An Experimental Comparison of Similarity Adaptation Approaches

Sebastian Stober and Andreas Nürnberger

Data & Knowledge Engineering Group
Faculty of Computer Science
Otto-von-Guericke-University Magdeburg, D-39106 Magdeburg, Germany
{Sebastian.Stober,Andreas.Nuernberger}@ovgu.de

Abstract. Similarity plays an important role in many multimedia retrieval applications. However, it often has many facets and its perception is highly subjective – very much depending on a person's background or retrieval goal. In previous work, we have developed various approaches for modeling and learning individual distance measures as a weighted linear combination of multiple facets in different application scenarios. Based on a generalized view of these approaches as an optimization problem guided by generic relative distance constraints, we describe ways to address the problem of constraint violations and finally compare the different approaches against each other. To this end, a comprehensive experiment using the *Magnatagatune* benchmark dataset is conducted.

1 Introduction

Similarity or distance measures are a crucial component in any information retrieval system in general. Particularly in multimedia retrieval, the objects of consideration can often be compared w.r.t. a multitude of facets. For instance, the distance of two music pieces could be computed based on the rhythm, tempo, lyrics, melody, harmonics, timbre or even mood. While sophisticated measures usually exist for each single facet, the question remains how to obtain a suitable combination of multiple facets that reflects the background and the retrieval goal of the user. In previous work, we have proposed various approaches for adapting individual distance measures as a weighted linear combination of multiple facets and demonstrated how they can be applied in several real-world interactive scenarios: organizing and exploring the work of the Beatles with the *BeatlesExplorer* user interface by re-arranging songs and correcting rankings [17], learning suitable similarity measures for folk song classification from expert annotations [2], and tagging photographs [18]. Each time, a different learning approach was taken to obtain the desired adaptation with different application-depending objectives. However, as described recently [19] and recapitulated in Section 2, there is a generalized view which is also consistent with various related works (Section 3). This formalization provides a unified model for our adaptation approaches taken so far which are briefly outlined in Sections 4.1 to 4.3. Additionally, we address the problem of constraint inconsistencies in Section 4.4 and furthermore propose alternative problem formalizations that allow constraint violations in Section 4.5. As the main contribution of this paper, Section 5 describes the experimental comparison of the approaches covered by Section 4 using the *Magnatagatune* benchmark dataset. Finally, Section 6 concludes this paper.

2 Formalization

To begin with, the concept of facet distances needs to be formalized assuming a feature-based representation of the objects of interest:

Definition 1. *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. Special care has to be taken, if extremely high facet distance values are present that express "infinite dissimilarity" or "no similarity at all". Such values introduce a strong bias for the mean of the facet distance and thus should be ignored during its computation.

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.*

Definition 2. *A relative 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 in [4]. Furthermore, the relative distance constraints are still rich enough to cover more complex forms of preference as summarized in [19]. Also note that such relative statements are usually much easier to formulate than absolute ones.

3 Relation to Other Approaches

An adaptive structuring technique for text and image collections using Self-Organizing Maps (SOMs) is described in [13]. The underlying weighted Euclidean distance is automatically adapted according to user feedback (changing the location of objects in the SOM). For the adaptation, no distance constraints as described by Equation 6 are used explicitly. Instead, weight update rules are applied based on the feature differences which correspond to the x_i in Equation 7.

Bade [1] applies metric learning for personalized hierarchical structuring of (text) collections where each document is represented by a vector of term weights [14]. To this end, structuring preferences are modeled by so-called *"must-link-before constraints"* – each referring to a triple (a, b, c) of documents. Such a constraint expresses a relative relationship according to hierarchy levels, namely that a and b should be linked on a lower hierarchy level than a and c. In hierarchical clustering, this means nothing else but that a and b are more similar than a and c. Consequently, *must-link-before constraints* can be considered as a domain-specific interpretation of the more generic relative distance constraints (Equation 6).

Cheng et al. [4] approach the metric learning problem from a case-based reasoning perspective and thus call the relative distance constraints *"case triples"*. In contrast to the previously outlined works, they address object representations beyond plain feature vectors: They model similarity as a weighted linear combination of *"local distance measures"* which corresponds to the facet concept and the aggregation function used here. As a major contribution, they also show how this formulation of the metric learning problem can be interpreted as a binary classification problem that can be solved by efficient learning algorithms and furthermore allows non-linear extensions by kernels.

The work of McFee et al. [12,11] on Metric Learning to Rank (MLR) – an extension of the Structural Support Vector Machine (SVM) approach [8] – is related in that their learning methods are also guided by relative distance constraints (which they call *"partial order constraints"*). They also combine features from different domains (acoustic, auto-tags and tags given by users) which could be interpreted as facets with the respective kernels corresponding to facet similarity measures. However, their approaches differ from the one addressed in this paper in that they aim to learn an embedding of the features into an Euclidean space and to this end apply complex non-linear transformations using kernels. Whilst their techniques are more powerful in the sense that they allow to model complex correlations, this comes at a high price: The high complexity is problematic when users want to understand or even manually adapt a learned distance measure. Here, the simplicity of the linear combination approach is highly beneficial.

