POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Learning a personalized similarity metric for musical content.

Supervisor: Prof. Augusto Sarti
Co-supervisor: Dr. Michele Buccoli

Master graduation thesis of :
Luca Carloni, matricola 837848

Academic year 2016/2017

POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

Apprendimento di una metrica di similarità personalizzata a partire dal contenuto.

Relatore: Prof. Augusto Sarti
Correlatore: Dr. Michele Buccoli

Tesi di Laurea Magistrale di:
Luca Carloni, matricola 837848

Anno Accademico 2016/2017

Abstract

Nowadays people have the chance to easily access a wide amount of music by means of many services. As users are not able to handle such wide music catalogs on their own, services need techniques able to automatically assist them. In order to do that, services rely on the evaluation of song similarity. As music is multi-faceted, users tend to evaluate similarity in different ways. Thus the concept of music similarity is highly subjective. Consequently, also services must evaluate music similarity in a personalized way.

As many users exploits music fruition services, we need scalable methods. So, we must adopt a content-based approach, as content-based approaches rely on content-related information computed from the audio track, thus always available. However, in order to elaborate a subjective similarity metric, content-based techniques need to be combined with music similarity informations provided by the user himself. So, in this thesis we present a hybrid model for personalized similarity modeling that relies on both content-based and user-related similarity information.

The goal is to elaborate a metric able to relate content-based and the similarity information provided by the user. To do so, we proposed a method that relies on a two-stage procedure. We first exploit a non-metric scaling technique to first elaborate a low-dimensional space (or embedding) which fulfills the similarity information provided by the user. Then we exploit a regression technique in order to learn a mapping able to relate content-based information and embedding-related information. We kernelize the regression procedure adopting a non-linear kernel function. In order to enhance the generalization properties of the method, we also combine the regression operation with a feature selection algorithm. The result is a novel content-based method for learning a personalized similarity metric for musical content.

The experiments that we conducted in order to assess the generalization properties of our method show that it is able to provide good performances,

even when data are divided adopting a rigid data division method.

Sommario

Oggi è possibile accedere ad una vasta quantità di musica attraverso vari servizi. La quantità di musica disponibile è tale per cui l'utente non è in grado di gestirla autonomamente. Per cui, i vari servizi devono dotarsi di tecniche in grado di aiutare ogni utente in maniera automatica. Per farlo, i servizi devono basarsi sulla stima delle similarità tra canzoni. Ma poichè la musica è un fenomeno complesso e multiforme, la similarità tra canzoni può essere valutata secondo varie prospettive, ed ogni utente tende ad avere la propria visione di similarità. Per soddisfare appieno gli utenti, dunque, la valutazione della similarità da parte dei servizi deve ricalcare quella dell'utente.

Poichè il numero di utenti a cui rivolgersi è elevato, occorrono tecniche di modellazione della similarità scalabili. Per cui, scegliamo un approccio basato sul contenuto, poichè l'informazione sul contenuto è sempre disponibile, data la canzone. Tuttavia, per poter apprendere una metrica di similarità personalizzata, tali metodi devono includere delle informazioni sulla similarità percettiva fornite dell'utente. Motivo per cui in questa tesi presentiamo un approccio ibrido in grado di combinare i due tipi di informazione : contenuto e similarità utente.

L'obiettivo è quello di apprendere una metrica in grado di mettere in relazione le informazioni di contenuto e la similarità percettiva. Il metodo proposto si basa su un approccio bifase. Una tecnica di non metric scaling si occupa prima di elaborare uno spazio geometrico modellato sulla base delle informazioni sulla similarità fornite dall'utente. Poi una tecnica di regressione apprende un mapping in grado di relazionare l'informazione di contenuto e lo spazio appreso. La procedura di regressione è sia resa non lineare tramite un kernel non lineare, sia combinata con una procedura di feature selection in modo da massimizzare le capacità di generalizzazione del metodo. Il risultato è un nuovo metodo capace di apprendere una metrica

di similarità.

Gli esperimenti condotti dimostrano che il metodo fornisce buone prestazioni di generalizzazione, anche quando viene applicata rigida una divisione tra dati di apprendimento e di test.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my family first, mum, dad and my sister for the unbelievable patience and support they gave me all along this journey as a university student. Then all my old-time friends, for all the wonderful memories and moments we shared along almost our entire life, in particular Lorenzo, Nicola, Ismaele, Lorenzo, Marco, Stefano, Joaquin and Alessio. Then to all friends i have met during the university years, special people with who i shared other great moments and memories. Then all parents, uncles, aunts, cousins and grandma. Thanks to Max and Michele for the precious help they provided during the thesis development and draft review. In the end some unusual acknowledgements. Thanks to my guitars for always being there in my loneliest and saddest moments, to Eric Clapton, Mark Knopfler and David Gilmour, for their inspiration and their music, that has been so helpful and healing along these years.

Chapter 1

Introduction

The current scenario concerning music fruition is characterized by the presence of wide music catalogs ubiquitously and instantly available to anyone through many services. Due to the size of such catalogs, users are not able to handle the amount of available musical information on their own. As a consequence, services must assist users according to their information needs. The possible user needs are *music retrieval*, *music browsing* and *music recommendation* [16]. With *music retrieval* we refer to the scenario in which the user has a specific music information need, that expresses by means of a query [16]. The query can be expressed either as a text or a song or a small piece of audio, and the system takes care of retrieving songs or musical items that match the query. With *music browsing* we refer to the scenario in which the user has an undirected need. The user does not want to retrieve a specific musical item but wants to explore the available musical collection. With *music recommendation* we refer to the scenario in which the user asks the system to filter the available musical collection in order to detect relevant items based on his/her preferences or actions [16].

All of the three scenarios rely on the concept of music similarity in order to satisfy user's needs. The definition and evaluation of such concept differ for each scenario. In the *retrieval scenario*, when the query is represented by a song or a piece of audio, the interest is focused on retrieving items that perfectly match the query. In the *browsing scenario*, songs within a catalog need to be organized according to their similarity. In particular, similar songs need to be close to each other while dissimilar songs need to be far apart. In the *recommendation scenario*, the system exploits either informations about user's preferences or the information coming from a reference song given by the user [16].

In the retrieval scenario, the similarity definition is objective, as the interest is focused on finding a perfect match [16]. On the other hand, for both browsing and recommendation scenario the definition of similarity is influenced by user's similarity perception. The modeling of *subjective similarity* perception is a complex task [38]. We have to consider that music is a rich and multi-faceted type of information. Therefore, it can be evaluated and perceived by means of several standpoints, to which people can assign more or less importance, according to their preference, taste, musical knowledge and experience [16]. At this point, two problems arise. One concerns the definition of a technique able to represent songs as a multi-faceted information type of information. The other concerns the definition of a function able to evaluate song similarity in a personalized way, i.e. according to the similarity perception of a specific user.

The research field able to characterize the musical information is *Music Information Retrieval* (MIR). MIR is a multidisciplinary research field that concerns the extraction, analysis, and usage of information coming from music [16]. In order to fully characterize musical information, MIR encompasses a wide variety of knowledge from others research communities. MIR community has proposed several approaches for evaluating song similarity. The existing approaches can be divided according to two criteria. One concerns the type of data such approaches rely on in order to represent songs, the other concerns the type of defined similarity function. According to the type of data, existing approaches can be divided into *Collaborative Filtering* (CF), *Context-based* and *Content-based.* Depending on the type of similarity function, approaches can be divided into *personalized* and *non-personalized.* Personalized approaches aim to learn similarity functions able to fit the user's characteristics, while non-personalized approaches aim to learn objective similarity functions.

Both CF and Context-based techniques rely on data provided by users in order to assess music similarity. In particular, CF exploits the information related to the listening habits of users in order to infer song similarity, while Context-based approaches infer song similarity exploiting the mechanism of tagging. Tagging is the process through which users assign semantically meaningful keywords to musical entities in order to characterize them. Both methods have proved to outperform non-personalized approaches in modeling similarity, whenever data they rely on are available [24]. However, as shown in section 2.1, they suffer from cold-start problems and scalability issues. Cold-start refers to the initial lack of information for new songs, which makes not possible to draw any information about similarity. With scalability issue we refer to the inability of these approaches to cope with an expanding amount of data.

As opposed to CF and context-based techniques, content-based methods are always able to evaluate song similarity as they rely on the information extracted from audio content. Content-based techniques evaluate song similarity by means of descriptors able to capture musical properties. Such descriptors (or *features*) are computed from the audio track by means of signal processing and machine learning techniques and can be categorized in three basic levels according to their semantic power [16]: *low-level*, *mid-level* and *high-level*. *Low-level* features refer to descriptors that are inferred directly from the audio content, but their semantic power is very low, as their meaning is far from the way people think of and perceive music similarity. Indeed, the meaning of low-level features can fully understood only by a signal processing expert [16]. Some examples of low-level features are Spectral Centroid (SC) and Zero Crossing Rate (ZCR) (section 3.1.2). *Mid-level* features refer to descriptors that can be obtained with the combination of several low-level features using prior knowledge on musical theory. The combining process can be empowered with the adoption of some psychoacoustic-related information [16]. Mid-level descriptors have more semantic than low-level descriptors, but are understandable by music experts or, more in general, by people having musical background or experience. Mid-level descriptors represent properties related to timber, rhythm and dynamics [10]. *High-level* features refer to those descriptors able to capture musical properties used by people in order to describe and perceive music such as instrumentation, genre, and mood [16]. In order to provide such meaningful descriptors, both low- and mid-level descriptors are combined by means of machine learning techniques.

Most of the content-based approaches evaluate song similarity considering solely information provided by descriptors. As these methods do not include any user-related similarity information within the process of similarity evaluation, they are non-personalized. However, there exist also content-based approaches that include user-related similarity information (like [53], [41], [44], and [45]) in order to learn a similarity function tailored on user's similarity opinions. All of these methods assume that there exists a linear relationship between descriptors and the user's perception of similarity. As a consequence, they develop several techniques able to learn a linear similarity function tailored on user's perception. As music similarity evaluation is a complex phenomenon and music is a context where non-linearity is definitely present and relevant ([38], [4]), linear similarity functions may not be appropriate for representing the subjective perception ([38], [4]).

The goal of this thesis is to extend existing content-based and personalized approaches. We introduce a method that combines both *content-based information* and *user-related data* but aims to learn a *non-linear* similarity model reflecting user's similarity perception. Our objective is to learn a non-linear mapping from descriptors to a low-dimensional spatial representation reflecting the subjective similarity of the user. We propose a *two-stage* approach that combines a non-metric scaling technique (*t-distributed Stochastic Triplet Embedding*, or t-STE), a feature selection algorithm and a machine learning technique (*Support Vector Regression*, or SVR), empowered by a non-linear kernel function. The two-stage formulation of the method aims to first learn, by means of t-STE, a low-dimensional space shaped according to user's perception. Then we aim to learn a non-linear mapping between the descriptors and the low-dimensional space by combining the kernelized version of SVR with the feature selection algorithm. The objective of the feature selection algorithm is to identify the feature subset that most enhances the generalization properties of the mapping. This is due to the fact that, once the learning phase is completed, we want to exploit the mapping in order to map new songs within the low-dimensional space, without having to re-train the method. Once the best feature subset is identified, the kernelized version of SVR takes care of learning the mapping between descriptors belonging to the optimal subset and the low-dimensional space. Once the training phase is ended, we obtain a mapping that allows to map new songs within a user-tailored space.

Content-based information is summarized by a wide set of low-level descriptors. According to [4], low-level descriptors are more suitable to model user preferences than high-level descriptors. But, in order to fully characterize the many facets of music, we need many descriptors in order to capture many musical properties. As user-related similarity information, we collect for each user a set of opinions about similarity between pairs of songs. Such data represent the similarity perception of the user and are needed in order to elaborate the user-tailored space. The effectiveness of our method relies on two facts. First, the learned low-dimensional space is generated from similarity assessments expressed by the user himself, thus the entire procedure is tailored on a single user. Then, SVR supports kernelized regression. By means of non-linear kernel functions, it is possible to perform linear regression in a new feature space induced by the kernel, in a way that it corresponds to perform non-linear regression in the original feature space. This allows to increase the flexibility of the similarity model, hence allowing to capture more complex relationships between descriptors and subjective similarity.

Our method extends the work proposed in [38]. The author of [38] adopts as user-related data a set of similarity scores between pairs of songs, thus resolving to a numerical similarity information. With regard to [38], we opt for translating the numerical information into ordinal similarity comparisons. According to [1] and [51], such type of similarity information is more reliable with respect to numerical information. Indeed, different users will likely use different internal scales to asses similarity. In addition, this setting of data gathering appear to be more natural for users [1]. In order to evaluate the generalization properties of our method, we conducted two types of experiment, each with a different data division strategy. Both experiments proved that the mapping learned by our method is able to map new songs in the low-dimensional space according to the similarity assessments concerning such new songs. Thus, our method provides good generalization properties independently on the adopted data division strategy.

Thesis outline In Chapter 2 we review the state of the art of music similarity modeling, showing the basic categories of similarity models and the most prominent methods of each category. Chapter 3 contains the theoretical background of our method: we discuss features characterization and the theoretical details of the techniques involved in our method. In Chapter 4 we describe in detail our similarity modeling approach, while Chapter 5 contains all the related experimental results. Chapter 6 draws some conclusions and presents some possible future developments .

Chapter 2

State of the art

In this chapter we review the state of the art concerning the modeling of music similarity. There are three main approaches to model music similarity: *Collaborative Filtering* (CF), *context-based* and *content-based. Collaborative Filtering* (CF) (section 2.1) relies on the assumption that similar users listen to similar songs. The similarity among users is inferred directly from their listening habits. Consequently, a similarity among songs is inferred by the amount of users who have listened both. *Context-based algorithms* (section 2.1) exploit contextual information assigned to songs in the form of tags. Tags are meaningful keywords that can be manually attributed to songs and give some sort of information about these latter: genre, velocity, associated emotions, listening context, country, etc. *Content-based approaches* (section 2.2) are methods that evaluate song similarity exploiting information extracted directly from the audio signal.

CF and context-based approaches are strictly related, since they both rely on user-related data in order to infer song similarity. For the same reason, they also are often combined. Content-based methods exploit content-related information in order to assess music similarity. In order to provide a personalized similarity metric, they need to include user-related similarity information. *Personalized* content-based approaches exploits user-related information, while *non-personalized* approaches do not, as they aim to learn objective similarity metrics.

2.1 Collaborative and Context-based models

Collaborative filtering (CF) is a strictly personalized approach to music similarity modeling, since it relies on information provided by the interactions of users with respect to musical tracks. Specifically, CF approaches collect for each user information about which songs he/she has listened to. The gathered information is then represented with a User-Song representation or a User-Artist representation [16]. Both representations are expressed as matrices in which rows constitute user profiles and columns represent songs (or artists).
Collaborative methods are often combined with *context-based techniques*, providing *hybrid models* for music similarity modeling ([29], [31], [15]). This is due to the fact that both approaches define song representations exploiting data coming from users. As CF exploits data concerning the listening activity of users, context-based models exploit the mechanism of tagging. Tagging is the operation of annotating a song with a set of meaningful keywords able to characterize its content [16].

When employed within large-scale systems for music recommendation, both CF and hybrid models approaches do not scale well [38], i.e. they are not able to properly handle an increasing data amount. Indeed, the memory consumption grows with the square of the number of users and songs in the system. In order to solve this issue, matrix factorization techniques (like *Singular Value Decomposition*, or SVD) are usually employed. We will first show standard approaches that do not rely on matrix factorization, and then matrix-factorization-based methods.

2.1.1 Standard approaches

The authors of [33] evaluate song similarity performing co-occurrence analysis of songs within user collections. They gather listening informations adopting a User-Song matrix and exploit it in order to characterize songs as numerical vectors of co-occurrences. Song similarity is assessed as the correlation between the co-occurrences vectors. An approach strictly related to [33] can be found in [3]. The authors present a method that perform co-occurrence analysis on user playlists. With regard to [33], they propose several alternatives to build the song representations, as well as several criteria for the similarity evaluation of the various song representations.

Also the work presented [18] relies on a User-Song matrices data structure. It computes song similarity exploiting the information coming from a former procedure of similarity estimation between users. For each user u, a set of k similar users is retrieved according to the correlation between user profiles. Then, for each user u, a list of novel songs considered similar to the examples within its collection is retrieved exploiting the information coming from the user profiles of the k most similar users.

