

**School of Electronic  
Engineering and  
Computer Science**

MSc Sound and Music Computing

Project Report 2015

**Hybrid music recommender  
using content-based and  
social information**

Paulo Esteban Chiliguano Torres

## Acknowledgements

I wish to express my sincere gratitude to Dr. Georgy Fazekas, Lecturer in Digital Media at Queen Mary University of London, for giving me the opportunity to work on this state-of-the-art field and for his guidance and valuable suggestions during the planning and development of this project. I also wish to acknowledge the supplementary assistance provided by Dr. Tony Stockman and Dr. Mathieu Barthet during my time as a student at Queen Mary University of London.

I am particularly grateful with National Government of the Republic of Ecuador for awarding me with a scholarship to study a postgraduate taught degree at a high-quality research university in the United Kingdom of Great Britain and Northern Ireland.

Finally, a special warm thanks goes to my parents, my brothers and Miss Ana Costilla for their support and encouragement throughout my studies.

## **Abstract**

There is a vast range of Internet resources available today, including songs, albums, playlists or podcasts, that a user cannot discover if there is not a tool to filter the items that the user might consider relevant. Several recommendation techniques have been developed since the internet explosion to achieve this filtering task. In an attempt to recommend relevant song to users, we propose an hybrid recommender that considers real-world users information and high-level representation for audio data. We use a deep learning technique, convolutional deep neural network, to represent the audio data in an abstract level. As our main contribution, we investigate a state-of-the-art technique, estimation of distribution algorithm, to capture the listening behaviour of an individual from the features of the songs that are interesting to the user. The designed hybrid music recommender outperforms the predictions compared with a traditional content-based recommender.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	2
1.2	Aims . . . . .	3
1.3	Thesis outline . . . . .	4
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Online Social Networks . . . . .	6
2.2	Music services platforms . . . . .	7
2.3	Recommender Systems . . . . .	8
2.3.1	Collaborative filtering . . . . .	8
2.3.2	Content-based filtering . . . . .	11
2.3.3	Item Representation . . . . .	12
2.3.4	User Modelling . . . . .	13
2.3.5	Hybrid recommender approaches . . . . .	13
2.4	Music Information Retrieval . . . . .	15
2.4.1	Genre classification . . . . .	15
2.4.2	Music recommender systems . . . . .	16
2.5	Deep Learning . . . . .	18

2.5.1	Deep Neural Networks . . . . .	18
2.5.2	Convolutional Deep Neural Networks . . . . .	20
2.6	Estimation of Distribution Algorithms . . . . .	23
2.6.1	A Hybrid Recommendation Model Based on EDA . . .	24
2.6.2	Continuous Univariate Marginal Distribution Algorithm	26
2.7	Summary . . . . .	27
<b>3</b>	<b>Methodology</b>	<b>28</b>
3.1	Data collection . . . . .	29
3.1.1	Taste Profile subset cleaning . . . . .	29
3.1.2	Fetching audio data . . . . .	30
3.1.3	Intermediate time-frequency representation for audio signals . . . . .	31
3.2	Data preprocessing . . . . .	34
3.2.1	Rating from implicit user feedback . . . . .	34
3.2.2	Standardise time-frequency representation . . . . .	35
3.3	Algorithms . . . . .	36
3.3.1	Probability of music genre representation . . . . .	36
3.3.2	User profile modelling . . . . .	39
3.3.3	Top-N songs recommendation . . . . .	41
3.4	Summary . . . . .	43
<b>4</b>	<b>Experiments</b>	<b>44</b>
4.1	Evaluation for recommender systems . . . . .	44
4.1.1	Types of experiments . . . . .	44
4.1.2	Evaluation strategies . . . . .	45

