Performances of low-level audio classifiers for large-scale music similarity

Julien Osmalskyj, Marc Van Droogenbroeck, Jean-Jacques Embrechts
INTELSIG Laboratory, Departement EECS
University of Liège, Belgium
{josmalsky, m.vandroogenbroeck, jjembrechts}@ulg.ac.be

***Abstract*—This paper proposes a survey of the performances of binary classifiers based on low-level audio features, for music similarity in large-scale databases. Various low-level descriptors are used individually and then combined using several fusion schemes in a content-based audio retrieval system. We show the performances of the classifiers in terms of pruning and loss and we demonstrate that some combination schemes achieve a better performance at a minimum computational cost.**

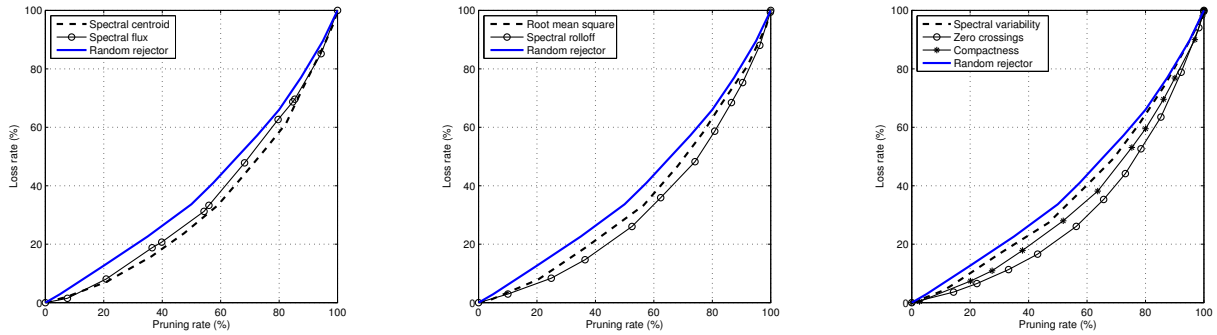
I. Introduction

Recent years have seen an increased interest in music and audio similarity in the Music Information Retrieval (MIR) community [1], [2], [3]. One example application is an audio matching system, whose goal is to retrieve similar items to queries, based on audio data only, and not on textual information such as tags. Such search systems are called Content-Based Retrieval Systems (CBRS) and can be implemented in two steps. The first one is a pruning step, whose goal is, given an audio query, to eliminate as many unrelated songs as possible without losing correct matches. The second step is a matching operation, on the remaining subset, which can be performed by an expert, human or computer based. This paper focuses on the first step. When dealing with large databases (typically more than 1 million items), the pruning stage should be efficient. Therefore, the features used for similarity computation should not be computationally expensive so that the retrieval cost is minimal [4]. However, most of the existing retrieval systems are based on relatively complex features such as chroma features [2] or Mel-frequency cepstral coefficients (MFCCs) [5]. In this paper, we investigate the performance of low-level features individually, as well as their combination, in order to determine if they can be useful in a retrieval system. We make use of weak classifiers called *rejectors* [3]. A rejector is a classifier, based on an audio feature, which aims at deciding whether two songs are considered similar (positive class C_+) or dissimilar (negative class C_-). Rejectors can be used for the pruning step of a CBRS. To evaluate the features, we use the Million Song Dataset (MSD) [6] coupled with the Second Hand Song Dataset (SHSD) which is a subset of the MSD composed of similar songs. As low-level features are not available by default in the MSD, we relied on the work of Schindler *et al.* [7], in which many other features have been computed on the Million Song Dataset, including low-level features (details are given in Section II). We denote the dataset provided in [7] by IFS dataset (IFSD). Combining the MSD, the SHSD and the IFSD therefore provides us a large-scale dataset suitable for similarity measures research. We show in this paper the performance of the features used individually and their combination using various fusion schemes. Combination methods from the literature are used. In particular, we implement boolean combinations using the union, intersection, and majority vote methods [8], [9], [10]. Techniques based on probabilities are also presented and investigated. We implemented a combination scheme based on the probabilistic product and the sum [11]. Finally, a composite classifier was learned using machine learning techniques, specifically the ExtraTrees algorithm [12]. We show that combination of low-level audio features does not necessarily improve the performance of a retrieval system. However, we show that some combining schemes bring better results than using individual features.

