Bayesian Aggregation for Hierarchical Classification

Zafer Barutcuoglu **Christopher R. DeCoro** **Robert E. Schapire** **Olga G. Troyanskaya**[*]

Department of Computer Science
Princeton University
35 Olden St.
Princeton, NJ 08540

Technical Report TR-785-07

Abstract

Large numbers of overlapping classes are found to be organized in hierarchies in many domains. In multi-label classification over such a hierarchy, members of a class must also belong to all of its parents. Training an independent classifier for each class is a common approach, but this may yield labels for a given example that collectively violate this constraint. We propose a principled method of resolving such inconsistencies to increase accuracy over all classes. Our approach is to view the hierarchy as a graphical model, and then to employ Bayesian inference to infer the most likely set of hierarchically consistent class labels from independent base classifier predictions. This method of Hierarchical Bayesian Aggregation (HBA) can work with any type of base classification algorithm. Experiments on synthetic data, as well as real data sets from bioinformatics, graphics, and music domains, illustrate the behavior of HBA under a range of conditions, and reliably demonstrates improvements in accuracy over all levels of a hierarchy.

1 INTRODUCTION

Large collections of classes are often organized into hierarchies in many domains, such as the Gene Ontology in bioinformatics, 3D object taxonomies in computer graphics, genres in music classification, and web directory categorizations. In the general multi-label classification setup, an example is allowed to belong to multiple classes, but a hierarchical organization implies that members of a class must also be members of all of its super-classes.

The popular conventional approach in multi-label classification scenarios is to decompose the problem into independent one-class tasks. This allows the flexibility of using any available well-understood binary classification algorithm as best suited for the data at hand, at the expense of ignoring correlations among classes. However, in the presence of a hierarchy, these correlations become hard constraints that are likely to be violated by independent predictors. Namely, an example may be predicted as positive for one class and negative for its parent class, so obviously at least one of them must be making a mistake. Not only does this pair of predictions require modification to be semantically valid with respect to the hierarchy, but also by making the right modification, overall accuracy may be improved.

In this paper, we propose constructing a Bayesian network from the hierarchy to infer the most likely set of hierarchically consistent class labels from the (possibly inconsistent) predictions of a set of independent base classifiers for a given example. This framework of Hierarchical Bayesian Aggregation (HBA) can accommodate any choice of base classifier, as long as estimates of their prediction accuracies are available on held-out validation data. The hierarchy itself can be any directed acyclic graph, not just a tree.

Other approaches to hierarchical classification include training the classifiers in a top-down manner, where the children are trained only on their parents' members, and are not consulted at all during evaluation if their parent predicts negative [Cesa-Bianchi *et al*. 2004]. This results in simpler classification tasks in the more specific classes, but the higher-level classes derive no benefit from the hierarchy despite the disproportionately large responsibility of their errors affecting all descendants. In contrast, our method allows classes at every level to be influenced by parents and children alike.

Another category of algorithms train all base classifiers in a correlated manner using the hierarchy, making the hierarchical information available during training as well as evaluation [Cai and Hofmann 2004, Dekel *et al*. 2004, Dumais and Chen 2000]. However, these algorithms are currently available as extensions to specific classification models such as perceptrons or SVMs, while our method transparently allows any choice of base classifier, and even different models in the same hierarchy.

In Section 2 we describe how to construct the Bayesian network, estimate its parameters, and perform inference. We also give extensions of the method to handle more constrained hierarchical scenarios, such as disallowing posi-

[*]OGT is also affiliated with Lewis-Sigler Institute for Integrative Genomics, Princeton University.

tive non-leaf predictions without a positive child prediction, or only allowing positive predictions in one class and its ancestors, as well as an adaptation for ordinal classification. In Section 3 we present experimental results on both synthetic data, and on three actual applications (from the Gene Ontology, the Princeton Shape Benchmark, and the MIREX 2005 Symbolic Music dataset) that empirically illustrate the behavior of our method, and show that it improves accuracy over all levels of a hierarchy.

2 METHOD

In this section we first formalize hierarchical consistency, and then describe how to construct a graphical model to infer the most likely hierarchically consistent labels for an example, given possibly inconsistent prior predictions.

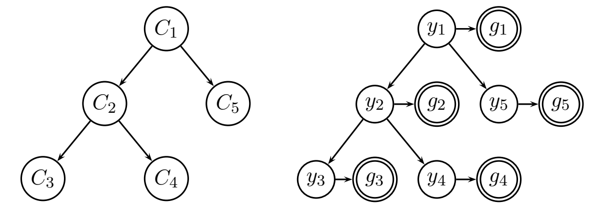
For a set of classes $C_1, \dots, C_n$ organized in a directed acyclic graph (DAG) representing a general-to-specific hierarchy, let $pa(i)$ denote the indices of the parents of class C_i. Given an example x, for each class C_i let the true label $y_i \in \{0, 1\}$ denote its membership in that class. The hierarchical consistency constraint dictates all members of a child class also belong to all of its parent classes; i.e., if $y_i = 1$, then $y_j = 1$ for all $j \in pa(i)$. We are also given an independent base classifier for each class C_i which outputs a prediction $g_i \in \{0, 1\}$ for the example x, with no regard to being hierarchically consistent among themselves. For a given test example x, we want to determine the most likely set of (consistent) true labels $y_1, \dots, y_n$ given the (possibly inconsistent) base classifier predictions $g_1, \dots, g_n$.

2.1 THE BAYESIAN NETWORK

Our goal is to model the joint probability distribution $P(y_1, \dots, y_n, g_1, \dots, g_n)$ in a compactly parameterizable way that allows efficient maximization of $P(y_1, \dots, y_n | g_1, \dots, g_n)$.

