On learning hierarchical classifications

Russell Greiner
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540-6632
greiner@scr.siemens.com

Adam Grove
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
grove@research.nj.nec.com

Dale Schuurmans*
Institute for Research in Cognitive Science
University of Pennsylvania
Philadelphia, PA 19104-6228
daes@linc.cis.upenn.edu

Abstract

Many significant real-world classification tasks involve a large number of categories which are arranged in a hierarchical structure; for example, classifying documents into subject categories under the library of congress scheme, or classifying world-wide-web documents into topic hierarchies. We investigate the potential benefits of using a given hierarchy over base classes to learn accurate multi-category classifiers for these domains.

First, we consider the possibility of exploiting a class hierarchy as *prior knowledge* that can help one learn a more accurate classifier. We explore the benefits of learning category-discriminants in a "hard" top-down fashion and compare this to a "soft" approach which shares training data among sibling categories. In doing so, we verify that hierarchies have the potential to improve prediction accuracy. But we argue that the reasons for this can be subtle. Sometimes, the improvement is only because using a hierarchy happens to constrain the expressiveness of a hypothesis class in an appropriate manner. However, various controlled experiments show that in other cases the performance advantage associated with using a hierarchy really does seem to be due to the "prior knowledge" it encodes.

Keywords: multicategory classification, hierarchical classifications, document classification

---

***Also:** NEC Research Institute, Princeton, NJ

1 Introduction

The vast majority of A.I. research into machine learning has been concerned with simple true/false classification tasks; that is, the problem of learning binary-valued concepts. However, in the large-scale domains typically considered by KDD, this is generally an inappropriate oversimplification. A typical KDD application will often involve learning a categorization scheme over a large number of class labels. This is especially the case in those applications which involve text or document classification; for example, when learning to insert a world-wide-web document at the appropriate node in a large concept hierarchy; to classify news stories by topic; or to choose one out of numerous possible diagnoses for an ill patient; and so on.

What is notable about these domains is not only that they involve a large collection of class *labels* (*i.e.*, possible classifications), but that these labels are also often semantically well-structured. For instance, WWW search engines often categorize items in a *topic* hierarchy which one can navigate in a top-to-bottom fashion; the children of a node tend to be specializations or subtopics of the parent.

The question motivating this research is this: can, and should, this semantic hierarchy be used when learning a classifier? Or is it just as effective to treat the set of labels as an unstructured set? This question is not a new one (we survey some of the related work in Section 4), but we feel there are many interesting unanswered questions.

In this preliminary report, we explain some of our early experiments and the issues we have encountered. Several of these issues do not seem to have received significant attention before, yet we believe that they may be important to research in the field. In particular, we consider (1) the need to control for expressive power, (2) how robust a technique is if a wrong or inaccurate hierarchy is used, and (3) whether the hierarchy is used in classification, or whether it is just used during learning. We have few definitive conclusions about any of these issues (except perhaps the first), but nevertheless hope that our results may stimulate discussion and deeper research. We ourselves are in the process of running several much more elaborate experiments, which we hope to include in the final version of this paper.

We begin with the issue of expressive power, because it is perhaps the the most interesting. Note that it is rather easy to construct "plausible" learning techniques that use hierarchies—and some of them even work! (See Section 3). However, it is important to understand that there are generally two distinct issues arising together here. In addition to the *semantic* contribution of the hierarchy (*i.e.*, as a source of *prior knowledge*), the use of a hierarchy tends to change the expressive power of the hypothesis class being considered by many learning techniques. For instance, consider a classifier that makes an irrevokable decision among the possible depth 1 children at the root of the hierarchy, and then recurses. That is, it then moves to the most likely child of the root node and tries to decide among *its* children, and so on, until a leaf is reached. We call this the *top-down, hard-decision, model*. Let us contrast this with a *flat-decision model*, which compares all leaf labels against each other and picks the best. Even without specifying the details of how these decisions and comparisons are made, it should be clear that these two possibilities will generally have different expressive power. There will be some classifiers that simply cannot be expressed in a top-down fashion relative to any particular hierarchy but which can easily be expressed in a flat fashion. The converse is generally, if not always, true as well. (We will see specific examples of this below.) The different expressive power often implies a difference in *model complexity* as well. We discuss examples later in which the flat model has many more parameters to be estimated that the seemingly analogous hierarchical classifier.

