# ISLANDS OF GAUSSIANS: THE SELF ORGANIZING MAP AND GAUSSIAN MUSIC SIMILARITY FEATURES

**Dominik Schnitzer[1,2], Arthur Flexer[1], Gerhard Widmer[1,2], Martin Gasser[1]**
[1]Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
[2]Department of Computational Perception, Johannes Kepler University, Linz, Austria
dominik.schnitzer@ofai.at, arthur.flexer@ofai.at,
gerhard.widmer@jku.at, martin.gasser@ofai.at

## ABSTRACT

Multivariate Gaussians are of special interest in the MIR field of automatic music recommendation. They are used as the de facto standard representation of music timbre to compute music similarity. However, standard algorithms for clustering and visualization are usually not designed to handle Gaussian distributions and their attached metrics (e.g. the Kullback-Leibler divergence). Hence to use these features the algorithms generally handle them indirectly by first mapping them to a vector space, for example by deriving a feature vector representation from a similarity matrix.

This paper uses the symmetrized Kullback-Leibler centroid of Gaussians to show how to avoid the vectorization detour for the Self Organizing Maps (SOM) data visualization algorithm. We propose an approach so that the algorithm can directly and naturally work on Gaussian music similarity features to compute maps of music collections. We show that by using our approach we can create SOMs which (1) better preserve the original similarity topology and (2) are far less complex to compute, as the often costly vectorization step is eliminated.

## 1. INTRODUCTION

Good content-based music recommendation systems are currently on the wish list of many music services since they help handling the massive audio databases which are currently emerging: be it simple music recommendations in the form of automatically generated playlists, advanced visualizations of the collections, or other new ideas for music discovery and listening.

One of the basic foundations of automatic content-based music recommendation systems is the ability to compute music similarity. In research it is not yet settled how to extract and represent good music similarity features that correspond well with the human perception of general music similarity. However, the currently best performing methods have one thing in common: A central component in most of the currently best working methods is a representation of timbre in terms of a multivariate Gaussian. Take for example, the top three algorithms in the MIREX [1] 2009 (Music Information Retrieval Exchange [4]) Automatic Music Recommendation evaluations. All used multivariate Gaussians and Kullback-Leibler-related divergences to describe and compare their similarity features. The basic idea of using a single multivariate Gaussian to model timbre similarity was first used by Mandel and Ellis [10]. In their case the Gaussian is computed over the Mel Frequency Cepstrum Coefficient representation [8] of the song.

We see the Gaussian representation of the music features as a very powerful way to describe the variability of the features in a song. With the Kullback-Leibler divergence and related divergence measures (Symmetrized Kullback-Leibler, Jensen-Shannon divergence) there also exist well founded ways to compute a distance/similarity value between the features (even though there are some problems related to the non-metricity of the divergences).

Things get interesting when we leave the path of simple feature representation and similarity computation. Standard algorithms for indexing, clustering or visualization are usually just not designed to work with Gaussian distributions and non-standard metrics like the one the Kullback-Leibler divergence induces. For instance, how would computing a simple average be performed with Gaussian features?

To work around that limitation the feature data is often artificially vectorized. In the domain of content based music recommendation techniques we have seen approaches computing the full distance matrix and using each row of the matrix as a feature vector [5, 14], or more venturesome ones reshaping the Gaussian covariance matrix and mean vector into a single long vector [11]. The first solution is expensive to compute the larger the music collection is, and the latter one, although fast, takes away the sense of using Gaussians.

In 2005 Banerjee et al. published an important paper in the machine learning literature where they show how to generalize the k-means clustering algorithm to the broad class of Bregman divergences [1]. This generalization practically opened all centroid-based algorithms to the

[1] http://www.music-ir.org/mirex/2009

wide range of Bregman divergences, which the Kullback-Leibler divergence is part of.

This paper builds on these findings and defines the weighted symmetrized Kullback-Leibler centroid to show how the Self Organizing Map (SOM) algorithm can work directly and naturally with Gaussians. The approach is able to create higher-quality two dimensional visualizations of music archives while retaining the nice scalability characteristics of the general SOM algorithm.

## 2. RELATED WORK

There already exists a wide range of publications dealing with visualizing acoustic music similarity features on SOMs. One of the first to do so were Rauber and Frühwirth [18] who use a very basic music similarity feature and a simple tabular grid which displays the clustered song titles on the map. This idea was extended by Pampalk et al. who use rhythmic similarity features and the Smoothed Data Histogram [16] (SDH) visualization to draw the SOM. Their visualization is inspired by geographical maps: blue regions (oceans) indicate areas onto which very few pieces of music are mapped, whereas clusters containing a larger quantity of pieces are colored in brown and white (mountains and snow). It was published under the name “Islands of Music” [15], which inspired the title of the presented paper.