Let us construct a Bayesian network from the hierarchy, with edges to each y_i from all $y_j : j \in pa(i)$, and edges to each g_i from the corresponding y_i (Figure 1). The network will have a particular configuration of value assignments for an example x (so technically all variables are also conditioned on x, which is omitted for clarity since it is given).

This structure assumes conditional independence of the prediction g_i from any other prediction g_j or true label y_j $(j \neq i)$ given its true label y_i. The hierarchical edges among the y nodes correspond to conditional class priors, and effectively encode hierarchical consistency, while the edges from y_i to g_i represent the predictive accuracy of the base classifier. A set of predictions for an example x corresponds to the g node values being observed, and any standard method of Bayesian network inference can be used to obtain the maximum-likelihood assignment to the unobserved y nodes (or the marginal probability of particular classes, if desired).



(a) Hierarchy of classes (b) Bayesian network

Figure 1: The class hierarchy (a) is transformed into a Bayesian network (b). The y nodes are the binary-valued hidden nodes representing actual membership to the class, and the corresponding g nodes are the observed classifier outputs.

2.2 PARAMETER ESTIMATION

Before inference can be performed, the parameters of this model need to be estimated from data, which are the conditional probability distributions for each variable given its parents in the network.

Specifically, these parameters are the hierarchical class priors $P(y_i | \mathbf{y}_{pa(i)})$ where $\mathbf{y}_{pa(i)}$ denotes all parent y nodes of y_i, and the base classifier output distributions $P(g_i | y_i)$. Although possible, it is not necessary to use the costly EM algorithm to estimate these. $P(y_i | \mathbf{y}_{pa(i)})$ can be straightforwardly estimated by frequency from training set labels. Indeed, only $P(y_i | \mathbf{y}_{pa(i)} = \mathbf{1})$ needs to be estimated (simply the ratio of parent positives that are also positive in the child y_i). Since training set labels must be hierarchically consistent by definition, the probability that y_i is positive when any of the parents is negative, $P(y_i = 1 | \mathbf{y}_{pa(i)} \neq \mathbf{1})$, will be zero, which is also what ensures hierarchical consistency of labels during inference. If Laplace smoothing is to be added, care must be taken to keep these probabilities zero.

The parameters $P(g_i | y_i)$ represent the base classifier output distributions to be expected on previously unseen examples, so estimating them over examples that have been used in training is likely to be severely biased. Assuming that the training data distribution reflects the test data distribution, part of the available data should be held out from training, so the base classifiers can be evaluated on this held-out validation set to provide a better estimate of their performance on new examples. If training data is too scarce to hold out completely, resampling methods such as k-fold cross-validation or bootstrapping can be used instead.

In the case of discrete base classifier outputs, $P(g_i | y_i)$ can be estimated using the confusion matrices over held-out data, where $P(g_i = 0 | y_i = 0)$ will be the ratio of negative examples classified correctly (TN/(TN+FP)), $P(g_i = 1 | y_i = 1)$ will be the ratio of positive examples classified correctly (TP/(TP+FN)), and so on.

If the base classifiers are able to output real-valued predictions such as probabilities or confidence ratings, the continuous distributions $P(g_i \in \mathbb{R} | y_i = 0)$ and $P(g_i \in \mathbb{R} | y_i = 1)$ can be modeled parametrically, for example as Gaussians. If $P(g_i \in \mathbb{R}|y_i = 0)$ is taken to be the Gaussian $\mathcal{N}(\mu_0, \sigma_0^2)$, its parameters μ_0 and σ_0^2 can be estimated as the mean and variance of the base classifier outputs for the held-out negative examples of that class. Similarly, $P(g_i \in \mathbb{R}|y_i = 1) = (\mu_1, \sigma_1^2)$ can be estimated over the held-out positive examples of the class. Instead of explicitly converting them into binary outputs by manual thresholding, providing real-valued base classifier outputs like this allows the system greater freedom during inference, as it will essentially be implicitly performing thresholding for all classes to maximize overall accuracy.

See Figure 2 for an example using Gaussians to model $P(g_i \in \mathbb{R}|y_i = 0)$ and $P(g_i \in \mathbb{R}|y_i = 1)$ where the base classifier output is the median of 10 unthresholded support vector machines for the "chromosome segregation" class in the Gene Ontology. The histograms depict the output distributions on positive and negative held-out examples, from which the mean and variance parameters are estimated for inference.

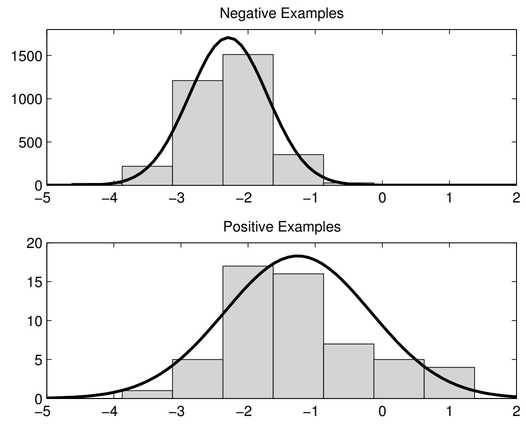


Figure 2: Modeling continuous classifier outputs using Gaussians ("chromosome segregation" class of the Gene Ontology). The histograms indicate the distribution of the median outputs of 10 unthresholded SVMs over positive and negative validation examples. (The y-axes have different scales.)