2.1.2 Factorization-based approaches

CF methods suffer from scalability issues as well as from data sparsity. This is due to the fact that it is not possible to gather information for each user-song pair, as the number of users and songs continuously grows in time [16]. In order to cope with these issues, CF methods often employ some kind of matrix factorization technique, as SVD (*Singular Value Decomposition*). With SVD it is possible to decompose a given User-Song matrix in a way such that users and songs are represented in terms of latent semantic concepts inferred from the original matrix and ordered in terms of relevance. Standard SVD works only with dense data matrices, but many different variations were proposed in order to deal with data sparsity ([17], [56]).

The authors of [50] present three hybrid models for predicting song similarities employing both a User-Song matrix and a User-Tag matrix. The objective of each of the three models is to present the user with a list of novel songs considered similar to examples within his/her collection. The novelty brought by these hybrid models consists in the fact that they allow to estimate song similarity fusing the information derived from the user's listening activity and the information related to tags that the user have given to musical items.

The work in [48] exploits purely CF data and matrix factorization as well. The method first elaborates a between-song correlation matrix exploiting only data coming from the listening activity of users and then decomposes it adopting a variant of SVD. Similarity between songs is estimated by evaluating the similarity between song representations induced by the decomposition.

The authors of [29] propose an hybrid CF-Context method to model the information coming from social tagging data with 3-order tensors, thus able to capture cubic correlations between a triple $< \text{users}, \text{tags}, \text{music items} >$. The discovery of latent structures in this model is performed with a variant of the *Singular Value Decomposition* (SVD), denoted as *Higher Order Singular Value Decomposition* (HOSVD).

Pure collaborative models (model relying only on CF data) has shown to provide good performances, since they rely on data which are directly linked to user's musical preference. However, they suffer from some disadvantages [16]. One issue is the *"gray sheep issue"*: if we compute song similarity exploiting information on between-user similarity, a user having uncommon preferences that differ from the great majority of users is unlikely to find similar users. Consequently, he will receive poor similarity predictions. [16]. Another issue is the *popularity bias*: if a song is very popular, it is going to be evaluated as similar to many other songs, and it is likely to be suggested more frequently by a recommender system. CF approaches also suffer from *cold-start* issue [16], which concerns the fact that it is not possible to draw any inferences for upcoming users or songs, since there is no information available. In addition, collaborative filtering relies only on user-related information. This could be an advantage in terms of flexibility and adaptability, but audio characteristics of songs are not considered at all. Thus, the similarity evaluation is independent on the similarity of song content [38].

Using tags for evaluating music similarity shows some drawbacks as well[16]. First, also context-based approaches suffer from both popularity bias and cold-start. The less popular songs risk not to receive enough keywords to be completely characterized, due to the fact that most of the users do not know them. Another issue concerns the semantic nature of tags: *polysemy* (words whose meaning changes according to the context), synonymy, different syntactical inflections of the same word (e.g. hip-hop and hiphop), and misspelled words can compromise the modeling task; natural language processing techniques need be applied to cope with this issues [16].

As opposed to CF and context-based methods, the main advantage of content-based models is that the similarity definition can be always computed, as these approaches do not rely on the contribution of users. Indeed, the similarity evaluation relies on the information extracted from audio contents of songs. Consequently, we decide to employ a content-based approach in order to evaluate music similarity.

2.2 Content-based models

Content-based approaches describe songs in terms of their audio characteristics. They exploit specific algorithms whose task is to process an audio track and to extract from it a set of descriptors able to capture and quantify musical properties. The set of descriptors is denoted as *feature vector.* In order to characterize the content of a song, we can adopt either low or mid- or high-level descriptors [16]. Low-level descriptors are extracted from the audio content of songs, but represent musical properties that are far apart from the concepts users adopt to evaluate similarity. Mid-level and high-level descriptors are usually obtained by combining several low-level features by means of machine learning techniques. As a consequence, they are characterized by a higher semantic power, i.e. their meaning is much more understandable by a user [38].

Content-based techniques can be divided into two main categories: *personalized* and *non-personalized.* Non-personalized techniques evaluates song similarity taking into account solely the information coming from audio contents, while personalized techniques incorporate user-related similarity information in order to compute a similarity measure that fits the similarity perception of the user.

2.2.1 Non-personalized approaches

Non-personalized music similarity modeling consists in creating distributions of features extracted from the audio, approximating these distributions with probability density functions, and evaluating song similarity adopting pairwise-distribution distance metrics.

The technique described in [2] first divides each song into musical phrases, i.e. fragments showing a continuity of audio content correlation. Then it extracts a series of MFCCs (*Mel Frequency Cepstral Coefficients*) for each phrase and models the MFCCs distributions using multivariate Gaussian models. The similarity of two songs is evaluated by means of the *Kullback-Leibler* distance between the generated distributions.

A similar approach to [2] is described in [22] and [3]. In [22], the series of MFCCs coefficients are clusterized. Then, informations related to the clusters (mean and covariance) are exploited in order to represent songs. Song similarity is then evaluated according to the *Earth's Mover Distance* (EMD) between song representations. In [3], the authors propose a different criterion (*Asymptotic Likelyhood Approximation*, or ALA) as a measure of distance between distributions.

The greatest drawback of non-personalized algorithms is the fact that they elaborate an objective similarity measure, as [2], [22], [3] show. Since they entirely rely on sets of objective features, they are unable to capture the subjective similarity perception of the user. In order to elaborate a personalized similarity model, it is necessary to include user-centric data into the method, thus resolving to a personalized approach [16].

2.2.2 Personalized approaches

Most of the personalized and content-based methods fall within the category of *Metric Learning algorithms.* Metric learning is a supervised machine learning field whose objective is to learn a subjective similarity measure that linearly depends on the set of song descriptors. Generally, the linear relationship between descriptors and similarity is computed by means of optimization techniques whose aim is to ensure that some predefined similarity constraints are respected. Such constraints are provided by users themselves. Specifically, the goal of metric learning is to learn a suitable matrix $\mathbf{W}$ such that the distance $d(\mathbf{x}, \mathbf{y})$ between two songs x and y characterized by feature vectors $\mathbf{x}$, $\mathbf{y}$ can be computed as

$$d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{W} (\mathbf{x} - \mathbf{y})$$

which is, essentially, a weighted Euclidean distance if we constrain $\mathbf{W}$ to be diagonal but different from the identity matrix. On the other hand, allowing $\mathbf{W}$ to be a full matrix, we accept some interactions between features.

The author of [41] compares several algorithms for learning $\mathbf{W}$. Some of these are based on second-order statistics (*Whitening*, *Linear Discriminant Analysis* or LDA, *Relevant Component Analysis* or RCA), others on procedures of optimization based on neighborhood relationships among songs (*Neighborhood Component Analysis* or NCA, *Large-Margin Nearest Neighbor* or LMNN).

Also the work in [53] compares some metric learning algorithms. The learning processes showed in the paper are driven by sets of pairwise-song similarity constraints provided by users themselves. In particular, some variants of both *Support Vector Machine* (or SVM) and *Metric Learning to Rank* (MLR [26]) are presented.

The authors of [24] propose a CF-based method for enhancing the performances of a non-personalized content-based method for song similarity evaluation. First, a User-Artist matrix is built. Then, the authors compute $\mathbf{W}$ through a constrained optimization procedure. The constraints derive from data gathered via CF, and concern the similarities of a generic song and the songs within the related set of relevant songs. The set of relevant songs is computed according to the algorithm described within [24]. Specifically, the constraints specify that the similarity between a generic song s_1 and another belonging to its "relevant set" has to be greater than the similarity between s_1 and another song not belonging to the "relevant set".

The work in [25] is a further development of the one proposed in [24], by the same authors. They rely on a User-Song representation using the same approach of [24], and evaluate the performances of their method starting from a different content-based approach.

There exist also personalized approaches that do not belong to the field of metric learning. In particular, both the work in [44] and the one in [45] aim to model user's similarity perception as a function depending on many facets. A facet is a distance measure based on feature subsets able to characterize a musical aspect [44]. The overall similarity function between two songs is defined as

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} w_i \cdot d_i(\mathbf{x}, \mathbf{y})$$

where N is the number of considered facets, w_i ($w_i \geq 0, \sum_{i=1}^{N} w_i = 1$) is the weight assigned to $d_i(\mathbf{x}, \mathbf{y})$, which is the distance operating on the $i - th$ subset of features. The objective is to learn a set of weights w_i such that the overall similarity function reflects the user's subjective similarity perception. To this purpose, the authors propose and compare many approaches relying on constrained optimization procedures. Among the revised optimization procedures, we find gradient descent, quadratic programming, maximal margin classifier, together with different slack formulations to allow violated constraints [44] [45].

The author of [43] presents a technique whose goal is to identify which subset of features more accurately approximates the similarity perception of the user. Once a song has been analyzed (and the relative features extracted), the technique employs a set of neural networks, each of which will generate a similarity measure involving a particular subset of features and return the most similar song according to the learned measure. The user evaluations about returned songs are used to refine the learning processes of the networks.

All of these methods show some drawbacks. The existing techniques for metric learning aim to learn a linear relationship between descriptors and user's perceived similarity. The linearity assumption constitutes a limit to the metric learning procedure, as music similarity evaluation is a complex phenomenon and music is a context where non-linearity is definitively present [38]. Both the work in [25] and the work in [43] rely on non-linear models for learning a personalized metric, but they have never been tested in a personalized way. In other words, their performances have never been evaluated on user-wise base, so their ability to fulfill a subjective similarity is not measured.

The objective of this thesis is extend the current scenario of personalized content-based approaches, and thus to provide a method able to detect potentially non-linear relationships between features and similarity. The work described in [38] constitutes the grounds of our proposed model. The objective of [38] is to learn a non-linear relationship between a set of descriptors and user subjective similarity. To do so, the author proposes a a two-stage personalized and non-linear method that combines a *metric Multidimensional Scaling* technique, a regression technique (SVR) and a feature selection algorithm. The two-stage formulation aims to first learn a low-dimensional space (or *embedding*) shaped by a set of user-related similarity constraints and then learn a non-linear mapping between a subset of acoustic features (identified by the feature selection algorithm) and song coordinates within the embedding. In order to detect non-linear mappings, the regression procedure is kernelized.

The author of [38] employs *numerical constraints* related to pure distances between songs in order to drive the learning process of the embedding. The distance between songs are obtained converting the similarity score provided by the user for the related pair of songs. However, as stated in [1], [8] and [53], pure numerical constraints are not suitable when trying to model similarity perception. This is due to the fact that different users will likely use different internal scales to asses similarity and that the magnitude of the input dissimilarities is often unreliable or too difficult to measure. On the other hand, *ordinal constraints*, i.e. constraints of the kind “A is more similar to B than to C”, are more straightforward to annotate and easy to obtain, even when the distance information is expressed through similarity scores between pairs of songs ([53], [1]). Due to these facts, we decide to include in our approach a *non-metric Multidimensional Scaling* technique as opposed to the metric technique of [38]. Indeed, non-metric scaling techniques employ ordinal similarity constraints as input for the scaling procedure. They preserve the ordinal relationship between distances rather than the pure distance values.

Chapter 3

Theoretical background

In this chapter we provide the theoretical details for the data and the techniques our model employ. We start with a general definition and characterization of MIR features. We then describe in detail each feature employed within our song representations, and then describe the measures for computing the distances between song representations. Afterwards, we show the foundations of the methods we adopted within our model: (1) *t-Distributed Stochastic Triplet Embedding* (t-STE) (2) *Support Vector Regression* (SVR) and adjusted R^2-based *feature selection.*

3.1 Features

3.1.1 Short-Time Fourier Transform

The *Short Time Fourier Transform* (STFT) is the mathematical tool used to transform a signal from the time domain to the frequency domain. STFT is meant to provide a local characterization of the frequency content of the audio track. In STFT, the signal to be transformed is divided into frames and multiplied by a finite-length window. For a signal x, the STFT is defined as

$$X_k{}^{(m)} = \sum_{n=0}^{L-1} x_{n+m\cdot N_h} \cdot w_n \cdot e^{-j\frac{2\pi kn}{L}},\ k = 0, 1, ..., L-1$$

where n is the sample number (i.e. x_n is the $n-th$ sample of x), k is the frequency bin, w_n is the weight of $n-th$ sample of x (according to the type of window involved), L is the window length, m is the frame number (that determines the position of the window), and N_h is the hop-size expressed in number of samples. Frames can be made overlapping, to reduce artifacts at the boundaries, by setting $N_h < L$.

The length of the window influences both time and frequency resolution. The wider the window length, the higher the frequency resolution and the lower the time resolution. The relation holds also if considered vice versa.

3.1.2 General feature characterization

MIR features are numerical descriptors, extracted from audio, able to capture and quantify musical properties. They can be categorized according to their *level of abstraction*. The abstraction level indicates the semantic power of the descriptors [16]. Thus, the higher the level of abstraction, the closer they are to the concepts users adopt for describing and evaluating music similarity. On the other hand, the higher the level of abstraction, the less objective the descriptors [38].

The level of abstraction of a feature is typically described by a scale of three levels: *low level*, *mid-level*, and *high level*. *Low-level features* are computed directly from the signal representation, but they represent non-trivial audio characteristics. *Mid-level features* capture aspects that are more musically meaningful, such as note- or beat-related properties, but their interpretation requires some musicological background. Corresponding mid-level feature extractors are frequently computed combining sets of low-level features [16]. On the highest level, we find features that are closer to the way people think of and describe music. Indeed, *high-level features* are comprehensible also by the average user. As well as mid-level features are obtained combining low-level descriptors, high-level descriptors are obtained through the combination of several low- and mid-level features.

Low-level features

Low-level features are directly computed from the representation of the audio signal in either the time domain or the frequency domain. All of the features are usually computed on the frame level, but it is possible to aggregate all values of a particular feature with respect to the full signal. This can be done by computing some statistical summarization function (such as mean or median value) or adopting probabilistic feature aggregation methods (like Gaussian mixture model or Maximum Likelyhood model [2]). Some examples of low-level features are the *Spectral Centroid* (SC) and the *Zero Crossing Rate* (ZCR). SC is a frequency-domain-related feature that represents the center of gravity of the magnitude spectrum, i.e. the frequency band where most of the energy is concentrated. This feature is used as a measure of brightness of a sound. On the other hand, ZCR is a time-domain-related feature that measures the number of times the amplitude value changes its sign within the current frame. It is used to detect percussive sounds and noise as well as an indicator of pitch for monophonic music signals, since higher ZCR values typically correspond to higher frequencies. Low-level features represent the most basic informations that can be extracted from audio. However their expressive power is very limited, as they represent musical properties that are far apart the way people evaluate song similarity. Thus they are understandable only by a person with a solid background on signal processing.

Mid-level features

Mid-level features are a combination of or extension to low-level features that incorporate musical theory, although boundaries between the abstraction levels are fuzzy and a strict categorization is difficult to make. Mid-level representations are considered closer to the concepts adopted by users to describe music, therefore allowing more meaningful comparisons between musical entities [10]. Mid-level features provide a more understandable representation of the musical content, and constitute a particular interest for people holding a musicological background. Among mid-level descriptors, we find features that represent properties related to timber, rhythm and dynamics [10]

High-level features

High-level features represent musical properties characterized by a high semantic level. They represent concepts whose meaning can be understood by the average listener [16], as they are designed to fill the semantic gap between low-level objective descriptors and the way people think of and perceive music similarity. Indeed, high-level descriptors are able to represent properties as instrumentation, genre, melody and mood [16]. The boost in the explanatory power of the features, however, comes to a cost. Indeed, they require a more complex elaboration, as they are computed by means of machine learning techniques that combine both low- and mid-level descriptors.

3.1.3 Feature details

In this section we provide the details about the features we chose for our work. We employ the same set of low-level descriptors used by [38], whose work constitutes the grounds for our model. We consider a wide set of features of different nature. This is due to the fact that music is a multi-faceted type of information and many low-level descriptors need to be employed in order to provide a suitable representation [4]. In particular, we employed four different types of low-level descriptors. *Energy-related* descriptors capture information about the energy distribution and evolution within a song. *Temporal* features analyze time-related audio aspects. Depending on the type of temporal information to be detected, they operate either on the original signal either on its spectral representation. *Spectral* features represent concepts that come from the spectrum analysis of a song, while *waveform-related* descriptors represent audio characteristics extracted from the raw song waveform, without any previous processing. We present the complete feature set according to such categorization.

The low-level descriptors we employ have different temporal scopes. The temporal scope of the features determines the way they are processed once they have been extracted from audio tracks. We follow the same temporal characterization used by [38]. Basically, we identify three main feature value types according to temporal scopes. *FrameSingleValued* and *FrameMultipleValued* features are computed on each frame, after signal windowing. The first has only one real value for each frame, while the second defines an entire vector per frame. Finally, *SingleValued* features consist of a single value computed on the entire track.