Of course, one of the most significant lessons arising out of machine learning research is the importance of representation expressivity and complexity issues. For any given training-set size, learning performance may be terrible both if the model is too complex (leading to *overfitting*) and also, on the other hand, if it is too simple and has insufficient expressive power. This is sometimes formalized as a *bias-variance* trade-off [10, 9, 17]. So the question is this: Whenever the application of a hierarchical constraint in training appears to help, is it because of the prior semantic knowledge encoded, or is it just that (in a rather indirect fashion) we have reduced the model complexity to a more appropriate level?

Several of our experiments were designed specifically for this, and the results surprised us. We control for expressivity in two ways. First, to control for the fact that imposing a hierarchical structure might simply reduce the expressiveness of the hypothesis class (without conveying any true information about the domain), we consider what happens if we use the same hierarchy but permute the leafs. In the extreme case, we permute the leafs randomly (so the *shape* of the tree looks identical, but the labels at leaf are not semantically correlated with the leaf's siblings, except by chance). Surprisingly, for some of our learning algorithms, the results remain almost as good. That is, sometimes expressivity is the main (or at least, a significant) reason that we see any win at all. This is important because, when this is so, one might perhaps do even better to confront the expressivity issue directly rather than hoping the hierarchy somehow find the right tradeoff for us. In particular, it suggests that sophisticated *model selection* techniques should be invoked [16, 29]. On the other hand, it is conceivable that using a tree of approximately the "right" size does tend to have some robust power to select the "right" model complexity automatically. If so, this would be quite a remarkable fact, and certainly deserving of more theoretical and experimental research.

However, when controlling for expressiveness in this fashion for other learning algorithms we see different results. In many cases, using an erroneous hierarchy greatly degrades performance as compared both to an analogous non-hierarchical method and to the use of the "good" hierarchy. This happens even if using the good hierarchy is still not as good as the non-hierarchical approach, which is the case for several algorithms we have investigated. This may seem paradoxical: whenever the bad hierarchy hurts a lot, should we not expect that the "good" hierarchy will help commensurately? We conjecture that many seemingly plausible algorithms are in fact rather brittle. Even if the hierarchy is only somewhat wrong, performance quickly degrades. As we discuss in Section 2 our test hierarchies generally did *not* perfectly conform to any clean semantics; we assembled them on the basis of intuition and seeming reasonableness. This was deliberate: a technique that relies on complete semantic correctness will be a useless one. However, it may be that the inevitable inaccuracy in our hierarchies is often enough to erase any potential advantage. This highlights the importance of looking for robust learning techniques.

The second way of exploring the question of expressive power concerns the third issue raised above; namely, *when* do we use the hierarchy. As we show, it is possible to use a flat decision model—*i.e.*, where the hierarchy plays no role in classification—but nevertheless use the hierarchy in a principled way during *training*, *i.e.*, to choose *which* flat-decision classifier to hypothesize. We present details in Section 2. Clearly, by doing this we are controlling for expressive power in a very direct fashion, because the underlying class of hypotheses is being held constant.

Our preliminary observations in this second setting are as follows. First, it seems extremely hard to gain a robust win using hierarchical learning. Of dozens of diverse techniques we have applied—some ad hoc, some that attempt to be statistically principled—only one displayed a small but convincing, cross-validated advantage on our test set.

Second, intriguingly, and in spite of the difficulty in improving on non-hierarchical methods, when we do find a win, we generally got results that were superior to results based on top-down hard-decision models. Perhaps part of the issue is that once a top-down model makes a bad decision, it cannot recover. And since such a model has to make several intermediate decisions before reaching its conclusion, the chance of error compounds quickly. This observation suggests a third model: top-down *soft*-decision hypotheses, which use the hierarchy during classification but do not irrevocably commit to decisions until the end. Probabilistic schemes such as Bayesian net models [23] and HME's [15] might perhaps be viewed this way. However, we have not investigated this idea further as yet.

The rest of this paper is structured as follows. The next section summarizes our experimental methodology and fills in some of the details of the particular learning algorithms and test domains we considered. Section 3 then illustrates the points made above with a subset of our experimental results. Section 4 is a brief survey of related work. We conclude in Section 5.