4.1.3	Decision based metrics . . . . .	46
4.2	Evaluation method . . . . .	47
4.2.1	Training set and test set . . . . .	47
4.2.2	Top-N evaluation . . . . .	47
<b>5</b>	<b>Results</b>	<b>48</b>
5.1	Genre classification results . . . . .	48
5.2	Recommender evaluation results . . . . .	49
<b>6</b>	<b>Conclusion</b>	<b>51</b>
6.1	Future work . . . . .	52
	<b>References</b>	<b>53</b>

# List of Figures

2.1	Collaborative filtering process (Sarwar et al. 2001) . . . . .	9
2.2	User based collaborative filtering ( <i>Recommendation Engine</i> 2013) . . . . .	9
2.3	Item based collaborative filtering ( <i>Recommendation Engine</i> 2013) . . . . .	9
2.4	Content-based filtering process (Blog.seagatesoft.com 2015) . .	11
2.5	Three-way aspect model (Yoshii et al. 2008) . . . . .	17
2.6	Schematic representation of a deep neural network (Brown 2014)	19
2.7	Convolutional deep neural network LeNet model (Deeplearn- ing.net 2015) . . . . .	22
2.8	Flowchart of estimation of distribution algorithm (Ding, Ding, and Peng 2015) . . . . .	23
3.1	Diagram of the cleaning process of the Taste Profile subset . .	29
3.2	Flowchart of the fetching audio process . . . . .	32
3.3	Flowchart for time-frequency representation process . . . . .	32
3.4	Diagram of the hybrid music recommender . . . . .	36

3.5	Diagram of CDNN for music genre classification (Kereliuk, Sturm, and Larsen 2015) . . . . .	37
-----	--	----



# List of Tables

5.1	Genre classification results . . . . .	48
5.2	Evaluation of recommender systems (N=5) . . . . .	49
5.3	Evaluation of recommender systems (N=10) . . . . .	49
5.4	Evaluation of recommender systems (N=20) . . . . .	50

# Chapter 1

## Introduction

Music has accompanied social activities on our daily lives and has influenced the shape of the technology landscape that we have today. Portable media players, mobile device applications or music streaming services enable us the access to a large volume of digital recorded music. This vast range of music tracks might include songs that are relevant or not to a listener, being necessary to develop facilities to bring out appropriate musical pieces to an user.

Recommender systems can be described as engines that guide the users to suitable objects from a large number of options in a particular domain such as books, films or music. The available information of users and items' attributes is analysed and exploited by the recommender systems to produce a list of previously unseen items that each user might find enjoyable. Depending on the analysed data, the design of a recommender can be focused on historical ratings given by users or similarities between the attributes of items that an user already rated.

## 1.1 Motivation

Due to the available information of relationship between users and items would be sparse, e.g., most part of the users tend to do not give enough ratings, the accuracy of predictions would decrease. Another disadvantage of traditional recommender systems, referred as *cold-start problem*, arises when a new item cannot be recommended until it gets enough ratings, or, equivalently, when a new user does not have any ratings (Melville and Sindhvani 2010). In order to alleviate the rating sparsity and cold-start problems, there is the motivation to combine two or more recommendation techniques into hybrid approaches.

Deep learning is an approach to artificial intelligence for describing raw data as a nested hierarchy of concepts, with each abstract concept defined in terms of simpler representations. For example, deep learning can describe high-level features of an image of a car such as position, colour or brightness of the object, in terms of contours, which are also represented in terms of edges. (Bengio, Goodfellow, and Courville 2015)

Inspired in natural evolution of species, estimation of distribution algorithms (EDAs) (Larranaga and Lozano 2002) are robust techniques developed during the last decade for optimisation in Statistics and Machine Learning fields. EDAs can capture the explicit structure of a population with a probability distribution estimated from the best individuals of that population.

## 1.2 Aims

We aim to design and implement a hybrid music recommender to mitigate the cold-start problem in a content-based recommendation strategy. The architecture of our hybrid recommender approach combines two fundamental tasks (*Recommender Systems* 2012): *user modelling* and *information filtering*. Both of these techniques require user-item data to learn user’s interest and select items based on their content description, respectively.