Wolff and Weyde [20] also apply MLR for learning a Mahalanobis distance that reflects a perceived or stated music similarity according to relative similarity ratings by users. Their experiments are based on the *Magnatagatune* dataset, which is also used here. However, they additionally incorporate genre tags of the corresponding albums from *Magnatune* as features and furthermore take a different approach to derive constraints (referred to as *"binary rankings"*) from the similarity judgments (cf. Section 5.2). Therefore, the results cannot be compared directly. In general, a higher number of satisfiable constraints can be expected as the Mahalanobis distance has more degrees of freedom for adaptation (a $n \times n$ matrix of covariances as opposed to n facet weights), However, the convergence behavior of the incremental adaptation remains unclear. For a meaningful comparison, both, the features and the evaluation methodology, need to be identical.

Slaney et al. [15] state that they *"use labels* [artist, label and blog] *to tune the Mahalanobis matrix so that similar songs are likely to be close to each other in the metric space."* Although this is not explicitly stated in their description, this also implies either relative distance constraints (as used here) or absolute constraints of the form *"Songs a and b have to be in the same cluster."* However, using the Mahalanobis distance, they require more restrictive plain vector representations as input. Furthermore, the resulting music similarity model – i. e., the covariance matrix of the Mahalanobis distance – is still harder to interpret and adapt manually.

Finally, the *SoniXplorer* [10] is in many aspects similar to the *BeatlesExplorer* prototype described earlier in [17]: The system covers multiple facets and uses a weighted linear aggregation for the underlying similarity measure. However, the adaptation is not guided by relative distance constraints. Instead, the system allows the users to specify distance information by manipulation of the terrain, i. e., the formation of new separating hills or their removal respectively. By numerical integration over the height profile, a target distance matrix for the learning algorithm is derived that contains *absolute* (quantitative) distance information.

4 Optimization Approaches

It is possible to look at the optimization problem introduced in Section 2 from different perspectives: tolerance w.r.t. constraint inconsistencies, stability, continuity and responsiveness. The following sections describe three different optimization approaches that have been taken in previous work – each one for a different application: a gradient descent approach (Section 4.1), a Quadratic Programming (QP) approach (Section 4.2), and a maximum margin approach (Section 4.3). Furthermore, Section 4.4 describes a generic way of dealing with constraint inconsistencies and Section 4.5 proposes several alternative QP problem formulations that allow constraint violations.

4.1 Gradient Descent

In the folk song classification experiments described in [2], weights are learned by a gradient descent approach similar to the work in [3]. 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:

$$w_i = w_i + \eta \Delta w_i \tag{9}$$

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

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

[1] Approaching the weight learning problem from the classification perspective using a perceptron for classification as described in [4] results in the same update rule.

weights are normalized to sum up to l. This algorithm can compute a weighting, even if not all constraints 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 iterations to increase responsiveness but this may result in some unsatisfied constraints.

4.2 Quadratic Programming: Minimizing Weight Change

For maximum continuity which is considered most important in the *BeatlesExplorer* application described in [17], the weights should change only as little as necessary to satisfy all constraints. This can directly be modeled as a Quadratic Programming (QP) 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 (Equation 4 and Equation 5) and the distance constraints (Equation 7) which can be used directly. The problem can be solved using the Goldfarb and Idnani dual QP algorithm for convex QP problems subject to general linear equality/inequality constraints [7]. For this original formalization of the weight learning problem, there is only a solution if all constraints are consistent. Section 4.5 proposes different ways to integrate slack variables which allow the violation of constraints.

4.3 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 [18] , the linear support vector machine algorithm as provided by *LIBLINEAR* [6] 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.

4.4 Dealing with Inconsistent Constraint Sets

Sometimes the set of constraints to be used for learning may be inconsistent because there are constraints that contradict each other. Reasons for this may be manifold – e. g., a user may have changed his mind or the constraints may be from different users or contexts in general. In such case, it is impossible to learn a facet weighting that satisfies all constraints – irrespective of the learning algorithm or the facets used. In order to obtain a consistent set of constraints, the constraint filtering approach described in [12] can be applied as follows:

1. A directed multigraph (i. e., a graph that may have multiple directed edges between two nodes) is constructed with pairs of objects as nodes and the distance constraints expressed by directed edges. For instance, for the distance constraint $d(b, c) < d(a, c)$, a directed edge from the node (b, c) to the node (a, c) would be inserted.
2. All cycles of length 2 are removed, i. e., all directly contradicting constraints. (This can be done very efficiently by checking the graph's adjacency matrix.)
3. The resulting multigraph is further reduced to a Directed Acyclic Graph (DAG) in a randomized fashion: Starting with an empty DAG, the edges of the multigraph are added in random order omitting those edges that would create cycles.
4. The corresponding distance constraints of the remaining edges in the DAG form a consistent constraints set.

This can be repeated multiple times as the resulting consistent set of constraints may not be maximal because of the randomized greedy approach taken in step 3. However, finding a maximum acyclic subgraph would be NP-hard.

