Hierarchical Classification: Combining Bayes with SVM

Nicolò Cesa-Bianchi CESA-BIANCHI@DSI.UNIMI.IT
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39, 20135 Milano, Italy

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
Dipartimento di Informatica e Comunicazione
Università dell'Insubria
via Mazzini 5, 21100 Varese, Italy

Luca Zaniboni ZANIBONI@DTI.UNIMI.IT
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
via Bramante 65, 26013 Crema (CR), Italy

Abstract

We study hierarchical classification in the general case when an instance could belong to more than one class node in the underlying taxonomy. Experiments done in previous work showed that a simple hierarchy of Support Vectors Machines (SVM) with a top-down evaluation scheme has a surprisingly good performance on this kind of task. In this paper, we introduce a refined evaluation scheme which turns the hierarchical SVM classifier into an approximator of the Bayes optimal classifier with respect to a simple stochastic model for the labels. Experiments on synthetic datasets, generated according to this stochastic model, show that our refined algorithm outperforms the simple hierarchical SVM. On real-world data, however, the advantage brought by our approach is a bit less clear. We conjecture this is due to a higher noise rate for the training labels in the low levels of the taxonomy.

1. Introduction

In hierarchical classification, class labels are arranged as nodes in a tree forest to represent a given taxonomy. Each data instance is labelled with a (possibly empty) set of the nodes which we call a *multilabel*. It is assumed that whenever a multilabel contains a certain node i in a tree of the taxonomy, then it must also contain all the nodes along the path connecting the tree root to node i.

The problem of hierarchical classification, especially of textual information, has been extensively investigated in past years (Dumais & Chen, 2000; Dekel et al., 2004a; Dekel et al., 2004b; Granitzer, 2003; Hofmann et al., 2003; Koller & Sahami, 1997; McCallum et al., 1998; Mladenic, 1998; Ruiz & Srinivasan, 2002; Sun & Lim, 2001). On the other hand, the more general case investigated in this paper, where the multilabel of an instance can include multiple and partial paths in the hierarchy forest, has received much less attention (Cesa-Bianchi et al., 2004; Cesa-Bianchi et al., 2005; Rousu et al., 2005; Szdemak et al., 2005).

In Cesa-Bianchi et al. (2005) the H-loss function is introduced as performance measure in hierarchical classification problems. Given a pair of multilabels, the H-loss is computed by examining the paths where the two multilabels differ (see Section 2 for a formal definition). As previoulsy noted in the literature, a simple hierarchical version of the standard Support Vector Machines (Cortes & Vapnik, 1995; Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002), called H-SVM, performs surprisingly well on the H-loss even if it is not designed to do so. This left open the problem of improving the classification performance using an algorithm explicitly designed for the purpose of mini-

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

mizing the H-loss.

In this paper, we introduce a new learning algorithm, called B-SVM, that differs from H-SVM mainly in the *evaluation* phase (namely, in assigning multilabels to instances). In fact, whereas H-SVM assigns labels to nodes following a top-down procedure, the evaluation of an instance in B-SVM is based on a recursive bottom-up procedure driven by partial evaluations of the H-loss in the subtrees of the hierarchy. This procedure is designed to make the B-SVM hierarchical classifier approximate the Bayes optimal classifier for the H-loss. The stochastic model we use to define Bayes optimality follows Cesa-Bianchi et al. (2005): Multilabels are generated in such a way that child nodes receive labels that are independent conditioned on the father's label.

Our Bayesian evaluation mechanism is based on a combination of good local classification power, obtained through the weights produced by the SVMs sitting at each node, and good probability estimation power, obtained by fitting sigmoids on the weights, as in Platt (1999).

It is worth emphasizing that since the Bayesian recursive evaluation is, to a large extent, independent of the training algorithm, we might replace SVM by other algorithms more geared at probability estimation, such as regularized logistic regression (Hastie et al., 2001). Thus, we believe the main contribution of this paper mainly affects Machine Learning practice: we propose and investigate a *modular* architecture for hierarchical classification based on off-the-shelf techniques.

The paper is organized as follows. In Section 2 we recall the main definitions, including the H-loss function. Section 3 presents the stochastic model for the generation of hierarchically labelled data and the corresponding Bayes optimal classifier for this model. In Section 4 we introduce an approximation of the Bayes optimal classifier based on SVM, and also recall the baseline algorithm H-SVM employed in previous work (Cesa-Bianchi et al., 2005; Rousu et al., 2005). Section 5 illustrates the datasets used in our experiments and discusses the empirical performance of the two algorithms. Section 6 is devoted to conclusions and ongoing research.

