Adaptive Distance Measures for Exploration and Structuring of Music Collections

Sebastian Stober[1]

[1] *Data & Knowledge Engineering Group, Faculty of Computer Science, Otto-von-Guericke-University Magdeburg, Universitaetsplatz 2, D-39106 Magdeburg, Germany*

Correspondence should be addressed to Sebastian Stober (stober@ovgu.de)

ABSTRACT

Music similarity plays an important role in many Music Information Retrieval applications. However, it has many facets and its perception is highly subjective – very much depending on a person's background or retrieval goal. This paper presents a generalized approach to modeling and learning individual distance measures for comparing music pieces based on multiple facets that can be weighted. The learning process is described as an optimization problem guided by generic distance constraints. Three application scenarios with different objectives exemplify how the proposed method can be employed in various contexts by deriving distance constraints either from domain-specific expert information or user actions in an interactive setting.

1. INTRODUCTION

One of the big challenges of computer science in the 21st century is the digital media explosion. Steadily growing hard-drives are filled with personal media collections comprising, e.g., music, photos and videos. With increasing collection size maintenance becomes a more and more tedious task, but without manual organization effort it gets harder to access specific pieces of media or even to keep an overview. Typically, a large portion of the digital content is just "collecting dust" because the user has simply forgotten about it. Automatic structuring is one means to ease access to media collections, be it for organization or for exploration. Such a structuring should reflect an individual user's personal preferences and needs in order to be intuitively understandable. Here, dealing with music in particular poses some additional challenges: Firstly, music can be described by a large variety of facets comprising, e.g., simple tags (artist, title etc.), content-based features ranging from simple loudness to complex timbre descriptions, tonality, meters and tempi, instrumentation, and lyrics but also information about the production and publishing process as well as the general reception in the public expressed in reviews or chart positions. Secondly, perception of music is highly subjective and may depend on a person's background. A musician, for instance, might especially look after structures, tonality or instrumentation (possibly paying – conscious- or unconsciously – special attention to his own instrument). Non-musicians will perhaps focus more on overall timbre or general mood. Others, in turn, may have a high interest in the lyrics as long as they are able to understand the particular language.

Given these considerations, the general idea of the approaches covered in this papers is as follows: Starting from various features that describe music as mentioned above, a complex multi-facet distance (or dissimilarity) measure is constructed where a distance facet is computed either on a single feature or a combination of features. Weighting each facet allows for adaptability. A weighting scheme then represents a user's preference for grouping similar songs together and can be applied for any similarity-based structuring approach. A user could explicitly adjust the facet weighting to fit his needs – but this is most likely a very difficult thing to do and some users might not even be aware of their own preferences. Alternatively, a weighting can also be learned by watching the user interact with the collection.

This paper brings together work in this field from the recent years [1, 17, 19]. introducing a generalized view with a unified model of the different adaptation approaches taken so far based on generic distance constraints (Section 2). This model is applied in three different scenarios with varying objectives (Sections 3–5). For each application, the problem specific modeling of the learning problem is described together with the essential findings from experiments.

2. GENERAL ADAPTATION APPROACH

2.1. Formalization

To begin with, the concept of facet distances needs to be formalized assuming a feature-based representation of the objects of interest (which are generally music pieces in the scope of this paper):

Definition Given a set of features F, let S be the space determined by the feature values for a set of objects O. A *facet* f is defined by a *facet distance measure* δ_f on a subspace $S_f \subseteq S$ of the feature space, where δ_f satisfies the following conditions for any $a,b \in O$:

- $\delta_f(a,b) \geq 0$ and $\delta_f(a,b) = 0$ if and only if $a = b$
- $\delta_f(a,b) = \delta_f(b,a)$ (symmetry)

Furthermore, δ_f is a distance metric if it additionally obeys the triangle inequality for any $a,b,c \in O$:

- $\delta_f(a,c) \leq \delta(a,b) + \delta(b,c)$ (triangle inequality)

In order to avoid a bias when aggregating several facet distance measures, the values need to be normalized. The following normalization is applied for all distance values $\delta_f(a,b)$ of a facet f:

$$\delta'_f(a,b) = \frac{\delta_f(a,b)}{\mu_f} \tag{1}$$

where μ_f is the mean facet distance with respect to f:

$$\mu_f = \frac{1}{|\{(a,b) \in O^2\}|} \sum_{(a,b)\in O^2} \delta_f(a,b) \tag{2}$$

