To be presented as a plenary lecture at the 2008 International Symposium on Music Information Retrieval, September 2008.

LEARNING A METRIC FOR MUSIC SIMILARITY

Malcolm Slaney
Yahoo! Research
2821 Mission College Blvd.
Santa Clara, CA 95054
malcolm@ieee.org

Kilian Weinberger
Yahoo! Research
2821 Mission College Blvd.
Santa Clara, CA 95054
kilian@yahoo-inc.com

William White
Yahoo! Media Innovation
1950 University Ave.
Berkeley, CA 94704
wwhite@yahoo-inc.com

ABSTRACT

This paper describe five different principled ways to embed songs into a Euclidean metric space. In particular, we learn embeddings so that the pairwise Euclidean distance between two songs reflects semantic dissimilarity. This allows distance-based analysis, such as for example straightforward nearest-neighbor classification, to detect and potentially suggest similar songs within a collection. Each of the six approaches (baseline, whitening, LDA, NCA, LMNN and RCA) rotate and scale the raw feature space with a linear transform. We tune the parameters of these models using a song-classification task with content-based features.

1 INTRODUCTION

Measuring the similarity of two musical pieces is difficult. Most importantly, two songs that are similar to two lovers of jazz, might be very different to somebody that does not listen to jazz. It is inherently an ill-posed problem.

Still, the task is important. Listeners want to find songs that are related to a song that they like. Music programmers want to find a sequence of songs that minimizes jarring discontinuities. A system based on measurements from hundreds of thousands of users is perhaps the ultimate solution [8], but there is still a need to find new songs, before an item-to-item system has enough data.

It is difficult to construct a distance calculation based on arbitrary features. A simple approach places the feature values into a vector and then calculates an Euclidean distance between points. Such a calculation implies two things about the features: their independence and their scale. Most importantly, a Euclidean metric assumes that features are (nearly) orthogonal so the distance along different axis can be summed. A Euclidean metric also assumes that each feature is equally important. Thus a distance of 1 unit in the X direction is perceptually identical to one unit in the Y direction. This is unlikely to be true, and this paper describes principled means of finding the appropriate weighting.

Much work on music similarity and search calculates a feature vector that describes the acoustics of the song, and then computes a distance between these features. In this work we describe six means of assigning weights to the dimensions and compare their performance. The purpose of this paper is not to determine the best similarity measure—after all evaluation of a personal decision such as similarity is difficult—but instead to test and compare several quantitative approaches that MIR practitioners can use to create their own similarity metric. West's recent paper provides a good overview of the problem and describes successful approaches for music similarity [11]. We hope to improve the performance of future systems by describing techniques for embedding features in a metric space.

We measure the performance of our system by testing identification ability with a k-nearest neighbor (kNN) classifier. A kNN classifier is based on distances between the query point and labeled training examples. If our metric space is "good" then similar songs will be close together and kNN classification will produce the right identification. In our case, we try to identify the album, artist or blog associated with each song.

A kNN classifier has several advantages for our task. A kNN classifer is simple to implement, and with large amounts of data they can be shown to give an error rate that is no worse than twice the optimal recognizer [2]. Simple classifiers have often been shown to produce surprisingly good results [5]. The nearest-neighbor formulation is interesting in our application because we are more interested in finding similar songs, than we are in measuring the distance between distant songs or conventional classification. Thus kNN classification is a good metric for measuring our ability to place songs into a (linear) similarity space.

2 DATA

Our data comes from the top 1000 most-popular mp3 blogs on the Web, as defined by music blog aggregator, The Hype Machine's "TOP MUSIC BLOGS On The Net" [1]. We analyzed each new mp3 track that was posted on these mp3 blogs during the first three weeks of March 2008.

In the ongoing quest to discover new music, music blogs provide an engaging and highly useful resource. Their cre-

[1] http://hypem.com/toplist

ators are passionate about music, given that they're blogging about it all the time. And unlike a traditional playlist or set of recommended tracks, music blogs also provide personal commentary which gives the blog consumer a social context for the music.

We're interested in comparing the similarity across music posted on the same blog to the similarity between different tracks by the same artist or from the same album.

One of the difficulties in gathering this type of data is the large amount of uncertainty and noise that exists within the metadata that describes mp3s. Our general experience analyzing Web mp3s has been that less than a third of the tracks we encounter have reliable (if any) metadata in their ID3 tags. Therefore the metadata we used for our artists and album calculations is limited to what we were able to parse out of valid ID3 tags or information we could infer from the filename, or the HTML used to reference the track.