‘Neptune’ [5] developed by Knees et al. improved Pampalk’s visualization by taking the two dimensional map into the third dimension. They add crawled meta-information and pictures from the web and allow a 3D walk through a music collection. They use a mix of rhythmic and timbre based similarity measures.

The ‘Globe of Music’ [7] by Leitich and Topf uses a GeoSOM [20] to map the music collection onto a globe for exploration. Lübbers et al. developed the ‘SoniXplorer’ [9] to navigate through music archives. They use a multimodal navigation model where the auralization of music supports the user on the SOM visualization of the music collection.

Mörchen et al. use the Emergent SOM algorithm to visualize and cluster music collections in their ‘Music Miner’ [12] system. For music similarity they use a large set of low-level features. In their paper they also point out that they cannot use Gaussian music similarity features as “they can not be used with datamining algorithms requiring the calculation of a centroid”. Solving that is the focus of the next sections of this paper.

## 3. PRELIMINARIES

To demonstrate how to use the SOM algorithm with Gaussian features we use the standard music similarity algorithm proposed by Mandel and Ellis [10]. This approach computes a single Gaussian music-timbre similarity feature. Similarity is computed with the symmetrized Kullback-Leibler divergence.

Since its publication this approach has been modified and improved in various ways, yet most stayed with the Gaussian feature representation. So everything presented here will of course work with the derived approaches too.

### 3.1 Music Features and Similarity Computation

To extract the timbre music similarity features we extract 25 Mel Cepstrum Frequency Coefficients [8] (MFCCs) for every $46ms$ of audio. This corresponds to a window size of $1024$ audio samples at $22.05kHz$. In this way a Gaussian timbre model $x$ finally consists of a 25-dimensional mean vector $\mu$, and a 25 × 25-dimensional covariance matrix $\Sigma$.

To compute the similarity between two Gaussians the Kullback-Leibler divergence ($kld$) can be used. There exists a closed form of this divergence [17] which allows the divergence to be computed between two $m$-dimensional Gaussians $x_{1,2} \sim \mathcal{N}(\mu, \Sigma)$.

$$\begin{aligned} 2\,kld(x_1||x_2) = \\ \log_e\left(\frac{\det\Sigma_2}{\det\Sigma_1}\right) + \operatorname{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + \\ (\mu_2-\mu_1)^\top\,\Sigma_1^{-1}(\mu_2-\mu_1) - m \end{aligned} \tag{1}$$

Since the $kld$ is asymmetric usually a symmetrized variant ($skld$) of the divergence is used for music similarity estimation:

$$skld(x_1||x_2) = \frac{kld(x_1||x_2) + kld(x_2||x_1)}{2} \tag{2}$$

### 3.2 Self Organizing Map (SOM) Algorithm

The SOM [6] is an unsupervised neural network that organizes multivariate data on a two dimensional map and is suited well for visualizations. It maps items which are similar in the original high-dimensional space onto locations close to each other on the map.

Basically the SOM consists of an ordered set of so-called *map units* $r_i$, each of which is assigned a *reference vector* (or *model vector*) $m_i$ in the feature space. The set of all reference vectors of a SOM is called its *codebook*. In the simplest case the codebook is initialized by a random strategy.

To compute a SOM, first the map dimensions and the number of training iterations ($t$) are fixed. Training is done in four basic repeating steps:

1. At iteration $t$ select a random vector $v(t)$ from the set of features.

2. Search for the best matching map unit $c$ on the SOM by computing the (Euclidean) distance of $v(t)$ to all $m_i$.

3. The codebook is updated by calculating a weighted centroid between $v(t)$ and the best matching unit $r_c$. Based on a neighborhood weighting function $h_{ci}(t)$ all map units participate in the adaptations depending on their distance on the two-dimensional outout

map. Equation 3 shows a standard Gaussian neighborhood function.

$$h_{ci}(t) = \alpha(t) \exp\left(-\frac{||r_c - r_i||}{2\alpha^2(t)}\right) \quad (3)$$

$$m_i(t+1) = m_i(t) + h_{ci}(t)\left[v(t) - m_i(t)\right] \quad (4)$$

4. The adaptation strength $\alpha(t)$ is decreased gradually with the iteration cycle $t$. This supports the formation of large clusters in the beginning and a finetuning toward the end of the training.