As a result, all facet distances have a mean value of 1.0. The actual distance between objects $a,b \in O$ w.r.t. the facets $f_1,\ldots,f_l$ is computed as weighted sum of the individual facet distances $\delta_{f_1}(a,b),\ldots,\delta_{f_l}(a,b)$:

$$d(a,b) = \sum_{i=1}^{l} w_i \delta_{f_i}(a,b) \tag{3}$$

This way, facet weights $w_1,\ldots,w_l \in \Re$ are introduced that allow to adapt the importance of each facet according to user preferences or for a specific retrieval task. These weights obviously have to be non-negative and should also have an upper bound, thus:

$$w_i \geq 0 \qquad \forall 1 \leq i \leq l \tag{4}$$

$$\sum_{i=1}^{l} w_i = l \tag{5}$$

They can either be specified manually or learned from preference information. In the scope of this work, all preference information is reduced to *relative distance constraints*. A distance constraint (s,a,b) demands that object a is closer to the seed object s than object b, i.e.:

$$d(s,a) < d(s,b) \tag{6}$$

With Equation 3 this can be rewritten as:

$$\sum_{i=1}^{l} w_i(\delta_{f_i}(s,b) - \delta_{f_i}(s,a)) = \sum_{i=1}^{l} w_i x_i > 0 \tag{7}$$

substituting $x_i = \delta_{f_i}(s,b) - \delta_{f_i}(s,a)$. Such basic constraints can directly be used to guide an optimization algorithm that aims to identify weights that violate as few constraints as possible [16]. Alternatively, the positive examples $(x,+1)$ and the negative examples $(-x,-1)$ can be used to train a binary classifier in which case the weights $w_1,\ldots,w_l$ define the model (separating hyperplane) of the classification as pointed out in [3]. At the same time, the relative distance constraints are still rich enough to cover more complex forms of preference as shown in the following sections. Also note that such relative statements are usually much easier to formulate than absolute ones.

2.2. Optimization Approaches

It is possible to look at the optimization problem from different perspectives: tolerance w.r.t. constraint inconsistencies, stability, continuity and responsiveness.

Inconsistency-tolerance: In some scenarios, it might happen that constraints or combinations thereof contradict each other. Reasons for this can be manifold. E.g., the collected preferences might stem from various users or reflect the preference of a single user at different points in time. It is also possible, that the user's comparison of the objects is based on features that are not covered by the distance facets. In any such case where inconsistencies occur, no weighting can be learned that satisfies all constraints. One way to deal with this problem is to detect and remove inconsistencies – e.g., by building a directed constraint graph, checking for cycles and removing appropriate edges as described in [10]. Alternatively, an optimizer may use soft constraints as opposed to hard constraints that have to be satisfied.

Stability: If there is a weighting that satisfies all constraints, it is most likely not a unique solution. I.e., for

the binary classification multiple hyperplanes might exist that separate the positive from the negative training examples. Choosing the maximum margin separating hyperplane – i.e. the one with the largest distance to the training examples – reduces the likelihood that small variances in the data lead to a different outcome.

Continuity: In an incremental learning setting where constraints are added over time, a gradual change of the weights might be desired. Here, the weighting with the smallest difference to the previous values is preferred rather than the one that results in the maximum margin.

Responsiveness: In interactive scenarios, the response time of the user-interface plays an important role. Here, the learning algorithm might need to guarantee that an acceptable solution is computed in restricted time bounds. So called anytime algorithms are especially suited for such task as they are able to return an approximate answer with the quality depending on the time available for computation.

In the work presented here, three different weight learning algorithms have been used:

2.2.1. Gradient Descent

One way of learning weights is to apply a gradient descent approach similar to the work described in [2]. During learning, all constraint triples (s,a,b) are presented to the algorithm several times until convergence is reached. If a constraint is violated by the current distance measure, the weighting is updated by trying to maximize

$$obj(s,a,b) = \sum_{i=1}^{l} w_i(\delta_{f_i}(s,b) - \delta_{f_i}(s,a)) \tag{8}$$

which can be directly derived from Equation 7. This leads to the update rule for the individual weights:

$$w_i = w_i + \eta \Delta w_i, \quad \text{with} \tag{9}$$

$$\Delta w_i = \frac{\partial obj(s,a,b)}{\partial w_i} = \delta_{f_i}(s,b) - \delta_{f_i}(s,a) \tag{10}$$

where the learning rate η defines the step width of each iteration.[1] To enforce the bounds on w_i given by Equations 4 and 5, an additional step is necessary after the update, in which all negative weights are set to 0 and then the weights are normalized to sum up to l.

[1] Interestingly, approaching the weight learning problem from the classification perspective using a perceptron for classification as described in [3] eventually leads to the same update rule.

This algorithm can compute a weighting, even if not all constraint can be satisfied due to inconsistencies. However, no largest margin is enforced. Using the current weights as initial values in combination with a small learning rate allows for some continuity but there may still be solutions with less change required. It is possible to limit the number of iteration to increase responsiveness but this may result in some unsatisfied constraints.