In these 1000 blogs, we found 5689 different songs from 2394 albums and 3366 artists. Just counting blogs for which we found labeled and unique songs, we were left with 586 different blogs in our dataset. After removing IDs for which we did not have enough data (less than 5 instances) we were left with 74 distinct albums, 164 different artists, and 319 blogs. The style of music represented in this collection differs from blog to blog. Many mp3 blogs could be broadly classified as "Indie" or "Indie Rock", but music shared on a specific blog is more representative of the blogger's personal taste than any particular genre.

3 FEATURES

We characterized each song using acoustic analysis provided via a public web API provided by The Echo Nest [4]. We send a song to their system, they analyze the acoustics and provide 18 features to characterize global properties of the songs. Although we did not test it, we expect that features from a system such as Marsyas [9] will give similar results.

The Echo Nest Analyze API splits the song into segments, each a section of audio with similar acoustic qualities. These segments are from 80ms to multiple seconds in length. For each segment they calculate the loudness, attack time and the other measures of the variation in the segment. There are also global properties such as tempo and time signature. The features we used are as follows [4]:

- segmentDurationMean: mean segment duration (sec.).
- segmentDurationVariance: variance of the segment duration (sec.^{2})—smaller variances indicate more regular segment durations.
- timeLoudnessMaxMean: mean time to the segment maximum, or attack duration (sec.).
- loudnessMaxMean: mean of segments' maximum loudness (dB).
- loudnessMaxVariance: variance of the segments' maximum loudness (dB^2). Larger variances mean larger dynamic range in the song.
- loudnessBeginMean: average loudness at the start of segments (dB).
- loudnessBeginVariance: variance of the loudness at the start of segments (dB^2). Correlated with loudnessMaxVariance
- loudnessDynamicsMean: average of overall dynamic range in the segments (dB).
- loudnessDynamicsVariance: segment dynamic range variance (dB^2). Higher variances suggest more dynamics in each segment.
- loudness: overall loudness estimate of the track (dB).
- tempo: overall track tempo estimate (in beat per minute, BPM). Doubling and halving errors are possible.
- tempoConfidence: a measure of the confidence of the tempo estimate (beween 0 and 1).
- beatVariance: a measure of the regularity of the beat (secs.^{2}).
- tatum: estimated overall tatum duration (in seconds). Tatums are subdivisions of the beat.
- tatumConfidence: a measure of the confidence of the tatum estimate (beween 0 and 1).
- numTatumsPerBeat: number of tatums per beat
- timeSignature: estimated time signature (number of beats per measure). This is perceptual measures, not what the composer might have written on the score. The description goes as follows: 0=None, 1=Unknown (perhaps too many variations), 2=2/4, 3=3/4 (eg waltz), 4=4/4 (typical of pop music), 5=5/4, 6=6/4. 7=7/4 etc.
- timeSignatureStability: a rough estimate of the stability of the time signature throughout the track

4 ALGORITHMS

We create a feature vector by concatenating the individual feature-analysis results (we used the order described in Section 3, but the order is irrelevant). Let us denote all input features as the matrix f, which is an mxn array of n m-dimensional feature vectors, one vector for each song's analysis results. Further let f_i be the i^{th} feature (column-)vector in f. To measure the distances between different feature vectors, we use learned Mahalanobis metrics [6].

A Mahalanobis (pseudo-)metric is defined as

$$d(f_i, f_j) = (f_i - f_j)^\top \mathbf{M} (f_i - f_j), \qquad (1)$$

where $\mathbf{M}$ is any well-defined positive semi-definite matrix. From Eq. (1) it should be clear that the Euclidean distance is a special case of the Mahalanobis metric with $\mathbf{M} = \mathbf{I}$, the identity matrix. We considered five different algorithms from the research literature to learn a Mahalanobis matrix to convert the raw features into a well-behaved metric space. Each of the algorithms either learns a positive semi-definite

matrix $\mathbf{M}$ or a matrix A, such that $\mathbf{M} = A^\top A$. We can uniquely decompose any positive semi-definite matrix as $\mathbf{M} = A^\top A$, for some real-valued matrix A (up to rotation). This reduces Eq. (1) to

$$d(f_i, f_j) = \|A(f_i - f_j)\|_2, \tag{2}$$

the Euclidean metric after the transformation $f_i \rightarrow Af_i$.