2 Methodology

2.1 Learning and Classification Algorithms

Our basic representation of examples, both in the training set and test set, is as a feature vector (assumed in the following to be of length k). The m'th component x_m is an integer specifying how often the m'th feature occurred in the example. This is motivated by our interest in textual classification, where an example is a document and the features are words; the vector then records how often each word occurs in the document.[1] This is perhaps the classic technique for representing documents for automatic text retrieval [26, 27, 21, 28].

Our representation is fairly generic, and thus there are of course countless standard learning algorithms and hypothesis classes that might be appropriate. However, recall that our goal here is *not* (yet) to achieve the best possible classification performance; rather, we are most interested in engineering a "fair" comparison between hierarchical and non-hierarchical learners. This dictates the use of rather simple models.

There are two core techniques we consider: *pairwise linear discriminants* (PLD) and what

[1] Thus, in this study, we are discarding information on the order of words.

we call the conditionally-multinomial attributes (CMA) model (similar to the well known *Naive Bayes*) model.

The PLD approach is as follows. Consider a collection of labels $L_1, \ldots L_n$ and a training-set of examples T. For *every* pair of labels i, j, let $T_{i,j}$ be the subset of examples whose label is either L_i or L_j. Using these examples only, we attempt to learn a good *linear* classifier that predicts L_i against L_j.[2] That is, we try to find coefficients $c_0, c_1, \ldots, c_k$ such that $c_0 + \sum_{m=1}^{k} c_m x_m \geq 0$ whenever the example has label L_i. Of course, it is unlikely that we can find a linear rule that predicts perfectly, but we try to find a rule that is correct as often as possible. Having learned all $n(n-1)/2$ such rules (one to distinguish between each pair of labels, e.g., as suggested in [14, 8]), we classify test examples using a voting scheme. That is, for each label L_i, we evaluate the predictors of L_i against L_j for all $j \neq i$, and count how many times L_i "wins". This count is L_i's vote. The label with the most votes is our prediction. Note that *if* the linear predictors were perfect, the real label must get a vote of $n-1$ (the maximum possible) and no other label can do better than $n-2$.

Our second core technique, CMA, is even simpler. For label L_i and feature m, let $p_{i,m}$ be the expected proportion of times feature m occurs in a document with label L_i. (Recall that the feature vector is a vector of feature *counts*, so this proportion is not necessarily $1/k$.) Under a very standard and simple probabilistic model—essentially, assuming that conditioned on the label, features are chosen with a multinomial distribution according to the probabilities $p_{i,j}$—one can show that the probability that a particular vector $\vec{x}$ has label L_i is a monotonic function of

$$S_i(\vec{x}) = w_{i,0} + \sum_{m=1}^{k} w_{i,m}\, x_m$$

where $w_{i,m} = \log p_{i,m}$ for $m > 0$, and $w_{i,0}$ is the logarithm of the proportion of documents that have label L_i. Thus, if we can estimate the $p_{i,m}$ somehow, one could simply choose for $\vec{x}$ that label L_i with the highest $S_i(\vec{x})$. Although this seems to be a very simpleminded rule and that the probabilistic assumptions being made are unlikely to ever really hold in practice, it has often been noted that such rules can be surprisingly effective anyway [26, 3, 27]. In fact, related "linear text classifier" representations are a basic technique in Information Retrieval [21, 26, 28].

Both of these techniques can be applied over the space of all attributes (*i.e.*, without using a hierarchy). We call this the non-hierarchical setting (NH). However, they can also be used with a hierarchy, in a fairly obvious top-down fashion. That is, one tries to learn (using PLD or CMA) how to distinguish between the children of the root of the hierarchy, and then, having done this, one separates the training data according to the predicted value, and recurses.[3] When one is classifying test examples, we use a similar top-down, hard-decision (TDHD) model—*i.e.*, classify at the root; go to the appropriate child and choose among *its* children, and so on until we reach a leaf.

[2] Although finding the optimal classifier is an NP-hard problem, there are many good, well-known, heuristics for this problem. We omit details of our particular approach.

[3] Note that, although in our examples documents are only labeled with leaf labels, this induces an obvious labeling with respect to internal tree nodes as well.

