of $L(\cdot, \cdot)$ is given. When L_{Δ} is used, we define $J(\mathbf{f})$ to be the reciprocal of the geometric margin of various form in the componentwise additive form. For instance, in linear learning with $f_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{x} + b_j$, $J(\mathbf{f}) = (2K)^{-1} \sum_{j=1}^K \|\mathbf{w}_j\|^2$, where $\|\cdot\|$ is a L_2 -norm; in nonlinear learning with $f_j(x) = (\mathcal{K}(\mathbf{x}, \mathbf{x}_1), \dots, \mathcal{K}(\mathbf{x}, \mathbf{x}_n))^T \mathbf{a}_j + b_j$, $J(\mathbf{f}) = (2K)^{-1} \sum_{j=1}^K \mathbf{a}_j^T \mathcal{K} \mathbf{a}_j$ with $\mathcal{K} = (\mathcal{K}(\mathbf{x}_i, \mathbf{x}_{i'}))_{i,i'=1}^n$.

The values of m and $\tilde{\mathbf{x}}_t$'s control the precision of the approximation of (7) to (5). Roughly, larger m yields better approximation at cost of an increased computational burden. For generating $\tilde{\mathbf{x}}_t$'s, it is trivial when \mathbf{X} is discrete. When \mathbf{X} is continuous, one may apply the convex hull scheme as in Shen, Shi and Wong (1999) by first generating the convex hull defined by the sample points $\mathbf{x}_1, \dots, \mathbf{x}_n$ and then placing uniformed spaced points inside the convex hull. As for the choice of m , we recommend m to be n , which yields adequate approximations as showed in Section 4.3.

As a technical remark, in the linear case, (5) can be substituted by

$$w_{par(j)} = w_j; \quad b_{par(j)} - b_j \geq 0, \quad (8)$$

based on the fact that (8) is equivalent to (5) when \mathcal{X} is unbounded. However, this is no longer true when \mathcal{X} is bounded as in many real applications, where (8) may impose more constraints than necessary for the hierarchical structure, and thus impede the classification performance. This aspect is illustrated numerically in Section 4. In the nonlinear case, it is unclear how to extend such a formulation.

3.2 Implementation

In this section, we implement (6) and (7) with the hinge loss $V(z) = (1 - z)_+$ and the ψ loss $V(z) = \psi(z)$, respectively. For the hinge loss, (6) subject to (7) is solved through its dual form and quadratic programming (QP). For the nonconvex ψ -loss, we develop a method based on difference convex (DC) programming (An and Tao, 1997), which was previously employed in Liu, Shen and Wong (2005) for supervised ψ -learning.

For simplicity, denote by $s(\mathbf{f})$ the cost function in (6) with $V(z) = \psi(z)$. Key to DC programming is decomposing $s(\mathbf{f})$ into a difference of two convex functions. Based on the decomposition, iterative upper convex approximations are constructed by replacing the second convex function with its tangent hyperplane. Minimizing the upper convex approximations yields a sequence of solutions converging to a stationary point, possibly an ϵ -global minimizer. We now construct the convex decomposition of $s(\mathbf{f}) = s_1(\mathbf{f}) - s_2(\mathbf{f})$, with

$$\begin{aligned} s_1(\mathbf{f}) &= \frac{C}{K} \sum_{i=1}^{n_l} \sum_{j=1}^K \psi_1(y_{ij} f_j(\mathbf{x}_i)) + J(\mathbf{f}) \text{ and} \\ s_2(\mathbf{f}) &= \frac{C}{K} \sum_{i=1}^{n_l} \sum_{j=1}^K \psi_2(y_{ij} f_j(\mathbf{x}_i)), \end{aligned}$$

where $\psi_1 = 2(1 - z)_+$ and $\psi_2 = 2(-z)_+$. Here ψ_1 and ψ_2 are obtained through a convex decomposition of $\psi = \psi_1 - \psi_2$ as displayed in Figure 1.

Figure 1 about here

With these decompositions, we treat the nonconvex minimization (6) by solving a sequence of quadratic problems iteratively. In step $k + 1$, we solve

$$\begin{aligned} \min_{\mathbf{f}} \quad & s_1(\mathbf{f}) - \langle \mathbf{w}, \nabla s_2(\hat{\mathbf{f}}^{(k)}) \rangle_{\mathcal{K}} \\ \text{subject to} \quad & f_{\text{par}(j)}(\mathbf{x}_i) - f_j(\mathbf{x}_i) \geq 0; \quad \forall i, j, \end{aligned} \quad (9)$$

where $\langle \cdot, \cdot \rangle_{\mathcal{K}}$ is the inner product with respect to kernel \mathcal{K} and $\nabla s_2(\hat{\mathbf{f}}^{(k)})$ is a gradient vector of $s_2(\mathbf{f})$ at the k -th step solution $\hat{\mathbf{w}}^{(k)}$, defined as the sum of partial derivatives of s_2 over each observation, with $\nabla \psi_2(z) = 0$ if $z > 0$ and $\nabla \psi_2(z) = -2$ otherwise. Note that $s_1(\mathbf{f}) - \langle \mathbf{w} - \mathbf{w}^{(k)}, \nabla s_2(\hat{\mathbf{f}}^{(k)}) \rangle_{\mathcal{K}}$ is a convex upper bound of $s(\mathbf{f})$ following the convexity of $s_2(\mathbf{f}^{(k)})$.

The detailed algorithm is given as follows.

Algorithm 1: (Sequential quadratic programming)

Step 1. (Initialization) Set initial $\hat{\mathbf{f}}^{(0)}$ to be the solution of $\min_{\mathbf{f}} s_1(\mathbf{f})$. Specify precision tolerance level $\epsilon > 0$.

Step 2. (Iteration) At iteration $k + 1$, compute $\hat{\mathbf{f}}^{(k+1)}$ by solving (9).

Step 3. (Stopping rule) Terminate when $|s(\hat{\mathbf{f}}^{(k+1)}) - s(\hat{\mathbf{f}}^{(k)})| \leq \epsilon$.

Then the estimate $\hat{\mathbf{f}}$ is the best solution among $\hat{\mathbf{f}}^{(k)}$; $k = 0, 1, \dots$

The convergence of **Algorithm 1** is guaranteed by the nonincreasing property of the upper envelopes of $s(\mathbf{f})$ with respect to iteration. The speed of convergence of **Algorithm 1** is superlinear, following the same proof as in Theorem 3 of Liu, Shen and Wong (2005) for ψ -learning. This means that the number of iterations required for **Algorithm 1** to achieve the precision ϵ is $o(\log(1/\epsilon))$. As a technical remark, we note that an ϵ -global minimizer

can be obtained when **Algorithm 1** is combined with the branch-and-bound method, as in Liu, Shen and Wong (2005). For a computational consideration, we shall not seek the exact global minimizer in here. Based on numerical experiments of An and Tao (1997), and Liu, Shen and Wong (2005), such a DC algorithm as in **Algorithm 1** usually leads to a good local solution even when it is not global. This is evident from the DC decomposition where s_2 can be thought of correcting the bias due to convexity imposed by s_1 that is the cost function of HSVM.

4 Numerical examples

This section examines effectiveness of the proposed hierarchical classifiers through two simulated examples and one real application to gene function prediction. The proposed classifiers with the hinge loss and ψ -loss, denoted as HSVM and HPSI, are compared against flat SVM (FlatSVM; solving (6) without constraint (7)) ignoring the hierarchical structure, parallel SVM (ParSVM; solving (6) with respect to constraint (8)) requiring classification functions to be parallel as well as Hoffman’s structured SVM (HoffSVM; Cai and Hoffman, 2004) that uses a hierarchical representation of classification functions. The top-down decision rule in Section 2.3 is applied to all classification functions to yield the final hierarchical classifiers.

A classifier’s generalization performance is measured by the test error averaged over 100 simulation replications, approximating the generalization error defined in (2). The test error can be written as

$$TE(\mathbf{f}) = \frac{1}{\#\{\text{test set}\}} \sum_{\text{test set}} l(\mathbf{Y}_i, \text{Sgn}(\mathbf{f}(\mathbf{X}_i))),$$

where l is an evaluation loss with four specific forms: l_{0-1} , l_{Δ} , l_H with c_j ’s defined in (3) and l_H with c_j ’s defined in (4). The corresponding test errors are denoted by TE_{0-1} , TE_{Δ} , TE_{sub} and TE_{sib} , respectively. Note that our focus here is TE_{Δ} as discussed in Section 2.2, although the other three losses are used as an evaluation reference.

For comparison, we use FlatSVM as a baseline. For a simulated example where the Bayes rule is available, the amount of improvement of a classifier over FlatSVM is defined as the percent of improvement in terms of the Bayesian regret

$$\frac{(TE(FlatSVM) - Bayes) - (TE(\cdot) - Bayes)}{TE(FlatSVM) - Bayes},$$

where TE can be TE_{0-1} , TE_{Δ} , TE_{sub} or TE_{sib} , and $Bayes$ denotes the corresponding Bayes error. The Bayes error is the ideal optimal performance and serves as a benchmark for comparison. For real application of gene function prediction where the Bayes rule is unavailable, the amount of improvement is defined as

$$\frac{TE(FlatSVM) - TE(\cdot)}{TE(FlatSVM)},$$

which may underestimate the actual percentage of improvement over FlatSVM.