2.2.2. Quadratic Optimization

For maximum continuity which is considered most important in the application described in Section 4, the weights should change only as little as necessary to satisfy all constraints. This can directly be modeled as a quadratic optimization problem demanding in the objective function that the sum over all (quadratic) deviations of the weights from their previous values should be minimal (with initial values 1):

$$\min_{(w_1,\ldots,w_l)\in\Re^l} \sum_{i=1}^{l} \left(w_i - w_i^{(old)}\right)^2 \tag{11}$$

subject to the constraints that enforce the weight bounds (Equations 4 and 5) and the distance constraints (Equation 7) which can be used directly. The problem can be solved using the Goldfarb and Idnani dual quadratic programming (QP) algorithm for convex QP problems subject to general linear equality/inequality constraints [5]. For this formalization of the weight learning problem, there is only a solution if all constraints are consistent. An additional introduction of slack variables would allow the violation of constraints. Note that in this case, the slack for the weight constraints has to be zero to avoid a violation of the weight bounds.

2.2.3. Maximal Margin Classifier

If stability is more important than continuity, the primary objective is to maximize the margin between the separating hyperplane and the positive and negative training samples (generated from the distance constraints as described in Equation 7). For the application described in Section 5, the linear support vector machine algorithm as provided by *LIBLINEAR* [4] is used. However, with this approach, a valid value range for the weights cannot be enforced. Specifically, weights can become negative. To reduce the chance of negative weights, artificial training examples are added that require positive weights (setting a single x_i to 1 at a time and the others to 0). These constraints may still be violated in favor of a larger margin or in case of general constraint inconsistencies.

3. APPLICATION I: FOLK SONG ANALYSIS

Folk song researchers detect and document relations between folk songs and their performances in order to understand oral transmission. At the *Meertens Institute*, they classify folk song variants into so called *melody norms*. This traditional classification captures both, aspects of musical similarity and historical relationships. There is only one class per tune and each class is represented by a *reference tune*. The *WITCHCRAFT* project supports the musicologists with a system for browsing tunes by musical content (in a symbolic representation). Given a query tune, the system ranks the tunes in the database according to a chosen similarity measure. Such a *tune ranking list* can be filtered by tune classification. This results in a *class ranking list* that contains for each melody norm only the most similar tune of that class. The latter is especially helpful when new tunes need to be classified. Several distance and similarity measures can be chosen for generating the ranked lists. However, a particular musicological way of classifying songs is usually not directly reflected by just a single one of these measures. Therefore, the multi-facet approach introduced in the previous section suggests itself for modeling tune similarity based on a range of basic distance and similarity measures. A detailed description of this application has been published in [1].

3.1. Modeling the Learning Problem

The challenge is to find an optimal facet weighting that produces tune and class ranking lists with tunes belonging to the same melody norm ranked first. Two scenarios are considered: Retrieving similar tunes for already classified tunes and querying with unclassified tunes.

Given a *classified* query tune q, it is known from the expert classification, which other tunes t_r belong to the same (relevant) class and which tunes t_i are irrelevant. As tunes of the same class should be ranked first and thus should be more similar, distance constraints (q, t_r, t_i) can directly be derived. Depending on the set of queries (and the resulting set of constraints) used, weighting for different levels of specificity can be learned:

- *individual tune weightings* $\mathbf{w}^{tune}$ – for a single tune (most specific and thus with the risk to overfit),
- *class weightings* $\mathbf{w}^{class}$ – for the set of tunes belonging to the same class (more general), and an
- *overall weighting* $\mathbf{w}^{all}$ – for all tunes (most general).

In case of an *unclassified* query tune without a specific individual- or class weighting, either the overall weighting has to be used or a case-based approach is taken to select a weighting in the following two steps – assuming that similar tunes have similar optimal weightings:

1. Find the closest (classified) tune t_{best}.
   (Here, the overall weighting is applied or the specific individual- or class weighting of each tune.[2])
2. Use the individual or class weighting of t_{best} to compute the ranking for the query.

In the following, $\mathbf{w}^b \circ \mathbf{w}^a$ denotes that $\mathbf{w}^a$ is used for case selection (step 1) and $\mathbf{w}^b$ for ranking (step 2).

3.2. Experiments

A dataset was provided that comprises 360 tunes – all well understood single melodic strophes – and their classification into one of 26 melody norms. Instead of the actual tunes, the dataset contains only the pairwise facet distances / similarities. 14 measures are considered of which 11 are taken from the *Simile* package.[3] These are *rawEd*, *diffEd*, *nGrSumCo*, *nGrUkkon*, *harmCorE*, *rhytFuzz*, *rhytGaus*, *opti1*, *opti3*, *accents_opti1* and *accents_opti2*. Note that the last four measures are themselves linear combinations of basic similarity measures in the sense of *Simile*. Two distance measures are based on the spectra of Laplacian and adjacency graphs [13] and one is an unpublished pitch sequence edit distance, implemented at *Meertens Institute*.