The remainder of this paper is organized as follows. Section II introduces the used features as well as the dataset used for evaluation. Section III describes the low-level based classifiers and presents their performance. In Section IV, we present the combination schemes used and the results obtained with fusion methods. In Section V, we discuss the results and finally, Section VI concludes the paper.

II. Experimental setup

In this section, we detail our experimental setup. Our goal is to experiment with large-scale audio retrieval systems. To this end, we use the only large-scale audio dataset available, namely the Million Song Dataset (MSD) [6]. The MSD contains textual and content-based features computed on one million songs. It represents a subset of the EchoNest database (http://echonest.com). Unfortunately, we do not have any information about the algorithms used to compute the features of the MSD, and therefore, they are difficult to reproduce. Moreover, the MSD provides the features only, and no audio data, making it unusable for computation of new features. In [7], Schindler *et al.* proposed the IFSD, another dataset derived from the MSD, in which they computed new features on the same songs (*e.g.* rhythm patterns, low-level features, *etc.*), using state of the art reproducible techniques. Note that the IFSD does not contain exactly one million songs, since the authors were not able to retrieve the audio data corresponding to each song of the MSD. However, more than 99% of the MSD is present in the IFSD ($994,623$ songs). In this paper, we



(a) (b) (c)

Figure 1: Performance of the individual classifiers based on low-level audio features in terms of pruning rate and loss rate using the IFSD. None of the feature achieves a convenient pruning rate. The zero crossing classifier achieves the best performance.

make use of a subset of the IFSD containing low-level audio features computed with the JMir extraction framework [13]. For each feature, the authors computed the average value on a range of analysis windows, as well as the standard deviation. As we experimentally noticed that the standard deviation does not bring any improvement over average, we decided to only use the average values of the features in our work. For this paper, we used the seven following features: *spectral centroid*, *spectral rolloff point*, *spectral flux*, *compactness*, *spectral variability*, *root mean square* and *zero crossings*. Each of them is described in [13].

For evaluation, we use the SHSD, which is a subset of the MSD organized in 4128 *cliques*. A clique gathers items of the MSD which are related to the same underlying song. Thus, a clique contains several versions of the same song, allowing us to evaluate features in the context of music similarity.

In the next section, we explain how we built classifiers on top of these low-level features and we show the performance obtained with these classifiers.

III. Elementary classifiers

A. Rejectors

For each feature of the IFSD, a *rejector* $\mathcal{R}$ takes two tracks T_1 and T_2 as an input and returns the class similar (C_+) or dissimilar (C_-). Each rejector is based on one low-level feature, therefore, we call our rejectors *weak rejectors*. To take its decision, a rejector makes use of a probability model learned using machine learning techniques. For each feature, we learned a model from the SHSD using the ExtraTrees [12] algorithm, which is a machine learning technique based on a forest on randomized trees. The model predicts the probability $\hat{P}_s$ that two tracks are similar. A rejector next classifies a pair of tracks in either the class "similar" C_+ or the class "dissimilar" C_- by thresholding the probability computed with the ExtraTrees algorithm. Each rejector was learned with a total of 1000 trees. To avoid overfitting of the models, the depth of the trees is limited and the optimal depth is found by maximizing the area under the *Receiver Operating Characteristic* (ROC). To fit the learning algorithm, we derived six attributes for each pair of features $(f1, f2)$: $min(f1, f2)$, $max(f1, f2)$, $abs(f1, f2)$, $abs(f1-f2)/(f1+f2)$, $(f1+f2)$ and $min(f1, f2)/max(f1, f2)$.