2.3 INFERENCE

Once the Bayesian network is constructed, any Bayesian inference algorithm can be used to compute the most likely label configuration or marginal probabilities. Depending on the hierarchy, an exact (e.g. junction trees), or approximate (e.g. Monte Carlo methods) inference algorithm may be preferable. In the case of tree hierarchies, an extension of the Viterbi algorithm to trees is available for efficient linear-time inference [Durand *et al.* 2004].

Instead of the hierarchy-wide most likely binary label configuration, it may also be desirable to compute an example's real-valued marginal membership probability for each class. These values can then be thresholded at different levels depending on the application, or can be used to rank all examples for a class by membership probability.

2.4 VARIATIONS AND EXTENSIONS

We next discuss some of the many possible variations on the basic technique outlined above.

2.4.1 Upward Edges

The hierarchical edges from each y node downward to its children encode the assumption of conditional independence of the children given the (positive) parent label. Reversing these edges to point upwards in the hierarchy (leaving the edges from y_i to g_i unaffected) still results in an acyclic graph, and allows sibling nodes to be correlated, which may make more causal sense in practical applications. This time the Bayesian network is parameterized over children, so $P(y_i|\mathbf{y}_{ch(i)})$ will be estimated (where $ch(i)$ denotes the child class indices of C_i), and $P(y_i = 0|\mathbf{y}_{ch(i)} \neq \mathbf{0})$ will be kept zero to ensure hierarchical consistency. However, most hierarchies have many more children than parents, so the increase in the number of conditional probability entries (exponential in the number of children) will also result in fewer data available to estimate each value, and the degradation in parameter accuracy may offset the benefits of better causal modeling, yielding worse inference results. In practice, both configurations should be compared on the data at hand.

2.4.2 Forcing Leaf Predictions

Some applications may dictate that positive labels for each example must propagate down to leaf nodes; that is, a non-leaf node cannot be positive when none of its children are positive. Here the children of a positive node obviously must be correlated, and using upward edges as described above with the additional setting of $P(y_i = 1|\mathbf{y}_{ch(i)} = \mathbf{0}) = 0$ extends our method to this case. Inference under these settings cannot result in label configurations with terminal positive predictions at non-leaf nodes.

2.4.3 One-Class Predictions

Another possible variation on hierarchical predictions is to require each example to belong to exactly one class, together with all of its ancestors (in the case of a tree, this is a single branch from the root, but not necessarily reaching a leaf). To accommodate this constraint, the binary y nodes in our Bayesian network can be converted to be multinomial, with y_i having $|ch(i)| + 2$ possible values, one for "negative", one for "terminal positive" (none of the children are positive), and one for each child being positive. The conditional probabilities will then be set to ensure that when y_i is "negative" or "terminal positive" all of its children are "negative", and when y_i is "child y_j is positive" its child y_j is not "negative" and all other children are "negative".

This constraint may also be combined with forcing leaf predictions, which will be akin to making a multi-class single-label prediction over the leaf classes. While it is possible to enforce the multinomial constraints using the upwards arrows in the above case instead of downwards, this is not necessary, as removing the "terminal positive" value from the multinomial variables is sufficient.

2.4.4 Ordinal Classification

An interesting special case to consider is a single-branch hierarchy, that is, a chain of classes, which may represent an array of predictors for binary "greater-than" decisions on an ordinal value range. For example, predicting user ratings for movies over the ordered label set of one to five, we may have the first classifier predicting whether the label is greater than one, the second classifier predicting whether it is greater than two, and so on. Ordinal consistency of these predictions is equivalent to hierarchical consistency down a single branch. Constructing our Bayesian network for this chain hierarchy yields a hidden Markov model, which makes the standard Viterbi algorithm directly applicable for inference.

3 EXPERIMENTAL RESULTS

In this section we present results from experiments on synthetic data to analyze the behavior of our method under a range of conditions, and we present three real data sets where we successfully applied it to improve accuracy and ranking performance.

3.1 EVALUATIONS USING SYNTHETIC DATA

To illustrate its behavior over different hierarchies, data distributions, and base classifier accuracies, we evaluated Hierarchical Bayesian Aggregation on a variety of synthetic data.

To provide different data distributions, we generated two types of synthetic data. In *bottom-up* data generation, every class initially contains a fraction p of the examples as positives, sampled independently. Then, given the hierarchy, each class also inherits all positives of its descendants. This results in the number of positive examples growing very quickly when going up in the hierarchy, especially for large numbers of children. In *top-down* generation, top-level classes contain a fraction q of the examples as positives, and every child class has the fraction q of its parents' positives, yielding q^{j+1} positives at a depth-j class in a tree hierarchy. This yields a relatively more gradual change of positive/negative ratio from one level to another, independent of the number of children.

A base classifier for each class is assumed to be available, yielding binary predictions on these examples, known to have a fixed accuracy a, the probability of predicting the correct label. We then construct our Bayesian network, and infer the corrected predictions for each example as the most likely true labels given the classifier predictions. The hierarchical edges in the Bayesian network are downward for the (uncorrelated) top-down data, and upward for the (correlated) bottom-up data.

Note that since our Bayesian network does not need to see the input features, and the labels and base classifier outputs are already available, it is not necessary to actually generate the input features and train real base classifiers on them. All we are concerned with is that these base classifiers have made their predictions with a known accuracy.