2. Definitions and notation

We assume data elements are encoded as unit-norm vectors $\boldsymbol{x} \in \mathbb{R}^d$, which we call *instances*. A *multilabel* for an instance $\boldsymbol{x}$ is any subset of the set $\{1, \ldots, N\}$ of all labels, including the empty set. We represent the multilabel of $\boldsymbol{x}$ with a vector $\boldsymbol{v} = (v_1, \ldots, v_N) \in \{0,1\}^N$, where $i \in \{1, \ldots, N\}$ belongs to the multilabel of $\boldsymbol{x}$ if and only if $v_i = 1$.

A *taxonomy* G is a forest whose trees are defined over the set of labels. A multilabel $\boldsymbol{v} \in \{0,1\}^N$ is said to *respect* a taxonomy G if and only if $\boldsymbol{v}$ is the union of one or more paths in G, where each path starts from a root but need not terminate on a leaf. See Figure 1. All the algorithms considered in this paper generate multilabels that respect a given underlying taxonomy.

We assume the data-generating mechanism produces pairs $(\boldsymbol{x}, \boldsymbol{v})$ such that $\boldsymbol{v}$ respects some fixed underlying taxonomy G with N nodes. The set of roots in G is denoted by $\mathrm{root}(G)$. We use $\mathrm{par}(i)$ to denote the unique parent of node i, $\mathrm{anc}(i)$ to denote the set of ancestors of i, $\mathrm{subtree}(i)$ to denote the set of nodes in the subtree rooted at i (including i), and $\mathrm{child}(i)$ to denote the set of children of node i. Finally, we denote by $\{\phi\}$ the indicator function of predicate ϕ.

The H-loss

A hierarchical loss function is used to measure the discrepancy between the predicted multilabel $\widehat{\boldsymbol{y}} = (\widehat{y}_1, \ldots, \widehat{y}_N)$ and the true multilabel $\boldsymbol{v} = (v_1, \ldots, v_N)$. The leading idea underlying our hierarchical loss function is: *if a parent class has been predicted wrongly, then errors in the children should not be taken into account.*

A loss function with these properties is the H-loss ℓ_H defined in Cesa-Bianchi et al. (2005) as

$$\ell_H(\widehat{\boldsymbol{y}}, \boldsymbol{v}) = \sum_{i=1}^{N} c_i \left\{ \widehat{y}_i \neq v_i \wedge \widehat{y}_j = v_j,\, j \in \mathrm{anc}(i) \right\} ,$$

where $c_1, \ldots, c_N > 0$ are fixed *cost coefficients*. In words, to compute the H-loss all paths in the taxonomy G from a root down to a leaf are examined and, whenever a node i is encountered such that $\widehat{y}_i \neq v_i$, then c_i is added to the loss, while all the other loss contributions from the subtree rooted at i are discarded. Pictorially, this allows us to associate with each error pattern a *contour* passing through the nodes contributing to the H-loss. Figure 1(c) gives an example.

3. A stochastic model for multilabels

The paper Cesa-Bianchi et al. (2005) introduced the following stochastic model to generate the multilabels associated with an arbitrary sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ of instances.

A probability distribution f_G over the set of multilabels is associated with a taxonomy G as follows. Each node i of G is tagged with a $\{0,1\}$-valued random variable V_i distributed according to a conditional proba-

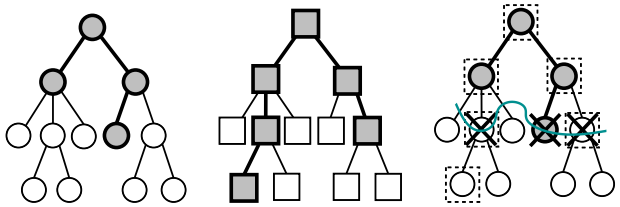

Figure 1. A one-tree forest (repeated three times). Each node corresponds to a class in the taxonomy G, hence in this case $N = 12$. Gray nodes are included in the multilabel under consideration, white nodes are not. (a), (b) Two generic multilabels that respect G. (c) Superposition of multilabel (a) on multilabel (b): Only the checked nodes contribute to the H-loss between (a) and (b). The H-loss contour passes through the checked nodes. Observe that, by definition of H-loss, if i is a contour node for the H-loss, then no other node in subtree(i) can be a contour node.

bility function $\mathbb{P}(V_i \mid V_{\mathrm{par}(i)}, \boldsymbol{x})$. To model the dependence between the labels of nodes i and $j = \mathrm{par}(i)$ we assume $\mathbb{P}\left(V_i = 1 \mid V_j = 0, \boldsymbol{x}\right) = 0$ for all nonroot nodes i and all instances $\boldsymbol{x}$.