In the next section, we describe our evaluation method using the IFSD and the SHSD.

B. Evaluation

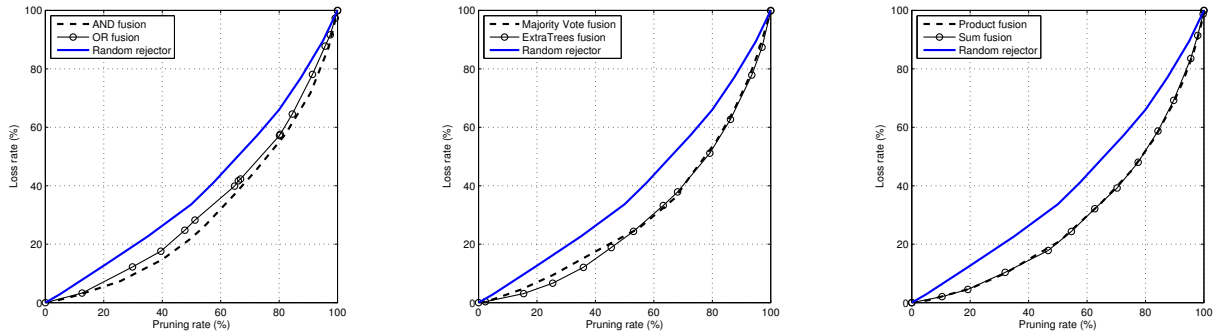
To evaluate our low-level features based classifiers, we use the IFSD coupled with the SHSD. For each value of a threshold Th and for each clique, we compare each song of the clique against every song of the IFSD using a rejector. Thus, for each track T_c of a clique, the rejector outputs a decision $\hat{s}$ for the pair (T_c, T_i), where $T_i \in$ IFSD such that $T_i \neq T_c$. It should be stressed that the decision $\hat{s}$ taken by a rejector is an approximation of the real class value s computed using the machine learning model. Using the rejector, we can next compute a *pruning rate* and a *loss rate* for a song and for a clique. The pruning rate corresponds to the mathematical expectation, over all possible queries, of the proportion of irrelevant samples from the IFSD that are pruned. We have

$$prune = p(\hat{s} = C_-|s = C_-) \quad (1)$$

The loss rate is the mathematical expectation, over all possible queries, of the probability to discard all matching samples in the IFSD. We therefore consider that there is a loss only if every correspondence to a query has been dropped from the pruned subset. For a query q, we have $\text{loss}(q) = [p(\hat{s} = C_-|s = C_+)]^{|c(q)|}$, where $|c(q)|$ corresponds to the size of the clique q belongs to. Thus, the total loss is computed as follows.

$$loss = \sum_{m=1}^{\infty} p\left(|c(q)| = m\right) [p(\hat{s} = C_-|s = C_+)]^m \quad (2)$$

Each rejector is therefore evaluated in terms of its pruning and loss rates for a range of values of a threshold Th. The results are plotted on *prune-loss* curves [3].



(a) (b) (c)

Figure 2: Performance of the composite classifiers in terms of pruning rate and loss rate using the IFSD. The improvements over the individual classifiers are not significant. The best combination is obtained with the product and sum rules.

C. Performance of elementary rejectors

The performance of each classifier is depicted in Figure 1 as prune-loss curves. Each curve is plotted against the random rejector curve, which takes its decisions completely randomly. As we can see, the performance of the elementary classifiers are better than the random rejectors, but clearly, the maximum pruning rate that can be achieved is quite small (around 20% of pruning for a maximum loss of less than 10%).

It is important to realize that the used features are very low-level and do not correspond to any musical feature and are therefore not much suited for music similarity, as it appears clearly in Figure 1. They are just weak descriptors directly computed from the spectrum of the audio signal. One feature, namely the zero-crossings average value, can however be related to the overall frequency of the audio signal, as it corresponds to the number of times the signal crosses the zero axis. This feature actually performs better than the other features in our music similarity experiment, as shown in Figure 1c.