We generated bottom-up data with $p = 0.01$ and top-down data with $q = 0.75$, over $a \in \{0.70, 0.80, 0.90\}$ on four tree hierarchies: `T2D4` and `T2D6` are binary trees of depth four and six, and `T5D2` and `T5D3` are 5-ary trees of depth two and three, respectively. Accuracies before and after Bayesian correction, averaged per node depth, are reported in Table 1. Note that in terms of the fraction of positive examples, `T2D4` is identical to the top depth-4 subtree of `T2D6` for top-down data, to the bottom depth-4 subtrees of `T2D6` for bottom-up data, and a similar relationship holds for `T5D2` and `T5D3`; hence the depth columns are aligned accordingly in the table.

3.1.1 Observations

Hierarchical Bayesian Aggregation clearly does improve average accuracy in almost all cases. Large accuracy improvements can be found at every level.

Comparing `T2D4` to `T2D6` and `T5D2` to `T5D3` on top-down and bottom-up data shows that the benefits from the addition of new nodes at the top or the bottom of the hierarchy gradually propagate to all other levels.

Closer inspection of the predictions `T5D3` with $a = 70\%$ on top-down data, as well as the local deteriorations in some levels of the other cases, reveals that starting out with relatively inaccurate classifiers hinders the improvement from the aggregation. Note that the term "inaccurate" is relative to the node. For example, the 70% accurate base classifier at the root of `T5D3` is worse than a constant positive prediction, as the root data is 75% positive.

To illustrate, in the absence of all hierarchical edges, inference on the isolated two-node subnetwork $y_i \rightarrow g_i$ will have one of four effects; y_i will always be equal to g_i, or always its reverse, or constant-positive, or constant-negative, whichever maximizes accuracy. If g_i has accuracy a and the percentage of positives in the data is p, y_i will effectively disregard the information from the base classifier unless $a > p$ and $a > 1 - p$. This condition is violated at the italicized levels in Table 1, and although less unforgiving in the presence of an actual hierarchy, improvement generally suffers with such inaccurate classifiers. With base classifiers that are generally better than constant predictions, accuracy improvements are substantial.

The difference between having a broad and shallow hierarchy versus a narrow and deep one can be observed by comparing `T2D4` to `T5D2` (both have 31 nodes in total) and `T2D6` (127 nodes) to `T5D3` (156 nodes). Except for the degenerate 70% accurate case, `T5D2` and `T5D3` have larger improvement at the root than their narrower, deeper binary tree counterparts. However, the binary trees have larger improvements towards the lower levels, and therefore better accuracy on average, as the (unweighted) average is dom-

Table 1: **Synthetic Data Results.** A_0 is the base classifier accuracy, A_1 is the accuracy of the most likely labels after inference, averaged over all classes, and $\Delta A = A_1 - A_0$, broken down by nodes at different depths in the tree, as percentages. Italicized levels indicate base classifiers that are less accurate than a constant negative or positive prediction.

NAME	$\mathbf{A_0}$	$\mathbf{A_1}$	$\mathbf{\Delta A}$	**ΔA BY DEPTH (ROOT → LEAF)**						
Top-down, 75% positive at root				ROOT						
T2D4	70	78.6	8.5	*-8.8*	-3.6	3.2	9.2	*12.1*		
T2D6	70	84.8	14.8	*-18.9*	-10.6	-1.7	5.3	*11.6*	*15.8*	*18.7*
T5D2	70	66.9	-3.2	*-14.2*	-8.6	-1.7				
T5D3	70	70.3	0.3	*-22.6*	-16.1	-5.6	2.3			
T2D4	80	89.4	9.4	5.8	7.2	8.6	9.4	10.1		
T2D6	80	93.7	13.7	7.4	9.7	10.9	12.2	13.3	*13.8*	*14.3*
T5D2	80	86.1	6.2	12.1	9.7	5.3				
T5D3	80	89.4	9.4	18.7	13.5	12.0	8.7			
T2D4	90	95.9	6.1	6.5	6.8	6.6	6.4	5.6		
T2D6	90	97.8	7.8	6.7	6.9	7.9	8.2	8.2	8.1	7.6
T5D2	90	94.4	4.3	9.4	7.1	3.6				
T5D3	90	95.9	5.8	10.0	9.6	8.0	5.2			
Bottom-up, 1% positive at leaf										LEAF
T2D4	70	96.1	26.1			3.9	*16.0*	*23.3*	*26.9*	*29.1*
T2D6	70	95.3	25.2	*-24.3*	*-7.6*	7.0	*17.1*	*23.4*	*27.1*	*28.7*
T5D2	70	97.4	27.4					3.7	*24.3*	*29.0*
T5D3	70	96.9	26.9				*-48.5*	3.1	*24.0*	*29.0*
T2D4	80	97.1	17.1			3.9	10.6	*15.3*	*17.6*	*19.0*
T2D6	80	97.1	17.1	*3.1*	*4.9*	7.6	12.6	*15.5*	*17.6*	*19.0*
T5D2	80	97.3	17.3					-6.2	*14.0*	*18.9*
T5D3	80	97.3	17.3				*-6.0*	-0.6	*14.3*	*18.8*
T2D4	90	98.9	9.0			5.1	6.5	*8.4*	*9.0*	*9.7*
T2D6	90	99.0	9.0	*7.1*	*6.4*	7.0	7.6	*8.5*	*9.0*	*9.5*
T5D2	90	98.6	8.6					1.4	*6.7*	*9.2*
T5D3	90	98.5	8.5				0.2	1.7	*6.8*	*9.2*

inated by the large number of leaves. In other words, to reap the maximum benefit from Hierarchical Bayesian Aggregation, it is better for a node to have immediate children than to have them arranged in deeper levels, but deep cascading benefits more nodes, and yields larger average improvement in accuracy.