4.5 Quadratic Programming Approaches with Soft Constraints

The underlying algorithm [7] of the QP solver used for the approach outlined in Section 4.2 solves convex QP problems of the form

$$\min_{\mathbf{x} \in \Re^n} \mathbf{a}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T G \mathbf{x} \tag{12}$$

subject to linear equality and inequality constraints

$$\mathbf{x}^T C_e = \mathbf{b}_e \tag{13}$$

$$\mathbf{x}^T C_i \geq \mathbf{b}_i \tag{14}$$

given the vectors $\mathbf{b}_e$ of dimension m_e, $\mathbf{b}_i$ of dimension m_i, and $\mathbf{a}$ of dimension n, and the matrices G of of dimension $n \times n$, C_e of dimension $m_e \times n$, and C_i of dimension $m_i \times n$. The matrix G has to be symmetric positive definite. In this case, a unique $\mathbf{x}$ solves the problem or the constraints are inconsistent. In the original modeling (Section 4.2), the objective is to minimize the weight change under some constraints given the previous weights $w_1^{(old)}, \ldots, w_l^{(old)}$. In particular, this approach can be used to determine facet weights that are closest to a uniform weighting and feasible under the given constraints by simply setting all previous weights to 1. Here, the elements of the vector $\mathbf{x}$ in the QP problem description correspond to the facet weights $w_1, \ldots, w_l$ (where l is the number of facets), and therefore n equals l. The objective function given in Equation 11 can be transformed into:

$$\min_{(w_1,\ldots,w_l)\in\Re^l} \sum_{i=1}^{l} w_i^2 - 2\sum_{i=1}^{l} w_i w_i^{(old)} + \sum_{i=1}^{l} {w_i^{(old)}}^2 \tag{15}$$

With respect to Equation 12, the first sum is captured by $\frac{1}{2}\mathbf{x}^T G\mathbf{x}$, the second sum is expressed by $\mathbf{a}^T\mathbf{x}$, and the third sum results in a constant value independent of the w_i and thus can be neglected. A single equality constraint is required to model the bound on the weight sum (Equation 5) by setting the respective coefficients in C_e to 1 and the value in $\mathbf{b}_e$ to l. Setting the i-th element of a row vector of C_i to 1 and the other elements and the corresponding value in $\mathbf{b_i}$ to 0 enforces the non-negativity of w_i. Thus, l inequality constraints are needed to express Equation 4. Finally, each distance constraint is represented by a single row vector of C_i where the value $c_{i,j}$ is the x_i from Equation 7 for the j-th distance constraint. The respective value in $\mathbf{b}_i$ has to be a value $\epsilon > 0$ because 0 would also allow the equality in Equation 6. Naturally, this value should be as small as possible w.r.t. machine precision. Greater values increase the stability of the solution but may also result in an inconsistent system if the solution space is trimmed too rigorously such that no feasible solution exists anymore.

In order to allow a distance constraint as formulated in Equation 7 to be violated, a slack variable $\xi \geq 0$ needs to be introduced such that:

$$\sum_{i=1}^{l} w_i(\delta_{f_i}(s,b) - \delta_{f_i}(s,a)) + \xi > 0 \tag{16}$$

This has to be done individually for all k distance constraints of the QP problem resulting in the respective slack variables $\xi_1,\ldots,\xi_k$. They are modeled as k additional dimensions of the vector $\mathbf{x}$ which is now $(w_1,\ldots,w_l,\xi_1,\ldots,\xi_k)$. (Consequently, the dimensionality of the modified QP problem is $l+k$ and as the number of distance constraints k can become quite big, this has a significant impact on the performance of the optimization algorithm.) For the constraints that ensure the weight bounds and the non-negativity of the weights, the added matrix columns are filled with zeros. For each of the k distance constraints, only the value in the column of the respective slack dimension is set to 1 while all others remain zero. Slack values other than 0 have to result in a penalty. To this end, the objective function needs to be extended. There are two possibilities to incorporate a slack penalty: either in the linear or the quadratic part of Equation 12. In the first case, the sum of the slack values is minimized, whereas in the second case, it is the sum of the squared slack values. In order to allow a balance between the (initial) objective function and the minimization of the aggregated slack, the latter is weighted by a constant κ. In case the sum of the squared slack values is used, both, negative and positive slack values are penalized. This way, the solver naturally avoids negative slack values. For the (linear) sum, however, the QP approach does not work if negative slack values are allowed because this introduces the problem that a single big negative slack value (of a constraint which is not violated) can compensate many small positive slack values of constraints that are violated. This results in a bias towards solutions with more violated constraints than necessary. Therefore, k inequality constraints that explicitly demand non-negative slack values have to be added to the scheme. Both approaches for incorporating a slack penalty into the objective function of the QP solver can be combined with the original primary objectives of minimizing the weight change. Furthermore, it is possible to have no primary objective and just minimize the slack penalty. The performance of these combinations is analyzed in the following section.

5 Experimental Comparison

In order to compare the different adaptation approaches covered above in a fully controlled environment, an experiment has been conducted using the publicly available *Magnatagatune* benchmark dataset [9]. The setup is explained in the following including the dataset and its pre-processing (Section 5.1), the constraint sets and adaptation algorithms (Section 5.2), and the evaluation methodology (Section 5.3). Results are presented and discussed in Section 5.4.