In this project, user-item information is obtained from the Taste Profile dataset, which is a complementary subset of the Million Song Dataset (Bertin-Mahieux et al. 2011) and provides real world listeners activity, i.e., play counts of a song. On the other hand, the items to consolidate the music library are obtained by using the unique identifier of each song to fetch its audio data from 7digital.

A convolutional deep neural network (CDNN), which is a deep learning model, is employed to describe the time-frequency content of each audio clip with a n-dimensional vector, whose dimensions represent the probability of a clip to belong to an specific music genre. In this project, we bound the number of music genres to 10.

As a primary contribution of this project, estimation of distribution algorithms (EDAs) are investigated to model user profiles in terms of probabilities of music genres preferences. The algorithms use play count and the content vector of each song in the user’s collection to optimise the profile. In addition, each dimension in the content vector is treated as a discrete and continuous variable, for evaluation purposes. To our knowledge, this is

the first approach that uses a continuous EDA for user profile modelling in recommender systems.

Each user profile then is compared with the vector representation of an audio clip to compute the similarity value between them. Recommendations for an user are built up by selecting the clips with highest similarity values.

The evaluation of our hybrid music recommender approach is assessed by comparing the results obtained with a traditional content-based recommender.

### **1.3 Thesis outline**

The rest of the report is organised as follows: Chapter 2 provides an overview in recommender systems. Recommendation process, associated challenges, and related work based on state-of-the-art techniques are discussed. In Chapter 3, we present our proposed hybrid recommendation approach and describe the stages and algorithms in detail. The experiments and evaluation protocols are to assess the performance of the hybrid recommender presented in Chapter 4. In Chapter 5, we proceed to discuss and analyse the results from the conducted experiments to evaluate the proposed hybrid music recommender. In Chapter 6, we present the conclusions and some thoughts for further research.

# Chapter 2

## Background

Recommender systems create opportunities and challenges for industry to understand consumption behaviour of users. In particular, for music industry, the development of recommender systems could improve digital music sales (Ringen 2015), and also, it could assist the listeners to discover new music through their habits (Hypebot.com 2015). However, when there is no priori information of a new introduced item in a recommender system, known as the *cold-start problem*, popular songs could be favoured in recommendation process instead of items in the *long tail*, i.e., songs that do not have enough ratings. Usually, content-based recommender systems are used to solve the cold-start problem because similarities between items are based on the content without regarding the ratings (Park and Tuzhilin 2008). Another solution to address the cold-start problem is to combine recommendation techniques to boost the strengths of each technique in an hybrid architecture. (Melville and Sindhvani 2010)

In this chapter, we present the importance of online social networks and

music services platforms for retrieving user-item information, in conjunction with related work on music recommender systems. Subsequently, a novel approach of an hybrid recommendation model based on estimation of distribution algorithms (EDAs) is introduced and examined.

## 2.1 Online Social Networks

Social network sites (boyd and Ellison 2007) are defined as:

“Web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system.”

During the last decade, online social networks, which are also identified as *social media* platforms, have become the outstanding technologies for retrieving and exchanging multimedia information (Putzke et al. 2014). Facebook, Twitter or YouTube, have enabled users to produce and share content on the internet, specially, customers around the world are renovating business models by sharing reviews and comments of products directly to companies. This produced content provides opportunities for research to track consumer’s behaviour. (Smith 2009)

In particular, Last.fm<sup>1</sup> is an online radio station that also have the facilities of a social media platform, where a user profile is built up by collecting the music tracks listened on multimedia players through a indexing process

---

<sup>1</sup><http://www.last.fm/>

called *scrobbling*. This profile may expose music consumption and listening behaviour. (Putzke et al. 2014)

## 2.2 Music services platforms

The Echo Nest<sup>2</sup> was a music intelligence company that offered solutions for music discovery and personalisation, dynamic curated sources, audio fingerprinting and interactive music applications. In 2014, The Echo Nest was acquired by Spotify<sup>3</sup>, which is a commercial music streaming service, where a user can browse and listen music tracks sorted by artists, albums, genres or playlists.