The distribution of improvement across the hierarchy varies with the label distribution and base accuracy. Although deeper nodes seem to get the largest improvement in general, the 90% accurate top-down data (the only case with no worse-than-constant classifiers) seems to favor mid-level classes in `T2D4` and `T2D6`, and higher-level classes in `T5D2` and `T5D3`.

3.1.2 ROC Analysis

Although we start out with base classifiers of equal accuracy on positives and negatives, their corresponding post-inference most-likely labels are affected by the skew of the data, specifically because Bayesian inference is improving accuracy. To observe this effect, it is useful to visualize classifiers on the ROC (receiver operating characteristics) axes, namely the true-positive ratio (accuracy on positives) against the false-positive ratio (error on negatives). Figure 3 shows an example, where each base classifier is a dot in the center cluster of 90% TP and FP ratios (variation is due to sampling), and each post-inference maximum-likelihood label, viewed as a predictor, is an asterisk.

The clustered spread of the post-inference predictors along an arc around the base classifiers can be explained by the property that points of equal accuracy (TP+TN) lie on parallel lines (*isoaccuracy* lines) on the ROC axes. While ROC coordinates have the property of being insensitive to the positive/negative skew of the data, skew changes the slope of the isoaccuracy lines, flatter than diagonal for positive majority, and steeper for negative majority. The effect of the Hierarchical Bayesian Aggregation is to make each classifier more accurate, moving it on the ROC axes perpendicularly to isoaccuracy lines, but since each level of the hierarchy has a different ratio of positives, these movements are in different directions, fanning out from the original common location. The post-inference asterisks in Figure 3 are visibly clustered by level, with the root at the top (flattest isoaccuracy line, most upward deflection) and the leaves in the bottom cluster (steepest isoaccuracy, most leftward deflection).

3.2 EVALUATIONS USING ACTUAL DATA

We next present experimental results on three real data sets. Because of different application goals, the three scenarios

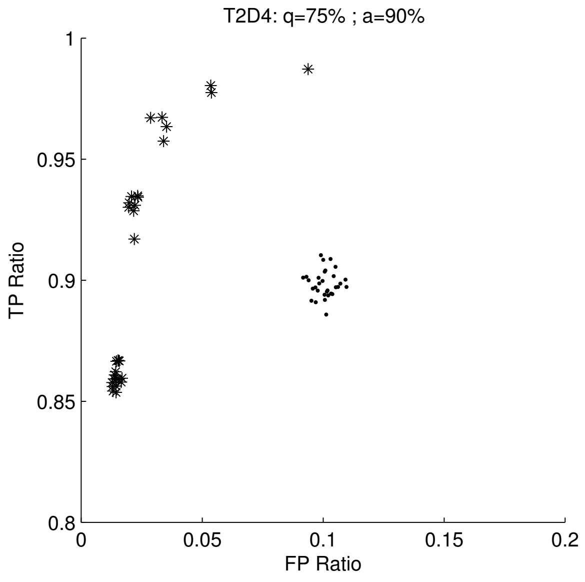


Figure 3: ROC plot detail of classes before and after HBA. Dots indicate base classifiers (variation due to sampling), and asterisks indicate post-inference predictions for each class (`T2D4` top-down data, $a = 90\%$). Post-inference coordinates are clustered along an arc because the increase in accuracy after inference moves each class in a direction determined by its positive/negative ratio, which depends on its depth in the hierarchy in our data.

are different in many ways, including the choice of base classifier, binary-*vs*-real valued outputs, parameter estimation strategy, and evaluation criteria. In all cases, Hierarchical Bayesian Aggregation provides significant improvement over independent base classifiers.

3.2.1 The Gene Ontology

The Gene Ontology project is a collaborative effort for annotating the genes of various model species over three structured controlled vocabularies (ontologies) corresponding to their biological processes, cellular components, and molecular function, where each ontology is described as a general-to-specific hierarchy of terms. The hierarchies are directed acyclic graphs, containing multiple parents and multiple top-level nodes. Each gene may be annotated to multiple classes in the hierarchy, as gene products can serve more than one purpose in the cell, and a gene may be annotated terminally to non-leaf nodes if no further information is available.

The common budding yeast, *Saccharomyces cerevisiae*, is a well-studied organism with plenty of information available on its gene products, and there has been increasing interest in predicting new Gene Ontology biological process annotations using various gene features from laboratory experiments [Chen and Yu 2004, Clare and King 2003, Karaoz *et al.* 2004, Lanckriet *et al.* 2004, Pavlidis *et al.* 2002, Troyanskaya *et al.* 2003], though these efforts typically focus on particular classes and ignore the hierarchical structure.

We used a 105-class subset of the biological process ontology, selected in consultation with a yeast geneticist for their practical value of new predictions and availability of data (at least one direct annotation and 15 total positives for yeast) in an April 2004 snapshot. The selected hierarchy has a maximum depth of seven, and includes multiple parents for many nodes. Using 3465 annotated yeast genes as positives (propagated up the hierarchy), and 5930 binary and real-valued input features for each gene from sources such as protein-protein interaction, colocalization, transcription factor binding sites, and coexpression microarray data, we trained 10 linear support vector machines for each class on bootstrap samples.

We used the median (*bagging* output [Breiman 1996]) of the 10 unthresholded SVM outputs as the g_i variables, and modeled $P(g_i \in \mathbb{R} | y_i = 0)$ and $P(g_i \in \mathbb{R} | y_i = 1)$ as Gaussians (see Figure 2) with parameters estimated using the "held-out" median output for each training example, i.e. the median of the outputs from classifiers whose training bootstrap sample did not include the example.