Implicit in our comments so far are 4 different algorithms; $\{\text{CMA, PLD}\} \times \{\text{TDHD, NH}\}$. However, the CMA model is particularly interesting because it allows another option. Thus far, we have not said how one estimates the probabilities $p_{i,j}$. Of course the obvious approach is to estimate these directly based on observed proportions in the training set.[4] However, another possibility is this: We can *estimate* the probabilities $p_{i,j}$ by using the hierarchy to bias the estimates in an appropriate manner, but then *classify* subsequent test examples using the CMA model in a "flat" fashion (*i.e.*, not using the hierarchy). As already noted, this allows us to control for expressive power of the class of potential hypotheses (*i.e.*, possible CMA classifiers)—and as we see in Section 3, this actually leads to our most successful technique. The intuition in using the hierarchy is that, when estimating probabilities, the siblings of a class node should (for *most* features) tend to have similar probabilities, and therefore we could perhaps get better estimates by taking into account (in an appropriately discounted fashion) all training samples with "nearby" labels. To the extent that this works, we would in effect have access to larger training sets, and therefore should get more accurate estimates.

We have tried many schemes of this sort, ranging from the purely ad-hoc to rather complex fully-Bayesian schemes. The only one we report on in this paper is a simple scheme based on the idea of *"Stein shrinkage"* [5, 6]. We first estimate the probability $p_{\cdot,m}$ at each node in the tree using a direct estimate, and also compute the variance of these estimates. These estimated probabilities, which do not uses hierarchical mixing in any way, are the *preliminary* probabilities. (For leaves, the variance of the estimate is essentially just the variance associated with binomial sampling; for internal nodes, we use the variances of the estimates among all the children of the node.) Then, in a top down fashion, we compute the *final* probability estimates at each node as a weighted average of the node's preliminary probability and the parent's preliminary probability, with the weights being inversely proportional to the variances. Intuitively, the more we trust an estimate, the higher its weight is. This, in effect, shares weights between sibling nodes, depending on the proximity and variability of their preliminary probability estimates. In later sections, we call this approach the Stein-CMA (SCMA) model.

2.2 The data and testing methodology

To date, we have considered two domains. The first is a collection of synthetic data-sets. Because this is less interesting, we omit a fuller description of the models; in essence we generate data from a probabilistic model contrived so that the labels are clustered into several sets (where the generators within a "cluster" are more similar to each other than to those outside); the clusters are grouped together as children of a common parent.

The second is a realistic and relatively large scale problem based around the Reuters-21578(distribution 1.0) collection of newswire articles [25, 30].[5] Many of these stories have been classified by topic from a fixed set of around 130 topic words. However, several of these topics occurred only a few times, and a few occurred thousands of times; so to avoid getting

[4] In practice, one should not use the observed proportions exactly as estimates. Rather, it is preferable to use a Bayesian estimate based on a suitable Dirichlet prior. However, for samples sizes as large as the one we are considering, the difference tends to be slight.

[5] We thank David Lewis, the maintainer of this collection, for suggesting this as a suitable domain for our work.

skewed results we eliminated both types of topics—restricting ourselves to the 49 topic words that appeared more than 20 and less than 500 times in the collection. Around 4000 documents were labeled by one or more of these topics.[6]

We constructed a hierarchy by hand over these 49 topics. As noted in the introduction, this construction was ad-hoc and there are certainly several debatable decisions made; this was in part deliberate, since we are only really interested in learning algorithms that display some robustness to errors and inaccuracies of this sort. Figure 1 shows the top level of the hierarchy, ignoring the leaves. So, for instance, the Reuter's topic COPPER would be classified under the internal node METALS, and the topic NAT-GAS would be under energy. Note that the internal nodes shown here *not* the 49 Reuter's topics; the latter appear as the leaves of the tree (not shown).