We now present the complete feature set, specifying the type of the descriptor as well as the temporal-scope related category of the descriptor. For FrameSingleValued- and FrameMultipleValued-type descriptors, all the computations are meant on single frames.

Energy-related features

Intensity ratio Value type: *FrameMultipleValued*. According to [21], if the signal is divided into several sub-bands, the Intensity ratio refers to the ratio between the sub-band's intensity and the overall intensity I. The intensity I is computed by summing all the amplitudes of the frequency components within the spectrum.

Loudness Value type: *FrameSingleValued.* The loudness is the characteristic by means of which music can be ordered on a scale extending from quiet to loud. Its value is computed with an approach built on psycho-acoustical theories that explain how the human ear perceive sounds [5].

RMS energy Value type: *SingleValued.* Root Mean Square (RMS) energy is a simple measure of the overall mean energy contained in a song.

Low energy Value type:*SingleValued.* According to [36], the signal is first segmented into frames. The low energy ratio is the percentage of frames whose RMS energy falls within a fixed threshold represented by the overall mean RMS energy.

Temporal features

BPM Value type: *FrameSingleValued.* It represents the estimated number of musical beats in each frame, expressed in beat per minute. The procedure for the estimation of the number of beats ([7], [6]) is composed of an onset analysis stage, used to derive a representation containing rhythm-based information, and a beat matching algorithm that computes short-term predictions of future beats in the audio.

Key mode Value type: *FrameSingleValued.* The descriptor represents a major-mode coefficient, given by the ratio between the time during which the song is estimated to have a major mode and the time during which it is estimated to have a minor mode. We employ the method depicted in [30], which allows to analyze the track and continuously estimates its key.

Onsets Aubio Value type: *FrameSingleValued.* The descriptor consists of several timecodes (or time instants) related to onset times within the audio signals. With onset times we refer to the beginning of discrete sound event [23].

Onsets OnsetsDS Value type: *FrameSingleValued*. Such feature represents the onset likelihood of the frame, computed adopting the Adaptive Whitening method employed in [46].

Onsets Queen Mary Value type: *SingleValued*. Similarly to the previous descriptor, such feature represents the onset likelihood of the frame. This time the likelihood is estimated using the algorithm in [12].

Tempo Value type: *SingleValued*. It represents an estimated overall tempo value (BPM) for the song. The algorithm in [12] calculates an overall energy rise function, locates peaks in the related auto-correlation function and converts from auto-correlation lag to the corresponding tempo.

Spectral features

Crest Value type: *FrameSingleValued*. A descriptor related to the flatness of the frame spectrum, i.e. to the noisiness-harmonicity ratio of the related signal [12].

Irregularity K Value type: *FrameSingleValued*. The feature measures the irregularity of the spectrum harmonics, which is empirically related to a perceived inharmonicity in the sound [19].

Irregularity J Value type: *FrameSingleValued*. Similarly to the previous descriptor, this feature is related to the variation of successive harmonic components of the spectrum [19]. Thus it represents a further indicator of sound inharmonicity.

MFCC coefficients Value type: *FrameMultipleValued*. Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound that relies on a non-linear psychoacoustic-based scale of frequencies (known as mel scale). Mel-frequency cepstral coefficients (MFCCs) are the coefficients that collectively summarize a MFC [40]. Generally speaking, MFCCs concern timbra-related characteristics of tracks. MFCCs are commonly derived as follows: first, the Fourier transform of each signal frame is taken. Then, the obtained spectrum is filtered using a Mel filter bank; afterwards, the logs of the powers at each of the Mel frequencies are computed.

Finally, the discrete cosine transform (DCT) is applied to the list of Mel log powers. The MFCCs are the amplitudes of the resulting spectrum. The procedure is represented in figure 3.1. Considering a psycho-acoustic scale leads to a better approximation of the auditory system, allowing a better modeling of the human timbre perception.

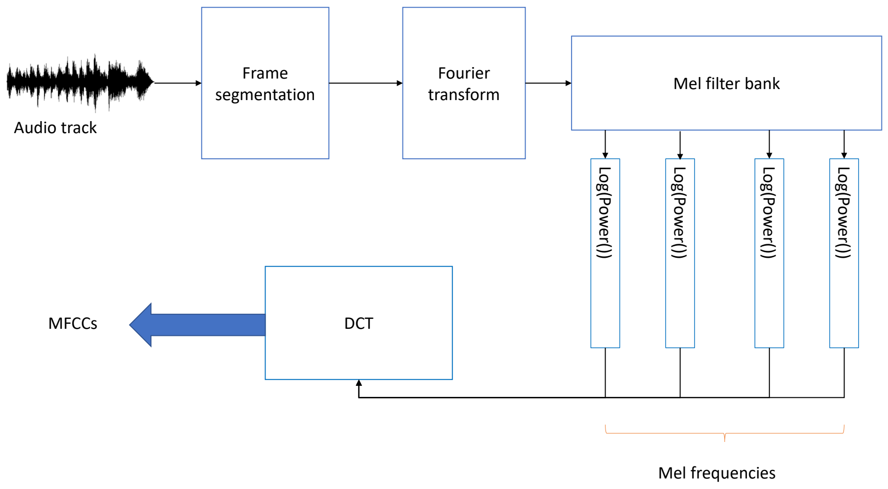


Figure 3.1: Overview of the MFCCs computation process.

Odd-even ratio Value type: *FrameSingleValued*. The odd-even ratio is defined as the ratio between odd and even harmonics of a general fundamental frequency.

Rhythm (Strength, Average onsets frequency, Auto-correlation, Mean correlation peak, Peak-valley ratio, Tempo) Value type (all): *FrameSingleValued*. The aim of this set of descriptors is to capture several characteristics able to define the rhythm of a song. The final goal is to detect the tempo (BPM) of the song, but intermediate steps provide useful information as well. We compute an *onset curve* adopting the algorithm in [21]. From the onset curve it is possible to define all the other parameters. In particular, the *rhythm strength* is the mean value of the peaks of the onset curve. The *average onsets frequency* is the total number of onsets divided by the length of the track in minutes. In addition, from the onset curve an *auto-correlation* signal is also computed, as this latter is used to detect peaks. The mean value of the selected auto-correlation peaks constitute the *mean correlation peak*, while the *peak-valley ratio* is the ratio between the mean correlation peak and the mean value of the valleys. Finally, the *tempo* is defined as the maximum common divisor of the detected peaks.

Rolloff Value type: *FrameSingleValued.* The feature represents the minimum frequency value such that a given percentage R (usually 95 %) of the spectrum energy stays below that frequency. It is a measure of the brightness of the sound [32]: the higher the rolloff, the more high-frequency components are present in the spectrum, denoting a brighter sound.

Spectral Centroid Value type: *FrameSingleValued.* Spectral centroid (SC) basically represents the center of mass of the spectrum [36]. More generally, it indicates if the spectrum is mostly composed by low or high frequency components, which it is often related to the perceived sound brightness.

Sharpness Value type: *FrameSingleValued.* Sharpness is the perceptual equivalent of the spectral centroid and it is based on psycho-acoustical models. It is computed according to the approach depicted in [28].

Spectral inharmonicity Value type: *FrameSingleValued.* The descriptor is a measure of the divergence of the spectrum components from the multiples of a detected fundamental frequency. The value obtained according to [36] represents an indicator of how much a sound is inharmonic.

Spectral contrast (mean, peak, valleys) Value type(all): *FrameMultipleValued.* Spectral Contrast represent a measure that reflects the distribution of the harmonic and non-harmonic components within the spectrum. According to the method in [14], the track is segmented into overlapping frames; then the spectrum is computed and filtered by an octave-scale filter that divides it into several sub-bands. For each of these bands, peak and valley are detected. For each sub-band, the spectral contrast is computed as the mean difference between peaks and valleys.

Spectral flux Value type: *FrameSingleValued.* Spectral flux is a measure of the change in energy between various frequency bands along a sequence of spectra measured from the audio data. As stated in [11], we consider only the positive values in the spectral difference, setting negative values to zero and leaves positive values unaltered.

Spectral kurtosis Value type: *FrameSingleValued.* Kurtosis is a statistical-based measure to determine the flatness of a probability distribution around its mean value; when applied to a spectrum becomes an indicator of the signal noisiness [36].

Spectral skewness Value type: *FrameSingleValued.* Skewness is a statistical-based measure as well. It is an indicator of the asymmetry of the spectrum around its mean value [36]. A positive value means that the spectrum is skewed towards the right, thus showing a long tail on lower frequency components; with negative values, the spectrum is skewed towards the left; for perfect symmetry, the feature takes the value of 0.

Spectral slope Value type: *FrameSingleValued.* Spectrum components tend to decrease towards higher frequencies. The spectral slope gives the rate of descent of the spectrum, obtained by computing a linear regression of the spectral amplitudes [36]. According to the definition, higher values are expected for dark songs, as opposed to lower values for bright songs.

Spectral smoothness Value type: *FrameSingleValued.* Smoothness is related to the differences between adjacent spectral components a_k. It is computed by evaluating the log of a component minus the average of the log of the surrounding components [36].

Spectral variance Value type: *FrameSingleValued.* The basic variance of spectral amplitudes. It constitutes an index of the noisiness of the sound. A small spectral variance means that all the spectrum energy is concentrated around the same frequency, thus the produced sound cannot be considered noise. Contrarily, a flat and distributed spectrum is typical of noisy sounds [36].

Tristimulus (Tristimulus 1, Tristimulus 2, Tristimulus 3) Value type (all): FrameSingleValued. Basically, each of these 3 descriptor defines an energy ratio, that respectively account for the normalized amplitude of the fundamental, the mid-range, and high-frequency harmonic content. They are computed according to the method within [39].

Waveform-related features

Average deviation Value type: *FrameSingleValued.* Computes the average deviation of the signal within each frame, i.e. the mean of the absolute deviations of each sample within the frame from the frame-wise sample mean.

Kurtosis Value type : *FrameSingleValued.* Its definition is analogous to the definition provided for spectral kurtosis. The difference relies in the fact that now it is applied to the time-domain representation of the signal.

Noisiness Value type : *FrameSingleValued.* According to what stated in [34], noise can contribute to alter the timber of a sound, when combined to harmonic components. We estimate the noisy contribution by removing the portion of the signal that is predictable with a simple linear predictor and considering the residual as noise. The assumption is that each sample of the signal is predictable by a linear combination of the previous samples. Finally, obtained values are averaged, to produce a single value that quantifies the amount of noise present in the frame.

Power curve (Raw power, Power slope, Smoothed power, Smoothed power slope) Value type (all): *FrameSingleValued.* A set of descriptors related to the *power curve* of the track. The average *raw power* is computed for each frame to be then converted to decibels since hearing loudness occurs on a logarithmic scale [36]. The *power slope* is simply the slope of the power curve. The *smoothed power* consists of a smoothed version of the raw power data, computed in order to have a more perceptually relevant view of the power curve. The *smoothed power slope* is defined analogously the simple power slope.

Type	Features
Energy	Intensity ratio, Loudness, Low energy, RMS energy.
Temporal	Beat tracker, Key detector, Onsets Aubio, Onsets OnsetsDS, Onsets Queen Mary, Tempo.
Spectral	Crest, Irregularity J, Irregularity K, MFCC coefficients, Odd-even ratio, Rhythm, Rolloff, Sharpness, Spectral inharmonicity, Spectral centroid, Spectral contrast, Spectral flux, Spectral kurtosis, Spectral skewness, Spectral slope, Spectral smoothness, Spectral variance, Tristimulus, Tuning.
Waveform	Average deviation, Kurtosis, Noisiness, Power curve, Skewness, Variance, Zero-crossings.

Table 3.1: Summary of low-lovel descriptors and their types.

Skewness Value type : *FrameSingleValued*. It has the same definition of spectral skewness descriptor, but it is applied to frame samples.

Variance Value type : *FrameSingleValued*. It has the same definition of spectral variance descriptor, but it is applied to frame samples.

Tuning Value type : *SingleValued*. The feature represents an estimation of the global tuning of the songs, computed according to the method within [9]. Integer values of the descriptor map to pitches using standard Pitch Class notation. E.g. a key value of 0 states that the song key is C, while 1 indicates the song is in $C_\sharp/D_\flat$, 2 that the song is in D, and so on.

Zero-crossings Value type : *FrameSingleValued*. Such descriptor counts the number of times the signal changes its sign. It is an indicator of the presence of noisy sounds in case of high ZCR values, as opposed to periodic sounds which are characterized by small ZCR values.

Table 3.1 summarizes the wide set of low-level descriptors according to their type, while table 3.2 summarizes the temporal scopes of the descriptors and the values that each type of descriptor induce.

Temporal scope	Effect
FrameSingleValued	Defines one value per frame
FrameMultipleValued	Defines a vector of descriptors per frame
SingleValued	Defines one value for the entire song

Table 3.2: Summary of the different temporal scopes of descriptors.

3.1.4 Measure of distances

Once two songs have been represented with a feature vector we can evaluate their similarity. According to [38], the similarity between two songs represented by two feature vectors $\mathbf{x}$ and $\mathbf{y}$) is evaluated by means of the distance between the feature vectors. The lower the distance between two songs, the more these two share similar audio characteristics. Usually, similarity and distances are normalized, such that the more the distance is close to 0, the more the similarity is close to 1 (meaning high similarity between the songs) [38]. Examples of distance functions between vectors are: Euclidean (3.1), Cosine (3.2), Manhattan (3.3), Chebychev (3.4), and Mahalanobis (3.5).

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \tag{3.1}$$

$$d(\mathbf{x},\mathbf{y}) = 1 - \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||} \tag{3.2}$$

$$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N} |x_i - y_i| \tag{3.3}$$

$$d(\mathbf{x},\mathbf{y}) = \max |x_i - y_i|, i = 1, ..., N \tag{3.4}$$

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T S^{-1}(\mathbf{x}-\mathbf{y})} \tag{3.5}$$

where x_i,y_i are the $i-th$ entries of the related feature vectors. Each of these distance measures support a strategy for the weighting of attributes, in such a way that different levels of significance can be given to the features composing the song representation.

3.2 Multidimensional Scaling

Multidimensional scaling (MDS) is a machine learning subfield whose objective is to learn a low-dimensional spatial representation (or *embedding*) which aims to preserve the distance relationships of a set of objects originally represented in a higher-dimensional space [20]. The higher the number of dimensions of the learned space (which is usually estabilished a-priori depending on the application), the easier is to preserve the distance relationships [1]. Depending on the type of distance relationships that needs to be preserved, multidimensional scaling algorithms can be categorized in two classes: *metric scaling algorithms* (section 3.2.1) and *non-metric scaling algorithms* (section 3.2.2). The former aim to preserve the *pure pairwise distance* between objects, while the latter aim to preserve a set of *ordering relationships* between objects.

3.2.1 Metric scaling algorithms

The input of methods belonging to this scaling category, like the algorithm presented in [20], is a self-distance matrix (SDM) computed within the original multi-dimensional space (usually adopting an Euclidean metric, formula 3.1). The objective is to learn an embedding in which numerical distances represented by SDM are preserved. Metric algorithms accomplish the objective with an iterative process for the minimization of a convex function. The convex function expresses the quantity of distance shifts introduced by the operation of dimensionality reduction. The convex optimization function to be minimized in [20] is

$$\sigma(\boldsymbol{X}) = \sum_{i=1}^{N}\sum_{j=1}^{N}(d(i,j) - d(i,j)_{(\boldsymbol{X})})^2 \tag{3.6}$$

where N is the number of objects within the representation, $d(i,j)$ is the Euclidean distance between object i and object j in the original space and $d(i,j)_{(\boldsymbol{X})}$ is the Euclidean distance between object i and object j within the embedding $\boldsymbol{X} \in \mathbb{R}^{N\times D}$. The function (3.6) is named *stress function*.

3.2.2 Non metric scaling algorithms

Non-metric algorithms aim to preserve a set of *ordering relationships* between objects. Thus, the input of such kind of techniques is a set Θ of ordering constraints usually expressed as *4-tuples* (i, j, k, l)

$$\Theta = \{(i, j, k, l) \mid d(i, j) < d(k, l)\}$$

It is also possible to employ *triplets* (i, j, k), as opposed to 4-tuples. The meaning of the two type of constraint is strictly related, as triplets can be thought as a particular case of 4-tuples constraints for which we have $k \equiv i$ and $l \equiv k$ [1]. The set of constraints becomes

$$\Theta = \{(i, j, k) \mid d(i, j) < d(i, k)\}$$

In our model we employ *t-distributed Stochastic Triplet Embedding* (t-STE) [51], a non-metric scaling that exploits set of constraints expressed as triplets.