5.1 Dataset and Pre-processing

The *Magnatagatune* dataset comprises 25863 clips – each one 29 seconds long – generated from 5405 source MP3s provided by the American independent record label *Magnatune* for research purposes. The clips are annotated with a combination of 188 unique tags that have been collected through the *TagATune* game [9]. Additionally, the dataset contains a detailed analysis of each clip computed using the *EchoNest* API.[2] The features comprise musical events, beats, structure, harmony, and various global attributes such as key, mode, loudness, tempo and time signature. Most importantly for the purpose of this evaluation, there is also a set of music similarity judgments. This information has been collected by showing a triple of clips and asking the player to choose the most different one. 533 such triples have been presented to multiple players resulting in 7650 similarity judgments.

In total, 110 facets are used to describe the distances between the clips in the experiment. An overview with brief explanations is given in Table 1. The facets comprise seven globally extracted features of which two – dancability and energy – are not contained in the original clip analysis information of the dataset but have become available with a newer version of the *EchoNest* API. Furthermore, the segment-based features describing pitch ("chroma") and timbre have been aggregated (per dimension) resulting in 12-dimensional vectors with the mean and standard deviation values. This has been done according to the approach described in [5] for the same dataset. The 188 unique tags used in the manual annotations have been preprocessed as follows:

[2] A detailed description of the extracted features can be found in the documentation of the *EchoNest Track Analyze* API under `http://developer.echonest.com/`

Table 1. Facet definition for the *Magnatagatune* dataset used in the experiment. Top rows: Globally extracted features. Middle rows: Aggregation of features extracted per segment. Bottom row: Manual annotations from *TagATune* game.

feature	dim	value description	distance measure
key	1	0 to 11 (one of the 12 keys) or −1 (none)	binary (exact match)
mode	1	0 (minor), 1 (major) or −1 (none)	binary (exact match)
loudness	1	overall value in decibel (dB)	absolute difference
tempo	1	in beats per minute (bpm)	absolute difference (& tempo doubling)
time signature	1	3 to 7 ($\frac{3}{4}$ to $\frac{7}{4}$), 1 (complex), or −1 (none)	binary; $\delta(3,6) = 0.5$
danceability	1	between 0 (low) and 1 (high)	absolute difference
energy	1	between 0 (low) and 1 (high)	absolute difference
pitch mean	12	dimensions correspond to pitch classes	Euclidean distance
pitch std. dev.	12	dimensions correspond to pitch classes	Euclidean distance
timbre mean	12	normalized timbre PCA coefficients	Euclidean distance
timbre std. dev.	12	normalized timbre PCA coefficients	Euclidean distance
tags (99 facets)	1	binary, one facet per tag, very sparse	binary (exact match)

1. Merging of singular and plural forms (e. g., "guitar" and "guitars").
2. Spelling correction (e. g., "harpsicord" → "harpsichord").
3. Combination of semantically identical tags (e. g., "funk" and "funky").
4. Creation of meta-tags with higher coverage for groups of tags that express the same concept. (e. g., "instrumental" = "instrumental" or "no vocal(s)" or "no voice(s)" or "no singer(s)" or "no singing").
5. Removal of unused tags (w.r.t. the relevant subset of *Magnatagatune*).

The resulting 99 tags are interpreted as one (binary) facet each.[3]

5.2 Constraint Sets and Algorithms

Two distance constraints can be derived from a single judgments that clip c is the most different of a triple (a, b, c), namely $d(a, b) < d(a, c)$ and $d(a, b) < d(b, c)$. However, the resulting set of constraints is inconsistent because there are constraints that contradict each other. This is most likely because the similarity judgments stem from multiple players of the *TagATune* game. Applying the filtering technique described in Section 4.4, a constraint graph with 15300 edges of which 1598 are unique is constructed. After the removal of length 2 cycles, 860 unique edges remain (6898 in total). The randomized filtering finally results in a DAG with 674 unique edges (6007 in total). Thus, the filtered consistent set

[3] Alternatively, it is possible to combine all annotations into a single facet or define facets for groups of related tags (e. g., all tags related to instrumentation) which significantly reduces the number of facets. However, this would also drastically reduce the size of the *selected constraints* set described in the following.

contains 674 constraints of which each is backed by 8.9 judgments on average. In the following, this set is referred to as *all constraints* set.

Even for the consistent *all constraints* set, it is impossible to learn a facet weighting that violates none of the constraints because the information captured by the facets is insufficient. I. e., players may have considered aspects in their judgments that are not covered by the features. From the classification perspective of Equation 7 this means that there is no hyperplane that clearly separates the positive from the negative examples. Or from the QP perspective, the system of equality and inequality constraints is inconsistent – i. e., it has no solution. Therefore, another set – in the following referred to as *selected constraints* – has been constructed by further filtering the *all constraints* set. To this end, the randomized approach of Section 4.4, step 3 has been applied again but this time constraints are only added to the set if the resulting QP problem has a solution. The *selected constraints* set obtained this way contains 521 constraints.

At first sight, the two sets of constraints seem to be quite large. After all, which user would like to answer several hundred questions of the form *"Which one of these three objects do you think is the most distinct one from the others?"* However, these distance constraints are in fact only the very atomic pieces of information used to guide the adaptation. As the example applications described in earlier work show, usually multiple such distance constraints are derived from a single action like moving an object [17], correcting a ranking [2] or adding a tag annotation [18].