One of the approaches—whitening—is unsupervised, i.e. the algorithm does not require any side-information in addition to the pure feature vectors f. The other four use labels to tune the Mahalanobis matrix A so that similar songs are likely to be close to each other in the metric space. In this study we use album, artist and blog labels for each song as a measure of similarity. We evaluate our algorithm by testing the performance of a nearest-neighbor classifier in these new spaces.

The output of our algorithms is $F = Af$. The learned matrix A is of size $m \times m$ in this paper, but also can be $m' \times m$, where $m' < m$, in which case it reduces the dimensionality of the output space. The result matrix F has n points arrayed so similar points are close together. We partition the algorithms that we discuss into two groups: algorithms based on second-order statistics, and algorithms based on optimization. We will discuss each in turn.

4.1 Algorithms based on second-order statistics

The first three algorithms learn a linear transformation of the input space based on second-order statistics of the feature vectors. These methods rely heavily on the spread of information as captured by an outer product in the covariance calculation

$$cov(f) = \frac{1}{n}\sum_i (f_i - \bar{f}_i)(f_i - \bar{f}_i)^\top \tag{3}$$

where $\bar{f}_i$ is the mean of the feature vector over all songs. This equation is used in two different ways. The within-class covariance function is calculated from all vectors within one class and the between-class covariance is calculated from the means of all class clusters.

Whitening

The easiest way to massage the data is to normalize each dimension of the feature vector so that they all have the same energy. A more sophisticated approach adds rotations so that the covariance matrix of the whitened data is diagonal. We do this by computing

$$A_w = [cov(f)]^{-1/2} \tag{4}$$

where $cov(\cdot)$ is the covariance of a matrix. The covariance of fA_w is the identity matrix. This approach is completely unsupervised because whitening does not take into account what is known about the songs and their neighbors. Whitening is important as it removes any arbitrary scale that the various features might have. To use whitening as a pre-processing for distance computation was originally proposed by Mahalanobis [6] and is the original formulation of the Mahalanobis metric. A potential draw-back of whitening is that it scales all input features equally, irrespective of whether they carry any discriminative signal or not.

LDA

Linear discriminant analysis (LDA) is a common means to find the optimal dimensions to project data and classify it. It is often used as a means of rotating the data and projecting it into a lower-dimensional space for dimensionality reduction.

LDA assumes that the data is labeled with (normally) two classes. It further assumes that the data within each class is distributed with a Gaussian distribution and further assumes that each class of data shares the same covariance matrix. This is likely not true in our case since some artists or albums are more diverse that others. In this work we use a multi-class formulation for LDA proposed by Duchene [3].

LDA optimizes class distinctions, maximizing the between-class spread while minimizing the within-class spread. This procedure is based on the assumption that each class is independently sampled from a single uni-modal distribution so the distribution is characterized by a single mean and variance, which may not apply in many more complicated real world scenarios.

RCA

Relevant component analysis (RCA) [1] is related to whitening and LDA as it is entirely based on second-order statistics of the input data. One can view RCA as local within-class whitening. Different from LDA, it does not maximize the between-class spread and therefore makes no uni-modal assumption on the data.

4.2 Algorithms based on optimization

The next two algorithms explicitly learn a matrix A by minimizing a carefully constructed objective function that mimics the kNN leave-one-out classification error.

The problem with optimizing a nearest-neighbor classifier is that the objective function is highly non-continuous and non-differentiable. Many changes to the solution, the A matrix in this case, make no change to a point's nearest neighbors and thus no change to the objective function. Then an infinitesimally small change to A will shift the nearest neighbors and the objective function will make a large jump. The two algorithms we consider next introduce two different surrogate loss functions whose minimization

loosely translates into the minimization of the kNN leave-one-out classification error.

