BAYESIAN AGGREGATION FOR HIERARCHICAL GENRE CLASSIFICATION

Christopher DeCoro **Zafer Barutcuoglu** **Rebecca Fiebrink**
Princeton University
Department of Computer Science

ABSTRACT

Hierarchical taxonomies of classes arise in the analysis of many types of musical information, including genre, as a means of organizing overlapping categories at varying levels of generality. However, incorporating hierarchical structure into conventional machine learning systems presents a challenge: the use of independent binary classifiers for each class in the hierarchy can produce hierarchically inconsistent predictions. That is, an example may be assigned to a class, and not assigned to the parent of that class. This paper applies a Bayesian framework to combine, or *aggregate*, a hierarchy of multiple binary classifiers in a principled manner, and consequently improves performance over the hierarchy as a whole. Furthermore, such an approach allows for an arbitrarily complex hierarchy, and does not suffer from classes that are too broad or too refined. Experiments on the MIREX 2005 symbolic genre classification dataset show that our Bayesian Aggregation algorithm provides significant improvement over independent classifiers, and demonstrates superior performance compared to previous work. Our method also improves similarity search by ranking songs by similarity of hierarchical predictions to those of a query song.

1 INTRODUCTION

Many musical concepts are inherently hierarchical. Some notion of hierarchy is implicitly or explicitly at play in concepts such as genre and mood (which have overlapping coarse and fine categories), instrument timbre (which is grouped by instrument families), and meter. When hierarchy has been accommodated in automatic classification of these concepts, it has generally been in quite simple ways (such as the top-down approach of [9]) or in ways that are acutely tailored to the learning task and/or classification method at hand (e.g. [8, 14]).

We have developed a technique called Bayesian Aggregation, which uses the output predictions of arbitrary independent classifiers (such as k-nearest neighbor, support vector machines, etc.) which we refer to as the *base classifiers*, and *aggregates* them in such a way as to take advantage of the hierarchical nature of the predictions to improve classification accuracy.

We demonstrate performance of this method on genre classification, a popular classification task in music information retrieval. Genre is a culturally relevant and practically useful concept, and genre classification systems have the potential to be quite useful in organizing and allowing efficient access to music databases as shown by McKay in [12]. Further, the same work argued that multiple class assignments and user-specified ontological structure are beneficial in principle; both of these are inherently supported by Bayesian Aggregation.

We describe the genre dataset and features used in Section 2.1, and provide background on classification algorithms in Section 2.2. We describe our algorithm in Section 3, and demonstrate in Section 4 that the use of Bayesian Aggregation allows for a significant improvement in genre classification accuracy when compared either to existing methods, or to the predictions of independent classifiers without aggregation. Further, it does so in a way that guarantees that the predictions will be consistent with the constraints of the hierarchy; that is, once an instance is assigned to a leaf class, it will also be assigned to the leaf's parent classes. We also demonstrate that the outputs of such a classification system may offer improvements to similarity search systems built on genre classifiers.

2 BACKGROUND

2.1 Genre classification

Genre classification has been a popular task in music information retrieval since originally posed by Tzanetakis [15]. In 2005, the MIREX contest featured both audio and symbolic musical genre classification tasks [11]. The winning participants of the symbolic genre classification task employed the Bodhidharma MIDI classification system (presented in [10] and expanded in [9]) that extracts 111 high-level features related to instrumentation, rhythm, dynamics, and chords. Bodhidharma also considers hierarchical relationships via a top-down classification approach, wherein classifier outputs at each branch of a hierarchy determine which child classifiers will be used to further refine the classification. In our evaluation, we used the Bodhidharma features from the MIREX 2005 38-leaf class hierarchy and 950-item symbolic genre dataset.

2.2 Classification

Classification is a well-studied problem in the machine learning literature. One popular choice of classifier is the

support vector machine (SVM), for which an accessible overview is given in [4]. Given a labeled training set, which includes both positive (members of the genre) and negative example vectors, the SVM finds the *maximally-separating hyperplane* in a kernel-transformed vector space between the two subsets. To classify a novel, unlabeled example, we can compute its distance to the maximally-separating hyperplane. SVMs have been used extensively, and there exists significant theoretical and empirical evidence of their classification efficacy.

In this work we exclusively use support vector machines as a base classifier, due to its ability to support wide ranges of data without assumptions on the distribution of values (kNN classifiers, by contrast, expect that the Euclidean L^2 norm is meaningfully defined over the feature vector space, which is not the case for these features). However, our method can be used to improve the results of any type of classifier; we have shown its utility with kNN classifiers in a previous work [1].