However, The Echo Nest API is still active for developer community and offers the access to artists, songs, taste profiles and playlists data. Particularly, The Echo Nest API is able to retrieve information limited to a particular music tracks catalogue such as 7digital<sup>4</sup>.

Both The Echo Nest and 7digital require to sign up for a free account to get unique keys for OAuth<sup>5</sup> authentication in order to retrieve desired information through their respective APIs. As well, free account has limited number of calls, in the case of Echo Nest is limited to 20 request per minute and in the case of 7digital is limited to 4000 request per day.

In this project, we use a The Echo Nest account to get music tracks identifiers for each song in the user-item dataset and we use a 7digital developer account to fetch audio for each music track catalogue identifier. The user-

---

<sup>2</sup><http://developer.echonest.com/>

<sup>3</sup><https://www.spotify.com/>

<sup>4</sup><http://developer.7digital.com/>

<sup>5</sup><http://oauth.net/>



item dataset consist of user - song - play count triplets of the Taste Profile<sup>6</sup> subset which contains real world listeners activity provided among Echo Nest partners including Last.fm.

## 2.3 Recommender Systems

Recommender systems are software or technical facilities to provide items suggestions or predict customer preferences by using prior user information. These systems play an important role in commercial applications to increase sales and convey user satisfaction. In general, recommender systems can be categorised in two major groups: collaborative filtering and content-based filtering (Melville and Sindhvani 2010).

Celma (2008) considers also another methods for music recommendation such as *demographic filtering* and *named context-based*.

### 2.3.1 Collaborative filtering

In collaborative filtering (CF) (Yao et al. 2015), a  $m \times n$  rating matrix (Figure 2.1) represents the relationships between  $m$  users and  $n$  items.

Recommendations are based on the computed similarities between rows (for users) or columns (for items), hence, CF can be further subdivided in the following neighbourhood models (Hu, Volinsky, and Koren 2008):

- **User based** collaborative filtering, produce a recommendation of a previously unseen item based on similarity between users (Figure 2.2).

---

<sup>6</sup><http://labrosa.ee.columbia.edu/millionsong/tasteprofile>

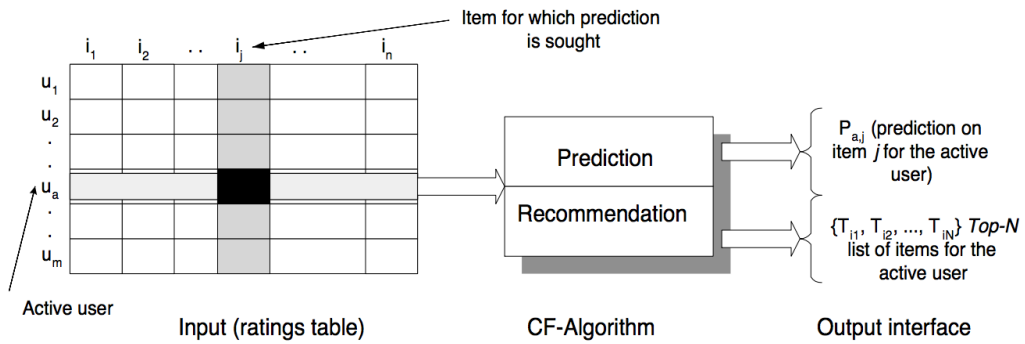


Figure 2.1: Collaborative filtering process (Sarwar et al. 2001)

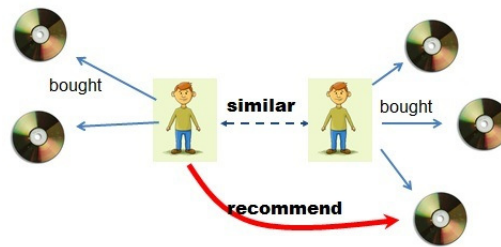


Figure 2.2: User based collaborative filtering (*Recommendation Engine* 2013)

- **Item based** collaborative filtering, produce a recommendation by comparing the similarities between a previously unseen item and the user's items (Figure 2.3).