NCA

In neighborhood component analysis (NCA) the hard decisions incumbent in identifying nearest neighbors are replaced with a soft decision based on distance to the query point [7]. Instead of defining the nearest neighbor as the closest feature vector, Goldberger et al. use a soft-neighborhood assignment. For a given feature vector f_i, the nearest neighbor f_j is picked at random with probability

$$p_{ij} = \frac{e^{-d(f_i,f_j)}}{\sum_k e^{-d(f_i,f_k)}}. \quad (5)$$

In other words, the probability that f_j is a nearest-neighbor of f_i decreases exponentially with the distance between them. Given this objective function, one can compute the probability, p_i, of a data point f_i within class C_i has a nearest neighbor within the same class:

$$p_i = \sum_{f_j \in C_i} p_{ij}. \quad (6)$$

The objective of NCA is to maximize the probability that each point has neighbors in the same class,

$$A_{nca} = \text{argmin} \sum_i p_i(A) \quad (7)$$

where the point probabilities depend implicitly on M. In words, the algorithm maximizes the expected number of classified input feature vectors under a probabilistic 1-NN classification. As Eq. (5) is continuous and differentiable so is the objective in Eq. (6), which can be maximized with standard hill-climbing algorithms such as gradient descent, or conjugate gradient.

The $O(N^2)$ cost of the calculation is mitigated because the exponential weighting falls off quickly so many distant pairs can be safely ignored.

LMNN

The last approach we investigated is large-margin nearest neighbor (LMNN) [10]. Similar to NCA, LMNN is also based on an optimization problem. However, instead of a smooth, non-convex objective, LMNN mimics the kNN leave-one-out classification error with a piecewise linear convex function. This minimization can be solved with well-studied semidefinite programs, which can be solved with standard optimiziation algorithms such as interior-point or sub-gradient descent. A key step in making the objective convex is to fix *target neighbors* for each input vectors prior to learning. These target neighbors must be of the same class and should be close under some reasonable metric. The objective tries to minimize the distance of an input vector to its target neighbors, while enforcing that no differently labeled inputs come closer than 1 unit from the target neighbor.

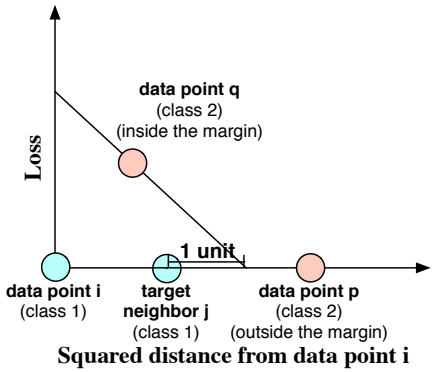

Figure 1. The loss function for LMNN. The loss (or error) increases rapidly for points not in class 1 that encroach too close to a point in the query class.

Partially inspired by support vector machines (SVM), the objective consists of two parts: One that forces target neighbors to be close, and a second that forces a margin between an input vector and differently-labeled vectors. Let $j \rightsquigarrow i$ denote that f_j is a target neighbor of f_i, then we write the objective as

$$\sum_{j \rightsquigarrow i} d(f_i, f_j) + \sum_{j \rightsquigarrow i} \sum_{k \notin C_i} [d(f_i, f_j) + 1 - d(f_i, f_k)]_+ , \quad (8)$$

where C_i is the class of f_i and $[a]_+ = max(a, 0)$. The second term of this objective function pushes dissimilar points at least one unit away, as illustrated in Figure 1.

As the objective function in Eq. (8) is piece-wise linear, it can be efficiently minimized over large data sets $N > 60000$. One potential weakness of LMNN is the fact that the target neighbors are fixed before the optimization and their choice significantly impacts the final metric. In this paper we chose them to be the nearest neighbors under the Euclidean metric after whitening.

5 EVALUATION

We evaluate each metric algorithm by testing a kNN classifier's ability to recognize the correct album, artist or blog that describe each song. We do this by organizing all data by ID, and then selecting enough IDs so that we have more than 70% of all songs in a training set. The remaining data, slightly less than 30%, is a test set. The classifier's task is to look at each test point, and see if at least 2 of its 3 neighbors have the desired ID.

Thus we train a Mahalanobis matrix on a large fraction of our data, and test the matrix by measuring identification

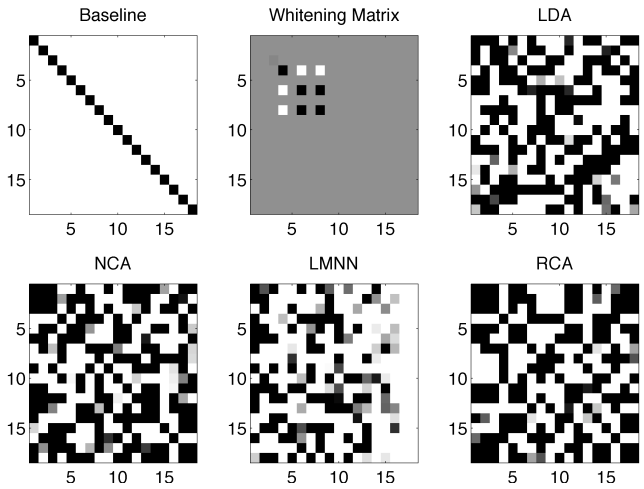

Figure 2. Summary of Mahalanobis matrices derived by each algorithm based on album similarity. The whitened matrix has both positive and negative terms centered around the gray of the background. The other matrices have been scaled so that white is at 0 and black represents the maximum value. All features are in the same order as described in Section 3.