3.2.3 t-distributed Stochastic Triplet Embedding (t-STE)

Let us suppose we are provided a set of N data objects, for which there exists a function $s(\cdot)$ able to define the similarity for each pair of objects $i, j \in N$. On the other hand, we are provided a set Θ of similarity comparisons, which constitute a realization of $s(\cdot)$, i.e. given a triplet $(i, j, k) \in \Theta$, $s(i, j) > s(i, k)$. Such realization is potentially affected by noise, i.e. contradictions in the constraints, as they come from similarity judgements expressed by people. Consequently, the goal of t-STE is to learn an embedding $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, with D equal to the number of embedding dimensions, able to represent the underlying function $s(\cdot)$ rather than its realization Θ. To do this, t-STE evaluates the goodness of $\boldsymbol{X}$ with respect to $s(\cdot)$ by computing a probability for each constraint, that represents how well the constraint is modeled within $\boldsymbol{X}$ [51]. In particular, the authors propose to use a heavy-tailed *t-Student kernel* with α degrees of freedom in order to model triplet probabilities as

$$p_{ijk} = \frac{\left(1 + \frac{||x_i - x_j||^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\left(1 + \frac{||x_i - x_j||^2}{\alpha}\right)^{-\frac{\alpha+1}{2}} + \left(1 + \frac{||x_i - x_k||^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}$$

The objective is therefore to learn an embedding $\boldsymbol{X}$ by maximizing a function based on log-probabilities of the constraints within Θ (equation

(3.7)). The maximization process relies on *projected gradient descent method* [51], a first-order iterative optimization algorithm for finding the extrema of a multi-variable function [1].

$$\max_{\boldsymbol{X}} \sum_{\forall (i,j,k) \in \Theta} \log(p_{ijk}) \tag{3.7}$$

Such formulation of triplet probabilities is due to the success of unsupervised dimensionality reduction techniques that employ heavy-tailed kernels, as [37] and [52]. Indeed, the use of a heavy-tailed kernel does have a major advantage with respect to the use of a Gaussian kernel [47] to compute triplet probabilities from an embedding. The resulting formulation does more than simply satisfy the constraints within Θ. This is due to the fact that the tails of the t-student distribution are not flat, and therefore t-STE decreases the distance between i and j (similar objects), even when the related constraint (i, j, k) is already satisfied. Similarly, it increases the distance between i and k (less similar objects), even when the related constraint is already satisfied [51]. In other words, t-STE collapses points whenever there are no constraints keeping the points apart and it separates points whenever there are no triplets keeping the points together. In addiction, as opposed to other non-metric algorithms relying on the definition of constraint probabilities as [47], t-STE is able to handle the noise in Θ. Indeed, the formulation of triplet probabilities allows to identify constraints that heavily contradict the underlying functions $s(\cdot)$, i.e. that contradict several other constraints. As opposed to [47] or [1], t-STE does not concern the correction of heavily contradicting constraints [51].

3.3 Support Vector Regression (SVR)

Support vector machines (SVM) are supervised machine learning techniques that aim to learn models able to analyze data either for *classification* or *regression* purposes [42]. Data samples are characterized by set of descriptors (or variables) and are associated to response values. According to the application, response values can be either a discrete value (denoted as label) or a continuous value. If the response value is discrete, we learn models for classification purposes, i.e. models able to predict labels for unseen data samples. If the response value is a continuous value, we learn models for regression purposes. These models are able to quantify the response values for unseen data. Indeed, regression techniques aim to discover the existing relationship between data descriptors and response values [42]. SVR belongs to the field of SVM-based regression techniques.

However, it may happen that not all the predictors have a statistical significance with respect to the response values. Or, equivalently, not all the predictors carry useful information in terms of prediction capability. Consequently, the set of predictors (or features) is usually preprocessed so that only informative feature are considered in the regression process [27]. Preprocessing techniques concerning the selection of descriptors fall within the category of feature selection algorithms.

3.3.1 Feature selection

In many machine learning settings, feature selection, also known as variable selection, is combined with regression techniques [27]. Feature selection is the process of selecting a suitable subset of features from an original set of features. Descriptors are selected such that the generalization performances of the associated predictive model are enhanced. Feature selection techniques are used for reasons as the simplification of complex models, in order to make them easier to interpret by researchers, and avoiding the *curse of dimensionality* [27]. The *curse of dimensionality* is a well-known phenomenon in the community of machine learning. Its basic effect is the following: when the data dimensionality increases, the volume of the representation space increases so fast that the available data become sparse. Sparsity is problematic for any method that requires statistical significance [27]. Indeed, data organization and research often relies on detecting areas where objects form groups with similar properties; in high dimensional data, however, all objects appear to be sparse and dissimilar in many ways, which prevents common strategies from being efficient. Feature selection is different from dimensionality reduction. Both methods aim to reduce the number of descriptors in the dataset, but dimensionality reduction methods like *Principal Component Analysis* (PCA), do so by creating new combinations of descriptors [27]. On the other hand, feature selection methods include and exclude descriptors without altering them.

The combination of feature selection and regression techniques allows to learn a predictive model depending only on the most appropriate features. We decide to select features according to their capability of describing the variance of the data to be predicted. We adopt the *adjusted* R^2 measure in order to evaluate the goodness of each feature. Such measure is a further development of an other performance measure named R^2 (denoted also as *coefficient of determination*). R^2 provides a measure of how well the observed outcomes are replicated by the model built considering a certain set of descriptors [27]; it measures the proportion of total variance in the observed data that is explained by the model. Given a N-sample dataset with observed response values $y_1, ..., y_N$, the general definition of R^2 is the following

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y})}$$

where N is the dataset size, $\hat{y_i}$ is the predicted value for sample i, and $\bar{y}$ is the mean of the observed response values ($\bar{y} = \sum_{i=1}^{N} y_i$). The higher it is, the more the model is able to perform accurate predictions. The maximum achievable value is 1. Lower values attest poorer prediction results and can also be negative, although this is quite uncommon [27]. However, the simple R^2 measure has an important drawback: every time a descriptor is added to the model, the related R^2 may increase, in a way that is not statistically related to the explanatory power of the new descriptor [27]. Consequently, a model including more descriptors may appear to perform better only because it involves a bigger amount of information; such feature selection procedure may easily lead to models containing descriptors not necessary meaningful in terms of prediction capability. In order to take into account this effect, an *improved version of the coefficient of determination* (*adjusted* R^2) has been proposed

$$\textit{adjusted } R^2 = 1 - (1 - R^2) \cdot \frac{N - 1}{N - p - 1}$$

with N the dataset size and p the number of descriptors actually belonging to the model. A candidate descriptor (a feature that could be included in the model) is actually included only if it improves the R^2 more than that would be expected by chance [38]. As its value is always smaller or equal to the simple R^2, also *adjusted* R^2 has maximum value of 1 and can be negative as well. *Adjusted* R^2 does not measure directly the ability of the model to

fit the data, since it embeds information about the dataset size and model dimensionality as well [27].
The process for the computation of the best subset of descriptors constitutes of iteratively including in the model one feature at a time computing every time the *adjusted* R^2. The peak value after which such measure systematically decreases provides the model including the best feature subset [38].

Regression techniques are able to learn both linear and non-linear relationships between data descriptors and response values. In our method we adopt non-linear regression in order to extend preexisting methods for similarity modeling. According to [38], as music similarity evaluation is a complex phenomenon, non-linear models are more appropriate to learn a similarity function able to reflect user's subjective similarity. SVR allows to learn non-linear functions using the same problem formalization that characterizes the linear case. Therefore, we discuss first the details related to the linear case.

3.3.2 Linear SVR

Suppose we have a training set of N samples. Each sample i is described by a feature vector $x_i \in \mathbb{R}^d$, where d in the number of features the vector consists of. We also define y_i as the response value related to x_i and $\hat{y}_i$ as the predicted response value related to x_i. We define the *prediction error* as the absolute difference between y_i and $\hat{y}_i$ that exceeds a chosen threshold ϵ. The goal of the learning process is to minimize the prediction errors, but we have to take into account that a system that perfectly learns the training set usually does not provide good generalization properties. It means that such system would generate a model so accurate in describing the training data that it would not be able to perform a good prediction when facing an unseen sample [38]. Instead, we are interested in a regressor that is able to generalize the information contained in the training set also to unseen data samples. For this reason, we look for a function $f(x)$ that minimizes the prediction errors but, at the same time, it is as flat as possible, as excessively wiggly functions tend to overfit the data [38]. We define the function $f(x)$ as the linear combination of the entries of the training vector x with a vector of coefficients $w \in \mathbb{R}^d$, plus a constant term $(b \in \mathbb{R})$

$$f(x) = \sum_{i=1}^{d} w_i \cdot x_i + b = \langle w, x \rangle + b \tag{3.8}$$

Requiring a function to be as flat as possible means requiring a small w, in terms of Euclidean norm. We can do it defining the following primal objective convex function:

$$\min \frac{1}{2}||w||^2$$

$$\text{subject to} \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \epsilon \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon \end{cases} \forall i = 1, .., N$$

Usually, such type of constrained problems are not solvable with this setting. To relax the constraints, we introduce some slack variables ζ_i, ζ_i^* to deal with unfeasible constraints. The slack variables represent the amount of error higher than ϵ provided by a prediction. In particular, ζ_i accounts the positive difference in the prediction error, while ζ_i^* accounts for the negative difference in the prediction error. Both slack variables are 0 if the prediction is correct, i.e. provides a prediction error lower than ϵ, and provide a linear penalization when a prediction error exceeds ϵ.

$$|\zeta_i| = \begin{cases} 0 & \text{if } |y_i - \langle w, x_i \rangle - b| < \epsilon \\ |y_i - \langle w, x_i \rangle - b - \epsilon| & \text{otherwise} \end{cases}$$

$$|\zeta_i^*| = \begin{cases} 0 & \text{if } |\langle w, x_i \rangle + b - y_i| < \epsilon \\ |\langle w, x_i \rangle + b - y_i - \epsilon| & \text{otherwise} \end{cases}$$

Figure 3.2 shows an intuitive picture of the effect that both the threshold ϵ and the slack variables have on the regression process. In practice, ϵ induces a range around the regression function (the region between the two blue parallel lines) within which prediction errors are not considered as such. Slack variables account for errors exceeding such range.

The introduction of these slack variables leads to the following optimization problem:

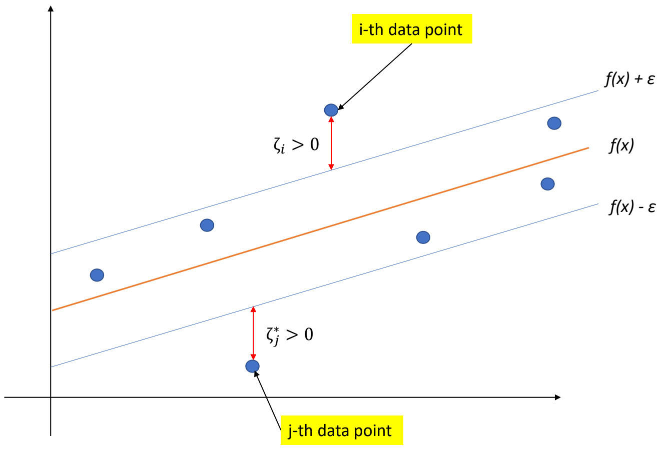


Figure 3.2: Effects of ϵ and slack variables ζ_i, ζ_i^* (linear SVR setting)

$$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\zeta_i + \zeta_i{}^*) \tag{3.9}$$

$$\text{subject to} \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \epsilon + \zeta_i \\ \langle w, x_i \rangle + b - y_i & \leq \epsilon + \zeta_i^* \quad ,\forall i = 1,..,N \\ \zeta_i, \zeta_i^* \geq 0 \end{cases}$$

where C plays the role of a constant value controlling the trade-off between the flatness of $f(x)$ (the first term of the objective function) and the quantity of tolerated prediction error. Starting from equation (3.9), we shape a suitable *Lagrange function*, by introducing a dual set of variables (a_i,a_i^*, η_i, η_i^*). The Lagrange function is obtained by taking the primal objective function and subtracting the product between each constraint and the corresponding dual variable (or *Lagrange multiplier*) [38]. It is possible to show that the optimal point can be reached either by minimizing with respect to the primal variables or by maximizing with respect to their dual counterparts; therefore, the function has a saddle point with respect to the primal and dual variables at the optimal solution. The resulting Lagrange function is

$$L = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\zeta_i + \zeta_i{}^*) - \sum_{i=1}^{N} a_i(\epsilon + \zeta_i - y_i + \langle w, x_i\rangle + b) \tag{3.10}$$

$$- \sum_{i=1}^{N} a_i{}^*(\epsilon + \zeta_i{}^* + y_i - \langle w, x_i\rangle - b) - \sum_{i=1}^{N}(\eta_i\zeta_i + \eta_i^*\zeta_i^*)$$

In order to find the saddle point, i.e. the optimal solution, we derive the function (3.10) with respect to the primal variables and we find the following conditions:

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N}(a_i^* - a_i) = 0 \tag{3.11}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N}(a_i - a_i^*)x_i = 0 \tag{3.12}$$

$$\frac{\partial L}{\partial \zeta_i^{(*)}} = C - a_i^{(*)} - \eta_i^{(*)} = 0 \tag{3.13}$$

It is possible to show that if we expand the original Lagrange function (3.10) and exploit the derivatives with respect to the primal variables, we get the following dual optimization problem [42].

$$\max \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{N}(a_i - a_i^*)(a_j - a_j^*)\langle x_i, x_j\rangle \\ -\epsilon\sum_{i=1}^{N}(a_i + a_i^*) + \sum_{i=1}^{N}(a_i - a_i^*)y_i \end{cases} \tag{3.14}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{N}(a_i - a_i^*) = & 0 \\ a_i, a_i^* & \in [0, C] \end{cases}$$

The dual variables η_i, η_i^* disappear from the final rewriting of the optimization problem. We have already seen that $w = \sum_{i=1}^{N}(a_i - a_i^*)x_i$ thus w can be computed as a linear combination of the training vectors. Using this fact, we obtain the following redefinition of equation (3.8), called *support vector expansion*

$$f(x) = \sum_{i=1}^{N}(a_i - a_i^*)\langle x_i, x\rangle + b \tag{3.15}$$

which is the final function we use to perform regression, after having trained the algorithm. b is computed exploiting the Karush-Kuhn-Tucker

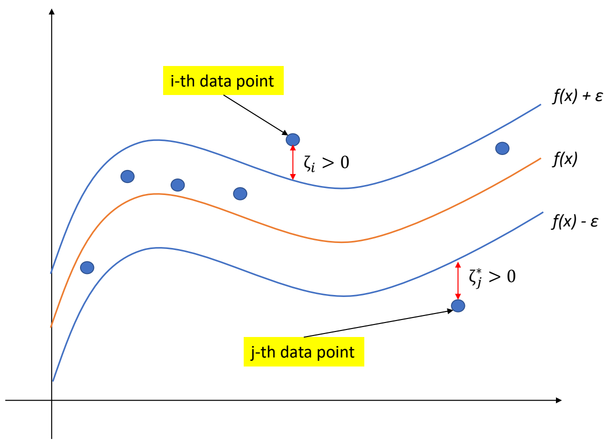


Figure 3.3: Effects of ϵ and slack variables introduction in non-linear SVR setting

conditions [42]. After the formalization of the linear problem, we now show how it is possible to learn non-linear functions exploiting the same formalization.

3.3.3 Non linear SVR

In order to get a non-linear regressor, we should preprocess the training vectors using a mapping function $\phi(x)$ able to characterize the original training vectors into a new feature space. To do so, we can substitute the dot product characterizing equation (3.8) with a kernel function $K(x_i, x_j)$ [42]. A kernel function is a function that, given two vectors in the original feature space, computes the value of their dot product in another feature space [38]. Kernel functions allow to perform regression on a new high-dimensional and non-linear space implicitly induced by the kernel itself, with no need to transform the original feature vectors. This constitutes an advantage. Indeed, the induced space can be easily made non-linear, allowing to perform non-linear regression using exactly the same infrastructure used for the linear case [42]. In addition, we can easily handle high-dimensional feature spaces without actually transforming the original vectors. Figure 3.3 shows the range induced by ϵ and the effect of slack variables in the non-linear case.