The quantity

$$f_G(\boldsymbol{v} \mid \boldsymbol{x}) = \prod_{i=1}^{N} \mathbb{P}\left(V_i = v_i \mid V_j = v_j,\, j = \mathrm{par}(i),\, \boldsymbol{x}\right)$$

defines a joint probability distribution on $V_1, \ldots, V_N$ conditioned on $\boldsymbol{x}$ being the current instance. This joint distribution puts zero probability on all multilabels $\boldsymbol{v} \in \{0,1\}^N$ which do not respect G.

Through f_G we specify an i.i.d. process $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots\}$ as follows. We assume that an arbitrary and unknown sequence of instance vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ is fixed in advance, where $\|\boldsymbol{x}_t\| = 1$ for all t. The multilabel $\boldsymbol{V}_t$ is distributed according to the joint distribution $f_G(\cdot \mid \boldsymbol{x}_t)$. We call each pair $(\boldsymbol{x}_t, \boldsymbol{v}_t)$, where $\boldsymbol{v}_t$ is a realization of $\boldsymbol{V}_t$, an *example*.

In what follows, when the taxonomy G is understood from the context, we use $p_i(\boldsymbol{x})$ to denote the probability $\mathbb{P}\left(V_i = 1 \mid V_{\mathrm{par}(i)} = 1, \boldsymbol{x}\right)$. When no ambiguity arises, we also write p_i instead of $p_i(\boldsymbol{x})$.

The Bayes-optimal classifier

We now recall, following Cesa-Bianchi et al. (2005), the Bayes-optimal classifier (H-BAYES) that results after combining the H-loss function with the stochastic multilabel model defined above. By definition, H-BAYES classifies any instance $\boldsymbol{x}$ with the multilabel

$$\boldsymbol{y}^* = \operatorname*{argmin}_{\boldsymbol{y} \in \{0,1\}^N} \mathbb{E}[\ell_H(\boldsymbol{y}, \boldsymbol{V}) \mid \boldsymbol{x}\,],$$

where the expectation is w.r.t. the random draw of $\boldsymbol{V}$, conditioned on $\boldsymbol{x}$ being the current instance.

Fix any unit-length instance $\boldsymbol{x}$ and let $\boldsymbol{y}$ be a multilabel that respects G. For each node i in G, recursively define

$$H_{i,\boldsymbol{x}}(\boldsymbol{y}) \;=\; c_i\left(p_i(1-y_i) + (1-p_i)y_i\right) + \\ + \sum_{k \in \mathrm{child}(i)} H_{k,\boldsymbol{x}}(\boldsymbol{y})\,,$$

where the sum is zero when child(i) is empty (i.e., when i is a leaf node).

The classification $\boldsymbol{y}^*$ performed by H-BAYES can be computed as follows. Initially, all nodes of G are assigned to a set S and then removed one by one. The value y_i^* of a node i is decided when i is removed from S. Node i can be removed from S only if i is a leaf or if all nodes j in the subtree rooted at i have already been removed. When i is removed, its value y_i^* is set to 1 if and only if

$$p_i \left(2 - \sum_{k \in \mathrm{child}(i)} H_{k,\boldsymbol{x}}(\boldsymbol{y}^*)/c_i\right) \geq 1\,. \qquad (1)$$

If y_i^* is set to zero, then all nodes in the subtree rooted at i are set to zero.

Note that if i is a leaf node then (1) is equivalent to $y_i^* = \{p_i \geq 1/2\}$. Also, since each $H_{i,\boldsymbol{x}}$ only depends on the nodes in subtree(i), the functions $H_{1,\boldsymbol{x}}, \ldots, H_{N,\boldsymbol{x}}$ are well defined, and can be computed in a standard bottom-up fashion. See Figure 2 for an example.