All numerical analyses are conducted in R2.1.1 for FlatSVM, ParSVM, HoffSVM, HSVM and HPSI. In linear learning, $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$; in Gaussian kernel learning, $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$, where σ is set to be the median distance between positive and negative classes to reduce computational cost of tuning σ^2 , c.f., Jaakkola, Diekhans and Haussler (1999).

4.1 Simulated examples

Example 1: A random sample, $(Y_{i1}, \dots, Y_{i8}, X_{i1}, \dots, X_{i10})$; $i = 1, \dots, 1000$, is sampled independently: $X_{ij} \sim U(-1, 1)$; $j = 1, 2$, $X_{ij} \sim N(0, 1)$; $j = 3, \dots, 10$, $Y_{i1} = 2I(X_{i1} \geq 0) - 1$, $Y_{i2} = 2I(X_{i1} < 0) - 1$ and $Y_{ij} = -1$; $j = 3, \dots, 8$. Secondly, 10% of the sample are chosen at random and Y_{i1} and Y_{i2} 's are flipped to generate the nonseparable case. Thirdly, randomly select 4/5 of the sample with $Y_{i1} = 1$ and $X_{i1} \geq 0$, 4/5 of that with $Y_{i1} = 1$ and $X_{i1} < 0$,

4/5 of that with $Y_{i2} = 1$ and $X_{i1} \leq 0$, 4/5 of that with $Y_{i2} = 1$ and $X_{i1} \leq 0$, and set their Y_{i3} 's- Y_{i6} 's to be 1, respectively. Finally, set $Y_{i7} = I(Y_{i3} = Y_{i4} = 1, 2X_{i1} + X_{i2} - 1 \geq 0)$ and $Y_{i8} = I(Y_{i5} = Y_{i6} = 1, 2X_{i1} - X_{i2} + 1 \leq 0)$. This yields the first simulated example defined by a DAG with multiple parents, in which 100 and 900 instances are used for training and testing.

Example 2: A random sample, $(Y_{i1}, \dots, Y_{i10}, X_{i1}, \dots, X_{i10}); i = 1, \dots, 1000$, is sampled independently: $X_{ij} \sim U(-1, 1); j = 1, 2, X_{ij} \sim N(0, 1); j = 3, \dots, 10, Y_{ij} = 2I(X_{i1} \geq -1 + 2j/11) - 1; j = 1, \dots, 10$. Secondly, randomly choose 10% of the sample with $Y_{ij} = 1$ and $Y_{i,j+1} = -1$, and change their $Y_{i,j+1}$'s 1 to generate the nonseparable case; $j = 1, \dots, 9$. This yields the second simulated example defined a deep tree, in which 100 and 900 instances are used for training and testing.

Figure 2 displays the hierarchical structures involved in Examples 1 and 2. Note that both the examples are multi-path since labeling is permitted for multiple classes, and the hierarchy in Example 1 is not tree, where nodes 7 and 8 have multiple parents.

Figure 2 about here

To eliminate dependency of each hierarchical method on tuning parameter C , the test errors are minimized over 61 grid points $\{10^{-3+k/10}; k = 0, \dots, 60\}$ through a grid search, and are averaged over 100 simulation replications. The results are summarized in Table 1, with the smallest test errors over the four competing classifiers boldfaced.

Table 1 about here

Evidently, HSVM and HPSI outperform FlatSVM and HoffSVM under all evaluation metrics and in both the linear and Gaussian cases. Interestingly, ParSVM yields the best performance in Example 2 when the true classification functions are parallel, but the worst performance in Example 1 when they are otherwise. HPSI performs better than HSVM in almost all the cases including Example 2 with Gaussian kernel where the improvement can be

seen in the fourth decimal, and significantly better in some cases. Note that HoffSVM and the H-loss are defined only for trees, and are not directly applicable to Example 1. In Example 2, the test errors TE_{0-1} and TE_{sib} become identical and so are the corresponding l_{0-1} and l_H , because the weights $c_j = 1$ for all j in (4) for the hierarchy defined in Figure 2. More specifically, the amounts of improvement of HSVM and HPSI over FlatSVM range from 2.9% to 63.1%, whereas those of HoffSVM and ParSVM over FlatSVM are from -91.4% to 28.8% and from -18.3% to 73.3% , respectively. This suggests that the structural representations of HoffSVM and ParSVM can impede the classification performance, whereas HSVM and HPSI improve over FlatSVM with the help of the hierarchical structure.

4.2 Application to gene function prediction

At present, almost half of the genes of yeast *S. cerevisiae* have not been annotated by a gene function annotation system MIPS, although the gene expression profiles are available for the entire genome. Therefore, predicting functional categories for unannotated genes is a central problem in biological research.

The gene function prediction problem can be cast into the framework of multi-path hierarchical classification, with a hierarchy defined by MIPS. Along a path, we extract general and specific information about one gene according to which functional category that it belongs to from the top to bottom of the hierarchy. This is achieved by using the information of known gene functions and their expression profiles. This is feasible because genes tend to have similar biological functions when they share similar expression profiles, c.f., Eisen et al. (1998). In this process, many genes are allowed to be annotated by multiple functional categories in MIPS. For instance, the gene YBL022C is annotated by three different functional categories: “TRANSCRIPTION”, “PROTEIN FATE (folding modification destination)” and “SUBCELLULAR LOCALIZATION” at the highest level.

This section applies the proposed method to predict gene functions through expression

data in Hughes et al. (2000), which are composed of expression profiles of a total of 6316 genes gathered from 300 microarray experiments for the yeast. In particular, we discriminate gene functional categories within two branches of the MIPS hierarchy, consisting of two functional classes at the highest level: “CELL CYCLE AND DNA PROCESSING” and “TRANSCRIPTION” and their corresponding offsprings, see Figure 4 for description. Given these two branches, only 23 functional categories that annotate 1103 genes according to MIPS as of May, 2003.

Figure 4 about here

Before we apply HSVM and HPSI to predict gene functions, we evaluate their performances based on repeated experiments. Specifically, we sample 303 genes from 1103 genes at random for training and use the remaining 800 genes for testing. This process is repeated over 100 times and the test errors under different evaluation losses are averaged over 100 replications, as described in Section 4.2. The results are summarized in Table 2. Again, HSVM and HPSI outperform FlatSVM, ParSVM and HoffSVM, under all the evaluation metrics, in both linear and Gaussian kernel cases. The amount of improvement of HSVM or HPSI over FlatSVM ranges from 0.3% to 10.4%. By comparison, ParSVM and HoffSVM perform worse than FlatSVM in almost all the cases, and HPSI performs nearly the same as HSVM in linear case and is significantly better than HSVM in nonlinear case.

The amounts of improvements of HSVM and HPSI in here are not as large as those in the simulated examples. This may be due to the high dimensionality $p = 300$ and relatively small sample size $n = 303$. In such a situation, the hierarchy provides only a limited amount of information. This aspect has been also noticed in the literature; c.f., Cai and Hoffman (2004), and Shahbaba and Neal (2007). However, in contrast to the published results for other high-dimension hierarchical problems (e.g., Rousu et al., 2006, and Cai and Hoffman, 2004), the above improvements are considered to be significant.

Table 2 about here

Next we apply the trained HPSI to our data as of May, 2003 to predict unknown gene functional categories that were not annotated by this version of MIPS. Our predicted gene functions will be cross-validated by a newer version of MIPS as of January, 2008, where more than 50% additional genes have been annotated by this newer version of MIPS, providing a good way to evaluate findings of our proposed method, c.f., Xiao and Pan (2005). Specifically, we construct HPSI classifier from the training set with 303 genes where HPSI yields the smallest error rate among the previous 100 simulations, and predict the unknown gene functions. Then we use most confident predictions for cross-validation, where the confidence is defined by the estimated conditional probability obtained through Wang, Shen and Liu (2008). The result is summarized in Table 3. Among the ten most confident predications, according to the newer version of MIPS, three are validated to be perfectly correct (+), two can not be verified (-) due to unknown gene functional categories in the 2008 version of MIPS, three are verified to one 1-level parent node (P1) and two are verified to one 2-level parent node (P2). Note that (P1) and (P2) are considered to be partially correct. For example, the 2008 version of MIPS indicates that gene YOR039W has annotated “G1/S transition of mitotic cell cycle”, whereas it is predicted to be annotated by “mitotic cell cycle and cell cycle control” (030301), which is the 2-level parent node of “G1/S transition of mitotic cell cycle”. In this case, the general hierarchical structure is correctly predicted whereas the detailed ones are missed. Overall, these ten predictions are confirmed by the new biological discovery given in the 2008 version of MIPS.