We chose this bagging approach for its use of all available data and reliability of performance. The alternatives included holding out some data for estimation and never using it for training, or first cross-validating to estimate parameters and then retraining a new classifier on all available data from scratch, for which no actual performance statistics would exist. Using the median over cross-validation folds instead of bootstrap samples does not provide enough diversity. For this real application where misclassification could waste costly laboratory trials, we found bagging to be a safe choice.

We used hierarchy edges in the upward direction which yielded slightly better results than downward. Since the Gene Ontology records very few actual negative annotations, we used every non-positive example as a negative in a class if it was not directly annotated to an ancestor of that class. This left open the possibility of further specializing known annotations while providing sufficiently many negatives to work with.

The application objective was to obtain a ranking of most probable gene annotations for particular classes to guide laboratory experimentation. Instead of the most likely hierarchy-wide binary label configuration for each example, we used inference to obtain the marginal membership probabilities for each class $P(y_i | g_1, \ldots, g_n)$, and compared these to the ranking accuracy of the corresponding base classifier output over all examples in terms of the *AUC score* (area under the ROC curve), which can be interpreted as the probability of ranking a random positive example higher than a random negative example. Figure 4 shows the scatterplot of AUC scores of post-inference marginal probabilities versus the AUC scores of bagged base classifier outputs for each class. 93 of 105 nodes improved, with an average AUC increase of 0.033 (4% relative improvement over the old AUC). Improvements were at all levels, though somewhat larger at the deeper nodes.

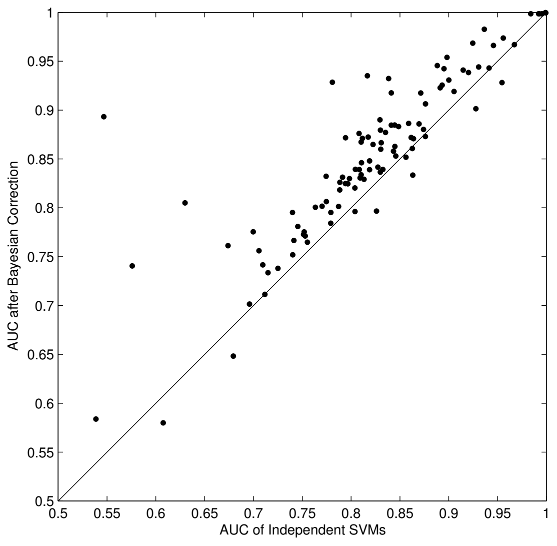


Figure 4: Scatterplot of AUCs for the 105 Gene Ontology classes, after *vs.* before HBA. Points above the diagonal correspond to improvements by our method.

Comparing Bayesian inference predictions to the median-bagged SVMs outputs on the 88 previously unannotated genes for which new Gene Ontology annotations that became available in July 2005 showed that, with the default threshold of zero, SVMs had 32% average precision (TP/(TP+FP)) and 7% recall (TP/(TP+FN)) over the 346 new gene-to-node pairs, whereas thresholding Bayesian marginals for comparable precision (31%) yielded 21% recall, and thresholding for the same 7% recall yielded 51% precision.

We examined in detail five unannotated genes for which Bayesian inference predicted functions related to mitosis, and laboratory testing found strong evidence for mitosis-related function for three genes in particular [Barutcuoglu *et al.* 2006]. All of these predictions were introduced by our hierarchical system, i.e. they were not predicted as positive by the individual base classifiers prior to inference.

3.2.2 Princeton Shape Benchmark

The Princeton Shape Benchmark [Shilane *et al.* 2004] is a computer graphics collection of 1814 three-dimensional object models arranged in a tree hierarchy of 170 classes. Several input feature representations are available in the benchmark, and we used the Spherical Harmonic descriptor (SHD) [Kazhdan *et al.* 2003] which has been shown to perform well for shape matching. SHD represents each object as a 544-dimensional vector, and is designed to encode the rotation-invariant similarity of two objects as the Euclidean distance of their feature vectors. This Euclidean distance property of the descriptor lends itself to using the k-nearest neighbors (kNN) algorithm without worrying about the relative importance of different features, where a test example is assigned the majority label among its k nearest neighbors from the training set.

To obtain held-out statistics without wasting training data, we used two-fold cross-validation (the scarcity of positives in many classes prevented using more folds). Namely, we split the examples of each class into two halves, built a binary-output kNN classifier for each half, and evaluated its performance on the other half. The two confusion matrices from the two halves were added up to yield one confusion matrix for that class to be used as the $P(g_i|y_i)$ estimate. The value of k for each class was chosen as the best $k \in \{1, 3, 5, 7, 9\}$ by leave-one-out cross-validation accuracy (134 of 170 classes chose $k = 1$, none chose $k = 9$).

We then built our Bayesian network and inferred each example's most likely set of binary labels, as well as its marginal probability for each class. Comparing the binary most-likely labels to the base classifiers, 132 of 170 classes improved in accuracy. Figure 5 shows a scatterplot of accuracies for each class before and after Hierarchical Bayesian Aggregation. 38 remained the same, probably due to the base classifiers being already over 99% accurate.

A more dramatic comparison was between the ranking performances of the binary base classifiers and the real-valued marginal probabilities from Hierarchical Bayesian Aggregation. Since a kNN has binary outputs, it corresponds to a single point on the ROC axes. However, for any two points (classifiers), one can interpolate a classifier anywhere on the line connecting them by choosing between their predictions with a Bernoulli distribution, so for each class, we defined the kNN's ROC curve as its single point connected to $(0, 0)$ and $(1, 1)$, and compared the area under it to the AUC of the marginal probabilities from inference. All classes except one improved, with the average AUC over all classes increasing from 0.70 to 0.84 [Barutcuoglu and DeCoro 2006].