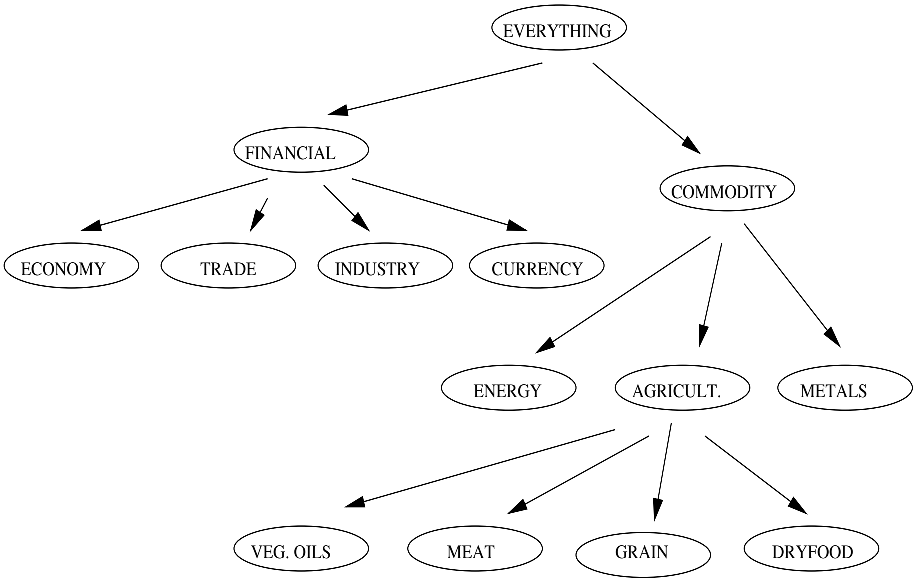

Figure 1: Non-leaf nodes in our Reuters hierarchy

The features we considered were a selection of 400 words, chosen by a very simple entropy-based measure as being useful to predictive accuracy. The words were take literally; there was no spelling correction or stemming. This, again, is consistent with our goal of comparing hierarchical to non-hierarchical techniques: the issue is not so much the quality of the features, but that we keep the features constant.

In all experiments, we partitioned the examples into a test set (of about 3000 documents) and a separate training set (the remainder) using a standard split suggested by the annotated Reuters collection. In addition, for certain techniques we also used a bootstrap-like cross-validation technique (where time permitted) which repeatedly partitioned the collection into

[6]A significant percentage of the documents were labeled with more than one topic. This caused some complications in the training and testing phases, but we omit further details here. We note, however, that the results in Section 3 are "optimistic" scores—counting as a success any test example in which we predicted any one of the true labels associated with the example. We have also used various more conservative scoring techniques, but these does not appear to change the results in any qualitative fashion.

random training and test samples. Averaging results obtained in this way generates more accurate statistics. Throughout, the training set included around 3000 documents.

3 Results

Here we show a very small subset of our results, chosen to illustrate the points made in the introduction. These tables show the percentage *error* achieved by the various algorithms.

The first table illustrates our point about expressive power. First note that PLD-TDHD performs significantly better than PLD-NH. One's first thought might be to attribute this to additional information being conveyed to the learning algorithm by the hierarchy. However, note that for PLD imposing *any* hierarchical structure on topic categories severely restricts the space of representable classifiers. In the flat case, we must learn to discriminate each of the 48 topics against the rest (thus there are $(49 \cdot 48)/2 = 1176$ classifiers to learn); in the hierarchical case we only need 133. Such an expressiveness change has drastic consequences of its own for training performance, independent of any prior knowledge expressed in the topic hierarchy. In fact, to illustrate this point, we considered the effect of training PLD-TDHD on two "bogus" hierarchies constructed by permuting the leaves of the original hierarchy. Here we see the surprising result that training PLD-TDHD on the bogus hierarchies also leads to a significant performance improvement over the flat PLD-NH model! In fact, the results are essentially as good as training with the proper hierarchy. So here it is not clear that in obtaining the original performance improvements, one is exploiting any of the domain knowledge expressed in the hierarchy at all.

	PLD-NH	PLD-TDHD with true hier.	PLD-TDHD bogus hier. 1	PLD-TDHD bogus hier. 2
Standard split	27.3	23.4	25.3	23.8

Now consider the probabilistic CMA model. Again, we contrast the results obtained by training under the flat NH model and the hierarchical TDHD model. Here we see a different phenomenon. Curiously, using a hierarchy never helps at all. But using a bad hierarchy is *much* more damaging that the true hierarchy. So sometimes the semantics is important, after all!