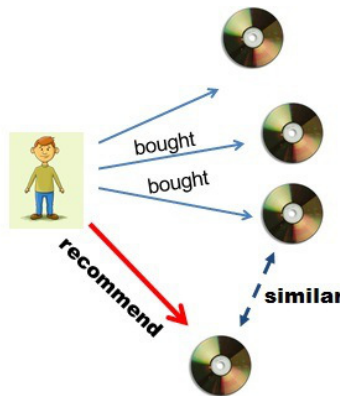


Figure 2.3: Item based collaborative filtering (*Recommendation Engine* 2013)

Similarities between a pair of users  $a, u$  are usually computed with Pearson correlation metric (Sarwar et al. 2001), given by Equation (2.1):

$$sim(a, u) = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2} \sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}} \quad (2.1)$$

where  $I$  is the set of items rated by both users,  $r_{u,i}$  is the rating given to item  $i$  by user  $u$ , and  $\bar{r}_u$  is the mean rating given by user  $u$ . Equivalently, for similarities between a pair of items  $i, j$ , the correlation is given by Equation (2.2):

$$sim(i, j) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}} \quad (2.2)$$

where  $U$  is the set of users who have rated both items,  $r_{u,i}$  is the rating given to item  $i$  by user  $u$ , and  $\bar{r}_i$  is the mean rating given to item  $i$ .

The strength of CF is that the recommendation process is independent of the item features (Burke 2002). On the other hand, CF would not be suitable technique when the user-item matrix is sparse. Moreover, CF considers only the most rated items, therefore, ignores the items in the long tail, and it is unable to handle the cold start problem. (Dai et al. 2014)

### **The cold start problem**

Recommendation process in CF might be difficult either for a user or an item with few ratings. (Burke 2002)

## The long tail phenomenon

The *long tail* items according to Yin et al. (2012) are referred to products with a low volume of sales but they can be more profitable than the popular items if they are recommended to the right consumers.

### 2.3.2 Content-based filtering

Content based (CB) filtering is based on the analysis of the features that describe the items. The recommendation component consists in matching up the attributes of the items that a user has already rated, usually referred as the *user profile* (Lops, Gemmis, and Semeraro 2011), against the attributes of a previously unseen products to produce a list of *top-N* recommendations. Figure 2.4 shows the architecture of content-based recommendation process.

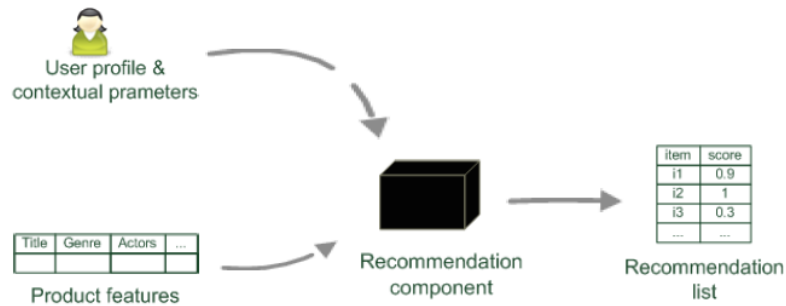


Figure 2.4: Content-based filtering process (Blog.seagatesoft.com 2015)

One of the strengths of CB filtering is that recommendation process is entirely based on the attributes of the items, thus, the recommendations produced for each user is independent from the other users information. Also, a CB recommender allows to recommend items that do not have any rat-

ings, therefore, they can diminish the effects of cold-start problem. (Lops, Gemmis, and Semeraro 2011)

### **Limitations of CB filtering**

One disadvantage of CB filtering is that personal reviews are not considered in the recommendation process, because this technique is limited to explicit representation of items (Celma 2008). Moreover, some representations limit the description to certain aspects only (Lops, Gemmis, and Semeraro 2011).