3.2.1. Querying with Classified Tunes

Figure 1 (left) contains the precision/recall curves for the three learned weightings and the two best basic measures, *rawEd* and *opti*1. Table 1 (top) shows the corresponding evaluation of the class ranking lists, ordered by best performance w.r.t. *average rank* of correct class (smaller is better), *average inverse rank* (larger is better), and classification *precision at 1st rank*. Not surprisingly, higher specificity leads to better performance in this scenario but is also more vulnerable to overfitting. The overall weight performs worse than the best basic similarity measure (*rawEd*) in most precision/recall regions (although only slightly) but *rawEd* produces the worst class

[2] Note that in this case different weightings are used to compute the distance to different database tunes, which leads to local distortions of the distance space around each case. While such a locally distorted metric is unsuitable for the computation of the entire ranking, it may still be useful to retrieve only t_{best}.

[3] http://doc.gold.ac.uk/isms/mmm/SIMILE_algo_docs_0.3.pdf

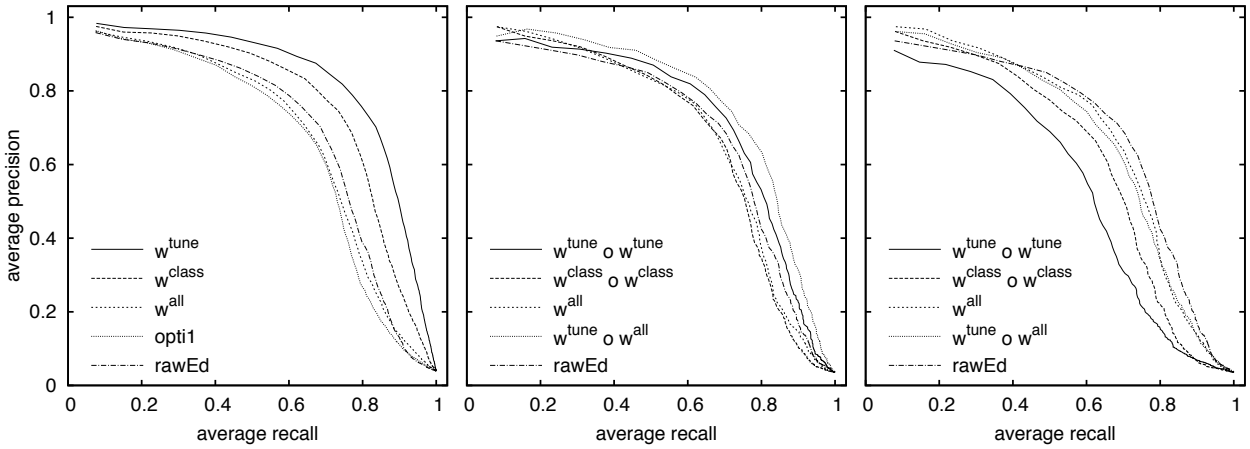

Fig. 1: Precision/Recall plots for tune-ranking-lists. **Left:** classified tunes. **Middle:** unclassified tunes of a known class. **Right:** unclassified tunes of an unknown class.

ranking lists. All this indicates that there might not be a single perfect overall measure that can be used in general but rather data/problem specific measures instead.

3.2.2. Querying with Unclassifed Tunes

In this more realistic scenario, two situations are possible: In the first case (Figure 1; middle), all but the query tune are used for learning – including other tunes of the same class. But of course, the system does not know during ranking which ones these are. In the other case (Figure 1; right) none of the tunes from the query tune's class is used for learning, simulating an entirely new class that shall be added to the database. This is of course a much harder case. For computation of the precision and recall values, *all* tunes were ranked according to the query tune, including the songs of the unknown class.[4]

Comparing the plots for both cases and taking the *rawEd* measure as baseline for reference shows that the performance of the learned measures is lower for the harder case – most notably for $\mathbf{w}^{tune} \circ \mathbf{w}^{tune}$. This implies that the case-based weight selection approach successfully chooses members of the same tune family if such are already in the database and otherwise resorts to less fitting weightings from other tune families. The overall weighting $\mathbf{w}^{all}$ turns out as most suitable for selecting the tune whose weighting is then used for ranking and even as best choice for the hardest case in general. For the very specific weightings there seems to be too much overfitting which makes them less applicable for other tunes. The baseline *rawEd* is better at the end of the ranking, while it is worse at the beginning. This is also reflected in the class ranking lists evaluation (Table 1) where *rawEd* performs worst. For automatic classification, $\mathbf{w}^{all}$ produces the fewest errors at the first rank.

[4] This methodology of simulating new tunes is very time-consuming as for each query the learning has to be redone without the respective information. Thus, only the reference melody and two random tunes are considered for each class resulting in 78 out of 360 queries.