Usually, the iterative training is continued until a convergence criterion is fulfilled or a preselected number of training iterations is finished. In the final step all items are assigned to the map unit they are most similar to.

A popular way of visualizing SOMs trained with music similarity features is the Smoothed Data Histogram (SDH) [16].

## 4. FROM VECTORS TO GAUSSIAN DISTRIBUTIONS

Although originally SOMs were defined for Euclidean feature vectors only, the algorithm per se is not limited to the vector space. Kohonen himself mentions this in the most recent edition of his standard work on Self-Organizing Maps [6]. This observation will be the basis for extending the SOM algorithm to the 'distribution space'.

A closer look at the SOM algorithm sketched in the last section, shows that there is only a single step where the algorithm in fact depends on vectors. It is the computation of the weighted centroid in Equation 4 [2]. We now rewrite Equation 4 so that it is more obvious that a centroid (a weighted mean of several vectors) is computed:

$$m_i(t+1) = (1 - h_{ci}(t))\, m_i(t) + h_{ci}(t) v(t) \quad (5)$$

The essence of this is if a weighted centroid can be computed for Gaussians and the symmetrized Kullback-Leibler divergence, the SOM algorithm would, without modifications, work for our data.

### 4.1 Weighted Symmetrized Kullback-Leibler Centroid

The Kullback-Leibler divergence is part of the broad family of Bregman divergences [3]. In 2005 Banerjee et al. showed that a unique centroid exists for any Bregman divergence and proved that the standard k-means (and with that basically any centroid-based) works in this rich family of divergences [1].

As Bregman divergences are asymmetric divergences, there exist three uniquely defined centroids for a divergence $D$ and a set of points $x_i$: the left-sided centroid $c_L$, right-sided centroid $c_R$ and symmetrized centroid $c_S$. The centroids are the optimizers of the minimum average distance:

$$\boldsymbol{c_L} = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} D(\boldsymbol{c}||x_i) \quad (6)$$

$$\boldsymbol{c_R} = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} D(x_i||\boldsymbol{c}) \quad (7)$$

$$\boldsymbol{c_S} = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \frac{D(x_i||\boldsymbol{c}) + D(\boldsymbol{c}||x_i)}{2} \quad (8)$$

As Nielsen and Nock show [13], no closed analytical form to compute the symmetrized centroid exists. In their paper they present an efficient geodesic walk algorithm to find the symmetrized Bregman centroid $c$ using the left $c_L$ and right $c_R$ centroids. In the last section they also define the three centroids for the Kullback-Leibler divergence and $x_i \sim \mathcal{N}(\mu_{x_i}, \Sigma_{x_i})$ multivariate Gaussians.

To use these centroids in the SOM algorithm we need to modify them and add a weighing term $\lambda_i$ (with $\sum_{i=1}^{n} \lambda_i = 1$) for each Gaussian. The individual weighted centroid definitions are given in the next paragraphs [3].

#### *4.1.1 Weighted Right-Sided Gaussian $kld$-Centroid*

The right-type $kld$-centroid Gaussian $c_R \sim \mathcal{N}(\mu_{c_R}, \Sigma_{c_R})$ coincides with the center of mass. For Gaussians $x_i$ and the $kld$ it is defined as:

$$\mu_{c_R} = \sum_{i=1}^{n} \lambda_i \mu_i \quad (9)$$

$$\Sigma_{c_R} = \sum_{i=1}^{n} \lambda_i \left(\mu_i \times \mu_i^T + \Sigma_i\right) - \mu_{c_R} \times \mu_{c_R}^T \quad (10)$$

with $\lambda_i$ as defined above.

#### *4.1.2 Weighted Left-Sided Gaussian $kld$-Centroid*

The left-type $kld$-centroid Gaussian $c_L \sim \mathcal{N}(\mu_{c_L}, \Sigma_{c_L})$ is obtained equivalently by minimizing the dual right-type centroid problem. For Gaussians $x_i$ and the $kld$ it is defined as:

$$\mu_{c_L} = \Sigma_{c_L} \times \sum_{i=1}^{n} \lambda_i (\Sigma_i^{-1} \times \mu_i) \quad (11)$$

$$\Sigma_{c_L} = \left(\sum_{i=1}^{n} \lambda_i \Sigma_i^{-1}\right)^{-1} \quad (12)$$