Table 3 about here

4.3 Sensitivity study

We now perform a sensitivity study to investigate the performance of HSVM and HPSI relative to the number of support points m . For illustration, we only report the result of HSVM. The study is conducted in Example 1, where m is set to be 0, 50, 100, 150, 200. Figure 3 indicates that the test error of HSVM, measured by l_Δ as a function of m , decreases as m increases in the linear and Gaussian kernel cases, indicating that an increase in m may improve the classification performance. However, the amount of improvement becomes negligible when m exceeds a certain level of thresholding, say 50. The other losses show the similar pattern as in Figure 3 and thus are omitted here.

Figure 3 about here

Next we perform a sensitivity study to investigate the degree of approximation of $\{\mathbf{f} \in \mathcal{F} : \mathbf{f} \text{ satisfies (7)}\}$ to $\{\mathbf{f} \in \mathcal{F} : \mathbf{f} \text{ satisfies (5)}\}$ with respect to m and p . The approximation accuracy is quantified by the proportion of inconsistent predictions on a test set, that is, $\#\{\mathbf{x} \in \{\text{test set}\} : \mathbf{f}(\mathbf{x}) \text{ does not satisfy (1)}\} / \#\{\text{test set}\}$. The study is conducted for the linear HSVM on the gene example in Section 4.2, with and without the top-down decision rule, where $m = 303, 600$ and $p = 30, 100, 300$. For the cases when $p < 300$, a prescreening procedure is applied for variable selection, as in Guyon, Weston and Vapnik (2002), and optimal C is determined by minimizing TE_Δ .

Table 4 displays the averaged proportion of inconsistent prediction as well as corresponding averaged TE_Δ over 100 independent replicates. Evidently, the proportions of inconsistent predictions are about 18% and 0%, with and without the top-down decision rule, when $m = 303$ and $p = 300$, implying an adequate approximation accuracy of (7). The proportions of inconsistent prediction decrease when p becomes smaller or m becomes larger. Intuitively, this is sensible because (7) enforces the hierarchical structure well when m is relative large comparing with p , but it may require a significantly increased computational cost. There-

fore, we recommend to choose a moderate size of m , and employ the top-down decision rule to enforce the final prediction to be consistent with the hierarchy.

Table 4 about here

5 Statistical learning theory

This section quantifies the asymptotic behavior of the generalization accuracy of the proposed classifier $\hat{\mathbf{f}}$ defined by (6) and (7). The generalization accuracy is measured by the Bayesian regret $e(\hat{\mathbf{f}}, \bar{\mathbf{f}}) = GE(\hat{\mathbf{f}}) - GE(\bar{\mathbf{f}}) \geq 0$ with $\bar{\mathbf{f}} = (\bar{f}_1, \dots, \bar{f}_K)$ and $\bar{f}_j(\mathbf{x}) = \text{sign}(P(Y_{(j)} = 1 | \mathbf{X} = \mathbf{x}) - 1/2)$, which is the difference between the actual performance of $\hat{\mathbf{f}}$ and the ideal performance of the Bayes rule $\bar{\mathbf{f}} = (\bar{f}_1, \dots, \bar{f}_K)$, with $GE(\mathbf{f}) = El_{\Delta}(\mathbf{Y}, \mathbf{f}(\mathbf{X}))$.

5.1 Asymptotic theory for large margin hierarchical classifiers

Bounds for $e(\hat{\mathbf{f}}, \bar{\mathbf{f}})$ will be derived in terms of complexity of the class of candidate decision functions defined by sample constraints $\tilde{\mathcal{F}} = \{\mathbf{f} : \mathbf{f} \in \mathcal{F} \text{ with } \mathbf{f} \text{ satisfying (7)}\}$, n , m , $K(\cdot, \cdot)$ and tuning parameter $\lambda = (nC)^{-1}$.

Corollary 1 *Under the assumptions of Theorem 1 in Appendix A,*

$$e(\hat{\mathbf{f}}, \bar{\mathbf{f}}) = O_p(\delta_n^{2\alpha}), \quad Ee(\hat{\mathbf{f}}, \bar{\mathbf{f}}) = O(\delta_n^{2\alpha}),$$

provided that $n(\lambda J^)^{2-\min(\beta, 1)}$ is bounded away from 0, where $\delta_n^2 = \min(1, \max(\epsilon_n^2, 2s_n) + 4T(T+1)(n+1)^{-1})$, and ϵ_n^2 , s_n , α , β , J^* and T are defined in Appendix A.*

Corollary 1 indicates that the imposed hierarchical constraints (5) play a key role in hierarchical classification because they may reduce the capacity of the candidate function class $\bar{\mathcal{F}} = \{\mathbf{f} : \mathbf{f} \in \mathcal{F} \text{ with } \mathbf{f} \text{ satisfying (5)}\} \subset \tilde{\mathcal{F}}$. As a result, better generalization accuracy can be realized than its flat counterpart, which is in contrast to Theorem 1 for multi-class

large margin classification in Shen and Wang (2007). More specifically, the rate $\delta_n^{2\alpha}$ becomes faster when the entropy of $\bar{\mathcal{F}}$ gets smaller, especially so when the number of levels of the involved hierarchy gets larger, c.f., Lemma 2.

5.2 Theoretical examples

We now apply Corollary 1 to one linear and one nonlinear learning examples to obtain the generalization error rates of HSVM and HPSI, as measured by the Bayesian regret $e(\mathbf{f}, \bar{\mathbf{f}})$. In either case, we will demonstrate that HSVM and HPSI outperform their flat counterparts FlatSVM and FlatPSI, when sufficient structural information is provided by the hierarchy. Technical details are deferred to Appendix C.

Linear learning: Consider linear classification where $\mathbf{X} = (X_{(1)}, X_{(2)})$ is sampled independently according to $U(-1, 1)$. Given \mathbf{X} , label $Y_{(j)}$ is 1 if $X_{(1)} > -1 + 2j/(K + 1)$ and -1 otherwise; $j = 1, \dots, K$. This defines a tree hierarchy: $1 \rightarrow 2 \rightarrow \dots \rightarrow K$. Then for $j = 1, \dots, 9$, 10% of the sample with $Y_{ij} = 1$ and $Y_{i,j+1} = -1$ are randomly chosen and their $Y_{i,j+1}$'s are changed to 1 to generate a nonseparable situation. Here \mathcal{F} consists of linear decision functions of the form $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_K(\mathbf{x}))$ with $f_j(\mathbf{x}) = b_j + \mathbf{x}^T \mathbf{w}_j$ for $\mathbf{w}_j \in \mathcal{R}^2$ and $\mathbf{x} = (x_{(1)}, x_{(2)}) \in \mathcal{R}^2$. The Bayes rule is $\bar{\mathbf{f}}(\mathbf{x}) = (\bar{f}_1(\mathbf{x}), \dots, \bar{f}_K(\mathbf{x}))$ with $\bar{f}_j(\mathbf{x}) = \text{sign}(x_{(1)} + 1 - 2j/(K + 1))$, yielding a sequence of vertical lines as the classification boundary for different classes.

An application of Corollary 1 yields that the convergence rates of HSVM and HPSI are $O((\min(K, n^{1/2})n^{-1} \log n)^{1/2})$ and $O(\min(K, n^{1/2})(n^{-1} \log n))$ respectively, while the convergence rates of FlatSVM and FlatPSI are $O((Kn^{-1} \log n)^{1/2})$ and $O(Kn^{-1} \log n)$ respectively, where K comes from the fact that the metric entropy of \mathcal{F} is K times as large as that of \mathcal{F}_1 . These rates are slower than those of HSVM and HPSI when $K > O(n^{1/2})$.

Nonlinear learning: Consider, in the preceding case, kernel learning with a different candidate decision function space defined by the Gaussian kernel. By the representa-

tion theorem of RKHS, c.f., Wahba (1990), it is convenient to embed a finite-dimensional Gaussian kernel representation into an infinite-dimensional space $\mathcal{F} = \{\mathbf{x} \in \mathcal{R}^2 : \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_K(\mathbf{x})) \text{ with } f_j(\mathbf{x}) = b_j + \mathbf{w}_j^T \phi(\mathbf{x}) = b_j + \sum_{l=0}^{\infty} w_{j,l} \phi_l(\mathbf{x}) : \mathbf{w}_j \in \mathcal{R}^{\infty}\}$, and $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = \mathcal{K}(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2 / (2\sigma_n^2))$ with σ_n^2 to be specified.

An application of Corollary 1 yields that the convergence rate of HSVM is $Ee(\hat{\mathbf{f}}, \bar{\mathbf{f}}) = O(\max(\min(K, n^{2/9})n^{-1}(\tau^2\sigma_n^2 + \log n + \sigma_n^{-2})^3, K\tau^{-1}))$, yielding $O(\min(K, n^{2/9})n^{-1/7})$ with $\tau \sim n^{1/7}$ when σ_n^2 is fixed, and $O(\min(K, n^{2/9})n^{-1/4})$ with $\tau \sim \sigma_n^{-2} \sim n^{1/4}$. Similarly, the convergence rate of HPSI is $Ee(\hat{\mathbf{f}}, \bar{\mathbf{f}}) = O(\max(\min(K, n^{2/9})n^{-1}(\log(n\tau^2\sigma_n^2) + \sigma_n^{-2})^3, K\tau^{-1})) = O(\min(K, n^{2/9})n^{-1}(\log n)^3)$ with $\tau \sim n(\log n)^{-3}$ when σ_n^2 is fixed, or $\tau \sim n(\log n)^{-3}$ and $\sigma_n^{-2} \sim \log n$. In contrast, the convergence rates of FlatSVM with fixed or optimally chosen σ_n^2 and FlatPSI to be $O(Kn^{-1/7})$, $O(Kn^{-1/4})$ and $O(Kn^{-1}(\log n)^3)$, respectively. Again, the rates are slower than those of HSVM with fixed or optimally chosen σ_n^2 and HPSI when $K > O(n^{2/9})$.