IV. Composite classifiers

In this section, we investigate the combination of our elementary classifiers and we analyze whether fusion of classifiers can increase or decrease the performance of the low-level features for music similarity. We created composite rejectors using several combination schemes. We analyzed fusion schemes based on boolean combinations, probabilistic combinations and one scheme based on machine learning. Note that composite boolean and probabilistic classifiers make the assumption that the features are independent.

A. Boolean fusion schemes

1) AND combination: The AND combination corresponds to the boolean intersection of each individual rejector. The composite rejector returns C_+ only if every single rejector returns C_+, and returns C_- otherwise. The performance of that combination is shown in Figure 2a.

2) OR combination: The OR combination corresponds to the boolean union of each individual rejector. The composite rejector returns C_+ only if at least one elementary rejector returns C_+, and returns C_- otherwise. The performance of the combination can be seen in Figure 2a.

3) Majority Vote (MV) combination: That combination scheme returns C_+ only if a majority of elementary rejectors returns C_+ and returns C_- otherwise. The performance is depicted in Figure 2b.

The boolean combination producing the best results is the AND combination, which is followed by the Majority Vote, and the OR fusion. However, none of the combinations outperforms the best results obtained with an elementary classifiers, namely the zero-crossings classifier.

B. Probabilistic fusion schemes

Probabilistic combination schemes are based directly on the probability that songs are similar or dissimilar. In these schemes, the elementary rejectors are therefore slightly modified to output a probability value rather than a class C_+ or C_-.

1) Product rule: The product rule multiplies the probabilities of the elementary rejectors and normalizes them according to [14]. Note that the final probability is thresholded as explained in Section III-B so that the composite rejector returns an output class C_+ or C_-. The composite probability for seven elementary rejectors is computed as

$$p_\times = \frac{\frac{1}{p(C_+)^6}\prod_{j=1}^{7} p(C_+, f_j)}{\frac{1}{p(C_+)^6}\prod_{j=1}^{N} p(C_+, f_j) + \frac{1}{p(C_-)^6}\prod_{j=1}^{7} p(C_-, f_j)} \tag{3}$$

where f_j corresponds to the feature used with rejector j. According to [15], the product rule is good if the individual classifiers are independent.

2) Sum rule: The sum rule is equivalent to the product rule, but it does perform a sum instead of a product. We have

$$p_+ = \frac{1}{7}\sum_{j=1}^{7} p_j \tag{4}$$

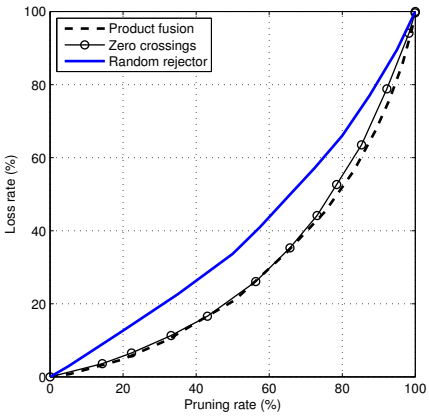


Figure 3: The improvement of the product classifier against the zero crossings classifier is not much significant.

where p_j is the probability returned by an elementary classifier. According to [15], the sum rule can be useful in reducing the noise in large-sets of weak classifiers. The results obtained with the product and sum rules are depicted in Figure 2c. Clearly, best performance is achieved with these combinations, compared to the other schemes. We can notice that the product and sum rules produce similar results.