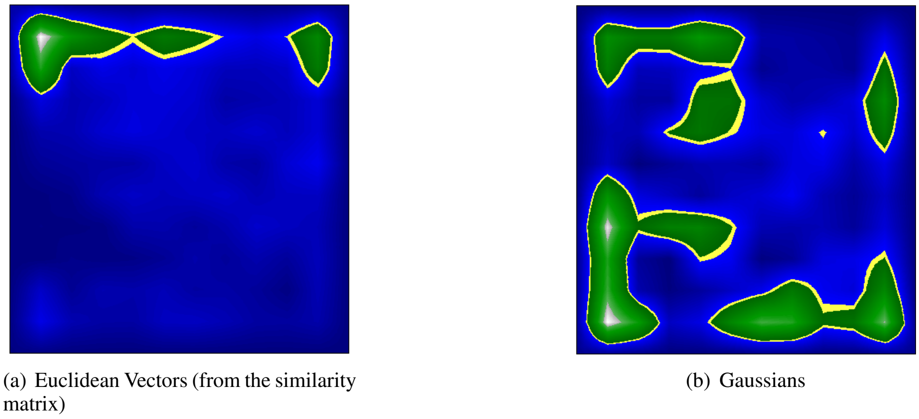
with $\lambda_i$ as defined above.

#### *4.1.3 Weighted (mid-point) Gaussian $skld$-Centroid*

To compute the weighted symmetrized Kullback-Leibler centroid, the weighted left and right centroids need to be computed. We then use the mid-point centroid approximation instead of computing the exact centroid. The midpoint empirically proved to be a good approximation of the true centroid of the $skld$ [19]. The approximation

[2] The distance $||r_c - r_i||$ in Equation 3 is computed in 'map space' and does not need to be modified.

[3] A detailed listing and explaination of the derivation of each centroid can not be given due to the length constraints of this paper.



(a) Euclidean Vectors (from the similarity matrix)

(b) Gaussians

**Figure 1**. A 10 × 10 SOM, computed with the two different approaches. The SOMs are visualized using the Matlab Smoothed Data Histogram Toolbox [16] clustering a collection of 1 000 music pieces. They were created using identical parameters for initialization and learning.

$c_S \sim \mathcal{N}(\mu_{c_S}, \Sigma_{c_S})$ merges the weighted left and right centroids in one step:

$$\mu_{c_S} = \frac{1}{2}\left(\mu_{c_L} + \mu_{c_R}\right) \tag{13}$$

$$\Sigma_{c_S} = \frac{1}{2}\sum_{i=\{L,R\}}\left(\mu_{c_i} \times \mu_{c_i}^T + \Sigma_{c_i}\right) - \mu_{c_S} \times \mu_{c_S}^T \tag{14}$$

### 4.2 The Generalized SOM

With the definition of the weighted *skld*-centroid everything is in place to use the SOM algorithm with Gaussian music-timbre models and the *skld*:

1. Initialization of the SOM and its $m_i$ is done by selecting random Gaussians from the music-timbre models.

2. Most importantly the iterative computation of the weighted centroid during the training of the codebook can now be replaced with the weighted symmetrized *kld* centroid.

3. The learning rate adaptation and neighborhood functions do not need to be changed. They are not dependent on the features.

4. In the final step the Gaussians are assigned to the nearest map units according to *skld*.

The approach of using the SOM directly with the Gaussian features is very close to the data and the original intention of the algorithm. We evaluate the generalized SOM in the next section.

## 5. EVALUATION

To compare SOMs generated with different approaches, we quantify how well the original neighborhood topology is preserved in a SOM mapping. As we need to compare SOMs using different metrics it is not possible to use standard SOM quality measures like the quantization error. Therefore we are using a rank distance. We search for the $n$ nearest neighbors of every item $x_i$ in the original space and check their location on the SOM. Ideally the nearest neighbors should also be mapped close to each other on the SOM. For a given number $n$ of nearest neighbors and a Gaussian $x_i$ this will be measured as the $n$ *nearest neighbor rank distance*:

1. Assign all Gaussians to their corresponding map unit on the SOM.

2. For Gaussian $x_i$ compute the Euclidean distance of its assigned map unit on the SOM to the map units assigned to all other Gaussians.

3. Sort the list of Euclidean distances in ascending order and transform it into a list of ranks.

4. Find the $n$ nearest neighbors of $x_i$ in the original space and average across their corresponding ranks.

The *average* $n$ *nearest neighbor rank distance* of all $x_i$ is a value describing the whole SOM. The lower its value, the better the preservation of the neighborhood on the SOM.