6 Discussion

This article proposed a novel large margin method for the symmetric difference loss in the hierarchical classification. In contrast to existing hierarchical methods, the proposed method integrates the hierarchical structure into classification through simple constraints, leading to better classification accuracy than its flat counterparts. Both theoretical and numerical analyses suggest that the generalization error of the proposed method becomes smaller than its flat counterpart due to the built-in hierarchical structure. The utility of the method is illustrated on an application to gene function prediction, where the proposed method successfully predicts the function categories of various genes that have not been annotated in the original data.

Appendix A: Theorem and Technical Assumptions

Let $\bar{\mathcal{F}} = \{\mathbf{f} : \mathbf{f} \in \mathcal{F} \text{ with } \mathbf{f} \text{ satisfying (5)}\} \subset \tilde{\mathcal{F}}$, $e_L(\mathbf{f}, \bar{\mathbf{f}}) = E(L_\Delta(\mathbf{f}, \mathbf{Z}) - L_\Delta(\bar{\mathbf{f}}, \mathbf{Z}))$ with $L_\Delta(\mathbf{f}, \mathbf{Z}) = K^{-1} \sum_{j=1}^K V(Y_{(j)} f_j(\mathbf{X}))$ the surrogate loss used in (6) and $L(\cdot)$ any margin loss. The following assumptions are made.

Assumption A. (Approximation error) For some positive sequence such that $s_n \rightarrow 0$ as $n \rightarrow \infty$, there exists $\mathbf{f}^* \in \bar{\mathcal{F}}$ such that $e_L(\mathbf{f}^*, \bar{\mathbf{f}}) \leq s_n$.

Assumption A is analogous to that in Shen et al. (2003), and ensures that the Bayes rule $\bar{\mathbf{f}}$ is well approximated by $\bar{\mathcal{F}}$.

Next, we define a truncated $L_\Delta^T(\mathbf{f}, \mathbf{Z}) = K^{-1} \sum_{j=1}^K V^T(Y_{(j)} f_j(\mathbf{X}))$ with

$$V^T(Y_{(j)} f_j(\mathbf{X})) = \begin{cases} V(Y_{(j)} f_j(\mathbf{X})) & \text{if } V(Y_{(j)} f_j(\mathbf{X})) \leq T, \\ T & \text{otherwise,} \end{cases}$$

for any $\mathbf{f} \in \tilde{\mathcal{F}}$ and some truncation constant T such that $\max_j \{V(Y_{(j)} \bar{f}_j(\mathbf{X})), V(Y_{(j)} f_j^*(\mathbf{X}))\} \leq T$; $\forall j$ almost surely, and $e_{L^T}(\mathbf{f}, \mathbf{f}^*) = E(L_\Delta^T(\mathbf{f}, \mathbf{Z}) - L_\Delta(\mathbf{f}^*, \mathbf{Z}))$.

Assumption B. (Conversion formula) There exist constants $0 < \alpha \leq \infty$, $\beta \geq 0$, $a_1 > 0$ and $a_2 > 0$ such that for any sufficiently small $\delta > 0$,

$$\sup_{\{\mathbf{f} \in \tilde{\mathcal{F}}: e_{L^T}(\mathbf{f}, \bar{\mathbf{f}}) \leq \delta\}} e(\mathbf{f}, \bar{\mathbf{f}}) \leq a_1 \delta^\alpha, \quad (10)$$

$$\sup_{\{\mathbf{f} \in \tilde{\mathcal{F}}: e_{L^T}(\mathbf{f}, \bar{\mathbf{f}}) \leq \delta\}} \text{Var}(L_\Delta^T(\mathbf{f}, \mathbf{Z}) - L_\Delta(\bar{\mathbf{f}}, \mathbf{Z})) \leq a_2 \delta^\beta. \quad (11)$$

Assumption B describes local smoothness of $e(\mathbf{f}, \bar{\mathbf{f}})$ and $\text{Var}(L_\Delta^T(\mathbf{f}, \mathbf{Z}) - L_\Delta(\bar{\mathbf{f}}, \mathbf{Z}))$ within a neighborhood of \mathbf{f}^* . The exponents α and β depend on the joint distribution of (\mathbf{X}, \mathbf{Y}) . Moreover, (11) is implied by the low noise assumption, c.f., Shen and Wang (2006).

Next, we define the L_2 -metric entropy with bracketing that measures the cardinality of $\bar{\mathcal{F}}$. Given any $\epsilon > 0$, denote $\{(\mathbf{f}_r^l, \mathbf{f}_r^u)\}_{r=1}^R$ as an ϵ -bracketing function set of $\bar{\mathcal{F}}$ if for any $\mathbf{f} \in \bar{\mathcal{F}}$, there exists an r such that $\mathbf{f}_r^l \leq \mathbf{f} \leq \mathbf{f}_r^u$ and $\|\mathbf{f}_r^l - \mathbf{f}_r^u\|_2 \leq \epsilon$; $r = 1, \dots, R$, where the

inequality sign is componentwise. Then the L_2 -metric entropy with bracketing $H_B(\epsilon, \bar{\mathcal{F}})$ is defined as the logarithm of the cardinality of the smallest ϵ -bracketing function set of $\bar{\mathcal{F}}$.

Let $\bar{\mathcal{F}}^L(t) = \{L_\Delta^T(\mathbf{f}, \mathbf{z}) - L_\Delta(\mathbf{f}^*, \mathbf{z}) : \mathbf{f} \in \bar{\mathcal{F}}(t)\}$ with $\bar{\mathcal{F}}(t) = \{\mathbf{f} \in \bar{\mathcal{F}} : J(\mathbf{f}) \leq J^*t\}$, $J(\mathbf{f}) = \frac{1}{2}\|\mathbf{f}\|_{\mathcal{K}}^2 = \frac{1}{2K} \sum_{j=1}^K \|f_j\|_{\mathcal{K}}^2$ and $J^* = \max(J(\mathbf{f}^*), 1)$.

Assumption C. (Metric entropy) For some constants $a_i > 0; i = 3, \dots, 5$ and $\epsilon_n > 0$,

$$\sup_{t \geq 2} \phi(\epsilon_n, t) \leq a_5 n^{1/2}, \quad (12)$$

where $\phi(\epsilon, t) = \int_{a_4 D}^{a_3^{1/2} D^{\beta/2}} H_B^{1/2}(w, \bar{\mathcal{F}}^L(t)) dw / D$, and $D = D(\epsilon, \lambda, t) = \min(\epsilon^2 + \lambda(t/2 - 1)J^*, 1)$.

Assumption D. (Smoothness) There exist some constants $a_6 > 0$, $a_7 > 0$ and $\delta_0 > 0$, such that for any $j = 1, \dots, K$, $f_j \in \mathcal{F}_j$ and $0 < \delta \leq \delta_0$,

$$\sup_{\|f_j\|_{\mathcal{K}} \leq a_6} \|\nabla_\delta f_j\|_1 \leq a_7,$$

where $\|\nabla_\delta f_j\|_1 = \delta^{-1} \sum_{r=1}^p E(|f_j(\mathbf{X} + \delta \mathbf{1}_r) - f_j(\mathbf{X})|)$ and $\mathbf{1}_r$ is a vector of length k with all elements being 0 but the r -th element being 1.

Assumption D requires that $f_j; j = 1, \dots, K$ satisfy a Lipschitz condition. This is usually met when \mathcal{F} is a RKHS according to the representer theorem (Kimeldorf and Wahba, 1971), which includes RKHS defined by linear, polynomial, Gaussian and spline kernels and so on.

Theorem 1 Under Assumptions A-D, for $\hat{\mathbf{f}}$ defined in (6) and (7), there exist constants $a_8 > 0$ and $a_9 > 0$ such that

$$P\left(e(\hat{\mathbf{f}}, \bar{\mathbf{f}}) \geq a_1 \delta_n^{2\alpha}\right) \leq 3.5 \exp(-a_8 n (\lambda J^*)^{2-\min(\beta, 1)}) + a_9 n^{-1}, \quad (13)$$

provided that $m \geq (2a_4^{-1} n \epsilon_n^{-2} \max_{1 \leq i, j \leq n} \|\mathbf{x}_i - \mathbf{x}_j\|_1)^p$ and $\lambda^{-1} \geq 2\delta_n^{-2} J^*$, where $\delta_n^2 = \min(1, \max(\epsilon_n^2, 2s_n) + 4T(T+1)(n+1)^{-1})$.