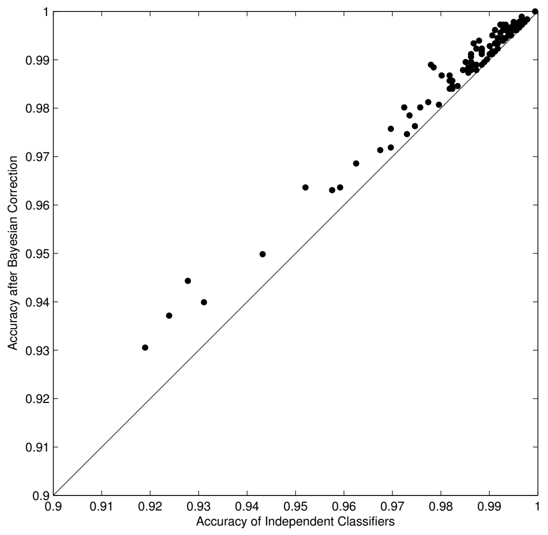


Figure 5: Scatterplot of accuracies on the Princeton Shape Benchmark classes, after *vs.* before HBA.

3.2.3 MIREX Symbolic Music Dataset

The annual MIREX competition is an event sponsored by the International Symposium on Music Information Retrieval to evaluate the state-of-the-art in methods for extracting high-level information from musical examples. In 2005, the contest featured both audio and symbolic (MIDI) genre classification tasks, in which 950 individual songs were to be classified into a given hierarchy of musical genres, with 38 leaf classes [McKay and Fujinaga 2005]. The winning participants of the symbolic classification task employed the Bodhidharma MIDI classification system (initially presented in [McKay and Fujinaga 2004], and described in depth in [McKay 2004]), which extracts 111 high-level features related to instrumentation, rhythm, dynamics and chords.

In our subsequent application of HBA to this classification task [DeCoro *et al.* 2007], we used the Bodhidharma features as inputs to SVM base classifiers (as the features are not Euclidean, kNN classification was less applicable). Probabilities were computed using 3-fold cross-validation, obtaining 3 SVMs for each class. Each example is used as training for two, and class prediction is given for that example by the third. Because of the large number of negative examples per class relative to positive examples (a characteristic known as *skew*, especially significant in the MIREX dataset), error is computed as *skew-insensitive accuracy*, which penalizes errors on the few positive examples proportionally higher than errors on the ample negatives. This measure is the average of sensitivity (accuracy on positive examples) and specificity (accuracy on negative examples):

$$0.5 \frac{\textit{true positives}}{\textit{true pos.} + \textit{false neg.}} + 0.5 \frac{\textit{true negatives}}{\textit{true neg.} + \textit{false pos.}}.$$

Average skew-insensitive accuracy over all 55 classes was 76.8% for independent SVMs, and improved to 85.1% after Bayesian Aggregation thresholded at $p > 0.5$. Figure 6 shows a scatterplot of each class before and after aggregation.

The MIREX 2005 contest reported a "raw accuracy," which relies on the one-leaf-only nature of song labels in this dataset and picks as the genre prediction the leaf node with the highest output. Under this multi-class single-label criterion, the winning Bodhidharma system scored 46.1; the median raw accuracy was 41.0. Using the same feature set, our SVM base classifiers achieved 56.0 raw accuracy (their system used a more complicated heuristic classifier). While SVMs already improved significantly relative to previous state-of-the-art, HBA accuracy further improved to 60.1.

In a related application, also pertinent in music information retrieval, is to search for songs "similar" in genre to a query song, or equivalently, rank all songs in a database by similarity to the query song using genre classification as a measure. For our experiments, we defined the similarity of two songs as the number of their equal binary labels in the hierarchy, which decreases as the path distance of their classes in the hierarchy increases. We computed the

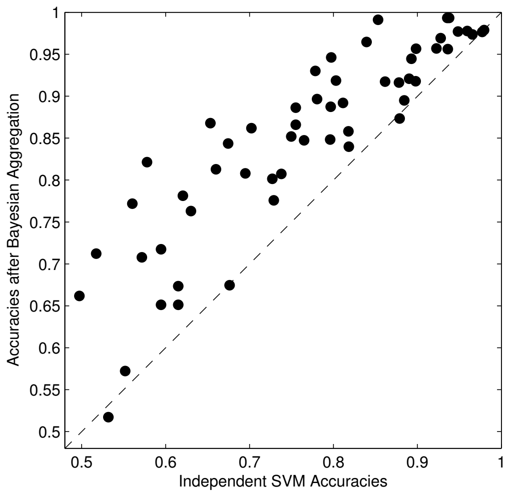


Figure 6: Scatterplot of MIREX skew-insensitive accuracies before *vs.* after HBA.

"true similarity" of every pair of songs using the actual labels, and the predicted similarities from both independent SVMs and subsequently the Bayes-aggregated predictions. To avoid selecting an arbitrary threshold, classes along the branch of the maximum-confidence leaf were selected as the positive labels for each example. Using each song as the query, all other songs were sorted by similarity, and the top predicted results were compared to the top results as given by true similarity. Across all examples, of the 100 most-similar songs an average of 52% were retrieved by independent SVMs, improved to 62% with aggregation. Similarly, of the top 50, an initial 46% retrieval rate improved to 52% after aggregation.

4 CONCLUSION

We outlined a general and principled method for resolving hierarchical inconsistencies among a set of independent classifiers, thus increasing accuracy over all classes. Its usefulness was demonstrated in experiments on synthetic and real data. We also described various extensions of the method to handle other constraints, as well as a special case for ordinal classification.