With the introduction of kernel functions, equation (3.15) simply becomes

$$f(x) = \sum_{i=1}^{N} (a_i - a_i^*) \cdot K(x_i, x_j) + b$$

which allows to rewrite the optimization problem defined by equation (3.14) as

$$\max \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{N} (a_i - a_i^*)(a_j - a_j^*) \cdot K(x_i, x_j) \\ -\epsilon \sum_{i=1}^{N} (a_i + a_i^*) + \sum_{i=1}^{N} (a_i - a_i^*) y_i \end{cases}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{N} (a_i - a_i^*) = & 0 \\ a_i, a_i^* & \in [0, C] \end{cases}$$

Chapter 4

Method overview

In this chapter we describe in details our method for the modeling of music similarity. Our method relies on two types of data and employs three fundamental techniques. Data are represented both by user-centric data, which are opinions about song similarities, and content-based data represented by song feature vectors. The three techniques employed are t-distributed Stochastic Triplet Embedding (t-STE, section 3.2.3), Support Vector Regression (SVR, section 3.3) and Feature Selection (section 3.3.1). Our method exploits t-STE in order to learn a low-dimensional space reflecting user's opinions (4.5.1) while SVR, in cooperation with the feature selection procedure, learns a mapping between features and such low-dimensional space (4.5.2). We have two datasets, each related to one of the data types. We have a dataset of songs and a user-centric dataset that contains one set of similarity opinions per user. In the following, we describe the complete method considering the training phase in single user scenario.

Figure 4.1 represents a high-level overview of our method. Songs are represented by means of feature vectors as described in section 4.4, while user's assessments are converted into similarity constraints as described in section 4.2. Then, feature vectors are divided into a training dataset and a validation dataset (section 4.3). After the division, feature vectors are processed as described in 4.4. Both datasets are exploited in the training stage (section 4.5) in order to learn the mapping function. The learned mapping function is then exploited in the *prediction stage* of our method, in order to map new songs within the low-dimensional space that reflects user's opinions. In this way, we are able to evaluate similarities for new songs within a space which has been tailored on the specific user.

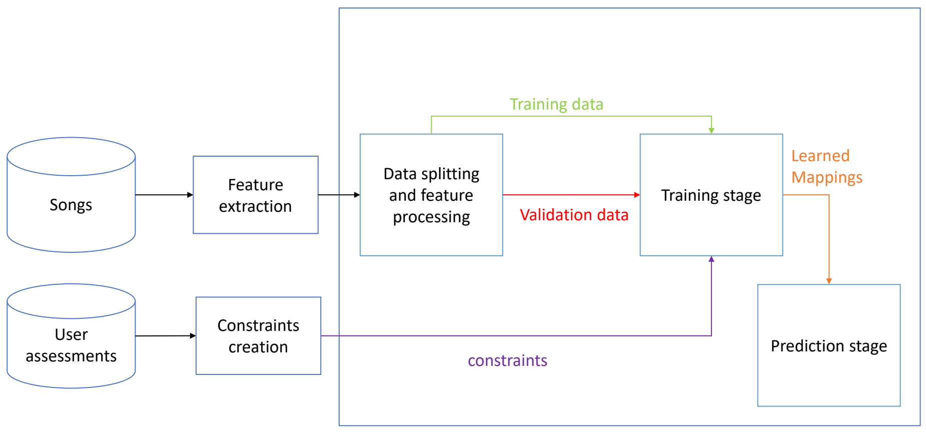


Figure 4.1: Single user scenario of our method.

4.1 Feature extraction

We represent each song within our dataset employing a vector of low-level descriptors, according to [38] and [4]. After the extracting process, descriptors are processed according to their temporal scopes (table 3.2). The goal of the feature processing operation is to build a global song representation, i.e. a feature vector representing the full track. What we do is to average the values of *FrameSingleValued* descriptors and *FrameMultipleValued* descriptors, as these descriptors are computed frame by frame. *FrameSingleValued* descriptors are descriptors that define one value per frame, so they are averaged along the full sequence of frames. *FrameMultipleValued* descriptors are descriptors that define a vector for each frame. Suppose that a *FrameMultipleValued* descriptor provides for each frame a vector $\mathbf{v}$ with P components

$$\mathbf{v} = \{v_0,\ v_1,\ \ldots,\ v_P\}$$

We treat each descriptor v_i, $i = 1, \ldots, P$ in the frame vector as a *FrameSingleValued* descriptor, so we average it along the full sequence of frames. Thus

FrameMultipleValued descriptors induce, after the processing operation, a vector $\bar{\mathbf{v}}$

$$\bar{\mathbf{v}} = \{\bar{v_0},\, \bar{v_1},\ \ldots,\, \bar{v_P}\}$$

whose $i-th$ value represents the average of $i-th$ descriptor $\in \mathbf{v}$. Such vector is appended to the sequence of processed descriptors. *SingleValued* descriptors are not processed, as they already represent one value for the full sequence of frames. At the end of the process, each song is characterized by a feature vector $\mathbf{f}_i$ such that each descriptor in $\mathbf{f}_i$ is a *SingleValued* descriptor.

4.2 Constraints creation

Each user is characterized by a set of similarity assessments, that can be represented as a set S of similarity scores between pair of songs.

$$S = \{s_{ij}\}$$

where s_{ij} denotes the similarity score that the user has assigned to the song pair (i, j). The similarity scores range between 1 and 10. The higher the score, the more songs are considered similar by the user. We translate the numerical information provided by the scores into ordinal information, according to the reasons stated in chapter 2 and chapter 3. In this way each user is finally characterized by a personal set Θ of similarity comparisons expressed as triplets (i, j, k). The meaning of each triplet $(i, j, k) \in \Theta$ is

$$s_{ij} > s_{ik}$$

The process of conversion from similarity scores to similarity constraints is the following. We first collect the similarity scores he/she provided into a self-similarity matrix $\mathbf{SSM} \in \mathbb{R}^{N\times N}$, with N being the cardinality of the song dataset. In this way, $\mathbf{SSM}[i, j] = s_{ij}$. We convert $\mathbf{SSM}$ into a set Θ of triplets with the following procedure. We start with $\Theta = \emptyset$.

(1) for the i-th row of $\mathbf{SSM}$, retrieve the set S_i of valid scores of the i-th row. Valid scores are scores that are greater or equal to 1.

$$S_i = \{s_{ij} \text{ such that } s_{ij} \geq 1\}$$

(1.1) for each score $s_{ij} \in S_i$, retrieve the set $S_i^{(j)}$ of valid scores in the i-th row that are lower than s_{ij}

$$S_i^{(j)} = \{s_{ik} \text{ such that } s_{ik} \in S_i \text{ and } s_{ij} > s_{ik}\}$$

(1.2) for each score $s_{ik} \in S_i^{(j)}$, create the triplet (i, j, k) and add it to Θ. I.e, $\Theta = \Theta \cup (i, j, k)$.

(2) repeate the procedure for each row of **SSM**.

Once Θ has been defined, we retrieve a set Φ of feature vectors $\mathbf{f}_i$ such that each vector $\mathbf{f}_i$ in Φ is related to a song mentioned by a constraint in Θ. Both data are then divided as described in section 4.3.

4.3 Data splitting

The procedure of data splitting is to create a training dataset and a validation dataset. Both datasets consist of both feature vectors and constraints, and we need the two datasets to be distinct, in order to elaborate a model with good generalization property. Both datasets are then exploited in the training phase according to the procedure described in section 4.5. We first split the constraints into a set $\Theta^{(tr)}$ of training constraints and a set $\Theta^{(vd)}$ of validation constraints, adopting the same procedure employed by [38]. Then, we define the set of training songs $\text{TR}^{(s)}$ and the set of validation songs $\text{VD}^{(s)}$ as follows

$$\text{TR}^{(s)} = \left\{ i, j, k \text{ such that } (i, j, k) \in \Theta^{(tr)} \right\}$$

$$\text{VD}^{(s)} = \left\{ i, j, k \text{ such that } (i, j, k) \in \Theta^{(vd)} \right\}$$

$\text{TR}^{(s)}$ consists of songs that are mentioned by constraints $(i, j, k) \in \Theta^{(tr)}$, as well as $\text{VD}^{(s)}$ consists of songs that are mentioned by constraints $(i, j, k) \in \Theta^{(vd)}$.

The two sets of songs help to define the set of training and validation vectors. Indeed, the set of training vectors $\Phi^{(tr)}$ includes feature vectors related to training songs, while the set of validation vectors $\Phi^{(vd)}$ includes feature vectors related to validation songs. We define $\Phi^{(tr)}$ and $\Phi^{(vd)}$ as

$$\Phi^{(tr)} = \left\{ \mathbf{f}_i \text{ such that } i \in \mathrm{TR}^{(s)} \right\}$$

$$\Phi^{(vd)} = \left\{ \mathbf{f}_i \text{ such that } i \in \mathrm{VD}^{(s)} \right\}$$

Both the sets of vectors are then processed according to the approach described in section 4.4 before being employed within the learning phase of our method. Figure 4.2 represents a close view of the process of data splitting.

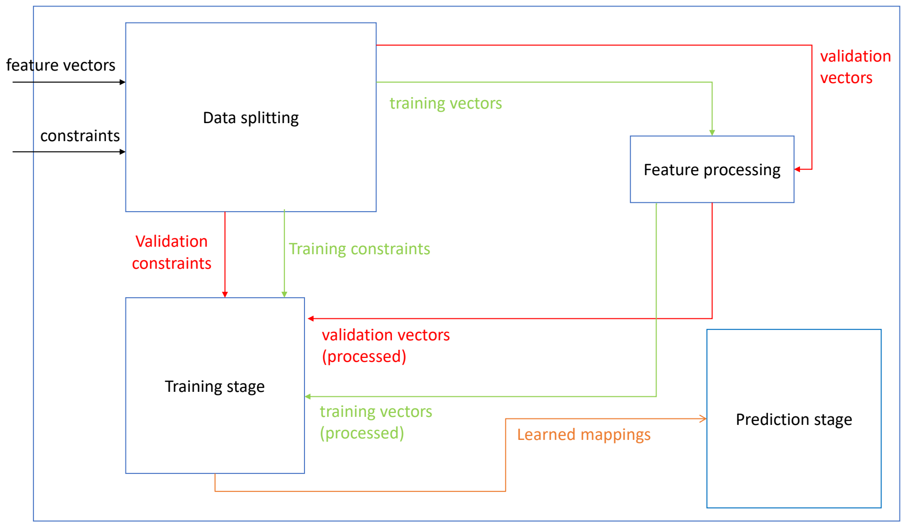


Figure 4.2: Closer view of data splitting and feature processing.

4.4 Feature processing

After the division, feature vectors are normalized according to a feature normalization criterion known as feature standardization [35]. Feature normalization is a feature refining operation often employed in data mining, due to wide range showed by pure raw data. Such technique allows to compare descriptors that have different scales and implicitly assigns the same importance to each feature by weighting all of them equally [35]. We employed feature standardization as it has proved to be useful for refining data used for regression [35]. Applying feature standardization implies processing each descriptor in each feature vector with the following criterion

$$f_i^{(n)} = \frac{f_i^{(n)} - \mu_n}{\sigma_n}$$

where $f_i^{(n)}$ is the $n - th$ descriptor of $\mathbf{f}_i$, μ_n is the mean for the $n - th$ descriptor and σ_n the standard deviation for the $n - th$ descriptor. The transformation acts such that each descriptor distribution is represented as a normal distribution with $\mu = 0$ and $\sigma = 1$.

We adopted the following methodology for standardizing feature vectors. We first applied features standardization for feature vectors in $\Phi^{(tr)}$. Then, we applied the same transformation for feature vectors in $\Phi^{(vd)}$, but employing statistics that were related to training vectors. I.e., we processed each descriptor for each vector $\mathbf{f}_i \in \Phi^{(vd)}$ as

$$f_i^{(n)} = \frac{f_i^{(n)} - \mu_n^{(tr)}}{\sigma_n^{(tr)}}$$

where $\mu_n^{(tr)}$ and $\sigma_n^{(tr)}$ are respectively mean and standard deviation for the n-th descriptor computed relatively to vectors in $\Phi^{(tr)}$.

4.5 Training stage

The full training stage of our proposed method is represented in figure 4.3. The full set of constraints Θ constitute the input for the non-metric scaling algorithm (t-STE), whose objective is to learn a low-dimensional space that fits the set of constraints as much as possible. The output of the scaling process is a set of low-dimensional vectors of song coordinates. In other words, each vector represent the coordinates of a song within the low-dimensional space. The coordinates have been computed in a way that the Euclidean distances (formula 3.1) among songs reflects as much as possible the similarity constraints presented as input. We will refer to the low-dimensional space learned by t-STE as *song space* (section 4.5.1). As for feature vectors, coordinates have been processed in order to feed the regressor with more easy-to-handle data. Coordinate vectors are then divided according to the former training-validation split applied to feature vectors, as described in (4.5.1).

Normalized song space coordinates and feature vectors constitute the input data of the regression procedure. The objective of SVR is to learn a suitable mapping from feature values to song space coordinates. The regression procedure is combined with a feature selection algorithm so that only the most suitable descriptors are considered within the learning process (section 4.5.2).

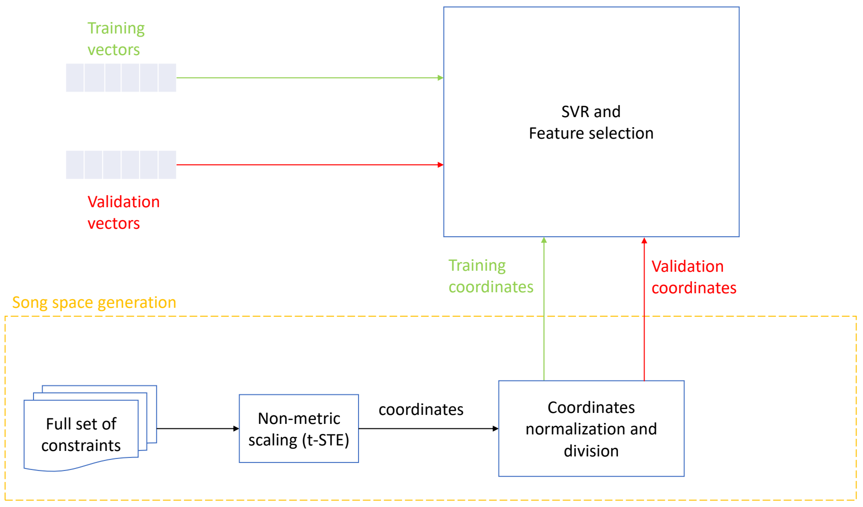


Figure 4.3: Block diagram of the training stage

4.5.1 Song space generation

Song space generation refers to the process of learning a low-dimensional space able to fit the input constraints. In our proposed method, input constraints are triplets $(i, j, k) \in \Theta$. As stated in section 3.2.3, t-STE is a constrained convex optimization problem solved through gradient descent algorithm. Therefore, according to the theory concerning gradient descent methods, the algorithm is initialized with a random matrix $\mathbf{X} \in \mathbb{R}^{s \times d}$, with s the number of songs within our dataset and d the desired dimensionality of the song space. This is due to the fact that, given a convex minimization function, the algorithm converges to the global minimum of such function whatever initialization has been provided. We denote the final set of coordinate vectors learned by the method as a set C of $d-$dimensional vectors

$\mathbf{c}_i$

$$C = \left\{ \mathbf{c}_i \text{ such that } i \in \mathrm{TR}^{(s)} \text{ or } i \in \mathrm{VD}^{(s)} \right\}$$

In other words, C contains one coordinate vector for each song mentioned in Θ. The effect of a single constraint on the algorithm is graphically showed in figure 4.4. Each input constraint induces a shifting stimulus to the related objects. In particular, each triplet (i, j, k) not yet fulfilled during the iterative process attracts j towards i while pulling k away from i, unless several other constraints contradict the actual triplet [51].

The final set of song coordinates is post-processed, in order to provide the regressor with more easy-to-handle response values. We normalized the set of coordinates dividing each vector $\mathbf{c}_i$ as

$$\mathbf{c}_i = \frac{\mathbf{c}_i}{c_{max}}$$

where c_{max} represents the maximum absolute value of the set of coordinates C. This way coordinates range between -1 and 1. We employed such normalization as it consists of a uniform scaling of the song space. Uniform scaling preserves the ordinal distance relationships. Therefore, constraints that were fulfilled in the unprocessed space will be preserved also in the processed space. The same occurs for unfulfilled constraints. The normalized set of coordinates is then divided according to the division that we applied for feature vectors. We define a set of training coordinates $C^{(tr)}$ and a set of validation coordinates $C^{(vd)}$ such that

$$\mathrm{C}^{(tr)} = \left\{ \mathbf{c}_i \text{ such that } i \in \mathrm{TR}^{(s)} \right\}$$

$$\mathrm{C}^{(vd)} = \left\{ \mathbf{c}_i \text{ such that } i \in \mathrm{VD}^{(s)} \right\}$$

Equivalently, $\mathrm{C}^{(tr)}$ contains coordinate vectors related to training songs, as well as $\mathrm{C}^{(vd)}$ contains coordinate vectors related to validation songs.