Table 2 lists the considered algorithms and their parameters. They comprise the three algorithms used in the applications described in previous work [17,2,18] and furthermore several variants of the QP approach with added slack dimensions that allow the violation of distance constraints (Section 4.5). As the GradDesc learner may get stuck in a local optimum, the computation is repeated up to 50 times if no solution could be found that satisfies all training constraints. Each run uses a different random order of the same training constraints. Finally, the solution which results in the lowest number of constraint violations is chosen.

5.3 Evaluation Methodology

The evaluation aims to answer the following questions:

- How good is the obtained adaptation (in terms of constraint violations)?
- How fast (with how much user effort) can it be learned?
- How stable is the quality of an adaptation if new constraints are added?

The number of violated distance constraints serves as a measure for how well the algorithm has adapted to the similarity preferences given some training constraints. All algorithms except QPmin(Δw) that cannot deal with inconsistencies are tested on both sets of constraints described in Section 5.2. For the *selected constraints* set, a solution satisfying all constraints is expected. Whereas, for the *all constraints* set, the behavior of the algorithms under constraints that cannot all be satisfied is tested. As the size difference between the two sets is 153,

Table 2. Algorithms covered in the comparison. Top: Algorithms used in applications described in earlier work (Sections 4.1 to 4.3). Bottom: Alternative QP problem formulations with added slack dimensions (Section 4.5).

abbreviation	algorithm	parameters
GradDesc	Gradient Descent	50 repeats with random permutations of training samples, dyn. learning rate
QPmin(Δw)	Quadratic Programming	minimal weight change, no slack
LibLinear	Maximal Margin Classifier (Java *LIBLINEAR* v1.5)	L2-regularized L2-loss SVC, $C = 10^7$, $\epsilon = 10^{-6}$, no bias term
QPmin(ξ)	Quadratic Programming	no primary objective, lin. slack $\kappa = 1$
QPmin(ξ^2)	Quadratic Programming	no primary objective, quad. slack $\kappa = 1$
QPmin(Δw+ξ)	Quadratic Programming	min. weight change, lin. slack $\kappa = 1$
QPmin(Δw+ξ^2)	Quadratic Programming	min. weight change, quad. slack $\kappa = 1$
QPmin(Δw+$10^5\xi^2$)	Quadratic Programming	min. weight change, quad. slack $\kappa = 10^5$

the optimal performance value for the *all constraints* set is expected to be close to 150 in terms of constraint violations. For each of the two sets, 100 random permutations of the distance constraints are generated. Each permutation is presented to the adaptation algorithm – one constraint at a time (i. e., stepwise) – until all constraints are used for training. After each step, the number of violated constraints in the whole set is determined. The values are averaged per step over the 100 permutations to reduce ordering effects.

5.4 Results

Figure 1 shows the detailed performance of all algorithms listed in Table 2. Each diagram combines the results of a single algorithm on both constraint sets – *all constraints* (top, red curve) and *selected constraints* (bottom, blue curve) – as these do not overlap. Both colored curves refer to the average of the 100 runs. Additionally, all performance values obtained for the 100 random permutations of the constraints are shown as points (in light gray). This gives an impression of the variance between the different runs. The two gray dotted horizontal lines indicate the baseline performance value obtained by the uniform facet weighting on all constraints (upper line) and the subset of selected constraints (lower line). The scaling of all plots is identical for better comparability. It is also the same for both axis.

Comparing the algorithms applied in earlier work (top rows of Table 2), the plots for the *selected constraints* set, where a weighting can be found that satisfies all constraints, are almost identical. For LibLinear the mean curve is a bit steeper, indicating slightly better early solutions. However, a little more variance can be observed – especially between 50 and 80 training constraints. Furthermore, it has to be noted that GradDesc and LibLinear converge on a solution that leaves a small number of constraints violated whereas QPmin(Δw) finds a weighting without constraint violations. The GradDesc learner still gets stuck in a local optimum