4. An SVM-based approximation

To implement H-BAYES we must derive estimates $\widehat{p}_i(\boldsymbol{x})$ of the probabilities $p_i(\boldsymbol{x})$ for all $\boldsymbol{x}$ and i. We do so by training an SVM at each node of the taxonomy. The SVM at node i is trained on the subset of the training set including all examples $(\boldsymbol{x}, \boldsymbol{v})$ such that $v_{\mathrm{par}(i)} = 1$. Let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N \in \mathbb{R}^d$ be the vectors[1] computed by these SVMs. To obtain the estimates $\widehat{p}_i$ from the vectors $\boldsymbol{w}_i$ we set

$$\widehat{p}_i(\boldsymbol{x}) = 1 \Big/ \left(1 + e^{\alpha_i\, \boldsymbol{w}_i^\top \boldsymbol{x} + \beta_i}\right)$$

and use Platt's algorithm (Platt, 1999) to fit the parameters $\alpha_1, \ldots, \alpha_N$ via cross-validation on the training set. (To avoid flipping the SVM classifications, we

[1] These vectors are meant to incorporate the biases computed by each SVM.

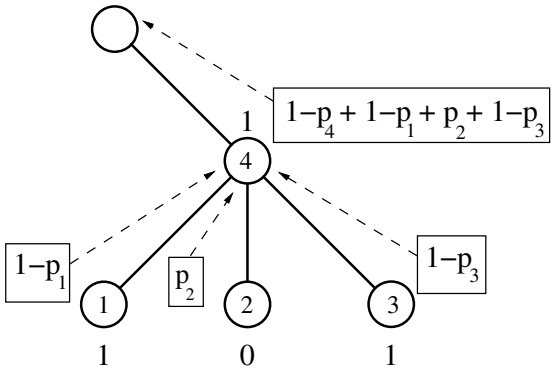

Figure 2. The bottom-up evaluation of H-BAYES as a message-passing procedure. For simplicity, assume $c_i = 1$ for all nodes i. Initially, leaves 1, 2, and 3 compute their optimal label $y_i^* = \{p_i \geq 1/2\}$ for $i = 1, 2, 3$. Then, each leaf i sends to the parent node the message $H_{i,\boldsymbol{x}} = 1 - p_i$ if $y_i^* = 1$ and the message $H_{i,\boldsymbol{x}} = p_i$ otherwise. The parent node 4 computes the sum $\Sigma_{\boldsymbol{x}} = H_{1,\boldsymbol{x}} + H_{2,\boldsymbol{x}} + H_{3,\boldsymbol{x}}$ of the children messages and sets $y_4^* = 1$ if and only if $p_4 \geq 1/(2 - \Sigma_{\boldsymbol{x}})$. Finally, if $y_4^* = 1$, then node 4 sends up to its parent the message $H_{4,\boldsymbol{x}} = 1 - p_4 + \Sigma_{\boldsymbol{x}}$; otherwise, the node sends the message $H_{4,\boldsymbol{x}} = p_4 + \Sigma_{\boldsymbol{x}}$. In the figure, $y_1^* = 1$, $y_2^* = 0$, and $y_3^* = 1$ so that we have $\Sigma_{\boldsymbol{x}} = (1 - p_1) + p_2 + (1 - p_3)$ and $H_{4,\boldsymbol{x}} = (1 - p_4) + \Sigma_{\boldsymbol{x}}$.

slightly modify Platt's technique and set $\beta_i = 0$ for each $i = 1, \ldots, N$.)

We have chosen Platt's sigmoid fitting algorithm for two reasons: First, keeping SVM as base classifier for our algorithms is useful to appreciate the potential benefit brought to the classification system by our Bayesian approximation. Second, Platt's technique, after all, is known to perform pretty well in practice (Caruana & Niculescu-Mizil, 2005).

We call B-SVM the algorithm that learns $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$ and $\widehat{p}_1, \ldots, \widehat{p}_N$ as described above, and then computes the classication of H-BAYES using estimates $\widehat{p}_i$ in place of p_i.