Another limitation of CB might be the collection of external data due to restricted access, e.g., the Million Song Dataset (Bertin-Mahieux et al. 2011) does not provide audio data due to copyright restrictions<sup>7</sup> and some preview clips are not available in the 7digital UK music catalogue.

In our project, a CB recommender is used as the baseline to show a improved performance in music recommendation. Please refer to Section 5.2 for more detail.

### **2.3.3 Item Representation**

Items require an accurate description to achieve upstanding results for recommending items to users (Celma 2008). In majority of the content-based filtering systems, item attributes are textual features extracted from web resources. (Lops, Gemmis, and Semeraro 2011)

In our approach, we describe the songs in terms of n-dimensional vectors. Each dimension in the vector represent the probability of the song to belong

---

<sup>7</sup><http://labrosa.ee.columbia.edu/millionsong/pages/can-i-contact-you-privately-get-audio>

to a music genre. The probability estimation is obtained from a music classifier implemented with a deep learning technique. The song representation process is illustrated in section 3.3.1

#### **2.3.4 User Modelling**

“User modeling [sic] is a discipline that deals with both how information about the user can be acquired and used by an automated system.” (*Recommender Systems* 2012)

Modelling a user profile consists of designing a structure for recording the interests which describe a user. There are several techniques for modelling an user profiles: vector, connexion, ontology and multidimensional representation. (Bouneffouf 2013)

In our project, we model each user profile through EDAs by minimising a fitness function. The parameters of the fitness function are the rating and similarity values of each song that a user has listened. The user profile is also represented in a n-dimensional vector of probabilities of music genres. This process is illustrated in section 3.3.2

#### **2.3.5 Hybrid recommender approaches**

An hybrid recommender system is developed through the combination of the recommendation techniques mentioned in the previous sections. Usually, hybrid approaches boost the advantages of CF by considering the user’s feedback and the advantages of CB by taking into count the item attributes.

According to Burke (2002), there are the following combination methods

to accomplish hybridisation:

- **Weighted** method, where a single item recommendation is computed as a linear combination of the recommendation value from each technique involved. The weight assigned to each recommender can be adjusted by considering additional feedback from the user.
- **Switching** method, where the hybrid system uses a criteria depending on the input data to switch between recommendation techniques implemented in the system.
- **Mixed** method, where recommendations from several different types of recommender are presented simultaneously.
- **Feature combination** method, where CF results are treated as additional attributes of a CB filtering recommender.
- **Cascade** method, where one recommender refines the coarse recommendations set given by the first recommender. This method is more efficient than the weighted method, because cascade implementation do not process every item at each stage.
- **Feature augmentation** method, where the rating of an item from one recommender is used as an input feature of another recommendation technique.
- **Meta-level** method, where a model generated for user's interest representation using one recommendation technique is used as the input of another recommender system. The advantage of this method is the

performance of the second recommender that uses the compressed representation instead of sparse raw data.

The hybrid music recommender approach in this project can be considered as implementation of feature augmentation method and a meta-level method. The general model of our hybrid recommender is explained in detail in Section 3.3.

## 2.4 Music Information Retrieval

Music Information Retrieval (MIR) (Casey et al. 2008) is a field of research for better human understanding of music data in an effort to reduce the *semantic gap* (Celma, Herrera, and Serra 2006) between high-level musical information and low-level audio data. Applications of MIR include artist identification, genre classification and music recommender systems (Weston et al. 2012; Yoshii et al. 2008).

### 2.4.1 Genre classification

Music classification is one of the main tasks in MIR for clustering audio tracks based on similarities between features of pieces of music. Automatic musical genre classification approach proposed by Tzanetakis and Cook (2002), which uses GTZAN genre dataset<sup>8</sup>, has been widely used in the past decade. The GTZAN dataset consists of a total of 1,000 clips, corresponding to 100 examples for each of the 10 music genres: blues, classical, country, disco,

---

<sup>8</sup><http://marsyas.info/downloads/datasets.html>



hiphop, jazz, metal, pop, reggae and rock. The total duration of each clip is 30 seconds.