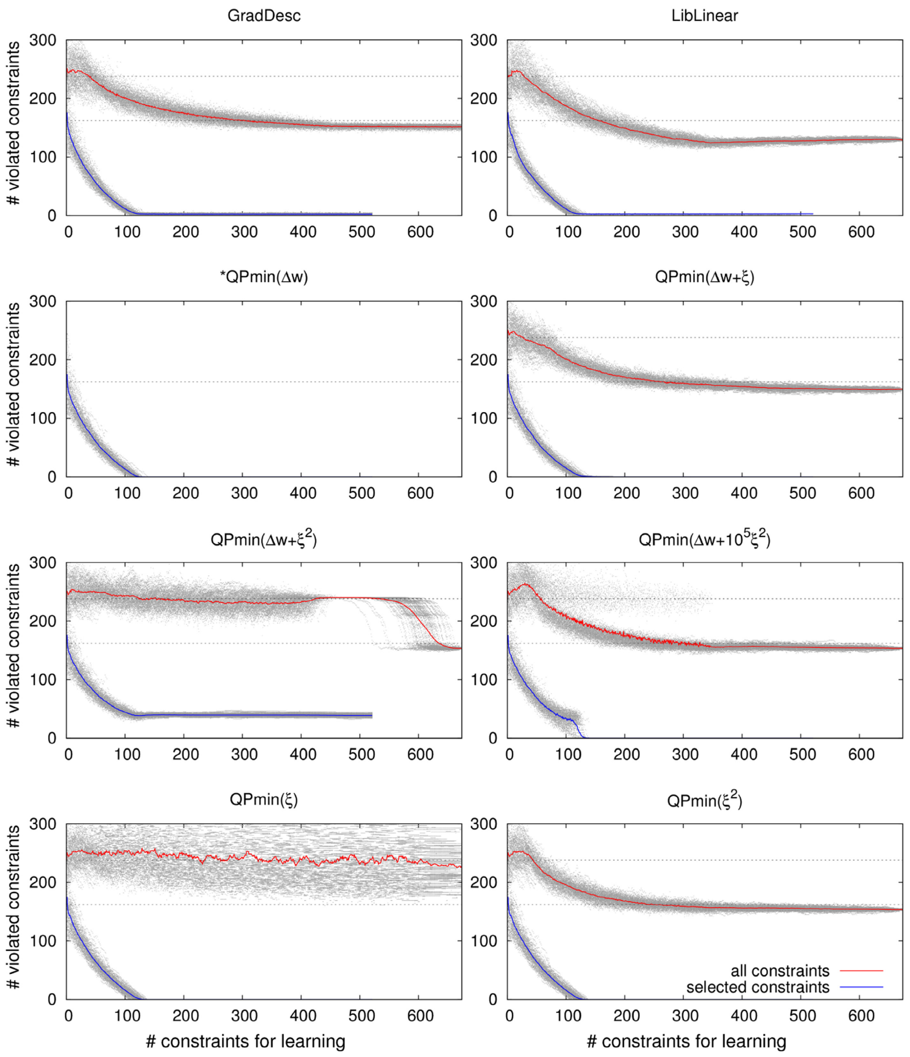

Fig. 1. Performance plots for the algorithms listed in Table 2 averaged over 100 random permutations of the *all constraints* set (red curves) and *selected constraints* set (blue curves). Baseline values for uniform facet weightings are shown as dotted gray horizontal lines. The gray point clouds visualize the value distributions within the 100 runs. (*QPmin(Δw) is not applicable on the *all constraints* set.)

possibly close to the global one. The problem of LibLinear is that it favors a large margin over small constraint violations, e. g., caused by small negative facet weights. For the larger *all constraints* set, QPmin(Δw) does not return a solution because the derived QP system to be solved is inconsistent. Comparing GradDesc and LibLinear which both can deal with constraint violations, the latter again shows faster convergence but slightly higher variance. Most notably, LibLinear leads to a solution that violates approximately 30 constraints less than GradDesc. The main reason for this is that it trades a few (slightly) violated weight bounds constraints against a larger number of distance constraints that are not violated. This results however in an invalid weighting.

Looking at the alternative QP approaches that aim to minimize only the slack without a primary objective, the plots for the *selected constraints* set look almost identical. They do not differ much from QPmin(Δw) even though the objective is much different here. Therefore, it can be concluded that both approaches work well if there is a solution that violates no constraints. However, the performance could not look much more different for the *all constraints* set: QPmin(ξ), i. e., modeling the slack in the linear part of the QP objective function (and leaving the quadratic part constant), seems not to work at all. There is almost no improvement compared to the baseline. At the same time, the variance increases which is much in contrast to all other approaches. However, QPmin(ξ^2), i. e., modeling the slack in the quadratic part of the objective function, produces a better solution than GradDesc. At the beginning, for up to 30 training constraints, there is no improvement. In fact both plots look here very much alike. But then the number of violations drops quickly and for 100 training constraints, it is already lower than for GradDesc.

For the alternative QP approaches that minimize the change of the facet weights with soft constraints, QPmin(Δw+ξ), has the best performance on both sets of constraints. This is surprising considering that QPmin(ξ) does not work at all for the *all constraints* set. Much in contrast, minimizing the quadratic slack penalty works only well without the primary objective (QPmin(ξ^2)). For the combination, QPmin(Δw+ξ^2), there seems to be a conflict between both objectives. This results in an unsatisfactory adaptation for the *selected constraints* set with more than 40 constraint remaining violated. The QPmin(Δw+ξ^2) plot for the *all constraints* set is very remarkable. It can be divided into three sections: In the first section up to roughly 440 constraints, there is high variance between the permutations and no significant improvement. Then, however, the values converge and until about 525 constraints, no variance can be observed. This point coincides with the size of the *selected constraints* set which is close to the maximal number of constraints that can be satisfied. Afterwards, in the last section, the number of violated constraints quickly decreases to a final value that is comparable to the other working approaches. This late adaptation suggests that the primary objective (minimizing the weight change) suppresses the minimization of the slack until the last section. Indeed, the facet weights have converged to 1 at the beginning of the second section which explains the performance close to the baseline (uniform facet weights). Only afterwards, the importance of the slack gains the upper hand – most likely because of the high number of slack dimensions caused by the many training constraints in this section. Choosing a high slack weight results in an earlier adaptation as shown for QPmin(Δw+$10^5\xi^2$) with $\kappa = 10^5$. However, the variance is very high and the performance is still inferior to QPmin(Δw+ξ). Even higher values of κ result in no significant improvement.