3 ALGORITHM DESCRIPTION

As mentioned, we perform hierarchical classification of music examples by aggregating the results of multiple independent classifiers according to a Bayesian framework, to perform collaborative error correction over their possibly-inconsistent predictions. We build the framework in a training phase using labeled example data, and subsequently use the pre-computed framework to assign labels to novel data in a classification phase. The reader is referred to [3] for details and analyses beyond those presented here.

3.1 Training Phase

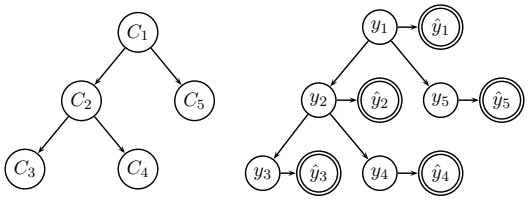
The first step to our algorithm is to train a set of individual base classifiers to predict memberships for each class in the hierarchy. We call this initial set of predictions the *base classification*. We use base classifications from the training set in order to generate a Bayesian network that describes the hierarchy. A Bayesian network involves a number of random variables, some of which are observed directly, while others are hidden [6]. Of these variables, some are assumed to be conditionally dependent on others. We can visually represent this as a graph, as in Figure 1. Nodes represent variables, and edges represent conditional dependence. Given values for observed nodes, Bayesian inference algorithms use this network to calculate probabilities for hidden node values, or find the most probable configuration of hidden node value assignments consistent with the observations.

For a given example, let y_i denote the actual binary membership to class i, $\hat{y}_i$ denote the base classifier prediction for that class, and $\vec{y}_{parents(i)}$ denote the actual membership to superclasses of i. For example, y_i might represent membership in the Smooth Jazz genre, with $\hat{y}_i$ the prediction of the base classifier trained to recognize that genre, and $\vec{y}_{parents}$ would then represent membership in the Modern Pop and Fusion genres.

After obtaining a set of (possibly inconsistent) $\hat{y}$ predictions from all base classifiers, we wish to find the most probable set of consistent y labels that may be underlying them. Therefore, for N nodes we need to find the labels $y_1 \dots y_N$ that maximize the conditional probability $P(y_1 \dots y_N | \hat{y}_1 \dots \hat{y}_N)$, which by Bayes rule equals

$$\frac{P(\hat{y}_1 \dots \hat{y}_N | y_1 \dots y_N) P(y_1 \dots y_N)}{Z}, \qquad (1)$$

where Z is a constant normalization factor. We propose a Bayesian network structure for this problem, as illustrated by Figure 1. The class hierarchy shown at left is transformed into a Bayesian network by adding extra nodes that correspond to the observed classifier outputs. The y-nodes are probabilistically dependent on their parent classes, and the $\hat{y}$-nodes are probabilistically dependent on their corresponding labels y.



(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 $\hat{y}$ nodes are the observed classifier outputs.

We enforce hierarchical consistency of labels using the edges among the y-nodes. The edges encode the conditional dependencies $P(y_i | \vec{y}_{parents(i)})$, where $\vec{y}_{parents(i)}$ is used to denote all parent y-nodes of node y_i, which we set to ensure that a label must be 0 if any of its parents is 0. The remaining entries $P(y_i | \vec{y}_{parents(i)} = 1)$ are inferred from the training set. We condition each y-node only on its parents, thereby limiting the complexity of the Bayes net to one that is both tractable to infer and constrained to avoid overfitting, producing the following simplification:

$$P(y_1 \dots y_N) = \prod_{i=1}^{N} P(y_i | \vec{y}_{parents(i)}). \qquad (2)$$

The edges from y to $\hat{y}$ reflect an important observation: for a given example, a classifier prediction $\hat{y}_i$ is conditionally independent of all other classifiers' predictions $\hat{y}_j$ and labels y_j ($i \neq j$) given its true label y_i. This simplifies Equation 1, since we can write

$$P(\hat{y}_1 \dots \hat{y}_N | y_1 \dots y_N) = \prod_{i=1}^{N} P(\hat{y}_i | y_i). \qquad (3)$$

$P(\hat{y}_i | y_i)$ consists of $P(\hat{y}_i | y_i = 1)$ and $P(\hat{y}_i | y_i = 0)$, which are the distributions for the base classifier predictions, and can be estimated during training by validation. We obtain a sampling of the distributions $P(\hat{y}_i | y_i = 1)$ and $P(\hat{y}_i | y_i = 0)$, using cross validation on the training data. We use the continuous SVM outputs, without thresholding at zero, as the observed $\hat{y}_i$ values, and model these two distributions as Gaussians, computing their means and variances for each class over the SVM outputs for positive and negative examples, respectively.