Table 1: Evaluation of the class-ranking-lists. **Top:** classified tunes. **Middle:** unclassified tunes of a known class. **Bottom:** unclassified tunes of an unknown class.

measure	average rank	avg. inv. rank	precision @ 1
$\mathbf{w}^{tune}$	1.042	0.989	0.983
$\mathbf{w}^{class}$	1.083	0.985	0.975
opti1	1.169	0.975	0.961
$\mathbf{w}^{all}$	1.172	0.974	0.961
rawEd	1.233	0.967	0.956
$\mathbf{w}^{tune} \circ \mathbf{w}^{all}$	1.218	0.969	0.949
$\mathbf{w}^{all}$	1.231	0.981	0.973
$\mathbf{w}^{tune} \circ \mathbf{w}^{tune}$	1.244	0.957	0.936
$\mathbf{w}^{class} \circ \mathbf{w}^{class}$	1.346	0.976	0.973
rawEd	1.410	0.946	0.936
$\mathbf{w}^{all}$	1.218	0.982	0.973
$\mathbf{w}^{tune} \circ \mathbf{w}^{all}$	1.244	0.971	0.961
$\mathbf{w}^{tune} \circ \mathbf{w}^{tune}$	1.282	0.942	0.910
$\mathbf{w}^{class} \circ \mathbf{w}^{class}$	1.359	0.970	0.961
rawEd	1.410	0.946	0.936

4. APPLICATION II: BEATLESEXPLORER

With the *BeatlesExplorer*, a first user-interface prototype for adaptive structuring and exploration of music collections has been developed and presented in [17]. The system uses a dataset containing 282 songs of The Beatles and a multi-facet distance measure with about 20 facets covering sound, harmonics, lyrics and information about the production process. The feature information is extracted (semi-automatically) from Wikipedia[5], LyricWiki[6], Alan W. Pollack's notes on The Beatles[7], manual chord annotations [6] and from the audio recordings using the frameworks CoMIRVA [14] and JAudio [9], A detailed description is given in [17].

Using initially uniform facet weights, a growing self-organizing map (SOM) is induced as described in [11] clustering similar songs of the music collection into hexagonal cluster cells. The result is a two-dimensional topology that preserves the neighborhood relations of the originally high dimensional feature space, i.e. not only songs within the same cluster cell are similar to each other but also songs of cells in the neighborhood are expected to be more similar than those in more distant cells. Using a growing approach as opposed to the common static approaches as, e.g., [12, 7]. ensures that only as many cells are created as are actually needed. Further, the approach is incremental: Songs may be added to the collection without having to relearn the whole map from scratch – instead, it is only extended. A screenshot of the prototype user interface is shown in Figure 2.[8] The cells of the generated hexagonal grid act as "virtual folders", each one containing a set of similar songs. Cells are labeled with the album cover(s) that are most frequently linked with the songs contained in the cell. Clicking on a cell opens a window that displays its content. For each song, the title and (if available) the album covers are listed.

4.1. Vectorization

SOMs belong to the category of prototype-based clustering approaches, i.e. each cluster is represented by a prototype. They require that the objects to be clustered and the cluster prototypes are vectors, i.e. elements of a vector space. The multi-facet feature representation used here does not adhere to this rather severe restriction and thus, some means of vectorization is required. An established vectorization approach that can be considered as common practice in related work (e.g., [12, 7], is to simply interpret each row of the distance matrix (containing the pairwise distances of all objects in the dataset) as a feature vector. I.e. the objects are described by the distances amongst each other. This approach has also been used originally in the BeatlesExplorer. However, recent investigations of the impact of this transformation on the distance relations [20] suggest an alternative approach based on Multidimensional Scaling (MDS) [8] that is especially suited for multi-facet representations like the one at hand.

Fig. 2: Screenshot of the *BeatlesExplorer* with the grid on the left (colored according to the number of songs in each cell) and two cell content windows on the right.

4.2. Modeling the Learning Problem

The initial clustering is done in a unsupervised manner. If the user does not agree on the cluster assignment for a specific song, he may change its location on the map by simple drag-and-drop actions. Furthermore, he can query the system for the ten most similar songs given a seed song and change the order of the resulting list.[9] Any of these actions, moving a song to another cell or modifying a ranking, results immediately in an adaptation of the distance measure and ultimately in a reassignment of all songs to the grid. This way, single manual actions may cause several automatic changes. While the user is interacting with the system in an iterative process of user action and resulting adaptation, the facet weights more and more converges towards to the (possibly only subconsciously existing) preferences of the user.

[5] http://en.wikipedia.org

[6] http://lyricwiki.org

[7] http://www.icce.rug.nl/~soundscapes/DATABASES/AWP/awp-notes_on.shtml

[8] A demo video is available at http://www.dke-research.de/aucoma/

[9] Such an action, e.g., through drag & drop on the ranked list, has not yet been incorporated into the graphical user interface. Currently, it can only be carried out through the simulation interface for evaluation.