### 5.1 Setup

To test how the SOM algorithm performs operating directly on the Gaussians we used a test collection of 16 754 songs. The songs are typical full three to five minutes songs of a mixed genre music collection. We compute the Gaussian timbre music similarity features for these songs (see Section 3.1) so that every song is characterized by a 25-dimensional Gaussian.

To compare the quality of the SOMs generated with our approach, we also generate SOMs with vectorized features. For each Gaussian feature we build a vector by computing the distance to all other features and normalizing this distance vector to zero mean and unit variance. This is equivalent to computing the full similarity matrix and using each row as a feature vector (done e.g. in [5, 14]).

As a baseline for our experiments we also use a randomly initialized SOM without any training. In our experiments we vary various SOM parameters to test different

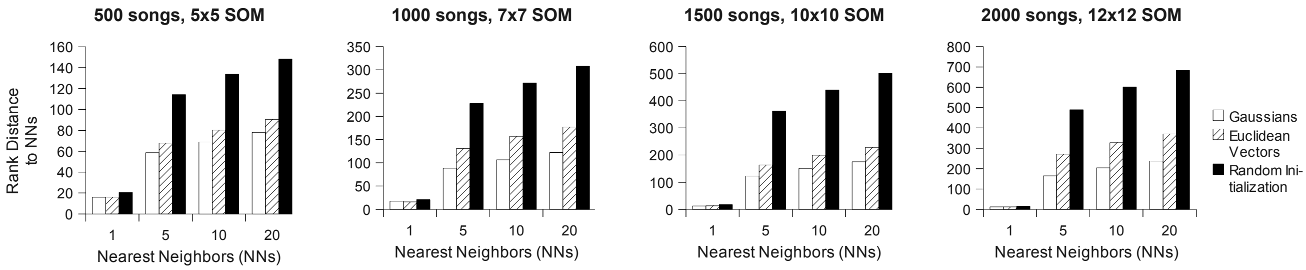


**Figure 2**. Four plots of the average $n$-nearest neighbor rank distance using different SOM configurations. Each plot compares the two strategies (Gaussian/Vectorized) to compute a SOM. As a baseline a randomly initialized SOM (black bar) is added. Lower rank distance values indicate that the original nearest neighbors (NNs) are mapped closer to each other on the SOM. The plots clearly shows that using Gaussians directly produces SOMs better preserving the neighborhood.

configurations: (1) we used SOM grid sizes of $5 \times 5$, $7 \times 7$, $10 \times 10$, $12 \times 12$, (2) and mapped 500, 1 000, 1 500 or 2 000 songs (randomly drawn from the base collection).

To ensure a fair evaluation we took the following precautions:

- In each run the same random seed is used for the random, vectorized and Gaussian SOM. This ensures identical random initialization and use of the same randomly chosen features during the training phase.
- The previously defined average $n$-nearest neighbor rank distance is computed for each map ($n = 1, 5, 10, 20$).
- Each unique experiment configuration is repeated ten times. Results are averaged.

### 5.2 Results

Figure 2 shows the average $n$-nearest neighbor rank distance for four selected SOM configurations which have been evaluated. In these experiments it can be seen that directly using Gaussians to train a SOM results in maps which are able to better preserve the neighborhood.

The results of all experiments conducted are summarized in Table 1. The table expresses the improvement of SOMs created with the Gaussian and vectorized approach relative to the randomly initialized SOMs. It confirms the results of the previous figure that throughout the configurations SOMs computed directly with the Gaussians produce higher-quality mappings and that this method should be preferred.

An illustration comparing two SOMs created with the two approaches is plotted in Figure 1. Albeit we can not make any judgements concerning the quality of the SOMs from the plots, we can see that a more structured SOM emerged from directly using the Gaussian features.