3.2 Classification Phase

The first step to assigning a genre to a novel example is to compute a base classification using the trained per-class SVMs. In our Bayesian network, this corresponds to observing values for the $\hat{y}_i$ nodes. A Bayesian inference algorithm will then find the most likely configuration of (consistent) hidden y labels for the given $\hat{y}$ predictions, or the marginal distribution $P(y_i|\hat{y}_1 \ldots \hat{y}_N)$ for each class separately. We use the marginal probabilities $P(y_i = 1|\hat{y}_1 \ldots \hat{y}_N)$ in our results so that we retain real-valued membership probabilities and can threshold them at different levels as desired. Among the many Bayesian inference algorithms available, in our experiments we used the *junction tree* algorithm for exact inference, although approximate inference with Monte Carlo methods such as Gibbs sampling may be more practical for more complex hierarchies. Descriptions and detailed references for these and other inference algorithms are available in [13].

4 EXPERIMENTAL RESULTS

4.1 Genre Classification

For class hierarchies in general, and in particular for the the MIREX 2005 dataset, each training example belongs to very few nodes, in comparison to all the other nodes for which it counts as a negative example. In consequence, for each node the number of negative examples is disproportionately larger than the number of positives, a characteristic known as *skew*. Since machine learning algorithms favor simpler models, when possible, to avoid overfitting the training data, failing to take skew into account can produce classifiers that unconditionally predict negative, as this is an ultimately simple model with seemingly very high accuracy. Instead, a skew-insensitive performance measure is necessary, both for optimizing during training and for analyzing results, which will penalize errors on the few positive examples proportionally higher than errors on the ample negatives. The standard skew-insensitive accuracy is the average of sensitivity (accuracy on positive examples) and specificity (accuracy on negative examples)

$$0.5\frac{\textit{true positives}}{\textit{true pos.} + \textit{false neg.}} + 0.5\frac{\textit{true negatives}}{\textit{true neg.} + \textit{false pos.}}. \quad (4)$$

On the MIREX 2005 symbolic genre classification dataset, we trained linear SVMs using the *SVMlight* software [7] with the appropriate cost factors to compensate for class skew. We used 3-fold cross-validation, obtaining three SVMs for each class. Each example is used as training for two, and a class prediction is given for that example by the third. The Bayesian network was then constructed using these distributions as previously described, and marginal probabilities were computed for the consistent hidden labels using Bayesian inference.

Average skew-insensitive accuracy over all 55 classes was 76.8% for independent SVMs, and 85.1% after Bayesian Aggregation thresholded at $p > 0.5$. Figure 2 shows a scatterplot of each class before and after aggregation.

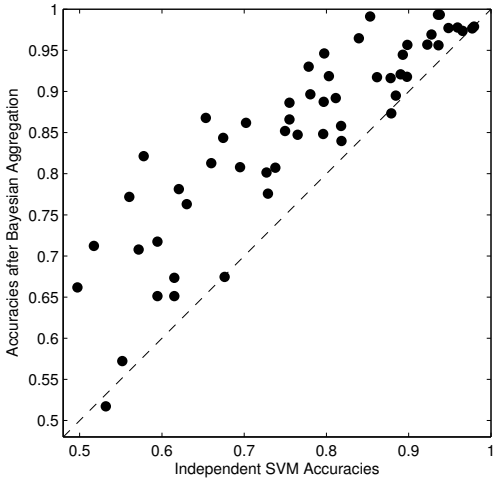


Figure 2. Scatterplot of skew-insensitive accuracies for each class before *vs.* after Bayesian Aggregation.

To provide a fair comparison of our results directly against previous work using this dataset, we also computed the "raw accuracy" statistic reported in the MIREX 2005 contest results, 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, 56.0% of all examples were labeled correctly by independent SVMs, and 60.1% after Bayesian Aggregation, compared to the MIREX 2005 contest entries in Table 1. Considering that this requires the choice of one correct class out of 38, where a random guess would have less than 3% accuracy, both the improvement over previous results and the improvement of Bayesian Aggregation over independent SVMs are significant.

Algorithm	Raw Accuracy
Bayesian Aggregation	60.1%
Independent SVMs	56.0%
Bodhidharma	46.1%
Basili et al. (NB)	45.0%
Basili et al. (J48)	41.0%
Li	39.8%
Ponce de Leon & Inesta	15.3%

Table 1. Independent SVMs and Bayesian Aggregation compared to MIREX 2005 symbolic genre classification contest entries by single-label multi-class "raw accuracy."

4.2 Similarity Search

A related application is to search for songs "similar" to a query song by genre, or equivalently, rank all songs in a database by genre similarity to the given song. Again it is assumed that only a small part of the available data is manually labeled, so the query song as well as the retrieved songs might have no certain genre label. While one could first classify each song as above into a discrete genre and then retrieve the other songs in that genre as the most similar, the hierarchy provides a means of defining inter-class similarity as well.