Remarks: Theorem 1 and Corollary 1 continue to hold when the L_2 -metric entropy in Assumption C is replaced by a uniform entropy, c.f., Theorem 1 of Shen and Wang (2007). Here the uniform entropy is defined as $H_U(w, \bar{\mathcal{F}}^L(t)) = \sup_Q H_{B,Q}(w, \bar{\mathcal{F}}^L(t))$, where $H_{B,Q}(w, \bar{\mathcal{F}}^L(t))$ is a L_2 -metric entropy with $\|g\|_2 = (\int g^2 dQ)^{1/2}$. The proof requires a slight modification, and the constant 3.5 on the right hand side of (13) becomes $1 + (20(1 - (32a_8n(\lambda J^*)^{2-\min(\beta,1)} - 1)^{-1})^{1/2}$.

Appendix B: Technical Proofs

Lemma 1 *A global minimizer of $El_\Delta(\mathbf{Y}, \mathbf{f}(\mathbf{X}))$ over all possible \mathbf{f} 's is $\bar{\mathbf{f}}$.*

Proof of Lemma 1: For l_Δ , note that the global minimizer of $El_\Delta(\mathbf{Y}, \text{Sgn}(\mathbf{f}(\mathbf{X}))) = K^{-1} \sum_{j=1}^K EI(Y_{(j)} \neq \text{sign}(f_j(\mathbf{X})))$ is identical to that of $EI(Y_{(j)} \neq \text{sign}(f_j(\mathbf{X})))$ for each j . It can be shown as in Lin (2002) that the global minimizer is $\bar{f}_j(\mathbf{x}) = \text{sign}(P(Y_{(j)} = 1 | \mathbf{X} = \mathbf{x}) - 1/2)$, satisfying (1).

Proof of Theorem 1: It follows from the definition of $\hat{\mathbf{f}}$ and $L_\Delta^T \leq L_\Delta$ that

$$\begin{aligned}
P(e_{L^T}(\hat{\mathbf{f}}, \bar{\mathbf{f}}) \geq \delta_n^2) &\leq P(\max_{j,r} |\hat{f}_j(\mathbf{X} + \delta \mathbf{1}_r) - \hat{\mathbf{f}}_j(\mathbf{X})| \geq n\delta) + \\
&P^* \left(\sup_{\{\mathbf{f} \in \tilde{\mathcal{F}}^n, e_{L^T}(\mathbf{f}, \bar{\mathbf{f}}) \geq \delta_n^2\}} n^{-1} \sum_{i=1}^n (\tilde{L}_\Delta(\mathbf{f}^*, \mathbf{Z}_i) - \tilde{L}_\Delta^T(\mathbf{f}, \mathbf{Z}_i)) \geq 0 \right) \\
&\leq P(\max_{j,r} |\hat{f}_j(\mathbf{X} + \delta \mathbf{1}_r) - \hat{f}_j(\mathbf{X})| \geq n\delta) + P(\mathbf{X} \notin \Omega) + \\
&P^* \left(\sup_{\{\mathbf{f} \in \tilde{\mathcal{F}}^n, e_{L^T}(\mathbf{f}, \bar{\mathbf{f}}) \geq \delta_n^2\}} n^{-1} \sum_{i=1}^n (\tilde{L}_\Delta(\mathbf{f}^*, \mathbf{Z}_i) - \tilde{L}_\Delta^T(\mathbf{f}, \mathbf{Z}_i)) \geq 0, \mathbf{X} \in \Omega \right),
\end{aligned} \tag{14}$$

where P^* denotes the outer probability measure on Ω , Ω is the convex hull spanned by X_1, \dots, X_n , $\tilde{\mathcal{F}}^n = \{\mathbf{f} \in \tilde{\mathcal{F}} : \max_{j,r} |f_j(\mathbf{X} + \delta \mathbf{1}_r) - \mathbf{f}_j(\mathbf{X})| \leq n\delta\}$, $n^{-1} \sum_{i=1}^n \tilde{L}_\Delta(\mathbf{f}, \mathbf{Z}_i)$ is the penalized cost function to be minimized with $\tilde{L}_\Delta(\mathbf{f}, \mathbf{Z}_i) = L_\Delta(\mathbf{f}, \mathbf{Z}_i) + \lambda J(\mathbf{f})$, and $\tilde{L}_\Delta^T(\mathbf{f}, \mathbf{Z}_i) = L_\Delta^T(\mathbf{f}, \mathbf{Z}_i) + \lambda J(\mathbf{f})$. We will proceed to bound the three terms in (14) separately.

To bound $P(\max_{j,r} |\hat{f}_j(\mathbf{X} + \delta \mathbf{1}_r) - \hat{\mathbf{f}}_j(\mathbf{X})| \geq \mathbf{n}\delta)$, note that $\sum_{j=1}^K EJ(\hat{f}_j)$ is bounded by some constant, c.f., Lemma 5 of Wang and Shen (2007). This, together with Assumption D, implies that there exists a constant $a_7 > 0$ such that $E \sum_{j=1}^K \|\nabla \hat{f}_j\|_1 \leq a_7$, implying that $E \max_{j,r} \delta^{-1} |\hat{f}_j(\mathbf{X} + \delta \mathbf{1}_r) - \hat{\mathbf{f}}_j(\mathbf{X})| \leq \mathbf{a}_7$. By Markov's inequality, $P(\max_{j,r} |\hat{f}_j(\mathbf{X} + \delta \mathbf{1}_r) - \hat{\mathbf{f}}_j(\mathbf{X})| \geq \mathbf{n}\delta) \leq \mathbf{a}_7/\mathbf{n}$.

To bound $P(\mathbf{X} \notin \Omega)$, we first consider the one-dimensional case, where Ω becomes an interval $[\min_i \mathbf{X}_i, \max_i \mathbf{X}_i]$. Then $P(\mathbf{X} \notin \Omega) = 2/(n+1)$ by noting that $\mathbf{X}, \mathbf{X}_1, \dots, \mathbf{X}_n$ are i.i.d. samples from the distribution of \mathbf{X} . Next, suppose that the dimension is greater than 1. Note that $\Omega = \prod_{i=1}^p [\min_i X_{i1}, \max_i X_{i1}]$, implying that $P(\mathbf{X} \notin \Omega) = 1 - P(\mathbf{X} \in \Omega) = 1 - (1 - 2/(n+1))^p \leq 2p/(n+1)$.

Finally, we bound the third term in (14) by applying Theorem 1 of Shen and Wang (2007). Here it suffices to verify Assumptions A-C in Shen and Wang (2007). First, Assumptions A-B are satisfied by replacing ϵ there by $\epsilon + 4T(T+1)/(n+1)$,

$$|e_{L^T}(\mathbf{f}, \bar{\mathbf{f}}) - \tilde{e}_{L^T}(\mathbf{f}, \bar{\mathbf{f}})| \leq 4T/(n+1) \quad \text{and}$$

$$|\text{Var}_{\Omega}(L_{\Delta}^T(\mathbf{f}, \mathbf{Z}) - L_{\Delta}(\bar{\mathbf{f}}, \mathbf{Z})) - \text{Var}(L_{\Delta}^T(\mathbf{f}, \mathbf{Z}) - L_{\Delta}(\bar{\mathbf{f}}, \mathbf{Z}))| \leq 4T^2/(n+1),$$

where $\tilde{e}_{L^T}(\mathbf{f}, \bar{\mathbf{f}}) = E_{\Omega}(L_{\Delta}^T(\mathbf{f}, \mathbf{Z})) - E_{\Omega}(L_{\Delta}^T(\bar{\mathbf{f}}, \mathbf{Z}))$, and E_{Ω} and Var_{Ω} are taken with respect to $\mathbf{X} \in \Omega$. Next, Assumption C is satisfied for $\bar{\mathcal{F}}$ by Assumption C in Appendix A. It remains to show that the entropy inequality (12) also holds for $\tilde{\mathcal{F}}^n$ or $\tilde{\mathcal{F}}$ through the relationship between $\tilde{\mathcal{F}}$ and $\bar{\mathcal{F}}$. Note that the support points $\tilde{\mathbf{x}}_t$ are uniformly spaced inside the convex hull Ω and $a_4 m^{1/p} \geq 2n\epsilon_n^{-2} \max_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_1$. For any point $\mathbf{x} \in \Omega$, there exists a support point $\tilde{\mathbf{x}}_t$ such that $\|\mathbf{x} - \tilde{\mathbf{x}}_t\|_1 \leq a_4 \epsilon_n^2 / 2n$, implying that $|f_j(\mathbf{x}) - f_j(\tilde{\mathbf{x}}_t)| \leq a_4 \epsilon_n^2 / 2$ and

$$f_{\text{par}(j)}(\mathbf{x}) - f_j(\mathbf{x}) \geq f_{\text{par}(j)}(\tilde{\mathbf{x}}_t) - f_j(\tilde{\mathbf{x}}_t) - |f_{\text{par}(j)}(\mathbf{x}) - f_{\text{par}(j)}(\tilde{\mathbf{x}}_t)| - |f_j(\mathbf{x}) - f_j(\tilde{\mathbf{x}}_t)| \geq -a_4 \epsilon_n^2,$$