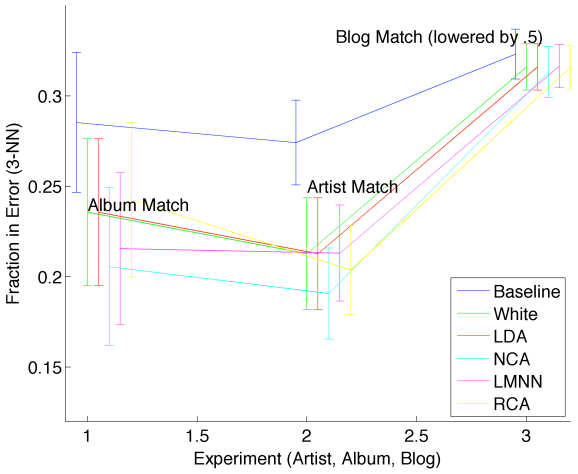

Figure 3. Summary of metric algorithms kNN performance. The results for each of the six metrics are offset horizontally, in the order shown in the legend, to facilitate comparison. Note, the performance of the blog-identification task was very poor, and we have reduced each blog error by 0.5 to fit them on the same graph as the others.

performance on data it has never seen. We do this for album and artist ID, and also try to predict the blog that mentions this song.

We found that adding a whitening stage before *all* algorithms improved their performance. Thus each of the four algorithms we investigated (LDA, NCA,LMNN, RCA) are preceded by a full whitening step. We also compare these algorithms to two strawman: a baseline using the original (unweighted) feature space, and the whitened feature space with no other processing.

We are interested in the robustness of these algorithms to noise. To test this, we measured performance as we added noisy features. Each feature is a zero-mean Gaussian random variable with a standard deviation that is 10% of the average for the real features. This level is arbitrary since all algorithms performed best, in testing, when the data is whitened first. This removes the level dependence on all but the baseline data.

6 RESULTS

Figure 2 shows all 6 Mahalanobis matrices. The four entries in the whitening matrix with the largest values are the four loudness features. Except for LDA, the other matrices make relatively small changes to the feature space.

Yet, these small changes in the Mahalanobis matrices have significant difference in performance. Figure 3 shows the performance of all six approaches on all three identification tasks. We performed 10 trials for each classification test, in each case choosing at random new (non-overlapping) training and testing sets. In these tests, NCA did best, but LMNN and RCA were close seconds on the album-match and artist-match tasks respectively. In our tests, both album and artist ID are relatively easy, but identifying the blogger who referenced the song was problematic. This suggests that album and artists are better defined than a blogger's musical interests.

Figure 4 shows the performance as we add noisy features. In this experiment we added noisy dimensions to simulate the effect of features that are not relevant to the musical queries. The error for the baseline quickly grows, while the other methods do better. This test was done for the album-recognition task. In this case NCA and RCA perform best, and this is significant because the computational cost of RCA is trivial compared to that of NCA and LMNN.

One explanation for the relative win of RCA over LMNN (and LDA) is that the later algorithms try to push different classes apart. This might not be possible, or even a good idea when there are as many classes as in our experiment. There just isn't any room for the classes to separate. Thus in our tests, especially when noise is added, the overlapping classes are not amenable to separation.