In order to verify the effectiveness of the Bayesian evaluation mechanism, we performed an experimental comparison between B-SVM and H-SVM. H-SVM uses the same vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N \in \mathbb{R}^d$ as B-SVM, but computes the label $\widehat{\boldsymbol{y}} = (\widehat{y}_1, \ldots, \widehat{y}_N)$ assigned to instance $\boldsymbol{x}$ according to the following top-down procedure:

$$\widehat{y}_i = \begin{cases} \{\boldsymbol{w}_i^\top \boldsymbol{x} \geq 0\} & \text{if } i \in \text{root}(G), \\ \{\boldsymbol{w}_i^\top \boldsymbol{x} \geq 0\} & \text{if } i \notin \text{root}(G) \wedge \widehat{y}_{\text{par}(i)} = 1, \\ 0 & \text{if } i \notin \text{root}(G) \wedge \widehat{y}_{\text{par}(i)} = 0, \end{cases}$$

where root(G) is the set of roots in the given taxonomy G. Note that H-SVM and B-SVM use exactly the same set of weights. The only difference is in the way the classifiers are defined in terms of these weights. In H-SVM, the multilabel is constructed via a top-down evaluation of the tree nodes, where the label $\widehat{y}_i$ assigned to node i (provided $\widehat{y}_{\text{par}(i)} = 1$) just depends on the sign of the margin $\boldsymbol{w}_i^\top \boldsymbol{x}$.

On the other hand, as hinted by Figure 2, B-SVM is a bottom-up procedure, where the label assigned to i (provided the label assigned to its parent is 1) depends on the interaction between the (estimated) node's probability $\widehat{p}_i$ and the probabilities of i's children $\widehat{p}_k$, $k \in \text{child}(i)$. (Recall that each $\widehat{p}_i$ is computed by passing the margin $\boldsymbol{w}_i^\top \boldsymbol{x}$ through a fitted sigmoid.)

Inspection of (1) in the definition of H-BAYES reveals that this interaction amounts to use at node i a classifier of the form $\{\boldsymbol{w}_i^\top \boldsymbol{x} - \tau_i\}$, where $\tau_i \geq 0$ is a threshold that depends on the (estimated) child probabilities.

A closer look at how the threshold is defined shows that τ_i is large (and positive) when the margins of i's children tend to be close to zero. Thus B-SVM tends to assign label 0 to those nodes i whose child nodes have small margins, independent of i's margin $\boldsymbol{w}_i^\top \boldsymbol{x}$. On the other hand, if i's children achieve large margin (either positive or negative) on the current instance $\boldsymbol{x}$, then τ_i tends to be close to zero. This allows node i to decide based solely on its own margin, independent of the sign of the margins of its children.

Finally observe that, due to its bottom-up dynamics, B-SVM needs to compute the margin of each node in order to build the multilabel of an instance. However this has to be done only in the *evaluation* phase (i.e., when the algorithm has to assign a multilabel to a test instance) not in the training phase. Thus, the additional computational effort of B-SVM compared to H-SVM is negligible.

5. Experiments

We tested H-SVM and B-SVM on four medium-size datasets, two of which were artificially generated. These datasets are described in detail below.

A remark about the cost coefficients for the H-loss: During the evaluation phase B-SVM constructed the multilabel of each instance using rule (1) with coefficients c_i set as follows: if i is a root, then $c_i = 1/|\text{root}(G)|$; otherwise, $c_i = 1/|\text{child}(j)|$, where $j = \text{par}(i)$. This setting normalizes the costs so as to remove any incentive towards building multilabels with short paths. In fact, whenever B-SVM decides on the label of node i, it has to balance the cost c_i of making a false negative on i with the total cost of mistakes made in the (potentially large) subtree rooted at i, given that the label of i has been guessed correctly.

However, since the total cost on the subtree is equal to the cost of the H-loss contour cutting through this subtree, the normalized cost setting makes this total cost not larger than c_i.

The synthetic datasets

The synthetic datasets were generated as follows. We defined a simple taxonomy consisting of three complete ternary trees each of depth 2, for a total of $N = 3^1 + 3^2 + 3^3 = 39$ nodes.

Let U_d be the uniform distribution on the surface of the unit sphere in $\mathbb{R}^d$. A stream of instances $\boldsymbol{x}_t$ was generated via independent draws from U_d. Multilabels were assigned using N vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$, also drawn independently from U_d. However, to give structure to the learning problem, we introduced some correlations between the vectors $\boldsymbol{u}_i$ and the distribution of instances. These correlations were introduced via the following filtering rules.