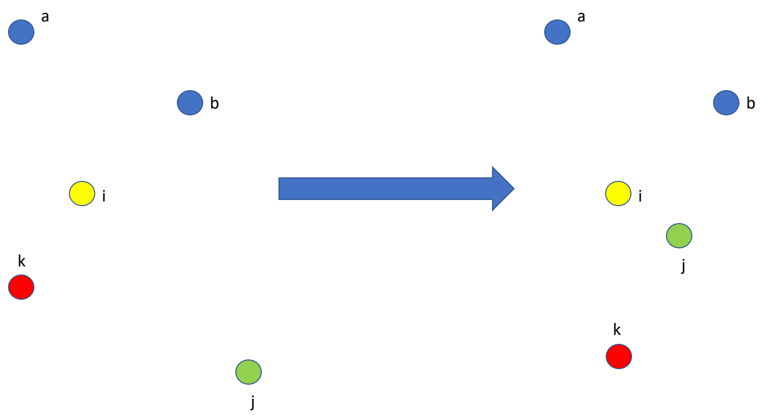


Figure 4.4: Effect of a input constraint (i, j, k). As we can see, initial conditions do not fulfill the constraint. Consequently, j is pulled towards i while k is pulled away from i.

4.5.2 Feature selection and regression

SVR is the machine learning tool accomplishing the task of learning a mapping between acoustic features and song space coordinates. Consequently, both data types serve as input for the regression procedure.

In particular, given a song space expressed as set of d-dimensional vectors, we aim to learn d mappings, i.e. one mapping per dimension. Therefore, the procedure of feature selection and subsequent regression is repeated d times. The input data for the $n-th$ execution $(n = 1, .., d)$ of the process are feature vectors as well as the song space coordinates with respect to the $n-th$ dimension. We denote the set of coordinates with respect to the $n-th$ dimension as $\mathrm{C}_{(n)}$

$$\mathrm{C}_{(n)} = \{c_i^n \text{ such that } \mathbf{c}_i \in C\}$$

After the training phase, we want to be able to map also new songs into the song space. For this reason, the feature selection stage takes care of identifying the feature subset that most enhances the generalization properties of the model to learn. To do so, the feature selcetion stage exploits both training and validation data. Then, SVR builds a mapping between such subset of features and the song coordinates with respect to the considered dimension. Figure 4.5 represents a closer view of a single execution.

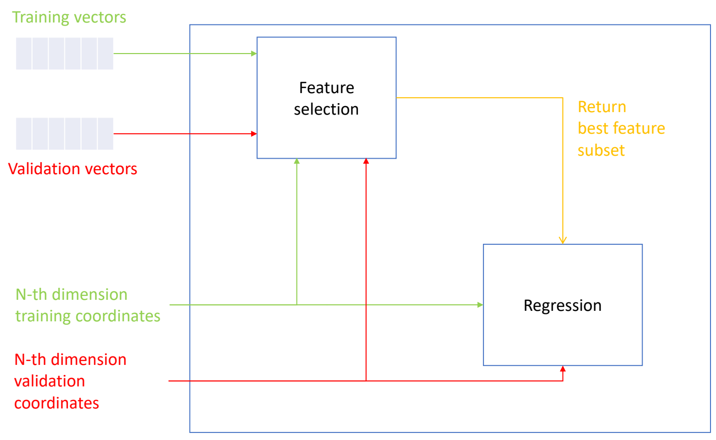

Training vectors
Validation vectors
Feature selection
Return best feature subset
Regression
N-th dimension training coordinates
N-th dimension validation coordinates

Figure 4.5: Block diagram of SVR and feature selection stages.

Feature selection

Feature selection stage is represented in figure 4.6. Since our proposed method exploits SVR in order to fit the data, feature selection stage exploits SVR as well. Therefore, its input data consist of both feature vectors and single-dimension song space coordinates. We denote training coordinates with respect to the $n-th$ dimension as $C_{(n)}^{(tr)}$, and validation coordinates with respect to $n-th$ dimension as $C_{(n)}^{(vd)}$.

As described in section 3.3.1, the best feature subset is built with an iterative procedure. We start defining two sets: CF and BM. CF represent the set of candidate features, and consists of features that have the chance of being included within the model. BM represent the set of features that have been included so far in the model. The two sets are initialized with the full set of original features and with an empty set respectively. For each candidate feature $f \in CF$ we build a temporary set of features TM which consists of BM extended with f, i.e. $TM = BM \cup f$. We learn a mapping from $\Phi^{(tr)}$ to $C_{(n)}^{(tr)}$ using a model that includes descriptors in TM. After having learned the model related to TM, we compute its adjusted R^2 score (section 3.3.1) on validation data, i.e. on the dataset composed by $\Phi^{(vd)}$ and $C_{(n)}^{(vd)}$. Once this operation as been accomplished for each f, we detect (if any) the candidate features that provided some increment with respect to the adjusted R^2 score achieved by the model related to BM (computed with the same procedure). In case at least 1 candidate has provided an increment, we include in BM the candidate f_{max} that provided the largest increment, i.e. $BM = BM \cup f_{max}$. We also delete f_{max} from the set of candidate features. If no candidates have provided an increment, the procedure stops and returns the actual set of descriptors BM. The stopping criterion agrees to what stated in section 3.3.1. Basically, a non-increasing or decreasing adjusted R^2 score indicates that the R^2 score of the model is not increasing as well or that it increasing in a way that is not statistically related to the explanatory power of the descriptors (section 3.3.1). Therefore, the model built until that moment relies on the best feature subset.

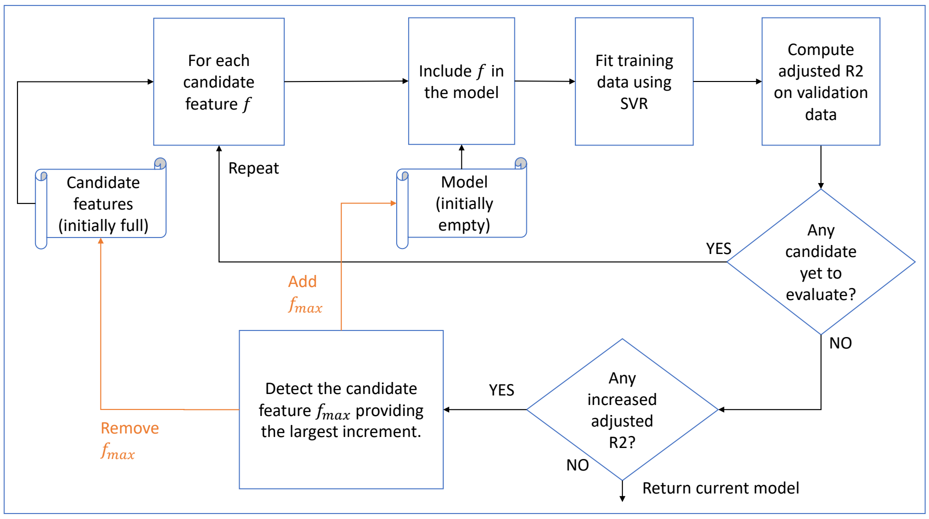

For each candidate feature f
Include f in the model
Fit training data using SVR
Compute adjusted R2 on validation data
Candidate features (initially full)
Repeat
Model (initially empty)
YES
Any candidate yet to evaluate?
Add f_{max}
NO
Detect the candidate feature f_{max} providing the largest increment.
YES
Any increased adjusted R2?
Remove f_{max}
NO
Return current model

Figure 4.6: Block diagram of the feature selection stage.

Regression

The goal of the regression operation is to learn a mapping between feature vectors and each set of single-dimension coordinates $C_{(n)}$. As $n = 1, \ldots, d$, we learn d mappings. So, input data for the regressor consist of both the full feature vector set Φ and BM and $C_{(n)}$. During the regression operation, vectors in Φ are truncated such that only descriptors belonging to BM are kept within the song representation. Our regressor exploits a kernelized version of SVR (section 3.3). We choose a non-linear kernel to empower the regression operation as done in [38], as this allows us to enhance the flexibility of the model and thus to capture more complex mappings with regard to a linear kernel.

After the training phase, we can exploit the mappings in order to map unseen songs in the learned song space. Indeed, all that we now require in order to map a song in the song space is its feature vector. Now we are able to evaluate song similarities for new songs in a space reflecting the similarity perception of the user. Figure 4.7 shows how to exploit the learned mappings, supposing to have $d = 2$ for simplicity.

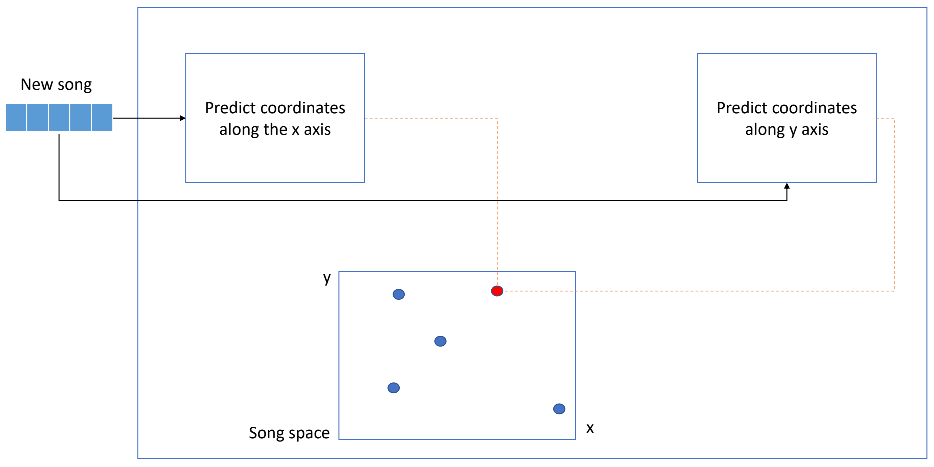

New song
Predict coordinates along the x axis
Predict coordinates along y axis
y
x
Song space

Figure 4.7: Exploiting the mappings in the prediction stage (2D case).

Chapter 5

Experimental setup and evaluation

In this chapter we provide the details concerning the experimental setup and the numerical results related to experiments that have been conducted. We conducted two types of experiment in order to test the generalization properties of our method. The two experiments differ in the way we divide data between training and test, but share both the general setting and the evaluation criterion. We start by describing the set of user-centric data that our methods exploits (section 5.1), and then discuss experimental setup for the two experiments (section 5.2). In particular, we first present the general setting shared by the two experiments in section 5.2.1, then discuss the two data division methods in section 5.2.2. In section 5.3 we first present the shared evaluation criterion and then show the numerical results in section 5.3.1.

5.1 Data collection

The song dataset employed within our work is the *CAL500 music database* [49]. It consists of 500 popular songs recorded in the last century, and contains a very heterogeneous set of tracks, allowing us to cover several musical genres and epochs. The user-related data we exploited come from the survey deployed by [38]. In order to collect information about the similarity from users, the author of [38] presents users a survey. In the survey, each user is presented several reference songs, and for each reference song, a list of 5

User ID	Number of scores
1	371
2	100
3	100
4	194
5	98
6	100
7	194
8	198
9	55
10	53
11	164
12	304
13	60
14	195
15	200
16	200
17	196
18	195
19	98

Table 5.1: Statistics of valid users.

songs is presented. User is asked to evaluate the similarity between the reference song and each song within the list. User can listen to the tracks and express his/her opinion about song similarity by means of a score ranging from 1 to 10. The higher the score, the more songs are considered similar by the user. Scores are collected into a self-similarity matrix **SSM**.

A total of 34 users has attended the survey. By a preliminary analysis of the data coming from the survey, we found that some users provided a small number of between-song similarity scores. As our approach requires a sufficient amount of data to learn an efficient similarity metric, we filter out users who provided less than 50 scores. We ended up with 19 valid users. Table 5.1 summarizes the details on the amount of scores annotated by each user.

5.2 Experimental setup

5.2.1 General setting

For both experiments, the dimensionality of the song space is set to $d = 2$, as previous experiments on the effectiveness of t-STE showed that it provides identical performances independently on the a priori established dimensionality. This requires SVR to learn only 2 dimension-wise mappings per user. We decided to employ a *Radial Basis Function* (RBF) kernel [13] in order to kernelize the regression procedure. The RBF kernel between two vectors $\mathbf{x}$ and $\mathbf{y}$ is computed as

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \left\|\mathbf{x} - \mathbf{y}\right\|^2)$$

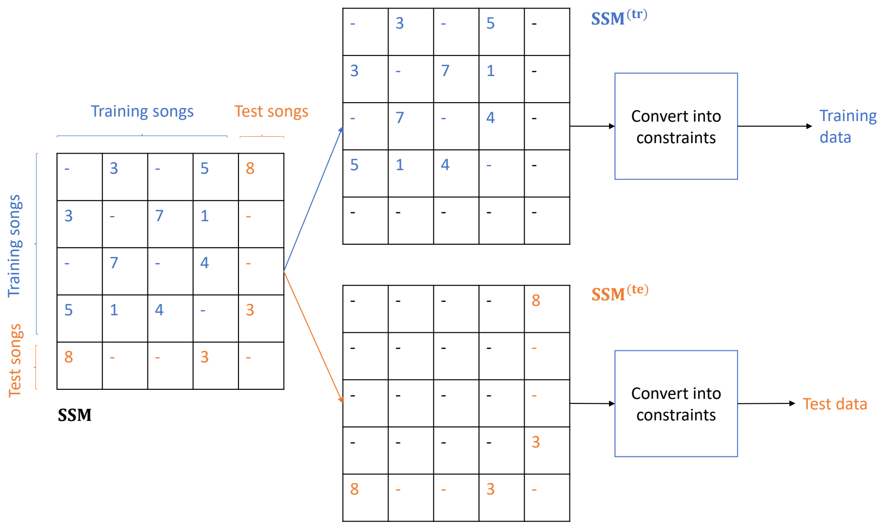
and we set $\gamma = 1$. RBF has been tested and compared with other kernel functions in a preliminary set of experiments. Other tested kernels were *Polynomial*, *Sigmoid* and *Laplacian.* RBF kernel has proved to greatly outperform both Polynomial kernel and Sigmoid kernel, while providing a light increase with respect to the Laplacian kernel. This is due to the fact that the Laplacian kernel represents a simpler variant of the original RBF formulation [13].

5.2.2 Data division methods

In order to test the generalization properties of our method, we need to divide training and test data into two distinct sets. We refer to the first data data division method as *song split*, while we refer to the second data division method as *constraint split.* Song split constitutes a rigid data division method as it allows to avoid data redundancy, while constraint split it is not able to avoid data redundancy. However, constraint split is the data division most frequently employed within the literature of content-based approaches that also exploit similarity constraints ([53], [55], [54] and [38]). So, we conducted the related experiment, for the sake of completeness. Our data consists of both constraints and feature vectors, so we need to divide both data types. The sets of vectors are defined accordingly to the related constraint sets. Training vectors are vectors related to songs mentioned by training constraints, while test vectors are vectors related to songs mentioned by test constraints. So, we first define the two sets of constraints and then define the two sets of vectors. We first describe in detail the song split procedure and highlight its capability of avoid data redundancy. Then we describe in detail the constraint split procedure and highlight the inability of such data division in avoiding data redundancy.

Song split

In this experimental setup we first split songs. We randomly divide songs into training songs and test songs adopting a 80-20 split. This means that we establish 80 % of the songs that have been annotated to be training songs and the remainder to be test songs. The division of songs implies a division of the original scores matrix **SSM** into a training matrix $\mathbf{SSM}^{(tr)}$ and a test matrix $\mathbf{SSM}^{(te)}$. $\mathbf{SSM}^{(tr)}$ contains similarity annotations that concern training songs, as well as $\mathbf{SSM}^{(te)}$ contains similarity annotations that concern test songs. Both matrices are then processed as described in 4.2, providing the two sets of training and test constraints. The set of training constraints consists of constraints (i, j, k) such that i and j and k are training songs. The set of test constraints consists of constraints (i, j, k) such that at least two songs i, j, k are test songs. The set of training vectors is composed by vectors of training songs, while the set of test vectors is composed by vectors of songs mentioned by test constraints. Figure 5.1 represents this experimental setup more closely.



-	3	-	5	8
3	-	7	1	-
-	7	-	4	-
5	1	4	-	3
8	-	-	3	-

-	3	-	5	-
3	-	7	1	-
-	7	-	4	-
5	1	4	-	-
-	-	-	-	-

-	-	-	-	8
-	-	-	-	-
-	-	-	-	-
-	-	-	-	3
8	-	-	3	-

Figure 5.1: Song split method.