for all j . Therefore, $\bar{\mathcal{F}} \subset \tilde{\mathcal{F}} \subset \{\mathbf{f} : f_j \in \mathcal{F}_j, f_{\text{par}(j)}(\mathbf{x}) - f_j(\mathbf{x}) \geq -a_4 \epsilon_n^2, \forall j\}$, and hence

$$H_B(w, \bar{\mathcal{F}}) \leq H_B(w, \tilde{\mathcal{F}}) \leq H_B(w, \{\mathbf{f} : f_j \in \mathcal{F}_j, f_{\text{par}(j)}(\mathbf{x}) - f_j(\mathbf{x}) \geq -a_4 \epsilon_n^2, \forall j\}) = H_B(w, \bar{\mathcal{F}}),$$

for all $w \geq a_4 \epsilon_n^2$, where $\tilde{\mathcal{F}}^L(t) = \{L_\Delta^T(\mathbf{f}, \mathbf{z}) - L_\Delta(\mathbf{f}^*, \mathbf{z}) : f \in \tilde{\mathcal{F}}(t)\}$, implying $H_B(w, \bar{\mathcal{F}}) = H_B(w, \tilde{\mathcal{F}})$. Similarly, $H_B(w, \bar{\mathcal{F}}^L(t)) = H_B(w, \tilde{\mathcal{F}}^L(t))$ for all $w \geq a_4 \epsilon_n^2$ with $\tilde{\mathcal{F}}^L(t) = \{L_\Delta^T(\mathbf{f}, \mathbf{z}) - L_\Delta(\mathbf{f}^*, \mathbf{z}) : \mathbf{f} \in \tilde{\mathcal{F}}, J(\mathbf{f}) \leq t\}$. Then the entropy inequality for $\tilde{\mathcal{F}}^L(t)$ follows from (12). The result then follows.

Appendix C: Verification of Assumptions in Section 5.2

Linear learning: We now verify Assumptions A-D for HSVM and HPSI. Assumptions A and B are met for HSVM according to Lemma 3 of Shen and Wang (2007) with $\mathbf{f}^* = \arg \inf_{\mathbf{f} \in \mathcal{F}} EL_\Delta(\mathbf{f}, \mathbf{Z})$, $\alpha = 1/2$ and $\beta = 1$ for HSVM, and $f_j^* = n(x_{(1)} + 1 - 2j/(K + 1))$, $\alpha = 1$ and $\beta = 1$ for HPSI. For Assumption C, it follows from Lemma 3 of Wang and Shen (2007) and Lemma 2(a) that $H_B(\epsilon, \bar{\mathcal{F}}^L(t)) \leq O(\min(K, (1/\epsilon)^{1/2}) \log(1/\epsilon))$. Easily, $\sup_{t \geq 2} \phi(\epsilon_n, t) \leq O((\min(K, (1/\epsilon_n^2)^{1/2}) \log(1/\epsilon_n^2))^{1/2}/\epsilon_n)$ in (12). Solving (12) yields $\epsilon_n^2 = \min(K, n^{1/2})n^{-1} \log n$ when $\lambda J^* \sim \epsilon_n^2$. Assumption D is satisfied for the linear kernel.

Kernel learning: We now verify Assumptions A-D for HSVM and HPSI, respectively.

HSVM: For Assumption A, let $f_j^* = 1 - (1 + \exp(\tau(x_{(1)} + 1 - 2j/(K + 1))))^{-1}$, then it can be verified that $e_L(\mathbf{f}^*, \bar{\mathbf{f}}) = O(K\tau^{-1})$ and $J(\mathbf{f}^*) = O(Ke^{\tau^2 \sigma_n^2})$. Assumption B is met for HSVM with $\alpha = \beta = 1$ by Lemmas 6 and 7 of Shen and Wang (2007). For Assumption C, it follows from Lemma 7 of Wang and Shen (2007) and Lemma 2(b) that $H_B(\epsilon, \bar{\mathcal{F}}^L(t)) \leq O(\min(K, (1/\epsilon)^{2/9})(\log((J^*t)^{1/2}/\epsilon) + \sigma_n^{-2})^3)$. Plugging it into (12), we have $\sup_{t \geq 2} \phi(\epsilon_n, t) \leq O((\min(K, (1/\epsilon_n^2)^{2/9})(\log((J^*)^{1/2}/\epsilon_n^2) + \sigma_n^{-2})^3)^{1/2}/\epsilon_n)$. Solving (12) yields $\epsilon_n^2 = \min(K, n^{2/9})n^{-1}(\log((J^*n/K)^{1/2}) + \sigma_n^{-2})^3$ when $\lambda J^* \sim \epsilon_n^2$. Assumption D is satisfied with the Gaussian kernel, c.f., Smola, Schölkopf and Müller (1998).

HPSI: For Assumption A, let $f_j^* = \tau(x_{(1)} + 1 - 2j/(K + 1))$ with $\tau > 0$ to be specified,

then it can be verified that $e_L(\mathbf{f}^*, \bar{\mathbf{f}}) = O(K\tau^{-1})$ and $J(\mathbf{f}^*) = O(K\tau^2\sigma_n^2)$. Assumption B is met with $\alpha = \beta = 1$ by Theorem 3.1 of Liu and Shen (2006). For Assumption C, as in the HSVM case, solving (12) yields $\epsilon_n^2 = \min(K, n^{2/9})n^{-1}(\log((J^*n/K)^{1/2}) + \sigma_n^{-2})^3$ when $\lambda J^* \sim \epsilon_n^2$. Assumption D is met with the Gaussian kernel.

Lemma 2 *In the example in Section 5.2, assume that the hierarchy is a nested sequence $1 \rightarrow 2 \rightarrow \dots \rightarrow K$ and $\mathcal{F}_1(t) = \dots = \mathcal{F}_K(t)$ with $\mathcal{F}_j(t) = \{f_j \in \mathcal{F}_j : \|f_j\|_{\mathcal{K}}^2 \leq J^*t\}$; $j = 1, \dots, K$.*

(a) *If $\mathcal{F}_1(t)$ is defined by polynomial kernel of order m , i.e., $\mathcal{K}(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^T \mathbf{z})^m$, then the L_2 ϵ -bracketing entropy of $\bar{\mathcal{F}}(t)$ is*

$$H_B(\epsilon, \bar{\mathcal{F}}(t)) \prec K \left(H_B(\epsilon, \mathcal{F}_1(t)) - \frac{1}{2}(m^2 + 3m) \log K \right),$$

where $u \prec v$ means u/v is bound by some constant.

(b) *If $\mathcal{F}_1(t)$ is defined by Gaussian kernel, then the L_2 ϵ -bracketing entropy of $\bar{\mathcal{F}}(t)$ is*

$$H_B(\epsilon, \bar{\mathcal{F}}(t)) \prec K \left(H_B(\epsilon, \mathcal{F}_1(t)) - \frac{1}{2}(M^2 + 3M) \log K \right),$$

where M is the minimal integer greater than $3 \log(T/\epsilon) + 160 \log 2/\sigma^2 + 5$.

Proof of Lemma 2: (a) Polynomial kernel: For any $\mathbf{f} \in \bar{\mathcal{F}}(t)$, constraints (5) imply that $f_1 \geq f_2 \geq \dots \geq f_K$. Without loss of generality, we assume that sets $\{\mathbf{x} : f_1(\mathbf{x}) < 0\}$ and $\{\mathbf{x} : f_K(\mathbf{x}) > 0\}$ are not empty. Otherwise, set $f_1(\mathbf{x}) = x_{(1)} + 1$ and $f_K(\mathbf{x}) = x_{(1)} - 1$. Choose points $\mathbf{u}_1, \dots, \mathbf{u}_S$ from $\{\mathbf{x} : f_1(\mathbf{x}) < 0\}$ and $\mathbf{v}_1, \dots, \mathbf{v}_S$ from $\{\mathbf{x} : f_K(\mathbf{x}) > 0\}$ with $S = (m^2 + 3m)/2$. By construction, there exist distinct \mathbf{w}_{sj} such that $f_j(\mathbf{w}_{sj}) = 0$; $j = 1, \dots, K$. This is because $f_j(\mathbf{x})$ is continuous in \mathbf{x} , and $f_j(\mathbf{u}_s) < 0$ and $f_j(\mathbf{v}_s) > 0$ for all j and s . Next we permute each row of $\mathbf{W} = (\mathbf{w}_{sj})_{S \times K}$ to obtain the permuted $\widetilde{\mathbf{W}}$. Note that each $\widetilde{\mathbf{W}}$ corresponds to an \mathbf{f} in $\mathcal{F}(t)$ in that the j -th column of $\widetilde{\mathbf{W}}$ uniquely determines

$f_j(\mathbf{x}) = 0$ because $f_j(\mathbf{x})$ is a polynomial having no more than S unknown coefficients. On the other hand, only \mathbf{W} corresponds to an \mathbf{f} in $\bar{\mathcal{F}}(t)$ given the constraints defined by the hierarchy. Therefore, an \mathbf{f} in $\bar{\mathcal{F}}(t)$ corresponds to $(K!)^S$ \mathbf{f} 's in $\mathcal{F}(t)$. This implies that $H_B(\epsilon, \bar{\mathcal{F}}(t)) \prec \log(e^{H_B(\epsilon, \mathcal{F}(t))}/(K!)^S) \prec H_B(\epsilon, \mathcal{F}(t)) - S \log(K!) \prec K(H_B(\epsilon, \mathcal{F}_1(t)) - S \log K)$.