First, we discarded all instances $\boldsymbol{x}_t$ such that $|\boldsymbol{u}_i^\top \boldsymbol{x}_t| < \theta_1$ for some $i = 1, \ldots, N$, where $\theta_1 > 0$ is a threshold parameter. Second, we constructed the multilabel $\boldsymbol{v}_t \in \{0,1\}^N$ associated with an instance $\boldsymbol{x}_t$ according to the following top-down procedure: If i has parent j and $v_{j,t} = 0$ then $v_{i,t}$ is set to 0; otherwise, $v_{i,t}$ is set to 1 with probability $1/\big(1 + e^{-\alpha \boldsymbol{u}_i^\top \boldsymbol{x}_t}\big)$. However, if $v_{i,t} = 1$ and $\boldsymbol{u}_i^\top \boldsymbol{x}_t < \theta_2$ then the instance is discarded. (Here $\alpha, \theta_2 > 0$ are additional parameters.)

Notice that, in this stochastic label assignment model, the noise on the label $v_{i,t}$ increases as the margin achieved by $\boldsymbol{u}_i$ on the current instance $\boldsymbol{x}_t$ decreases. Hence, the first margin condition $|\boldsymbol{u}_i^\top \boldsymbol{x}_t| < \theta_1$ imposes a uniform upper bound on the noise at each node. The second margin condition $\boldsymbol{u}_i^\top \boldsymbol{x}_t < \theta_2$ is used to control the noise on $v_{i,t}$ conditioned on the event that $v_{i,t} = 1$.

We generated two datasets, called `Synth1` and `Synth2`: `Synth1` is a rather noisy dataset, with a relatively large number of paths per multilabel, while `Synth2` is less noisy and has multilabels that span fewer paths. The two datasets have been generated through the following parameter setting:

	d	θ_1	θ_2	α
`Synth1`	100	0.01	0.03	20
`Synth2`	100	0.001	0.07	25

`Synth1` turns out to have on average 2.656 paths per multilabel, while `Synth2` has on average 1.277. Both of them consist of 40,000 examples divided into 4 ordered chunks. Each algorithm was trained on 3 consecutive chunks and tested on the remaining one (where the fourth chunk is followed by the first one in a circular way).

The RCV1 dataset

This dataset consists of the first (in chronological order) 100,000 newswire stories from the Reuters Corpus Collection (Reuters, 2000). The associated taxonomy of labels, which are the topics of the documents, has 101 nodes organized in a forest of 4 trees. The forest is shallow: the longest path has length 3 and the the distribution of nodes, sorted by increasing path length, is $\{0.04, 0.53, 0.42, 0.01\}$. The average number of paths per multilabel is 1.5. We divided this dataset into 5 equally-sized chunks, trained the algorithms on one chunk, and tested them on the next chunk.

The OHSUMED dataset

This dataset includes the documents classified in the nodes of the subtree rooted in "Quality of Health Care" (MeSH code N05.715) of the `OHSUMED` corpus of medical abstracts (Hersh, 1994). Since `OHSUMED` is not quite a tree but a directed acyclic graph, and since the H-loss is defined for trees only, we removed from this `OHSUMED` fragment the few nodes that did not have a unique path to the root. This produced a hierarchy with 94 classes and a data set with 55,503 documents. The choice of this specific subtree was motivated by its structure only; in particular: the subtree depth is 4, the distribution of nodes (sorted by increasing path length) is $\{0.26, 0.37, 0.22, 0.12, 0.03\}$, and there is a reasonable number of partial and multiple path multilabels (the average number of paths per instance is 1.53). We ran 5 experiments by randomly splitting the corpus in a training set of 40,000 documents and a test set of 15,503 documents.

For both the `RCV1` and the `OHSUMED` datasets we used a standard bag-of-words vectorization with TF-IDF, and then normalized the resulting vectors to unit length. Finally, in all cases test set performance is measured as an average over chunk experiments.

Results

The results of our experiments are summarized in Tables 1, 2, and 3 below.

The results are expressed in terms of the H-loss computed with all cost coefficients c_i set to 1. We did not report H-loss results with the normalized cost coefficients used in the evaluation phase of B-SVM, since these coefficients compress the range of H-loss values making it harder to appreciate performance differences. In this sense, we view the H-loss with normalized coefficients just as an auxiliary loss function

that helps B-SVM to come up with "balanced" multilabels.

Table 1. Average test error (H-loss) and standard deviation on the four datasets.