Finally, Table 3 lists some empirically determined values for the processing time of the different algorithms. Especially in interactive settings, a short response time is important. For GradDesc, the times refer only to a single repetition. Generally, these measurements can only give an impression of the processing time for the adaptation as no special preparations of the testing system, the runtime environment or the compiler have been made. The values are averaged over 100 random permutations and have been measured for 10, 100 and all available training constraints of the two sets. Values for the *all constraints* set are expected to be higher because of the unavoidable constraint violations that occur here. For GradDesc and LibLinear, this is surprisingly not the case. Possibly, finding a solution for the *selected constraints* set is harder for these algorithms. However, it has to be noted that LibLinear is the only approach that in the current implementation of the library interface requires slow hard disk access to read the problem description from a temporary file. The actual processing times for LibLinear are therefore much lower. In the adaption experiment described in [18] with much larger constraint sets, LibLinear has already shown that its runtime scales well. The QP approaches could run into problems here – especially QPmin(Δw+ξ) which requires even more constraints for the non-negativity of the slack variables. In contrast to the other algorithms, GradDesc, which is rather slow (but also the only algorithm in the evaluation that does not rely on highly optimized library code) could be interrupted during computation and still return a satisfying adaptation.

A direct comparison of all tested approaches is shown in Figure 2. For the *selected constraints* set, all approaches except those using the square slack penalty work almost equally well. I. e., if a solution exists that satisfies all constraints, one is found. Only GradDesc gets stuck a little too early and LibLinear favors the larger margin. Of the approaches with square slack penalty, QPmin(Δw+ξ^2) does not work well leaving roughly 40 unsatisfied constraints and QPmin(Δw+$10^5\xi^2$) converges too slowly. For the harder *all constraints* set, LibLinear can be considered as the "disqualified winner" of the competition. It shows the overall quickest convergence requiring less steps then the other approaches for the adaptation and returns weightings that violate significantly fewer constraints. However, the latter is only possible because of "cheating" as the weights violate the essential non-negativity constraint (Equation 4) and thus cannot be interpreted as intended. Given these results and the good scalability for large problems, an internal modification of *LIBLINEAR* that ensures non-negative weights looks promising. However, this is not a trivial task and intended for future work. QPmin(Δw+ξ) has the best final performance value for a valid adaptation on *all constraints* which is even slightly below 150. However, its adaptation is a bit slow

Table 3. Processing times (in seconds) for the adaptation depending on the number of training constraints measured on both constraint sets. Values averaged over 100 repetitions on a consumer notebook (2.4 GHz Intel Core 2 Duo, 4GB RAM). Algorithms without satisfying adaptations in the evaluation are in gray.

algorithm (abbreviation)	selected constraints 10	100	521	all constraints 10	100	674
GradDesc	<0.01	3.34	13.29	<0.01	5.41	8.45
QPmin($\varDelta$w)	<0.01	0.02	1.08			
LibLinear	0.13	1.21	2.54	0.38	0.46	1.19
QPmin(ξ)	<0.01	0.06	5.84	<0.01	0.19	19.62
QPmin(ξ^2)	<0.01	0.03	1.15	<0.01	0.05	5.95
QPmin($\varDelta$w+ξ)	<0.01	0.06	6.59	<0.01	0.06	27.73
QPmin($\varDelta$w+ξ^2)	<0.01	0.02	0.82	<0.01	0.03	3.05
QPmin($\varDelta$w+$10^5\xi^2$)	<0.01	0.02	1.12	<0.01	0.04	4.59

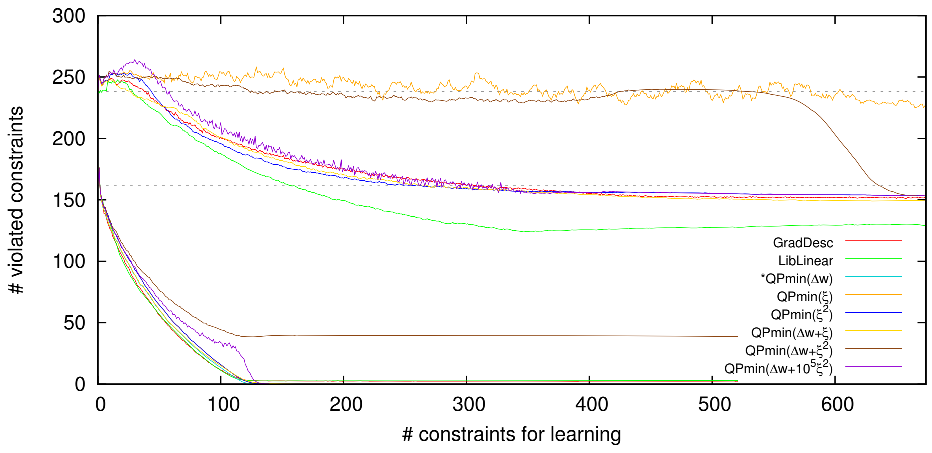

Fig. 2. Direct comparison of all approaches tested in the experiment. All values are averaged over 100 random permutations of the constraints. (*QPmin($\varDelta$w) is not applicable on *all constraints* set.)

in the beginning. For the first 70 steps, GradDesc would be a better choice and in the middle section, QPmin(ξ^2) does slightly better. In the end, the performance difference of these three approaches is only very small. Finally, QPmin(ξ) does not work at all for *all constraints* and QPmin($\varDelta$w+ξ^2) converges only in the end which is not acceptable either. These combination should therefore not be used.