The problem of dragging songs as described above can be mapped to a problem of cluster reassignment: Let P be the set of prototypes representing all the cluster cells of the SOM. Following the "winner takes all" principle, each object is assigned to the cluster with the most similar prototype. A *position constraint* (o,t) means that If the user moves a song o to a target cluster t, a *position constraint* is generated. Such a constraint can be expressed by multiple distance constraints as the distance of o to the respective prototype p_t of t has to be smaller than to any other prototype:

$$d(o,p_t) < d(o,p) \qquad \forall p \in P \setminus \{p_t\} \qquad (12)$$

If the user corrects a retrieved top-10 list $o_1,\ldots,o_{10}$ for a seed song s by moving the song at rank i to rank j, a ranking constraint $(s,o_1,\ldots,o_{10},i,j)$ is generated. Such a constraint is interpreted as follows: If j is less than i, it can be concluded that $o_j,\ldots,o_{i-1}$ are less similar and that o_{j-1} (if such exists) is more similar than o_i w.r.t. the seed song s. In the other case, i.e. if j is greater than i, the songs with rank $i < r < j$ are more similar than o_i and o_j is less similar w.r.t. the seed song s. This translates to the following distance constraints:

$$\begin{aligned} d(s,o_i) &> d(s,o_{j-1}) \\ \wedge\, d(s,o_i) &< d(s,o_r) \quad \forall j \le r < i \quad \text{if } j < i \quad (13) \\ d(s,o_i) &< d(s,o_j) \\ \wedge\, d(s,o_i) &> d(s,o_r) \quad \forall i < r < j \quad \text{if } i < j \quad (14) \end{aligned}$$

This way, the set of constraints increases with each user action. In this incremental learning scenario, where a gradual change of the weights is desired to avoid too abrupt changes of the cluster assignments, the quadratic optimization approach (Section 2.2.2) has been applied.

4.3. Experiments

By simulating the user interaction with the system, it is possible to objectively evaluate the usefulness of the adaptation approach. This section outlines the experimental setup and the most important findings. Additional details beyond the scope of this paper can be found in [17]. The basic idea is to assume for each simulation a fixed random facet weighting that the simulated user has "in mind". The following three scenarios are considered: fully random weights, random binary weights (with additional normalization to satisfy Equation 5), and random single facet only (the weights for all facets except for a single random facet are zero). The resulting distance measure (which is unknown to the system) is used to identify misplaced songs or the correct order of a ranking. During a simulation, the user either only rearranges misplaced songs or corrects rankings. In each simulation step, one randomly selected misplaced song is moved or the top-10 ranking for a random seed song is corrected respectively (only considering the result with the highest rank discrepancy). Afterwards, the weights are adapted incorporating the new constraints. A simulation terminates when no misplaced object can be moved anymore or no more top-10 rankings can be corrected respectively. Each scenario is tested on multiple SOMs (learned on the same data but with different random initializations) for multiple users (different facet weights) and multiple simulations each (different songs chosen to be moved or as seeds for ranked lists). Two measures are computed for evaluation: the *average top-10 precision (Prec@10)* which measures how much the nearest neighbors agree, and the *object position error (OPE)*. The latter is defined as the aggregated Euclidean distance on the cell grid between the current and the correct cell coordinates for each object in the dataset [17]. It thus allows a much finer assessment of the object misplacement than the bare number of misplaced objects.

4.4. Results

Of the three scenarios, the fully random weighting is closest to the uniform initialization. Only few iterations for adaptation are required and the OPE is surprisingly small. Scenario 3 is the hardest with an initial Prec@10 of about only 10% and nearly all songs misplaced.

Using position constraints, the adaptation converges much quicker and requires far less iterations than for ranking constraints. A possible cause is that more distance constraints can be derived from a position constraint and thus the adaptation algorithm has more information in fewer steps. It becomes evident from looking at the final Prec@10 that the termination criterion is met earlier: For rearranging misplaced songs, the value would never come even close to 100% – even though all songs finally are at their correct position – and is much smaller than what can be achieved by using ranking constraints. The explanation for this "glass ceiling" effect lies in the nature of the SOM. All objects are assigned to the correct cluster, as long as there is no other cluster that is more similar. If this criterion is met, the simulation stops because no more position constraints can be generated that would add information for the optimizer.

At this point, however, there may still be many valid weighting schemes. The system has just not enough evidence, to decide which one is the right one, and chooses the one that minimizes the objective function (c.f. Equation 11). Especially with a large solution space, the chosen weighting may significantly differ from the real one which results in the "glass ceiling" for the Prec@10.

Using the ranking constraints directly optimizes the Prec@10 and only indirectly affects the OPE through the adapted distance measure. Consequently, a significantly higher number of iterations is necessary in these simulations until all songs are correctly arranged, independently from the scenario of user weight initialization. Apart from the fact that less information can be derived from a ranking constraint, the resulting distance constraints may also be not as restrictive because they mostly refer to songs that are considered similar. In contrast, the variance amongst the cluster prototypes (which are considered for each position constraint) is high because they cover the whole collection.