	CMA-NH	CMA-TDHD with true hier.	CMA-TDHD bogus hier. 1	CMA-TDHD bogus hier. 2
Standard split	19.0	23.4	30.5	30.0
Cross-validated[7]	18.3	22.2	28.0	31.6

However, as our final illustration, we consider the SCMA model, which uses the hierarchy to help learn the probabilities, but classifies in a "flat" fashion. (*i.e.*, this is the same as CMA-NH, except for the way the probabilities are estimated.) Below we compare the results of CMA-NH with using Stein-shrinkage in a degenerate one level tree. Here we see that merely applying Stein-shrinkage in a naive manner does not help, and in fact hurts classification performance significantly. However, our results with applying SCMA to the natural hierarchy are positive.

[7] All cross-validated results are averaged over 100 random training/test splits of the Reuters database.

Here we see an improvement of around 2% over CMA-NH. Note that cross-validation is over 100 trials, and so these numbers are quite accurate; this gain seems genuine, albeit small. It is important to note that since this test controls for expressive power, this gain is presumably due to the semantic information in the hierarchy. Confirming this, we observe that if we use a bad hierarchy the performance advantage vanishes.

	CMA-NH (from above)	SCMA degenerate	SCMA with true hier.	SCMA bogus hier. 1	SCMA bogus hier. 2
Original split	19.0	22.8	18.1	18.6	17.8
Cross-validated	18.3	21.5	16.7	18.1	18.2

Thus, we conclude that using a hierarchy on class labels does indeed have the potential to improve classification performance, but this is not necessarily trivial to achieve—naively applying hierarchical learning also has the potential to degrade classification performance. In this short paper, we have not been able to present all our results (such as, for instance, the results on synthetic data), but the pattern we have briefly shown is typical. Our results do show that if one asks *why* hierarchies have the effect they do, the answer can be surprisingly subtle.

4 Related Work

The work closest to ours is the earlier paper of Koller and Sahami [18]. They have independently considered the importance of topic hierarchies for document classification, and in fact have also experimented with classifying Reuters news stories into a topic hierarchy. (Although the topic hierarchy they consider is considerably smaller than our own.) They exploit a topic hierarchy in a hard top-down fashion to achieve performance improvements in their learning algorithms. Their work differs from ours in important respects, and we view our work as complementary. The techniques in [18] seem to be principally motivated by the desire to improve accuracy (*i.e.*, to obtain good performance in an absolute sense). Their learning algorithms were therefore much more sophisticated than the algorithms we have considered here; particularly with regard to the feature selection strategy, which is the focus of their results. They did indeed discover an advantage in using a hierarchies, just as we do, in spite of the difference in algorithms being considered.

On the other hand, our goal is to uncover and delineate the possible sources of improvement from exploiting a hierarchy, and so unlike [18] we have tried very hard to control for expressive power in a rigorous fashion; one price we pay being able to make "clean" comparisons is that we have used simpler algorithms. The second important difference is that, in addition to top-down hard-decision models, we also consider a "soft" approach to training which uses a hierarchy only as a bias for training a base-level classifier (*i.e.*, under the Stein-shrinkage, or SCMA approach). Our initial motivation for this was as another way of addressing the expressive power issue. But we also find that these soft training techniques, which connect to a large and important literature on Bayesian and empirical Bayesian statistics [6, 5, 22, 11], actually seem to hold promise for being an effective technique for exploiting hierarchies in a robust and general fashion.

To distinguish this general approach, note that there is other work, for instance [7], which considers *learning* a hierarchy on classes in an unsupervised fashion (see also “hierarchical clustering” [4]). However, this does not address the issue of how best to exploit the presumptive knowledge expressed by a *given* hierarchy to learn a more accurate classifier. Our expectation is that one should be able to do better by exploiting a given hierarchy rather than attempting to bootstrap on an artificial hierarchy learned from the same data.

Hierarchies also arise in many other ways in the KDD and ML research. For instance, it is common for research in KDD to consider hierarchies on attribute values [13, 12]. But work of this type generally considers learning a relation over database tuples (a “generalized relation” [12]), classifies tuples in a binary fashion as being “in” or “out” of the learned relation. Here, generalization or specialization on attribute values is simply a technique for expanding or restricting the extent of the learned relation in an appropriate way. Although an important technique in database mining applications, this is somewhat orthogonal to the issue we address here.

There is a significant literature on learning flat multicategory classifications. For example, this is predominant in the connectionist literature on perceptual learning tasks such as speech and handwriting recognition [19, 20]. The most important recent advance in this area perhaps is Dietterich and Bakiri’s approach to arranging binary classifiers in error correcting output codes to achieve high classification accuracy in multicategory domains [2]. However, none of these techniques exploit the existence of a prior hierarchical structure over the class labels.