(b) Gaussian kernel: As showed in Zhou (2002), for any $f \in \mathcal{F}_1(t)$, there exists a polynomial function g of order M such that $\|g - f\|_K \leq \epsilon$. Therefore, the L_2 ϵ -bracketing entropy of $\bar{\mathcal{F}}(t)$ with Gaussian kernel can be bounded by that of $\bar{\mathcal{F}}(t)$ with polynomial kernel of order M . The result then follows from (a).

References

- [1] AN, L. AND TAO, P. (1997). Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *J. Global Optimization*, **11**, 253-285.
- [2] CAI, L. AND HOFMANN, T. (2004). Hierarchical document categorization with support vector machines. In *ACM Conf. on Information and Knowledge Management (CIKM)*.
- [3] CESA-BIANCHI, N., GENTILE, C. AND ZANIBONI, L. (2006). Incremental algorithm for hierarchical classification *J. Mach. Learn. Res.*, **7**, 31-54.
- [4] CESA-BIANCHI, N., CONCONI, A. AND GENTILE, C. (2004). Regret bounds for hierarchical classification with linear-threshold functions. *Proc. of Annual Conf. on Comp. Learn. Theory*, 93-108.
- [5] CORTES, C. AND VAPNIK, V. N. (1995). Support-vector networks. *Mach. Learn.*, **20**, 273-97.

- [6] EISEN, M., SPELLMAN, P., BROWNDAGGER, P. AND BOTSTEIN, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, **95**, 14863-14868.
- [7] GU, C. (2000). Multidimension smoothing with splines. *Smoothing and Regression: Approaches, Computation and Application*, Schimek.
- [8] GUYON, I., WESTON, J. AND VAPNIK, V. (2002). Gene selection for cancer classification using support vector machine. *Mach. Learn.* **46**, 389-422.
- [9] HUGHES, T., MARTON, M., JONES, A., ROBERTS, C., STOUGHTON, R., ARMOUR, C., BENNETT, H., COFFEY, E., DAI, H., HE, Y., KIDD, M., KING, A., MEYER, M., SLADE, D., LUM, P., STEPANIANTS, S., SHOEMAKER, D., GACHOTTE, D., CHAKRABURTTY, K., SIMON, J., BARD, M. AND FRIEND, S. (2000). Functional discovery via a compendium of expression profiles. *Cell*, **102**, 109-126.
- [10] JAAKKOLA, T., DIEKHANS, M. AND HAUSSLER, D. (1999). Using the Fisher kernel method to detect remote protein homologies. In *Proc. of Int. Conf. on Intelligent Systems for Molecular Biology*, 149-158.
- [11] JOACHIMS, T. (1998). Text categorization with support vector machines: learning with many relevant features. In Nedellec, C., and Rouveirol, C., editors, Proc. of the 10th European Conf. on Machine Learning (ECML1998), **1398**, 117-142. Springer Verlag.
- [12] KIMELDORF, G. AND WAHBA, G. (1971). Some results on Tchebychean spline functions. *J. Math. Anal. Applic.*, **33**, 82-95.
- [13] LEWIS, D. (1998). Naiver (Bayes) at forty: The independence assumption in information retrieval. In *Proc. of European Conf. on Mach. Learn. (ECML1998)*, 4-15.

- [14] LIN, Y. (2002). Support vector machines and the Bayes rule in classification. *Data Mining and Knowledge Discovery*, **6**, 259-275.
- [15] LIU, S., SHEN, X. AND WONG, W. (2005). Computational development of ψ -learning. *Proc. SIAM Int. Data Mining Conf.*, 1-12.
- [16] LIU, Y. AND SHEN X. (2006). Multicategory ψ -learning. *J. Amer. Statist. Assoc.*, **101**, 500-509.
- [17] MASON, P., BAXTER, L., BARTLETT, J. AND FREAN, M. (2000). Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, **12**, 512-518. The MIT Press.
- [18] MEWES, H., ALBERMANN, K., HEUMANN, K., LIEBL, S. AND PFEIFFER, F. (2002). MIPS: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Research*, **25**, 28-30.
- [19] Rousu, J., Saunders, C., Szedmak, S. and Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *J. of Mach. Learn. Res.*, **7**, 1601-1626.
- [20] SCHAPIRE, R., SINGER, Y. AND SINGHAL, A. (1998). Boosting and Rochio applied to text filering. In *Proc. of Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 215-223.
- [21] SCHÖLKOPF, B., SMOLA, A., WILLIAMSON, R. AND BARTLETT, P. (2000). New support vector algorithms. *Neural Computation*, **12**, 1207-1245.
- [22] SHAHBABA, B. AND NEAL, R. (2007). Improving Classification When a Class Hierarchy is Available Using a Hierarchy-Based Prior. *Bayesian Analysis*, **2**, 221-238.
- [23] SHEN, X., SHI, J. AND WONG, W. H. (1999). Random sieve likelihood and general regression models. *J. Amer. Statist. Assoc.*, **94**, 835-846.

- [24] SHEN, X., TSENG, G., ZHANG, X. AND WONG, W. H. (2003). On ψ -learning. *J. Ameri. Statist. Assoc.*, **98**, 724-734.
- [25] SHEN, X. AND WANG, L. (2006). Discussion of 2004 IMS Medallion Lecture: “Local Rademacher complexities and oracle inequalities in risk minimization”. *Ann. Statist.*, **34**, 2677-2680.
- [26] SHEN, X. AND WANG, L. (2007). Generalization error for multi-class margin classification. *Electronic J. of Statist.*, **1**, 307-330.
- [27] SMOLA, A., SCHÖKOPF, B. AND MÜLLER, K. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, **11**, 637-649.
- [28] TSOCHANTARIDIS, I., JOACHIMS, T., HOFMANN, T. AND ALTUN Y. (2005). Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, **6**, 1453-1484.
- [29] WANG, J. AND SHEN, X. (2007). Large margin semi-supervised learning. *J. Mach. Learn. Res.*, **8**, 1867-1891.
- [30] WANG, J., SHEN, X. AND LIU, Y. (2008). Probability estimation for large margin classifiers. *Biometrika*, **95**, 149-167.
- [31] WAHBA, G. (1990). *Spline models for observational data*. CBMS-NSF Regional Conf. Series, Philadelphia.
- [32] XIAO, G. AND PAN, W. (2005). Gene function prediction by a combined analysis of gene expression data and protein-protein interaction data. *J. Bioinformatics and Computat. Biology*, **3**, 1371-1390.

- [33] YANG, Y. AND LIU, X. (1999). A reexamination of text categorization methods. In Proc. of the 22nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 42-49.
- [34] ZHOU, X., KAO, M. AND WONG, W. (2002). Transitive functional annotation by shortest-path analysis of gene expression data. *Proc. Nat. Acad. Sci.*, **99**, 12783-12788.
- [35] ZHU, J. AND HASTIE, T. (2005). Kernel logistic regression and the import vector machine. *J. Comp. and Graph. Statist.*, **14**, 185-205.

Figure 1: Plot of ψ , ψ_1 and ψ_2 for the DC decomposition of $\psi = \psi_1 - \psi_2$. Solid, dotted and dashed lines represent ψ , ψ_1 and ψ_2 , respectively.

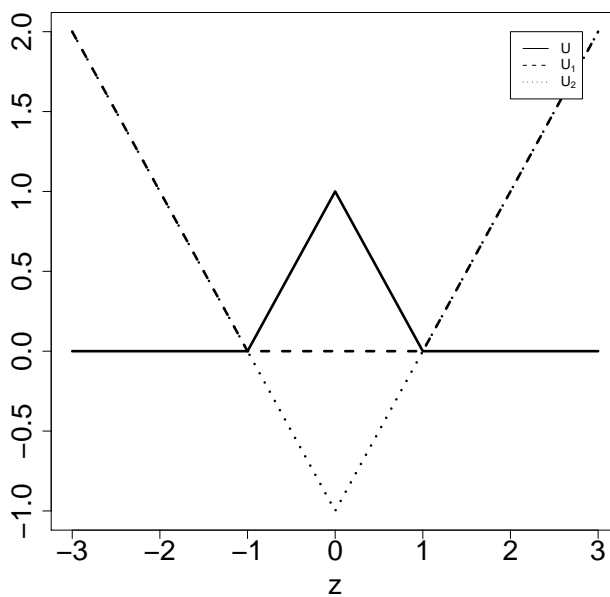


Table 1: Averaged test errors as well as estimated standard errors (in parenthesis) of HSVM, HPSI, HoffSVM, ParSVM and FlatSVM, over 100 pairs of training and testing samples, in Examples 1 and 2. The smallest test error in each row is boldfaced. Here TE_{0-1} , TE_{Δ} , TE_{sub} and TE_{sib} are different generalization error rates, Bayes is the Bayes error and Impro. is the amount of improvement.