Besides producing higher-quality SOMs, we emphasize that this approach is also far less complex to compute than a variant working with vectorized features: (1) it is almost impossible to compute the full similarity matrix on a large collection of songs (i.e. over 100 000 songs) and (2) a SOM with 100 000-dimensional (or larger) vectors would be very expensive to compute. By using random projections [2] one can overcome that, but that would probably come with a loss of mapping quality. A SOM computed directly with the Gaussians, on the other hand, requires only a fraction of the computational effort, as the full similarity matrix does not need to be computed and the original features are used as intended.

| | | *Nearest Neighbors* | | | |
|---|---|---|---|---|---|
| *Features* | *Type* | 1 | 5 | 10 | 20 |
| 500 | Gaussians | **0.90** | **0.41** | **0.41** | **0.43** |
| | Vectors | 0.97 | .55 | 0.56 | 0.54 |
| 1000 | Gaussians | **0.80** | **0.40** | **0.40** | **0.41** |
| | Vectors | 1.07 | 0.64 | 0.63 | 0.62 |
| 1500 | Gaussians | **0.75** | **0.38** | **0.38** | **0.40** |
| | Vectors | 0.75 | 0.60 | 0.60 | 0.60 |
| 2000 | Gaussians | **0.76** | **0.41** | **0.42** | **0.43** |
| | Vectors | 0.77 | 0.70 | 0.70 | 0.69 |

**Table 1**. The table shows the average $n$-nearest neighbor rank distance of a SOM in relation to a randomly initialized one. Lower ratios indicate a better neighborhood topology preservation. It can be seen that in each configuration the Gaussian approach produces better mappings.

## 6. DISCUSSION & FUTURE WORK

We have shown how to compute a weighted symmetrized Kullback-Leibler centroid on multivariate Gaussians and on top of that how to directly and naturally compute a SOM with Gaussian music similarity features. The SOMs computed with that approach are shown to produce better mappings and omit the so far necessary step of vectorizing the data to compute a SOM.

The approach easily fits into the large number of existing SOM visualizations using Gaussian music similarity features together with a Kullback-Leibler related divergence and is of course not limited to music similarity. By using it the quality of the produced SOM should increase and the application can scale to larger collections of features.

Besides computing SOMs we also gave a clear definition of how to compute the weighted symmetrized Kullback-Leibler centroid so that it can be re-used to solve different problems where the features are also parametric Gaussians: for example to do a k-means cluster analysis in music collections directly with the Gaussian features. Maybe the centroid could also be of use to build an indexing algorithm for faster nearest neighbor retrieval.

## ACKNOWLEDGMENTS

This research is supported by the Austrian Research Fund (FWF) under grants L511-N15, P21247 and the Vienna Science and Technology Fund (WWTF). The Austrian Research Institute for Artificial Intelligence (OFAI) acknowledges financial support form the Austrian Federal Ministries BMVIT and BMWF.

## 7. REFERENCES

[1] A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

[2] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM New York, NY, USA, 2001.

[3] L.M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[4] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[5] P. Knees, M. Schedl, T. Pohle, and G. Widmer. Exploring music collections in virtual landscapes. *IEEE multimedia*, 14(3):46–54, 2007.

[6] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[7] S. Leitich and M. Topf. Globe of music: Music library visualization using geosom. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR07), Vienna, Austria*, 2007.

[8] B. Logan. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*, 2000.

[9] D. Lübbers. SoniXplorer: Combining visualization and auralization for content-based exploration of music collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR05)*, 2005.

[10] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), London, UK*, 2005.

[11] M. Mandel and D. Ellis. Labrosas audio music similarity and classification submissions. *Music Information Retrieval Information Exchange (MIREX)*, 2007.

[12] F. Mörchen, A. Ultsch, M. Nöcker, and C. Stamm. Databionic visualization of music collections according to perceptual distance. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR), London, UK*, 2005.

[13] F. Nielsen and R. Nock. Sided and symmetrized Bregman centroids. *IEEE Transactions on Information Theory*, 55(6):2048–2059, 2009.

[14] E. Pampalk. Computational models of music similarity and their application in music information retrieval. *Docteral dissertation, Vienna University of Technology, Austria*, 2006.

[15] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the 10th ACM international conference on Multimedia*, page 579. ACM, 2002.

[16] E. Pampalk, A. Rauber, and D. Merkl. Using smoothed data histograms for cluster visualization in self-organizing maps. *Lecture notes in computer science*, pages 871–876, 2002.

[17] W.D. Penny. Kullback-Leibler divergences of normal, gamma, Dirichlet and Wishart densities. *Wellcome Department of Cognitive Neurology*, 2001.

[18] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. *Lecture Notes in Computer Science*, pages 402–414, 2001.

[19] R. Veldhuis. The centroid of the symmetrical Kullback-Leibler distance. *IEEE signal processing letters*, 9(3):96–99, 2002.

[20] Y. Wu and M. Takatsuka. Spherical self-organizing map using efficient indexed geodesic data structure. *Neural Networks*, 19(6-7):900–910, 2006.