Finally, we note that decision tree classifiers [1, 24], which classify examples in a hierarchical fashion and naturally encode multicategory classification schemes, are not designed to exploit a hierarchy on *class labels* to obtain better performance.

So, to conclude, while other uses of hierarchies are common in the KDD and ML literatures, the idea of exploiting hierarchies over domain categories (*i.e.*, regarding the hierarchy as prior knowledge that can improve learning performance) seems to be comparatively unexplored.

5 Conclusions

In this paper, we have presented some ideas and preliminary results concerning hierarchical classification, concentrating on the issue of expressive power versus semantic knowledge. We close by briefly noting two of the many other interesting research questions in this area. First, it would be useful to have a deeper theoretical understanding (from the standpoint of computational learning theory) of the advantages and disadvantages of hierarchically based classifiers. A more practical issue is the following: What if one is allowed to classify to internal nodes in a tree, and not just leaves? One can imagine various plausible *loss* models (which would specify the penalties associated with not giving a maximally specific classification, versus the penalty for being wrong). Can one find a good learning algorithm designed to minimize expected loss in this sense?

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[2] T. G. Dietterich and G Bakiri. Solving multiclass learning problems via error-correcting output codes. *JAIR*, 2:263–286, 1995.

[3] P. Domingos and M. Pazzani. Beyond independence: Conditions for optimality of the simple bayesian classifier. In *Proceedings 13th International Conference on Machine Learning (ML '96)*, pages 105–122, 1996.

[4] R. O. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[5] B. Efron and C. Morris. Stein's estimation rule and its competitors—an empirical Bayes approach. *J. Amer. Stat. Assoc.*, 68:117–130, 1977.

[6] B. Efron and C. Morris. Stein's paradox in statistics. *Scientific American*, pages 119–127, 1977.

[7] D. Fisher. Knowledge acquisition via incremental concept clustering. *Machine Learning*, 2:139–172, 1987.

[8] J. Friedman. Another approach to polychotomous classification. Technical report, Stanford University, Statistics Department, 1996.

[9] J. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. Technical report, Stanford University, Statistics Department, 1996.

[10] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Comp.*, 4:1–58, 1992.

[11] E. George, U. Makov, and A. Smith. Fully Bayesian hierarchical analysis for exponential families via Monte Carlo computation. In P. Freeman and A. Smith, editors, *Aspects of Uncertainty*. Wiley, New York, 1994.

[12] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: an attribute-oriented approach. In *VLDB-92*, pages 547–559, 1992.

[13] J. Han, Y. Fu, W. Wei, J. Chiang, W. Gong, K. Koperski, D. Li, and Y. Lu. Dbminer: a system for mining knowledge in large relational databases. In *KDD-96*, pages 250–255, 1996.

[14] D. J. Hand. *Discrimination and Classification*. Wiley, New York, 1981.

[15] M. I. Jordan and Jacobs R.A. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, pages 181–214, 1994.

[16] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI-95*, 1995.

[17] R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *ML-96*, pages 275–283, 1996.

[18] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. Technical report, Stanford University, Computer Science Department, 1997.

[19] K. Lang, G. Hinton, and A. Waibel. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[20] Y. le Cun, B. Boser, J. S. Denker, D. Henderson, D. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comp.*, 1:541–551, 1989.

[21] D. Lewis, R. Schapire, J. Callan, and R. Papka. Training algorithms for linear text classifiers. In *SIGIR-96*, pages 298–306, 1996.

[22] J. Maritz and T. Lwin. *Emprical Bayes Methods*. Chapman and Hall, London, 1989.

[23] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, 1988.

[24] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[25] 1997. Text Categorization Test Collection, http://www.research.att.com/ lewis/reuters21578.html.

[26] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[27] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *CACM*, 18(11):613–620, 1975.

[28] H. Schuetze, D. Hull, and J. Pederson. A comparison of classifiers and document representations for the routing problem. In *SIGIR-95*, pages 229–237, 1995.

[29] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA, 1991.

[30] Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, NV, 1995. ISRI; Univ. of Nevada, Las Vegas.