Song split allows to avoid data redundancy. We have to recall that each constraint (i, j, k) defines two distances, $d(i, j)$ and $d(i, k)$. We refer to the distances defined by training constraints as *training distances*, as well as we refer to the distances defined by test constraints as *test distances*. Training distances represent the information for training the mapping learning. Indeed, training constraints are used as input for t-STE, and the mapping is trained such that it is able to reflect the distances induced by training constraints. Test distances represent the information we want to predict, as we want the mapping to be able to fulfill test constraints. In this setting, the two sets are completely distinct. Indeed, training distances are distances that are defined only for pairs of training song. As test constraints consist of constraints in which at least two songs are test songs, each test constraint induces two unseen pair-wise distances. As a consequence, each test distance constitute an a priori unpredictable information for the mapping. So, the experiment provides an ideal setting for the evaluation of the generalization properties of the method, as it completely separates the training information from the test information.

Constraints split

Adopting constraint split, we first convert the whole self-similarity matrix **SSM** into a set of constraints as described in section 4.2. Then we split the whole set of constraints into a training set and a test set adopting a random selection that reserves 80 % of constraints to training and the remainder to test. Figure 5.2 represents the data division process.

In this setting, training songs and test songs are defined from the related set of constraints, while with song split the reverse scenario occurs. Thus, with this setting the set of test songs is not separated by the set of training songs. As test distances are pairwise-distances of test songs, and training distances are pairwise-distances of training songs, several test distances will be already defined in the set of training distances. In other words, the set representing training information partially includes the set representing test information. This causes the method to exploit data redundancy. As a consequence, the performances of our method are boosted in a way that it is not related to its generalization properties.

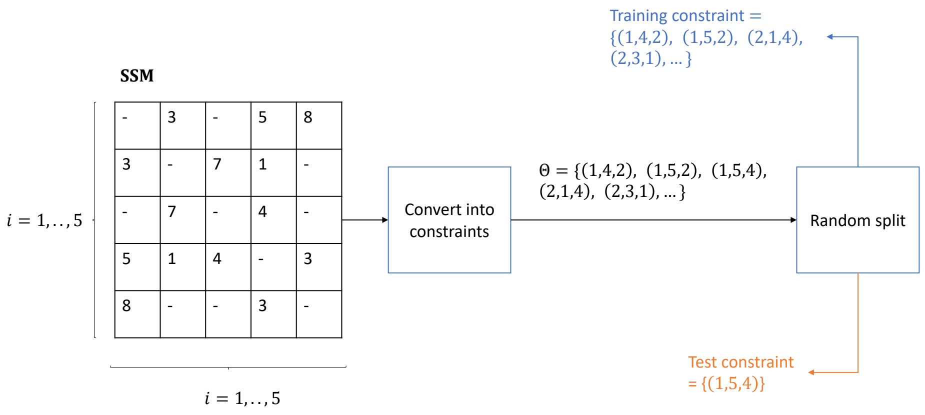

SSM
- 3 - 5 8
3 - 7 1 -
- 7 - 4 -
5 1 4 - 3
8 - - 3 -
i = 1,..,5
i = 1,..,5
Convert into constraints
Θ = {(1,4,2), (1,5,2), (1,5,4), (2,1,4), (2,3,1), … }
Random split
Training constraint = {(1,4,2), (1,5,2), (2,1,4), (2,3,1), … }
Test constraint = {(1,5,4)}

Figure 5.2: Constraints split.

5.3 Evaluation

The evaluation of the generalization properties is the same for both methods. We adopt the mappings learned in the training phase and use them in order to predict test song coordinates given test song feature vectors. In particular, as represented in figure 5.3, we exploit each dimension-wise mapping in order to predict song coordinates with respect to that specific dimension. We then evaluate the goodness of the predictions in terms of test constraints fitting, measured as the percentage of test constraints that the predicted coordinates fulfill. For each test constraint (i, j, k), we check that

$$d(\mathbf{c}_i, \mathbf{c}_j) < d(\mathbf{c}_i, \mathbf{c}_k)$$

In other words, we check if, within the song space, the Euclidean distance between i and j is smaller than the Euclidean distance between i and k. We denote the percentage of fulfilled test constraints as *test accuracy*. As we need to properly evaluate our model on a user-wise base, each experiment is repeated a certain number of times for each user. In particular, we repeat each experiment 30 times per user, in order to exploit 30 performance values in order to assess the overall performance of our method. Fo each user, we collect the 30 test accuracies related to the 30 repetitions of the experiment, and represent their distribution by means of a boxplot. The accuracy distributions will help us to graphically evaluate the goodness of our method. The baseline distribution to which we compare to is the random distribution, i.e. a distribution with mean equal to 0.5 and a wide variance. We expect our method to provide for both experiments mappings that are better than the random mapping, thus we expect each user-related distribution to be characterized by a mean higher than 0.5 and a small variance.

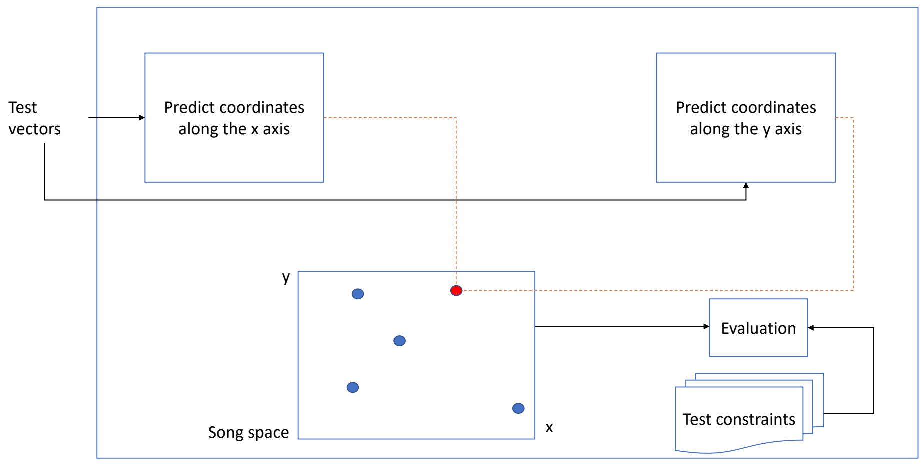

Test vectors
Predict coordinates along the x axis
Predict coordinates along the y axis
y
x
Song space
Evaluation
Test constraints

Figure 5.3: Evaluation of test constraints fitting. For simplicity, a 2D scenario is represented.

5.3.1 Numerical results

Figure 5.4 shows the accuracy distributions obtained from the song split experiment, while table 5.2 summarizes some statistics related to the same experiment. As we can see, even when a rigid data division is applied, the method is able to provide good generalization properties with respect to each user. The characteristics of each distribution are far apart from the characteristics of the random scenario. We can observe that users have distributions with different characteristics. In particular, users who provided most data are characterized by distributions with a smaller variance, while users who provided less data are characterized by distributions with a higher variance. This is due to the fact that the more data are provided, the more the learned mappings tend to match their performance as their parameters tend to match as well. Indeed, with many data, the training sets of the various repetitions tend to be similar, and the mapping learning will provide similar mappings as well. With less data, we have the opposite effect. With less data, training sets of different repetitions tend to differ, and as a consequence the learned mappings differ as well. Due to the diversity of the learned mappings, the accuracy distributions present a larger variance.

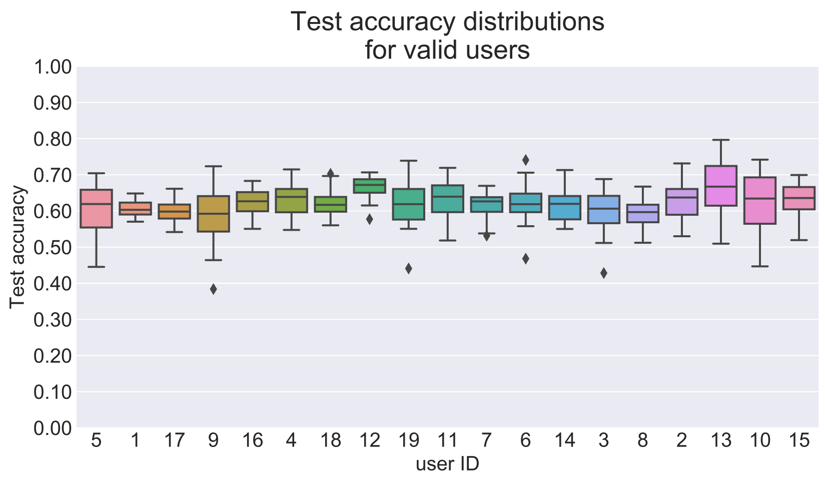


Figure 5.4: Test accuracy distributions for song split experiment

Figure 5.5 shows the accuracy distributions obtained from the constraint split experiment, while table 5.3 summarizes some statistics related to the same experiment. As expected, even in this case the accuracy distributions greatly differ from a random distribution. According to the previous experiment, distributions still relate with the amount of data provided by the related user. As we can see, with this experimental setup, that involves a less rigid data division, the method greatly increases its performances. The boost in the performances is also related to the amount of redundant data that the methods can exploit with this setting.

The relationship between the data amount and the accuracy distribution can be further proved by observing the feature histograms related to each

User ID	Accuracy μ	Accuracy σ	Number of triplets
5	0.61	0.07	225
1	0.61	0.02	1276
17	0.60	0.03	474
9	0.59	0.08	91
16	0.62	0.03	487
4	0.63	0.04	473
18	0.62	0.04	489
12	0.66	0.03	628
19	0.62	0.06	202
11	0.63	0.05	369
7	0.62	0.04	441
6	0.62	0.06	198
14	0.62	0.05	489
3	0.60	0.06	207
8	0.59	0.04	433
2	0.63	0.06	209
13	0.66	0.08	92
10	0.63	0.08	113
15	0.63	0.05	456

Table 5.2: Summary of statistics related to song split experiment.

user. For each user, we computed the related feature histogram counting the times that each feature has been selected by the feature selection algorithm along his/her sequence of experiments. Our method learns mappings that take into account only the set of descriptors identified by the feature selection algorithm. So, counting the times each feature is selected we count the times a feature has been included in the mapping learning. As we stated, mappings related to a user who provided most data tend to match. In other words, these mappings tend to be defined by similar set of descriptors. So we expect the feature histogram for a user who provided most data to be characterized by several peaks and valleys. Peaks correspond to those descriptors belonging to the similar feature subsets computed along the experiment sequence. Figure 5.6 represent the feature histogram for user 1, who provided the largest quantity of constraints. As we can see, peaks and

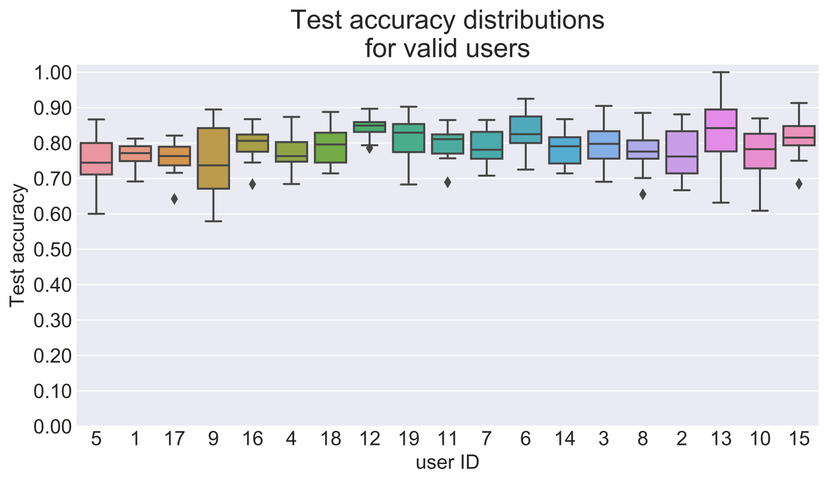


Figure 5.5: Test accuracy distributions for constraint split.

valleys are clearly identifiable. On the other hand, we expect a user who provided less data to be characterized by a flat feature histogram, as his/her mappings tend to be characterized by different set of descriptors along the sequence of experiments. So different descriptors tend to be selected along the experiment sequence, providing a more flat distribution of occurrences. Figure 5.7 represents the feature histogram for user 9 which is the one who provided less data. The figure confirms the previous statements, as distinct descriptors are more hard to detect.

Feature histograms also allow us to confirm that fact that different users tend to evaluate music similarity according to different similarity criteria. The diversity of similarity criteria can be assessed by observing the distribution of feature histograms. Different histogram distributions mean that the mappings of the two users rely on different sets of descriptors, thus suggesting their similarity criteria to be different. Figure 5.8 and 5.9 represent some examples of two users showing different histogram shapes. This confirms the necessity of resolving to personalized techniques in order to model the complex phenomenon of music similarity.

User ID	Accuracy μ	Accuracy σ	Number of constraints
5	0.75	0.07	225
1	0.77	0.03	1276
17	0.76	0.04	474
9	0.75	0.10	91
16	0.80	0.04	487
4	0.78	0.04	473
18	0.79	0.05	489
12	0.85	0.03	628
19	0.81	0.06	202
11	0.80	0.04	369
7	0.79	0.05	441
6	0.83	0.05	198
14	0.78	0.04	489
3	0.80	0.05	207
8	0.78	0.05	433
2	0.77	0.08	209
13	0.83	0.10	92
10	0.77	0.07	113
15	0.82	0.05	456

Table 5.3: Summary of statistics related to constraint split experiment.

Exploiting the information provided by feature histograms, we identified for each user the descriptors that are most frequently employed by the mapping learning procedure, exploiting the information provided by feature histograms. We denote the set of most frequently used descriptors as the set of relevant descriptors for the related user. We consider relevant descriptors to be the descriptors that at least half of times along the experiment sequence are selected to represent the mapping. In other words, we select descriptors whose histogram value is higher than 0.5. Table 5.4 presents for each user the related set of relevant descriptors. As we can see, for users who provided most data, a wide set of descriptors, related to histogram peaks, are detected. Users who provided less data, as they are characterized by flat histograms, are defined by a small set of relevant descriptors.

User ID	Most frequent descriptors
5	aubio tempo, smoothness, spectral contrast peaks, sharpness
1	aubio tempo, chordino harmonicchange, irregularity j, rhythm autocor, spectral contrast peaks, spectralflux thresholdfunction, spectralreflux scaledspectralreflux, spectralreflux spectralrefluxonsets, sharpness, MFCCs
17	aubio tempo, odd even ratio, powercurve smoothpowerslope, spectralreflux scaledspectralreflux, sharpness, MFCCs
9	aubio tempo, sharpness
16	aubio tempo, chordino chordchangerate, intensity, odd even ratio, spectral contrast valleys, sharpness
4	aubio tempo, chordino chordchangerate, intensity, rhythm autocor, spectralreflux scaledspectralreflux, spectralreflux spectralrefluxonsets, sharpness, MFCCs,
18	aubio onsets, aubio tempo, chordino harmonicchange, spectralreflux scaledspectralreflux, sharpness, MFCCs
12	aubio tempo, chordino chordchangerate, intensity, odd even ratio, powercurve smoothpowerslope, rhythm autocor, spectralreflux spectralrefluxonsets, sharpness, MFCCs
19	aubio tempo, chordino chordchangerate, sharpness, MFCCs
11	aubio tempo, chordino chordchangerate, spectral contrast valleys, spectralreflux scaledspectralreflux, sharpness, MFCCs
7	aubio tempo, chordino chordchangerate, chordino harmonicchange, spectralreflux scaledspectralreflux, spectralreflux spectralrefluxonsets, sharpness, MFCCs
6	aubio tempo, chordino chordchangerate, sharpness, MFCCs
14	aubio tempo, rhythm autocor, spectral contrast valleys, spectral contrast valleys, sharpness, MFCCs
3	aubio tempo, chordino harmonicchange, powercurve powerslopeproduct, qm onsets, spectralreflux scaledspectralreflux
8	aubio onsets, aubio tempo, chordino chordchangerate, onsetsds onsets, spectralreflux scaledspectralreflux, spectralreflux spectralrefluxonsets, sharpness, MFCCs
2	aubio tempo, chordino chordchangerate, irregularity j, rhythm autocor, sharpness
13	sharpness
10	sharpness
15	aubio tempo, chordino chordchangerate, spectralreflux scaledspectralreflux, spectralreflux spectralrefluxonsets, sharpness, MFCCs

Table 5.4: List of relevant descriptors for each user.