Future work on Hierarchical Bayesian Aggregation might provide extensions where the inference algorithm optimizes other performance criteria than overall accuracy, such as explicitly maximizing AUC scores. The sensitivity of the method to parameter estimation errors may also be explored.

Acknowledgments

The authors would like to thank Camelia Chiriac and Rajiv Ayyangar for laboratory work; Philip Shilane, Rebecca Fiebrink, Corey McKay, Kara Dolinski and members of the Functional Genomics Laboratory for their contributions of additional datasets and reviews on earlier drafts of this text; and Professor Szymon M. Rusinkiewicz of the Princeton Graphics Lab for providing additional funding and support.

CRD was supported as an ATI/AMD Technologies Fellow. OGT is an Alfred P. Sloan Fellow. This research was supported by NSF grant IIS-0513552 and partially supported by NIH grant R01 GM071966 to OGT, and NSF grant IIS-0513552 to SMR.

References

[Barutcuoglu *et al*. 2006] Barutcuoglu,Z., Schapire,R.E., Troyanskaya,O.G. (2006) Hierarchical multi-label prediction of gene function, *Bioinformatics*, **22**:830-836.

[Barutcuoglu and DeCoro 2006] Barutcuoglu Z., DeCoro,C. (2006) Hierarchical Shape Classification Using Bayesian Aggregation, *Proc. IEEE Intl. Conf. on Shape Modeling and Applications (SMI'06)*.

[Breiman 1996] Breiman,L. (1996) Bagging Predictors, *Machine Learning*, **24(2)**:123-140.

[Cai and Hofmann 2004] Cai,L., Hofmann,T. (2004) Hierarchical Document Categorization with Support Vector Machines, *ACM 13th Conference on Information and Knowledge Management*, 2004.

[Cesa-Bianchi *et al*. 2004] Cesa-Bianchi,N., Conconi,A., Gentile,C. (2004) Regret Bounds for Hierarchical Classification with Linear-Threshold Functions. *The 17th Annual Conference on Learning Theory*, pp. 93-108. LNAI 3120, Springer.

[Chen and Yu 2004] Chen,Y., Yu,D. (2004) Global protein function annotation through mining genome-scale data in yeast *Saccharomyces cerevisiae*, *Nucleic Acids Res.*, 2004 Dec 7;**32(21)**:6414-24.

[Clare and King 2003] Clare,A. and King,R.D. (2003) Predicting gene function in *Saccharomyces cerevisiase*, *Bioinformatics* **19(2)**:1142-49.

[DeCoro *et al*. 2007] DeCoro,C., Barutuoglu,Z., Fiebrink,R. (2007) Bayesian Aggregation for Hierarchical Genre Classification, *International Symposium on Music Information Retrieval*, 2007

[Dekel *et al*. 2004] Dekel,O., Keshet,J., Singer, Y. (2004) Large margin hierarchical classification, ÎCML'04, pp. 209-216.

[Dumais and Chen 2000] Dumais,S.T., Chen,H. (2000) Hierarchical classification of Web content, *Proc. 23rd ACM Intl. Conf. on Research and Development in Information Retrieval (SIGIR'00)*, pp. 256-263.

[Durand *et al*. 2004] Durand,J.-B., Gonalvs,P., Gudon,Y. (2004) Computational Methods for Hidden Markov Tree ModelsAn Application to Wavelet Trees, *IEEE Transactions on Signal Processing*, Sep 2004; **52(9)**:2551-2560.

[Karaoz *et al*. 2004] Karaoz,U., Murali,T.M., Letovsky,S. *et al.* (2004) Whole-genome annotation by using evidence integration in functional-linkage networks, *Proc. Natl. Acad. Sci. USA*, **101(9)**:2888-93.

[Kazhdan *et al*. 2003] Kazhdan,M., Funkhouser,T., Rusinkiewicz,S. (2003) Rotation invariant spherical harmonic representation of 3D shape descriptors, *Symposium on Geometry Processing*, pp. 167-175.

[Lanckriet *et al*. 2004] Lanckriet,G.R., Deng,M., Cristianini,N. *et al.* (2004) Kernel-based data fusion and its application to protein function prediction in yeast, *Proc. 9th Pacific Symposium on Biocomputing*, 6th-10th Jan., Hawaii, pp. 300-311.

[McKay 2004] McKay, C. "Automatic Genre Classification of MIDI Recordings," *M.A. Thesis*, McGill University, Canada, 2004.

[McKay and Fujinaga 2004] McKay, C. and I. Fujinaga. "Automatic Genre Classification Using Large High-Level Musical Feature Sets," *Proc. ISMIR*, 2004.

[McKay and Fujinaga 2005] McKay, C. and I. Fujinaga. "The Bodhidharma system and the results of the MIREX 2005 symbolic genre classification contest," *Proc. ISMIR*, 2005.

[Pavlidis *et al*. 2002] Pavlidis,P., Weston,J., Cai,J., Noble,W.S. (2002) Learning gene functional classifications from multiple data types, *J. Comput. Biol.* 2002;**9(2)**:401-11.

[Shilane *et al*. 2004] Shilane,P., Kazhdan,M., Min,P., Funkhouser,T. (2004) The Princeton Shape Benchmark, *Proc. Shape Modeling International*, Genoa, Italy.

[Troyanskaya *et al*. 2003] Troyanskaya,O.G., Dolinski,K., Owen,A.B. *et al.* (2003) A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiase*), *Proc. Natl. Acad. Sci. USA*, **100(14)**:8348-8353.