Nonetheless, the GTZAN dataset has inaccuracies (Sturm 2012), it still provides an useful baseline to compare genre classifiers.

## 2.4.2 Music recommender systems

### Collaborative retrieval music recommender

Weston et al. (2012) proposed a latent *collaborative retrieval* algorithm using the *Last.fm Dataset - 1K users*<sup>9</sup> dataset. For each (artist, song) tuple in the dataset, they computed the audio data using 39-dimensional vector corresponding to 13 Mel Frequency Cepstral Coefficients (MFCCs), and the first and the second derivatives. The vectors obtained are used to build up a dictionary using the K-means algorithm. Each audio frame is represented with a vector that contains the number of occurrences of a dictionary vector in the frame. The collaborative retrieval algorithm present outperforming results compared with the Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF) methods used on collaborative filtering recommendation tasks.

### Hybrid music recommender

Yoshii et al. (2008) proposed a hybrid recommender system considering rating scores collected from Amazon.co.jp and acoustic features derived from the signals of musical pieces corresponding to Japanese CD singles that were

---

<sup>9</sup><http://www.dtic.upf.edu/ocelma/MusicRecommendationDataset/lastfm-1K.html>

ranked in weekly top-20 from April 2000 to December 2005. Acoustic features for each piece are represented as a *bag-of-timbres*, i.e., a set of weights of polyphonic timbres, equivalent to a 13-dimensional MFCC representation. Bags of timbres are computed with a Gaussian Mixture Model, considering the same combination of Gaussians for all the pieces.

A three-way aspect model (see Figure 2.5) is used to decompose the joint probability of users  $U$ , pieces  $M$  and features  $T$  into a set of latent genre variables  $Z$ . It is assumed that user  $u$  stochastically choose a genre  $z$  according to their preferences and then the genre  $z$  stochastically generates a piece of music  $m$  and an acoustic feature  $t$ .

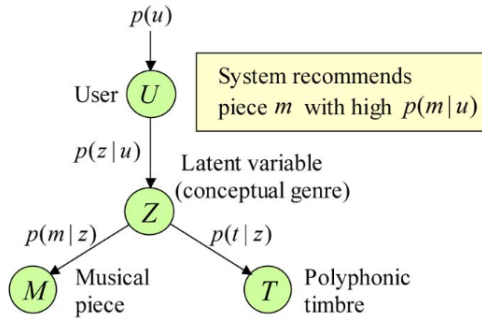


Figure 2.5: Three-way aspect model (Yoshii et al. 2008)

The results of the comparative experiments revealed that three-way aspect hybrid method outperformed the CF and CB recommendation techniques in terms of accuracy considering  $|T| = 64$  features and  $|Z| = 10$  latent variables.

## 2.5 Deep Learning

High-level features that help us make sense of an observed data., e.g. genre, mood or release time in a music library could be difficult to compute. Deep learning algorithms allows us to build complex concepts out of simpler concepts (Bengio, Goodfellow, and Courville 2015). Deep learning can solve the difficulty of representing high-level features, e.g., perceived genre in a piece of music, by expressing them in terms of low-level signal features, e.g. spectrum, frequency or pitch.

In MIR, deep learning methods capture the attention of researchers for the following reasons (Kereliuk, Sturm, and Larsen 2015):

- Hierarchical representations of structures in data.
- Efficient feature learning and classification algorithms.
- Open and publicly available implementations, e.g., *Theano* (Bastien et al. 2012; Bergstra et al. 2010) library for Python.