All of these algorithms have the ability to do feature selection and reduce the dimensionality of the feature space. LDA and RCA order the dimensions by their ability to predict the data clouds, so it's natural to cut off the smaller dimensions. Both NCA and LMNN are essentially optimization problems, and by posing the problem with a rectangular instead of a square A matrix one can calculate a low-

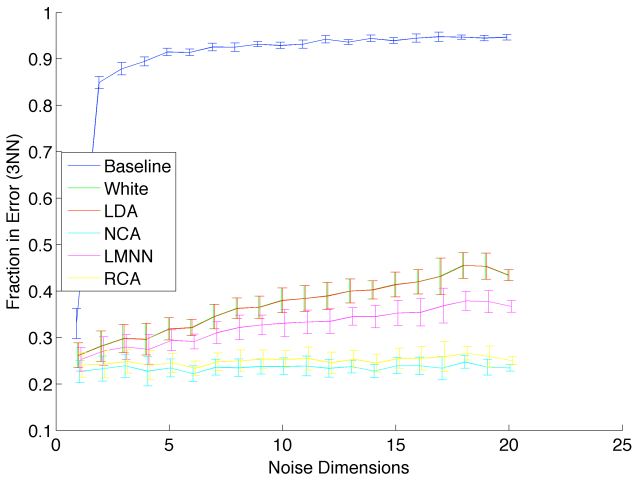

Figure 4. Summary of metric algorithms kNN performance with additional noisy features. We have augmented the original 18 features with additional purely noise features to measure the ability of each algorithm to ignore the noisy dimensions. The results for each of the six metrics are offset horizontally, in the order shown in the legend, to facilitate comparison.

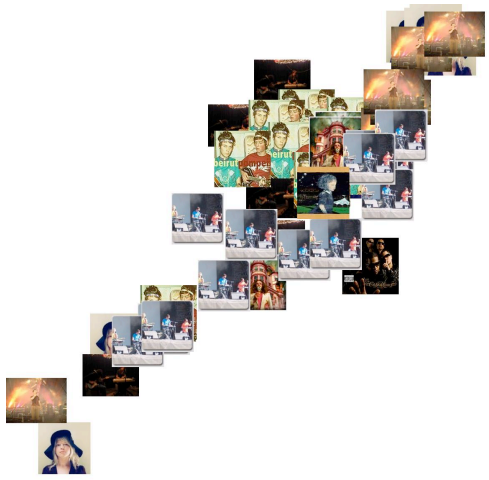

Figure 5. We used NCA to find the optimal projection of the 9 most common artists. Each image indicates the location of one or more songs by that artist in a two-dimensional space.

dimensional embedding. We illustrate this kind of dimensionality reduction in Figure 5. Dimensionality reduction can be important in a nearest-neighbor recognizer because one must store all the prototypes, and the dimensionality of the feature space directly links to the amount of memory needed to store each song's feature vector.

7 CONCLUSIONS

In this paper we have described and demonstrated 6 different means to embed acoustic features into a metric space. In the best-performing cases the algorithms use meta data about the songs—in our case album, artist, or blog IDs— to tune the space so that songs with the same ID are close to each other. With our data, more than 5000 songs described on music blogs, we found that all algorithms lead to a significant improvement in kNN classification and, in particular, NCA and RCA perform by far most robustly with noisy input features. More work remains to be done to verify that these results produce sensible playlists and pleasing transitions. We would also like to investigate which features are important for these problems.

8 REFERENCES

[1] A. Bar-Hillel, T. Hertz, N. Shental, D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *J. of Machine Learning Research,* 6, pp. 937–965, 2005.

[2] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. in Information Theory, IT-13*, pp. 21–27, 1967.

[3] J. Duchene and S. Leclercq. An Optimal transformation for discriminant principal component analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 10(6), Nov. 1988.

[4] The Echo Nest Analyze API. http://developer.echonest.com/docs/analyze/xml xmldescription, downloaded March 31, 2008.

[5] R. Holte. Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, 11(1), pp. 63–90, 1993.

[6] P. Mahalanobis. On the generalized distance in statistics. *Proc. Nat. Inst. Sci. India (Calcutta)*, 2, pp. 49–55, 1936.

[7] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NIPS),* 2004.

[8] M. Slaney and W. White. Similarity based on rating data. *Proc. of the International Symposium on Music Information Retrieval*, Vienna, Austria, 2007.

[9] George Tzanetakis and Perry Cook. Music analysis and retrieval systems. *Journal of American Society for Information Science and Technology,* 55(12) 2004.

[10] Kilian Q. Weinberger, John Blitzer, Lawrence K. Saul. Distance metric learning for large margin nearest-neighbor classification. in *Advances in Neural Information Processing Systems 18,* Vancouver, BC, Canada, December 5–8, 2005.

[11] Kris West and Paul Lamere. A model-based approach to constructing music similarity functions. *EURASIP Journal on Advances in Signal Processing,* vol. 2007.