DATASET	SYNTH1	SYNTH2
H-SVM	1.454 (±0.007)	0.350 (±0.007)
B-SVM	**1.269** (±0.008)	**0.322** (±0.003)

DATASET	RCV1	OHSUMED
H-SVM	0.716 (±0.024)	1.171 (±0.005)
B-SVM	**0.712** (±0.023)	**1.158** (±0.005)

Table 1 summarizes the results on the test set for each algorithm and dataset. These numbers are averages over four experiments for the synthetic dataset, and over five experiments for the real-world datasets. Note that B-SVM beats H-SVM in all four cases. The difference in H-loss performance is clear on the two synthetic experiments, significant on the OHSUMED dataset, but only marginally significant in the case of RCV1.

Table 2. Test error (H-loss) on each chunk of the RCV1 and OHSUMED datasets.

	RCV1				
CHUNK #	1	2	3	4	5
H-SVM	0.702	0.727	0.727	0.741	0.682
B-SVM	**0.701**	**0.727**	**0.712**	**0.741**	**0.681**

	OHSUMED				
CHUNK #	1	2	3	4	5
H-SVM	1.176	1.163	1.171	1.170	1.171
B-SVM	**1.164**	**1.152**	**1.158**	**1.157**	**1.161**

Table 2 reveals that, nonetheless, there is some definite statistical trend in the performance on the RCV1 and OHSUMED data. Though the difference in the averages is small, B-SVM is *always* beating H-SVM on the ten experiments.

Finally, Table 3 shows the distribution of the H-loss mistakes per test example across hierarchy levels (again, averaged over four experiments for synthetical data and over five experiments for real-world ones). An H-loss mistake at level k means that, along a path in the taxonomy, the classifier made its first mistake at level k. For example, the entry 0.432 for depth 0 in the H-SVM column of Synth1 means that, averaging over test set chunks and test examples within each chunk, the number of wrong roots in the Synth1 taxonomy (out of the three) was 0.432. The entry 0.572 for depth 1 in the same column means that, on average, the number of nodes which were wrong at depth 1, given that the corresponding roots were correct, was 0.572.

Table 3. Breakdown of the H-loss performance across hierarchy levels for datasets Synth1, Synth2, RCV1 and OHSUMED.

	SYNTH1		SYNTH2	
DEPTH #	H-SVM	B-SVM	H-SVM	B-SVM
0	**0.432**	0.445	0.123	**0.121**
1	0.572	**0.519**	0.184	**0.166**
2	0.757	**0.678**	0.187	**0.183**

	RCV1	
DEPTH #	H-SVM	B-SVM
0	**0.190**	0.192
1	**0.471**	0.474
2	0.131	**0.126**
3	0.097	**0.093**

	OHSUMED	
DEPTH #	H-SVM	B-SVM
0	**1.010**	1.018
1	0.316	**0.297**
2	0.160	**0.152**
3	**0.099**	0.101
4	0.413	**0.389**

The results in Table 3 provide some information about the differences in the behavior of the two algorithms. During the evaluation of a test instance, errors propagate following the direction of the evaluation mechanism: top-down for H-SVM and bottom-up for B-SVM. At root level, the two algorithms perform very similarly, although H-SVM does slightly better. Considering what we said above, we might view this phenomenon as a consequence of the bottom-up propagation of error taking place in B-SVM (in fact, B-SVM performs best on Synth2, which is the less noisy of the two synthetic datasets). One can see that, as the depth increases, B-SVM tends to perform better than H-SVM. However, at depths 3 and 4 of OHSUMED this advantage turns out to be nearly lost. We suspect that this is due to the fact that labels assigned by humans at deep levels of the hierarchy are rather noisy.

In order to reduce the effect of noise propagation, one could combine top-town and bottom-up evaluation schemes into a hybrid system that works top-down close to the roots and bottom-up close to the leaves. We have actually implemented one of them and tested it on the above mentioned datasets. Though this hybrid algorithm tends to be wrong on *different* test examples than those observed for H-SVM and B-SVM, it has not lead to a significant improvement in H-loss performance.

6. Conclusions and ongoing research

We have shown how to improve the performance of a baseline hierarchical SVM algorithm by replacing its top-down evaluation procedure with a recursive bottom-up scheme that arises from the analysis of the Bayes optimal hierarchical classifier for the H-loss.