Consequently, the optimal strategy for a user to minimize manual effort would be a combination of both, rearranging and ranking. Moving misplaced songs can be used for a rough adaptation of the system in only a few steps. Afterwards, the correction of ranking provides a means for fine-tuning.

5. APPLICATION III: MUSICGALAXY

The user-interface prototype presented in the previous section has several limitations: Apart from the required vectorization (Section 4.1), there are several parameters that have to be tuned carefully to obtain a good SOM such as the initialization of the prototypes, the learning rate, the termination criterion for iteration, the initial structure, and the rules by which the structure should grow. Moreover, if the distance measure is changed drastically during user-adaptation, the structure of the SOM which was learned using the initial facet weights may no longer be appropriate. In the worst case, all songs might end up in the same cluster cell. Another problem is the scalability w.r.t. the collection size: The initial generation of the SOM already takes several seconds for the rather small Beatles corpus. Furthermore, for large collections, the SOM will consist of many cells or cells that contain many songs – and both limits the usefulness of the visualization.

Moreover, approaches that like SOMs project a dataset onto two dimensions while trying to preserve the topology, share a common fundamental problem: The inherent dimensionality reduction inevitably causes some loss of information. This leads to a distorted display of the original neighborhoods such that some objects will appear closer than they actually are, and on the other hand some objects that are distant in the projection may in fact be close neighbors. Fixing one distorted neighborhood is generally not possible without damaging others. However, instead of trying to globally repair projection errors, the neighborhood *in user focus* can be temporarily fixed.

5.1. Focus-Adaptive Distortion

MusicGalaxy [18] is an interface prototype for exploring large music collections that uses such a *focus-adaptive* distortion technique. It takes a different approach than the *BeatlesExplorer* addressing the above mentioned problems. Figure 3 shows a screenshot of the interface visualizing a music collection. Within *MusicGalaxy*, each track is a star and a spatially well distributed subset of frequently played tracks is additionally displayed as a album cover for orientation. The arrangement is computed using Multidimensional Scaling (MDS) [8] based on a multi-facet distance measure. MDS aims to preserve the distances between the tracks in the projection, this way minimizing the overall distance distortions but still facing the problem that some neighborhoods may not be projected correctly. To this end, an adaptive distortion technique is applied based on a complex overlay of multiple fish-eye lenses divided into *primary* and *secondary focus*. While the user can control the fish-eye lens of the primary focus to zoom into regions of interest, the secondary focus is automatically adapted. It consists of a varying number of smaller fish-eye lenses. When the primary focus changes, a *neighbor index* is queried with the object closest to the center of focus. If nearest neighbors are returned that are not in the primary focus, secondary lenses are added at the respective positions. As a result, the overall distortion of the visualization temporarily brings the distant nearest neighbors back closer to the focused region of interest. This way, distorted distances introduced by the projection can to some extend be compensated.

The user-interface has been evaluated in a study as reported in [15]. The participants had to solve an exploratory image retrieval task: finding and tagging representative images for several topics.[10] The evaluation

[10] Images were used instead of music tracks because (1) it could be guaranteed that the collection is unknown to all users and (2) visual similarity and relevance are much quicker assessed.

facet	projection	distortion
dynamics	0.0	1.0
rhythm	1.0	0.0
timbre	0.0	1.0

Fig. 3: Top: *MusicGalaxy* (inverted color scheme for print). Bottom right: corresponding lens distortion resulting from (user-controlled) primary focus (red) & (adaptive) secondary lenses (blue). Bottom left: facet weights for the projection and distortion distance measures.

showed that the participants indeed frequently used the secondary focus to find other photos belonging to the same topic as the one in primary focus. However, some photos in secondary focus did not belong to the same topic. Follow-up experiments [19] investigated whether it is possible to automatically adapt the neighbor index during the exploratory search process to return more relevant neighbors for the primary focus topic. The approach and findings are summarized in the following.

5.2. Modeling the Learning Problem

The required preference information is deduced from the annotations already made by the user: Given a tagged object in primary focus all other objects with the same tag are considered as relevant. Assuming that they share some common feature(s), the distance measure of the neighbor index that is queried for the distortion has to be adapted accordingly, such that more relevant objects are amongst the nearest neighbors. This *distortion distance measure* can be adapted independently of the *projection distance measure* which is used by the MDS and left untouched here to not confuse the user by a changing arrangement. The distance constraints for the adaptation can be derived much like for the folk song classification (Section 3): A pair of objects, a and b, annotated with the same tag T should be more similar to each other than to any other object c not belonging to T:

$$d(a,b) < d(a,c) \;\wedge\; d(a,b) < d(b,c) \quad \forall (a,b,c) | a,b \in T \wedge c \notin T \qquad (15)$$

As for the folk songs, facet weightings with different levels of specificity can be learned depending on the scope of the constraints: individual, per-tag or overall. Here, the problem of learning to tag is modeled as classification task using the maximum margin approach (Section 2.2.3) for a robust classification model that can deal with constraint inconsistencies. Continuity is less important here, because the adaptation has to impact on the position of the objects in the galaxy.