6 Conclusions

Based on a generalized view that brings together our work from recent years in the field of adaptive similarity and distance measures based on weighted linear combinations, we have conducted an experimental comparison of the different approaches. To this end, we have utilized the publicly available *Magnatagatune* benchmark dataset. Pre-processing of this dataset comprised the refinement of the tag annotations, the definition of facets, the generation of distance constraints from the similarity judgments, and the filtering of the constraints to obtain a consistent set. As performance measure, the number of constraint violations has been plotted against the number of constraints used for training. In general, the evaluation methodology is not confined to approaches that model distance as a weighted linear combination of facets. Basically, any algorithm that uses relative distance constraints can be tested this way. Thus, it is possible to extend the comparison and also cover approaches that, e. g., use the Mahalanobis distance or complex kernel-based models which is planned for future work.[4] Further plans include to modify *LIBLINEAR* such that weights cannot become negative.

Acknowledgments. This work was supported in part by the German National Merit Foundation and the German Research Foundation (DFG) under the project AUCOMA.

References

1. Bade, K.: Personalized Hierarchical Structuring. PhD thesis, Otto-von-Guericke-University Magdeburg (2009)
2. Bade, K., Garbers, J., Stober, S., Wiering, F., Nürnberger, A.: Supporting folk-song research by automatic metric learning and ranking. In: Proc. of the 10th Int. Conf. on Music Information Retrieval (ISMIR 2009) (2009)
3. Bade, K., Nürnberger, A.: Creating a cluster hierarchy under constraints of a partially known hierarchy. In: Proc. of the 2008 SIAM Int. Conf. on Data Mining (2008)
4. Cheng, W., Hüllermeier, E.: Learning similarity functions from qualitative feedback. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 120–134. Springer, Heidelberg (2008)
5. Donaldson, J., Lamere, P.: Using visualizations for music discovery. In: Tutorial at the 10th Int. Conf. on Music Information Retrieval (ISMIR 2009) (2009)
6. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research 9, 1871–1874 (2008)
7. Goldfarb, D., Idnani, A.: A numerically stable dual method for solving strictly convex quadratic programs. Mathematical Programming 27(1), 1–33 (1983)
8. Joachims, T.: A Support Vector Method for Multivariate Performance Measures. In: Proc. of the Int. Conf. on Machine Learning (ICML 2005) (2005)

[4] The constraint sets derived from the *Magnatagatune* dataset can be provided upon request. Please contact `sebastian.stober@ovgu.de`

9. Law, E., von Ahn, L.: Input-agreement: a new mechanism for collecting data using human computation games. In: Proc. of the 27th Int. Conf. on Human Factors in Computing Systems (CHI 2009) (2009)
10. Lübbers, D., Jarke, M.: Adaptive multimodal exploration of music collections. In: Proc. of the 10th Int. Conf. on Music Information Retrieval (ISMIR 2009) (2009)
11. McFee, B., Barrington, L., Lanckriet, G.: Learning similarity from collaborative filters. In: Proc. of the 11th Int. Conf. on Music Information Retrieval (ISMIR 2010) (2010)
12. McFee, B., Lanckriet, G.: Heterogeneous embedding for subjective artist similarity. In: Proc. of the 10th Int. Conf. on Music Information Retrieval (ISMIR 2009) (2009)
13. Nürnberger, A., Klose, A.: Improving clustering and visualization of multimedia data using interactive user feedback. In: Proc. of the 9th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002) (2002)
14. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Information Processing & Management 24(5), 513–523 (1988)
15. Slaney, M., Weinberger, K.Q., White, W.: Learning a metric for music similarity. In: Proc. of the 9th Int. Conf. on Music Information Retrieval (ISMIR 2008) (2008)
16. Nürnberger, A., Stober, S.: User modelling for interactive user-adaptive collection structuring. In: Boujemaa, N., Detyniecki, M., Nürnberger, A. (eds.) AMR 2007. LNCS, vol. 4918, pp. 95–108. Springer, Heidelberg (2008)
17. Stober, S., Nürnberger, A.: Towards user-adaptive structuring and organization of music collections. In: Detyniecki, M., Leiner, U., Nürnberger, A. (eds.) AMR 2008. LNCS, vol. 5811, pp. 53–65. Springer, Heidelberg (2010)
18. Stober, S., Nürnberger, A.: Similarity adaptation in an exploratory retrieval scenario. In: Detyniecki, M., Knees, P., Nürnberger, A., Schedl, M., Stober, S. (eds.) AMR 2010. LNCS, vol. 6817, pp. 144–158. Springer, Heidelberg (2012)
19. Stober, S.: Adaptive distance measures for exploration and structuring of music collections. In: Proc. of AES 42nd Conf. on Semantic Audio (2011)
20. Wolff, D., Weyde, T.: Combining Sources of Description for Approximating Music Similarity Ratings. In: Detyniecki, M., García-Serrano, A., Nürnberger, A., Stober, S. (eds.) AMR 2011. LNCS, vol. 7836, pp. 114–124. Springer, Heidelberg (2013)