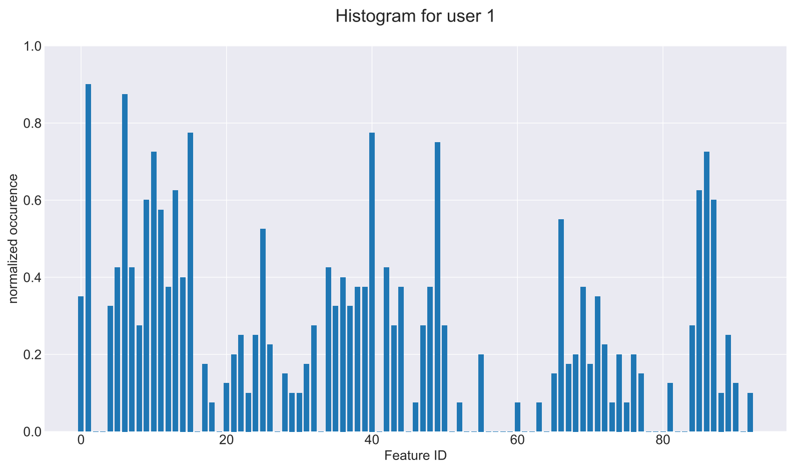


Figure 5.6: Feature histogram for user 1, who provided the highest amount of data. Occurrences are normalized.

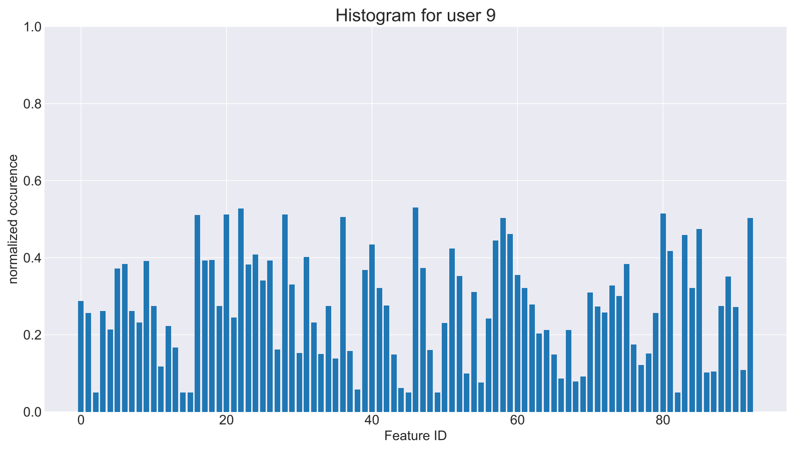


Figure 5.7: Feature histogram for user 9, who provided the smallest amount of data. Occurrences are normalized.

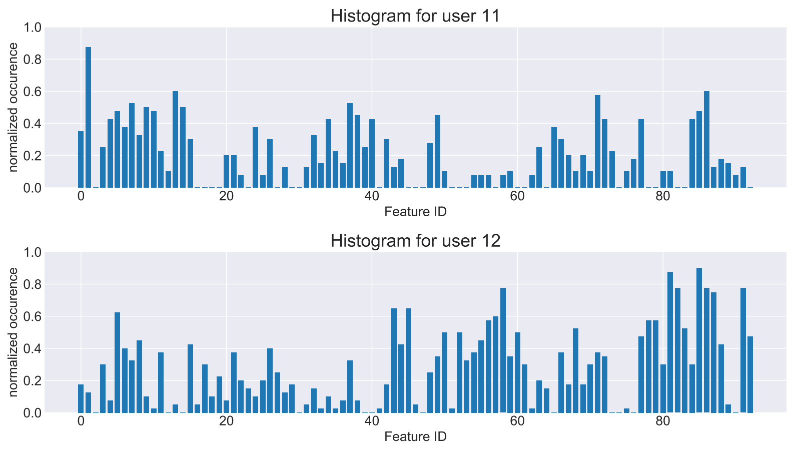


Figure 5.8: Feature histogram for user 11 and 12.

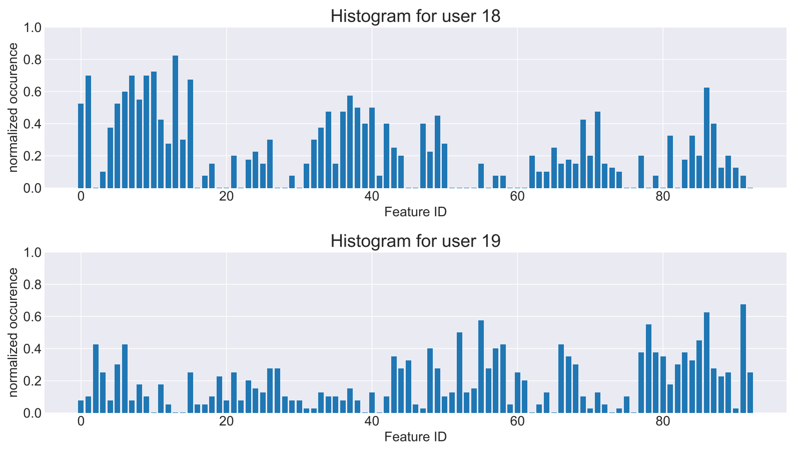


Figure 5.9: Feature histogram for user 18 and 19.

Chapter 6

Conclusions

In this thesis, we presented a novel approach for learning a personalized similarity metric for musical content. As we need a scalable method in order to handle a wide amount of users, we adopted a content-based approach. In addition, as we need to evaluate music similarity in a personalized way, we developed a hybrid method that relies both on content-based data and similarity information provided by users. We modeled content-related information adopting sets of low-level descriptors and collected similarity information as sets of song similarity constraints. Our method relies on a two-stage procedure. We first learn a low-dimensional space tailored on user's perceived similarity. Then, we relate the low-space-related information with the content-based information exploiting a regression technique. In order to learn a non-linear mapping between content-based and low-space information, the regression is kernelized by means of a non-linear kernel. The mapping represents the user's metric. The regression procedure is also combined with a feature selection algorithm, whose objective is to identify the feature subset able to enhance the generalization properties of the method. We conducted two experiments in order to evaluate the generalization properties of our method, each of the two characterized by a different data division algorithm. Our method has proved to perform well with both experimental setups, achieving good generalization performances even when evaluated with a rigid data division method.

Our method presents some advantages with regard to the existing approaches. First, the process of learning the personalized metric is fast and requires a low computational burden. Once data about the similarity are available, all we require is to run a scaling procedure and then a regression procedure. In addition, once the training phase is completed, both the learned mappings and the low-dimensional space can be exploited offline. They can be used in order to handle typical user requests as music recommendation, playlist generation and browsing. For each of these three tasks, mappings can be employed in order to map new songs within the low-dimensional space. For recommendation and playlist generation, this allows us to evaluate song similarity in a space shaped according to user's opinion. For browsing, the user can exploit both the low-dimensional space and the mappings in order to elaborate a personal musical collection that has been organized according to their opinion. In addition, the mappings reduce all the information coming from the audio to a small set of coordinates, so evaluating song similarity reduces to evaluate distances between low-dimensional points. So, for both recommendation and playlist generation, the computation of similarity is straightforward. A further advantage is the fact that the mappings rely only on subsets of features. As we know the set of descriptors that characterize the mapping related to a user, it is sufficient to compute that set of descriptors in order to map a new song within the low-dimensional space related to that user. So, we do not need to extract a wide set of descriptors from new songs in order to map them.

Our method also presents some possibilities for improvements and future works. The first possible development concerns the adoption of implicit data coming Collaborative Filtering. Collaborative Filtering (2.1) collects for each user information about their listening habits. The idea is to adopt CF data in order to estimate the user similarity information that our method needs in order to learn the mapping. This gives the possibility to implicitly learn the user metric without requiring the user to supply any similarity information. This could provide a content-based method not requiring any user feedback or interaction to learn the metric. CF data can also be helpful in order to upgrade the process of learning the metric and make it "time-adaptive". As music is a complex phenomenon, users may change their preferences over time. As CF data are related to user's listening habits, we can exploit them in order to learn a new similarity metric able to reflect the current preferences of the user, providing a method able to continuously update a specific user metric.

An other development of our method is represented by the chance of including a process for user profiling. The idea is to adopt a user profiling process in order to collect some user-related characteristics as age, musical preferences, musical experience or musical theory background. The goal is to investigate if user's characteristics are somehow related to the set of descriptors that define their similarity metric. If such relationship exists, we expect that similar profiles will tend to rely on similar set of descriptors for the metric. As a consequence, we could think of grouping similar users, simply by exploiting the information coming from their profiles. This could be an advantage when facing a new user, for which a personalized metric need to be learned. Just by exploiting the information coming from their profile, we could think of first characterizing the user with a metric related to a user similar to him/her. This allows us to provide new users, for which CF data are not available, with an initial metric that is somehow related with their subjective similarity. This could be an efficient solution to the cold start issue of CF approaches. Once CF data for new users are available, we can learn their"true" personalized metric.

Bibliography

[1] S. Agarwal, J. Wills, L. Cayton, G. Lanckriet, D. Kriegman, and S. Belongie. Generalized non-metric multidimensional scaling. In *JMLR W and CP*, volume 2, pages 11–18, 2007.

[2] Jean-Julien Aucouturier, François Pachet, and Mark Sandler. The way it sounds: Timbre models for analysis and retrieval of music signals. In *IEEE Transactions On Multimedia*, volume 7, pages 1028–1035. Springer, 2005.

[3] Adam Berenzweig, Beth Logan, Daniel PW Ellis, and Brian Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.

[4] Dmitry Bogdanov. *From music similarity to music recommendation. Computational approaches based on audio features and metadata.* PhD thesis, Universitat Pompeu Fabra, Barcellona, 2013.

[5] Brian R Glasberg Brian C.J. Moore. Modeling binaural loudness. *The Journal of the Acoustical Society of America*, 2007.

[6] Matthew E.P. Davies, Paul M. Brossier, and Mark D. Plumbley. Beat tracking towards automatic musical accompaniment. In *Audio Engineering Society Convention*, 2005.

[7] Matthew EP Davies and Mark D Plumbley. Causal tempo tracking of audio. In *ISMIR*. Citeseer, 2004.

[8] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[9] Bas de Haas, Jose Pedro Magalhaes, and Frans Wiering. Improving audio chord transcription by exploiting harmonic and metric knowledge. In *ISMIR proceedings*, 2012.

[10] Christian Dittmar, Christoph Bastuck, and Matthias Gruhne. Novel mid-level audio features for music similarity. In *Proceedings of the International Conference on Music Communication Science (ICoMCS)*, pages 38–41, 2007.

[11] Simon Dixon. Onset detection revisited. In *Proceedings of the 9th International Conference on Digital Audio Effects*, 2006.

[12] Chris Duxbury, Juan Pablo Bello, Mike Davies, and Mark Sandler et al. Complex domain onset detection for musical signals. In *Proc. Digital Audio Effects Workshop*, 2003.

[13] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[14] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings of 2002 IEEE International Conference*, 2002.

[15] Peter Knees and Markus Schedl. A survey of music similarity and recommendation from music context data. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 10(1), 2013.

[16] Peter Kness and Markus Schedl. *Music Similarity and Retrieval - An Introduction to Audio-and Web-based Strategies*. Springer, 2016.

[17] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 426–434. ACM, 2008.

[18] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 145–186. Springer, 2011.

[19] Jochen Krimphoff, Stephen McAdams, and Suzanne Winsberg. Caracterisation du timbre des sons complexes. analyses acoustiques et quantification psychophysique. *Le Journal de Physique, 4*.

[20] Jan De Leeuw and Patrick Mair. *Multidimensional scaling using majorization: SMACOF in R*. Department of Statistics, UCLA, 2011.

[21] Dan Liu Lie Lu and Hong-Jiang Zhang. Automatic mood detection and tracking of music audio signals. In *IEEE Transactions on audio, speech, and language processing*, 2006.

[22] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *ICME*, pages 22–25, 2001.

[23] Paul Masri. *Computer modelling of sound for transformation and synthesis of musical signals*. PhD thesis, University of Bristol, 1996.

[24] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning similarity from collaborative filters. In *11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.

[25] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. In *IEEE Transactions on Audio, Speech and Language Processing*, volume 20, pages 2207–2218, 2012.

[26] Brian McFee and Gert R Lanckriet. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 775–782, 2010.

[27] Luis Carlos Molina, Lluis Belanche, and Angela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002.

[28] Brian C.J. Moore, Brian R. Glasberg, and Thomas Baer. A model for the prediction of thresholds, loudness, and partial loudness. *Journal of the Audio Engineering Society*, 1997.

[29] Alexandros Nanopoulos, Dimitrios Rafailidis, Panagiotis Symeonidis, and Yannis Manolopoulos. Musicbox: Personalized music recommendation based on cubic analysis of social tags. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2):407–412, 2010.

[30] Katy Noland and Mark Sandler. Signal processing parameters for tonality estimation. In *Audio Engineering Society Convention*, 2007.

[31] Andreas Nurnberger and Sebastian Stober. User modelling for interactive user-adaptive collection structuring. In *International Workshop on Adaptive Multimedia Retrieval*, pages 95–108. Springer, 2007.

[32] Bee Suan Ong. *Towards automatic music structural analysis. Identifying characteristic within-song excerpts in popular music.* PhD thesis, Citeseer, 2005.

[33] Francois Pachet, Gert Westermann, and Damien Laigre. Musical data mining for electronic music distribution. In *Web Delivering of Music, 2001. Proceedings. First International Conference on*, pages 101–106. IEEE, 2001.

[34] Tae Hong Park. *Salient feature extraction of musical instrument signals.* PhD thesis, Dartmouth College, Hanover, New Hampshire, 2000.

[35] S Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[36] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project.

[37] M.A. Carreira Perpinan. The elastic embedding algorithm for dimensionality reduction. In *ICML*, pages 167–174, 2010.

[38] Luca Poddigue. A personalized content-based music similarity function. Master's thesis, Politecnico di Milano, 2014.

[39] Howard F Pollard and Erik V Jansson. A tristimulus method for the specification of musical timbre. *Acta Acustica united with Acustica*, 51(3):162–171, 1982.

[40] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition.* PTR Prentice Hall Englewood Cliffs, 1993.

[41] Malcolm Slaney, Kilian Weinberger, and William White. Learning a metric for music similarity. In *International Conference of Music Information Retrieval (ISMIR)*, 2008.

[42] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. In *. Statistics and computing*, volume 14, pages 199–222, 2004.

[43] Dionysios N. Sotiropoulos, Aristomenis S. Lampropoulos, and George A. Tsihrintzis. Musiper: a system for modeling music similarity perception based on objective feature subset selection. *Springer*, 2007.

[44] Sebastian Stober. Adaptive distance measures for exploration and structuring of music collections. In *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*. Audio Engineering Society, 2011.

[45] Sebastian Stober and Andreas Nürnberger. An experimental comparison of similarity adaptation approaches. In *Adaptive Retrieval. Large-Scale Multimedia Retrieval and Evaluation*, pages 96–113. Springer, 2013.

[46] Dan Stowell and Mark Plumbley. Adaptive whitening for improved real-time audio onset detection. In *Proceedings of the International Computer Music Conference*, 2007.

[47] O. Tamuz, C. Liu, S.J. Belongie, O. Shamir, and A.T. Kalai. Adaptively learning the crowd kernel. In *ICML*, pages 673–680, 2011.

[48] Andreas Töscher, Michael Jahrer, and Robert Legenstein. Improved neighborhood-based algorithms for large-scale recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*. ACM, 2008.

[49] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic description using the cal500 dataset. In *ACM Special Interest Group on Information Retrieval Conference*, 2007.

[50] Andreu Vall, Marcin Skowron, Peter Knees, and Markus Schedl. Improving music recommendations with a weighted factorization of the tagging activity. In *ISMIR*, pages 65–71, 2015.

[51] Laurens van der Maaten and Kilian Weinberger. Stochastic triplet embedding. In *IEEE International Workshop On Machine Learning For Signal Processing*, September 2012.

[52] L.J.P. van der Maaten. Learning a parametric embedding by preserving local structure. In *JMLR W and CP*, volume 5, pages 384–391, 2009.

[53] Daniel Wolff and Tillman Weyde. Adapting metrics for music similarity using comparative ratings. In *ISMIR*, pages 73–78, 2011.

[54] Daniel Wolff and Tillman Weyde. Combining sources of description for approximating music similarity ratings. In *Adaptive Multimedia Retrieval. Large-Scale Multimedia Retrieval and Evaluation*, 2013.

[55] Daniel Wolff and Tillman Weyde. Learning music similarity from relative user ratings. *Information retrieval*, 17(2):109–136, 2014.

[56] Dan Yang, Zongming Ma, and Andreas Buja. A sparse svd method for high-dimensional data. *arXiv preprint arXiv:1112.2433*, 2011.