C. Machine learning fusion

We have also investigated a combination technique using a trained classifier. Instead of using solely the results of each individual rejector, we built a new composite feature set derived from each individual feature. Thus, for each of the seven features, we compute six attributes as explained in Section III-A, and we store all the 42 values in the dataset. Based on this new dataset, we computed a composite model using the ExtraTrees [12] learning algorithm containing 100 trees, and we used it in our evaluation method by thresholding the resulting probabilities. The performance of the ExtraTrees composite rejector can be seen in Figure 2b. For lower pruning rates, it outperforms the Majority Vote fusion scheme. However, for higher pruning rates, the performance is equivalent to the Majority Vote scheme. Note however that this classifier is the slowest one from a computational point of view.

V. Discussion

From our experiments, we can realize that no combination rule really outperforms the performance obtained with the zero-crossing elementary classifier, which achieves the best results. The composite classifier achieving the best performance is the product classifier which only outperforms the zero-crossing rejector slightly, as can be seen in Figure 3.

The small difference between these results could be explained by the fact that, for some fusion schemes, the combination works under the assumption of independence of the features, which is hardly the case with our low-level features. It should be stressed, however, that despite the low improvements, the product combination rule (and the sum as well since their results are similar) always produces best results.

VI. Conclusion

In this paper, we presented an evaluation of several low-level audio features in the context of music similarity. We created weak-classifiers called rejectors and evaluated elementary and composite classifiers in a similarity framework, in terms of pruning and loss rates. The conclusion of our observation is two-fold: first, the maximum achievable pruning rate with low-features is small. Clearly, they are not well suited for similarity measures as we cannot achieve a high pruning with these features. Secondly, the combination of weak classifiers does not bring much improvement over elementary classifiers. Indeed, the best results are obtained with the product and sum rules, which slightly decrease the loss rate (at a fixed pruning rate), but not significantly.

However, we believe that the product rule coupled with stronger features, more related to musical information, such as chroma features, rhythm patterns of MFCC coefficients, could lead to a significant increasing of the pruning rate, while keeping the loss minimal.

References

[1] M. Casey and M. Slaney, "Fast recognition of remixed audio," in *Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, 2007.

[2] F. Kurth and M. Muller, "Efficient index-based audio matching," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 382–395, 2008.

[3] J. Osmalskyj, S. Piérard, M. Van Droogenbroeck, and J. Embrechts, "Efficient database pruning for large-scale cover song recognition," in *Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, Vancouver, Canada, May 2013, pp. 714–718.

[4] H. Turtle and J. Flood, "Query evaluation: strategies and optimizations," *Information Processing & Management*, vol. 31, no. 6, pp. 831–850, 1995.

[5] M. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proc. of IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[6] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Int. Symp. Music Inform. Retrieval (ISMIR)*, 2011.

[7] A. Schindler, R. Mayer, and A. Rauber, "Facilitating comprehensive benchmarking experiments on the million song dataset," in *Int. Symp. Music Inform. Retrieval (ISMIR)*, 2012, pp. 469–474.

[8] R. Duin and D. Tax, "Experiments with classifier combining rules," in *Multiple Classifier Systems*, ser. Lecture Notes in Comp. Science. Springer, 2000, vol. 1857, pp. 16–29.

[9] T. Ho, J. Hull, and S. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66–75, Jan. 1994.

[10] L. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[11] L. Kuncheva, J. Bezdek, and R. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition*, vol. 34, no. 2, pp. 299–314, Feb. 2001.

[12] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

[13] C. McKay, I. Fujinaga, and P. Depalle, "jaudio: A feature extraction library," in *Proceedings of the International Conference on Music Information Retrieval*, 2005, pp. 600–3.

[14] J. Kittler, M. Hatef, and R. Duin, "Combining classifiers," in *IEEE Int. Conf. Pattern Recognition (ICPR)*, vol. 2, Vienna, Austria, Aug. 1996, pp. 897–901.

[15] R. Duin, "The combining classifier: to train or not to train?" in *IEEE Int. Conf. Pattern Recognition (ICPR)*, vol. 2, Quebec City, Canada, Aug. 2002, pp. 765–770.