Our modular approach is likely to provide new useful tools for Machine Learning practitioners engaged with classification problems on taxonomical data.

Although our preliminary experiments show encouraging results, this research leaves open several important issues. In the following we mention two of them.

• On real-world data (the `RCV1` and `OHSUMED` experiments) B-SVM is only slightly better than H-SVM. We do not know exactly why this happens. On the one hand, it is plausible that in hierarchical data labelled by humans, labels at deep levels of the taxonomy are too noisy for the Bayesian mechanism to work well. On the other hand, the independence assumption, which stands at the basis of our stochastic model, could be simply too naive to accurately model real data.

• Our Bayesian evaluation scheme is based on a blend of good classification power (the weights $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$) and good probability estimation (the estimators $\hat{p}_1, \ldots, \hat{p}_N$ obtained by fitting sigmoids on these weights). Since the Bayesian recursive evaluation tends to be independent of the underlying training algorithm, it is possible to replace SVM by other algorithms more suitable for probability estimation, such as the regularized logistic regression of Hastie et al. (2001).

Acknowledgments

The authors would like to thank the anonymous reviewers for their help in improving the presentation of this paper. In particular, we would like to thank one of them for pointing out several mistakes in our previous version of Figure 2.

The authors gratefully acknowledge partial support by the PASCAL Network of Excellence under EC grant no. 506778. This publication only reflects the authors' views.

References

Caruana, R., & Niculescu-Mizil, A. (2005). Predicting good probabilities with supervised learning. *Proceedings of the American Meteorology Conference*. San Diego.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2004). Regret bounds for hierarchical classification with linear-threshold functions. *Proceedings of the 17th Annual Conference on Learning Theory* (pp. 93–108). Springer.

Cesa-Bianchi, N., Gentile, C., Tironi, A., & Zaniboni, L. (2005). Incremental algorithms for hierarchical classification. *Advances in Neural Information Processing Systems 17* (pp. 233–240). MIT Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press.

Dekel, O., Keshet, J., & Singer, Y. (2004a). An efficient online algorithm for hierarchical phoneme classification.

Dekel, O., Keshet, J., & Singer, Y. (2004b). Large margin hierarchical classification. *Proceedings of the 21st International Conference on Machine Learning*. Omnipress.

Dumais, S., & Chen, H. (2000). Hierarchical classification of web content. *Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval* (pp. 256–263). ACM Press.

Granitzer, M. (2003). *Hierarchical text classification using methods from machine learning*. Doctoral dissertation, Graz University of Technology.

Hastie, T., Tishbirani, R., & Friedman, J. (2001). *The elements of statistical learning*. Springer.

Hersh, W. (1994). The ohsumed test collection. Available electronically at `http://medir.ohsu.edu/pub/ohsumed/`.

Hofmann, T., Cai, L., & Ciaramita, M. (2003). Learning with taxonomies: Classifying documents and words.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. *Proceedings of the 14th International Conference on Machine Learning* (pp. 170–178). Morgan Kaufmann Publishers.

McCallum, A., Rosenfeld, R., Mitchell, T., , & Ng, A. (1998). Improving text classification by shrinkage in a hierarchy of classes. *Proceedings of the 15th International Conference on Machine Learning* (pp. 359–367). Morgan Kaufmann Publishers.

Mladenic, D. (1998). Turning yahoo into an automatic web-page classifier. *Proceedings of the 13th European Conference on Artificial Intelligence* (pp. 473–474).

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers* (pp. 61–74). MIT Press.

Reuters (2000). Reuters corpus volume 1. Available electronically at `http://about.reuters.com/`.

Rousu, J., Saunders, C., Szdemak, S., & Shawe-Taylor, J. (2005). Learning hierarchical multi-category text classification models. *Proceedings of the 22nd International Conference on Machine Learning* (pp. 745–752). Omnipress.

Ruiz, M., & Srinivasan, P. (2002). Hierarchical text categorization using neural networks. *Information Retrieval*, *5*, 87–118.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press.

Sun, A., & Lim, E. (2001). Hierarchical text classification and evaluation. *Proceedings of the 2001 International Conference on Data Mining* (pp. 521–528). IEEE Press.

Szdemak, S., Saunders, C., Shawe-Taylor, J., & Rousu, J. (2005). Learning hierarchies at two-class complexity. Nips 2005 Workshop on Kernel methods and structured domains.