For our experiments, we defined 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 "true similarity" of every pair of songs using the actual labels, and the predicted similarities from independent SVMs outputs and then for the Bayes-aggregated predictions. To avoid selecting an arbitrary threshold, the 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, while the aggregated predictions retrieved 62%. Similarly, of the top 50, SVMs retrieved 46%, compared to 52% after Bayesian Aggregation.

West and Lamere have used Euclidean distance between genre classifier soft outputs to perform similarity computation for playlist generation and collection visualization, but without regard to hierarchical class organization [16]. Our results suggest that applying Bayesian Aggregation to such a system could improve on their approach.

5 CONCLUSIONS AND FUTURE WORK

Bayesian Aggregation offers an elegant and generalizable means of taking hierarchy into account in performing a classification task, and thus offers several benefits to music information retrieval researchers. Our experiments demonstrate that Bayesian Aggregation is able to significantly improve genre classification accuracy, both compared to the base classifiers used alone, as well as to previous work. This is done at the minor cost of training and evaluation of parent-node classifiers in addition to leaf-node classifiers; the time required for additional evaluations is on the order of milliseconds. By allowing for a soft assignment to related classes, our algorithm is able to effectively expand the training set for each class, which is particularly important for datasets such as this with a limited number of examples.

This has the important consequence that our method enables arbitrarily refined or broad taxonomies; it remains robust even when considering classes for which the individual classifiers may be inaccurate, thereby decoupling taxonomy design from concerns of system performance. The algorithm does this by discovering inaccuracies during the training phase and implicitly discounts such classifiers' predictions on new data. Furthermore, this approach is superior to heuristic top-down or bottom-up approaches that place excessive responsibility on the root or leaf classifiers, respectively, without providing any means to correct bad predictions at those levels. Most importantly, Bayesian Aggregation is able to improve accuracy for any type of base classifier, as suited to the task. Although we used SVMs for this application, the system is independent of this choice, and one can use any other classification algorithm as applicable, such as neural networks or kNN.

Given these characteristics of our algorithm, there exist many avenues of future work, which we intend to explore. Our algorithm is currently capable of handling multi-label classification problems, such as genre or mood, where a song can be labeled to multiple classes at various levels. We therefore intend to empirically demonstrate its efficacy on such tasks. Additionally, our algorithm can be specialized to make single-leaf predictions, potentially further improving performance. We are excited about the potentials that hierarchy-aware methods present, and have presented this work as a significant first step.

6 ACKNOWLEDGMENTS

We would like to thank Cory McKay for sharing the Bodhidharma MIDI dataset used for evaluation of this work, and for his helpful advice and opinions. This work was partially supported by an ATI/AMD Technologies Fellowship and NSF grant IIS-0513552.

7 REFERENCES

[1] Barutcuoglu, Z. and C. DeCoro. "Hierarchical Shape Classification Using Bayesian Aggregation," *Proc. Shape Modeling International*, 2006.

[2] Barutcuoglu, Z., R.E. Schapire and O.G. Troyanskaya. "Hierarchical Multi-label Prediction of Gene Function," *Bioinformatics*, January 2006.

[3] Barutcuoglu, Z., C. DeCoro, R.E. Schapire and O.G. Troyanskaya. "Bayesian Aggregation for Hierarchical Classification," *Technical Report TR-785-07*, Princeton University, Department of Computer Science, 2007

[4] Burges, C.J.C. "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, 1998.

[5] Flach, P.A. "The geometry of ROC space: understanding machine learning metrics through ROC isometrics," *Proc. 20th International Conference on Machine Learning,* 2003.

[6] Heckerman, D. "A Tutorial on Learning with Bayesian Networks," *Learning in Graphical Models,* MIT Press, Cambridge, MA, 1999.

[7] Joachims, T. "Making large-Scale SVM Learning Practical," *Advances in Kernel Methods - Support Vector Learning*, MIT-Press, Cambridge, MA, 1999.

[8] Klapuri, A., A. Eronen and J. Astola. "Analysis of the meter of acoustic musical signals," *IEEE Trans. Speech and Audio Processing* 14(1), 2006.

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

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

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

[12] McKay, C. and I. Fujinaga. "Musical genre classification: Is it worth pursuing and how can it be improved?" *Proc. ISMIR*, 2006.

[13] Murphy, K. "The Bayes Net Toolbox for MATLAB," *Computing Science and Statistics,* 2001.

[14] Rauber, A., E. Pampalk and D. Merkl. "Using pycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity," *Proc. ISMIR,* 2002

[15] Tzanetakis, G., G. Essl and P. Cook. "Automatic musical genre classification of audio signals," *Proc. ISMIR*, 2001.

[16] West, K. and P. Lamere. "A Model-Based Approach to Constructing Music Similarity Functions," *EURASIP Journal on Advances in Signal Processing*, 2007.