Metric	FlatSVM	HoffSVM	ParSVM	HSVM	HPSI	Bayes
<i>Example 1: Linear</i>						
TE_{0-1}	.760(.0062)	–	.790(.0055)	.617(.0052)	.598(.0059)	53/125
Impro.	–	–	8.9%	42.6%	48.2%	–
TE_{Δ}	.176(.0016)	–	.186(.0012)	.153(.0015)	.147(.0012)	243/2000
Impro.	–	–	-18.3%	42.2%	53.2%	–
<i>Example 1: Gaussian</i>						
TE_{0-1}	.797(.0054)	–	–	.718(.0048)	.713(.0048)	53/125
Impro.	–	–	–	21.2%	22.5%	–
TE_{Δ}	.221(.0029)	–	–	.204(.0023)	.203(.0024)	243/2000
Impro.	–	–	–	17.1%	18.1%	–
<i>Example 2: Linear</i>						
TE_{0-1}	.685(.0081)	.613(.0010)	.363(.0049)	.414(.0121)	.409(.0121)	9/110
Impro.	–	11.9%	53.4%	44.9%	45.8%	–
TE_{Δ}	.116(.0032)	.085(.0022)	.037(.0005)	.049(.0026)	.048(.0026)	9/1100
Impro.	–	28.8%	73.3%	62.1%	63.1%	–
TE_{sub}	.351(.0061)	.316(.0022)	.197(.0033)	.229(.0077)	.228(.0075)	9/220
Impro.	–	11.3%	43.3%	39.3%	39.7%	–
TE_{sib}	.685(.0081)	.613(.0010)	.363(.0049)	.414(.0121)	.409(.0121)	9/110
Impro.	–	11.9%	53.4%	44.9%	45.8%	–
<i>Example 2: Gaussian</i>						
TE_{0-1}	.828(.0022)	.822(.0013)	–	.806(.0022)	.806(.0021)	9/110
Impro.	–	0.8%	–	2.9%	2.9%	–
TE_{Δ}	.169(.0015)	.316(.0035)	–	.157(.0012)	.157(.0012)	9/1100
Impro.	–	-91.4%	–	7.5%	7.5%	–
TE_{sub}	.503(.0025)	.576(.0067)	–	.485(.0019)	.485(.0018)	9/220
Impro.	–	-15.8%	–	3.9%	3.9%	–
TE_{sib}	.828(.0022)	.822(.0013)	–	.806(.0022)	.806(.0021)	9/110
Impro.	–	0.8%	–	2.9%	2.9%	–

Table 2: Averaged test errors as well as estimated standard errors (in parenthesis) of HSVM, HPSI, HoffSVM, ParSVM and FlatSVM, over 100 pairs of training and testing samples, in the gene prediction example. The smallest test error in each row is boldfaced. Here TE_{0-1} , TE_{Δ} , TE_{sub} and TE_{sib} are different generalization error rates, and Impro. is the amount of improvement.

Metric	FlatSVM	ParSVM	HoffSVM	HSVM	HPSI
<i>Linear</i>					
TE_{0-1}	.979(.0030)	.972(.0019)	.989(.0010)	.956(.0042)	.958(.0005)
Impro.	—	0.7%	-1.0%	2.3%	2.1%
TE_{Δ}	.187(.0006)	.307(.0014)	.288(.0007)	.181(.0006)	.182(.0005)
Impro.	—	-64.2%	-54.0%	3.2%	2.7%
TE_{sub}	.626(.0018)	.665(.0017)	.693(.0014)	.601(.0018)	.596(.0009)
Impro.	—	-6.2%	-10.7%	4.0%	4.8%
TE_{sib}	.628(.0018)	.688(.0017)	.738(.0013)	.603(.0018)	.600(.0010)
Impro.	—	-9.6%	-17.5%	4.0%	4.5%
<i>Gaussian</i>					
TE_{0-1}	.986(.0019)	—	.990(.0017)	.946(.0017)	.883(.0016)
Impro.	—	—	-0.4%	4.1%	10.4%
TE_{Δ}	.167(.0003)	—	.329(.0032)	.165(.0005)	.161(.0005)
Impro.	—	—	-97.0%	1.2%	3.6%
TE_{sub}	.625(.0016)	—	.708(.0035)	.623(.0024)	.608(.0008)
Impro.	—	—	-13.3%	0.3%	2.7%
TE_{sib}	.622(.0025)	—	.742(.0033)	.620(.0016)	.611(.0010)
Impro.	—	—	-19.3%	0.3%	1.8%

Table 3: Verification of top 10 gene predictions made by HPSI using the 2008 version of MIPS, where corresponding nodes are given in parenthesis.

Gene	Predicted function category	Verified
YDR013W	DNA synthesis and replication (030103)	+
YDR279W	DNA synthesis and replication (030103)	+
YGL015C	cell cycle (0303)	—
YML118W-a	mitotic cell cycle and cell cycle control (030301)	P2
YOR039W	mitotic cell cycle and cell cycle control (030301)	P2
YLR445W	mRNA synthesis (040501)	—
YNL023C	mRNA synthesis (040501)	P1
YPR009W	mRNA synthesis (040501)	P1
YML069W	mRNA synthesis (040501)	+
YPL007C	mRNA synthesis (040501)	P1

Table 4: Averaged proportion of inconsistent predictions and test errors TE_{Δ} on test set as well as estimated standard errors (in parenthesis).

	$m = 300$ $p = 30$	$m = 300$ $p = 100$	$m = 300$ $p = 303$	$m = 600$ $p = 303$
<i>Without top-down rule</i>				
Proportion	2.5E-4(7.9E-5)	2.3E-2(9.1E-4)	1.8E-1(2.6E-3)	9.1E-2(1.3E-3)
TE_{Δ}	.196(.0004)	.192(.0003)	.190(.0004)	.184(.0004)
<i>With top-down rule</i>				
Proportion	0	0	0	0
TE_{Δ}	.196(.0004)	.191(.0003)	.181(.0006)	.179(.0004)

Figure 2: The left panel displays the hierarchical structures in Example 1 and the right panel displays the hierarchical structures in Example 2.

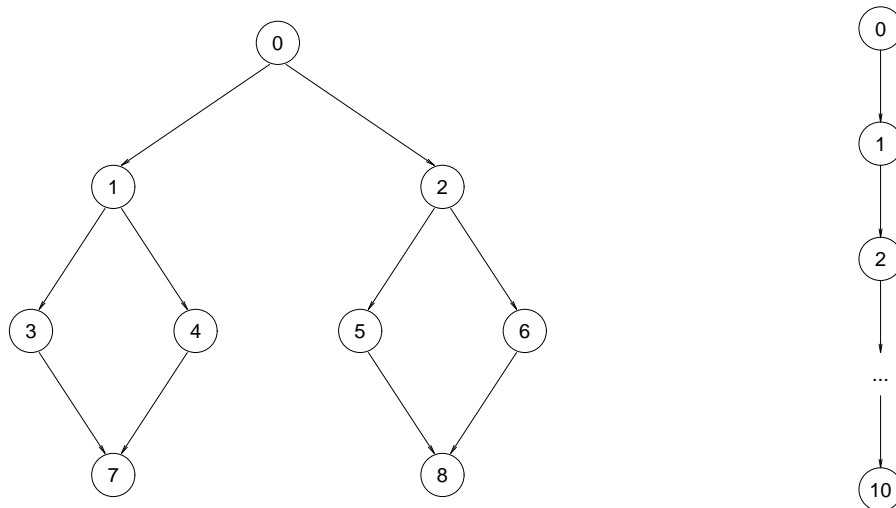


Figure 3: Sensitivity study for various values of m in Example 1.

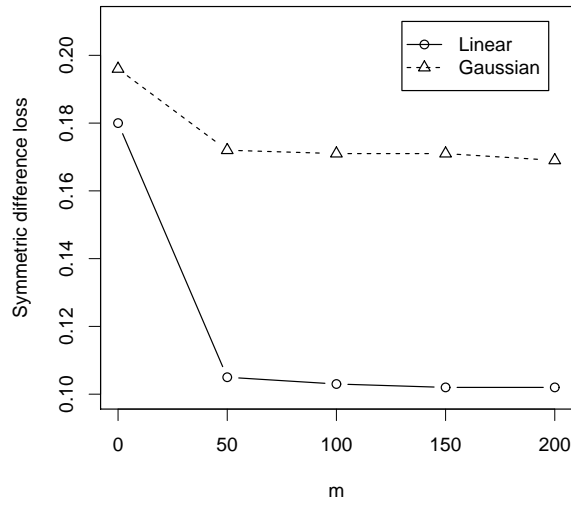


Figure 4: Hierarchical structure in gene function prediction example, where the number inside each circle stands for the function class within its parent class. To identify the exact function class for each node, we need to combine all numbers from its ancestors at the first level to itself. For example, the grey node stands for function class 03010501, which is “DNA repair” according to MIPS.