5.3. Observations & Outlook

The experiments with the photo collection show that the adaptation of the distortion distance measure indeed increases the number of relevant (and still untagged) objects retrieved by the neighbor index. – In *MusicGalaxy*, this would not only be helpful for tagging but, e.g., also for generating playlists by navigating the secondary focus. However, the initial number of only three facets did not provide enough degrees of freedom for the adaptation algorithm. This problem was solved by decomposing one facet (based on a color histogram) into 64 sub-facets (one per bin) which allowed a finer level of adaptation. At the same time, this increased the risk of overfitting as well – especially for the individual weightings. Similar problems are likely to occur for *MusicGalaxy* which currently also considers only a small number of facets. Therefore, the variety of facets needs to be increased before further experiments with real users are conducted.

Further work includes the integration of adaptive structuring by rearranging and ranking correction as described in Section 4, i.e. changing the projection distance measure used for the MDS. While ranking correction is directly supported, the absence of prototypes as reference points in the structure makes the interpretation of rearrangements a challenging task.

6. CONCLUSIONS

The presented approach for adapting distance computation is broadly applicable in various contexts: Firstly, it only puts little restriction on the underlying distance facets. Secondly, the learning process is formulated as common constrained optimization problem and thus, it is possible to employ a wide range of generic solvers with different objectives that fit to the application scenario. And finally, the distance constraints that guide the adaptation can easily be derived from a wide range of information – either provided by an expert or inferred from the actions of a user in an interactive setting. For future work, a performance comparison of the different optimization approaches on a benchmark dataset is planned.

7. ACKNOWLEDGMENTS

This work has been supported by the German National Merit Foundation, the German Research Foundation (DFG) under the project AUCOMA, and the European Commission under FP7-ICT-2007-C FET-Open, contract no. BISON-211898. The author would further like to thank everyone who contributed to this work and the organizers of the AES 42nd International Conference for their kind invitation to present it.

8. REFERENCES

[1] K. Bade, J. Garbers, S. Stober, F. Wiering, and A. Nürnberger. Supporting folk-song research by automatic metric learning and ranking. In *Proc. of ISMIR'09*, 2009.

[2] K. Bade and A. Nürnberger. Creating a cluster hierarchy under constraints of a partially known hierarchy. In *Proc. of SIAM'08*, 2008.

[3] W. Cheng and E. Hüllermeier. Learning similarity functions from qualitative feedback. In *Proc. of ECCBR'08*, 2008.

[4] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27(1):1–33, 1983.

[6] C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proc. of ISMIR'05*, 2005.

[7] P. Knees, T. Pohle, M. Schedl, and G. Widmer. Exploring Music Collections in Virtual Landscapes. *IEEE MultiMedia*, 14(3):46–54, 2007.

[8] J. Kruskal and M. Wish. *Multidimensional Scaling*. Sage, 1986.

[9] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle. jAudio: An feature extraction library. In *Proc. of ISMIR'05*, 2005.

[10] B. McFee and G. Lanckriet. Heterogeneous embedding for subjective artist similarity. In *Proc. of ISMIR'09*, 2009.

[11] A. Nürnberger and A. Klose. Improving clustering and visualization of multimedia data using interactive user feedback. In *Proc. of IPMU'02*, 2002.

[12] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, 2006.

[13] A. Pinto, R. H. van Leuken, M. F. Demirci, F. Wiering, and R. C. Veltkamp. Indexing music collections through graph spectra. In *Proc. of ISMIR'07*, 2007.

[14] M. Schedl. The CoMIRVA Toolkit for Visualizing Music-Related Data. Technical report, Johannes Kepler University Linz, 2006.

[15] S. Stober, C. Hentschel, and A. Nürnberger. Evaluation of adaptive springlens - a multi-focus interface for exploring multimedia collections. In *Proc. of NordiCHI'10*, 2010.

[16] S. Stober and A. Nürnberger. User modelling for interactive user-adaptive collection structuring. In *Proc. of AMR'07*, 2007.

[17] S. Stober and A. Nürnberger. Towards user-adaptive structuring and organization of music collections. In *Proc. of AMR'08*, 2008.

[18] S. Stober and A. Nürnberger. MusicGalaxy - an adaptive user-interface for exploratory music retrieval. In *Proc. of SMC'10*, 2010.

[19] S. Stober and A. Nürnberger. Similarity adaptation in an exploratory retrieval scenario. In *Proc. of AMR'10*, 2010.

[20] S. Stober and A. Nürnberger. Analyzing the impact of data vectorization on distance relations. In *Proc. of AdMIRe'11*, 2011. (submitted).