These advantages of deep learning methods enable us to learn abstractions from music low-level content in order to reduce the *semantic gap* (Celma, Herrera, and Serra 2006) in MIR. Additionally, feature extraction does not require significant domain knowledge compared to *hand-crafted* engineering. Nonetheless, deep learning implementations require a lot of data.

### 2.5.1 Deep Neural Networks

A deep neural network (DNN) (Hinton et al. 2012) is defined as a feed-forward artificial neural network (ANN), or multi-layer perceptron (MLP),

with more than one layer of hidden units between the input and the output layer (see Figure 2.6).

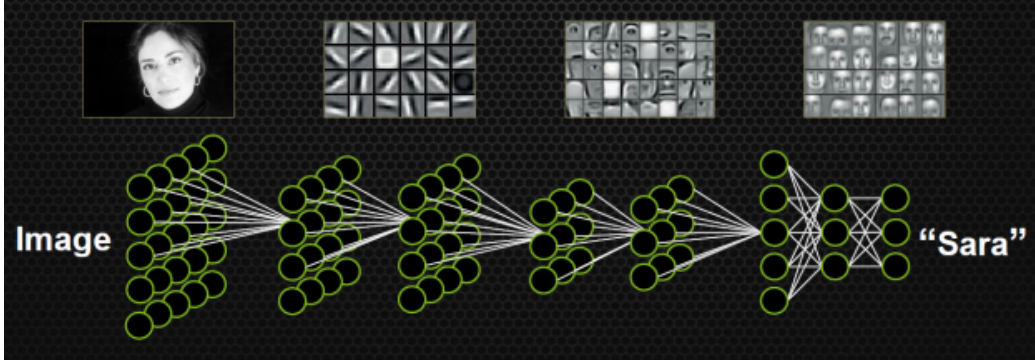


Figure 2.6: Schematic representation of a deep neural network (Brown 2014)

Each hidden unit  $j$  maps its total input from the layer below  $x_j$ , given by Equation (2.3)

$$x_j = b_j + \sum_i y_i w_{ij} \quad (2.3)$$

where  $b_j$  is the bias of unit  $j$ ,  $i$  is an index over units in the layer below, and  $w_{ij}$  is the weight on a connection to unit  $j$  from unit  $i$  in the layer below, to a scalar value  $y_j$  that is directed to the layer above. The activation function of hidden units can be hyperbolic tangent, logistic or rectifier linear activation function. For classification, output unit  $j$  converts its total input  $x_j$  into a class probability  $p_j$  by using the *softmax* nonlinearity, given by Equation (2.4)

$$p_j = \frac{\exp x_j}{\sum_k \exp x_k} \quad (2.4)$$

where  $k$  is the number of classes.

## Music Feature Learning

Sigtia and Dixon (2014) examined and compared DNNs to discover features from the GTZAN dataset and the ISMIR 2004 genre classification dataset<sup>10</sup>, using rectifier linear units (ReLU) and dropout regularisation. The ReLU activation function is defined as  $\max(x, 0)$ . ReLU provides better convergence without pre-training. Dropout regularisation reduces the problem of overfitting.

First, the GTZAN dataset was divided into four 50/25/25 train, validation, test parts. For each audio clip of the dataset, they calculated the Fast Fourier Transform (FFT) on frames of length 1,024 samples (22,050 kHz sampling rate) with a window overlap of 50%. Next, they used the magnitude of each FFT frame resulting in a 513 dimensional vector. And then, each feature dimension is normalised to have zero mean and unit standard deviation.

For the deep neural network, the 500 hidden units were trained with stochastic gradient descent (SGD) with a learning rate of 0.01, a patience of 10 and a dropout rate of 0.25.

The system classifies the GTZAN data with an accuracy of  $83 \pm 1.1\%$ , a value of the same order of results obtained with hand-crafted features.

### 2.5.2 Convolutional Deep Neural Networks

Inspired in the behaviour of animal visual processing system (Deeplearning.net 2015), a convolutional deep neural network (CDNN) (Bengio, Good-

---

<sup>10</sup>[http://ismir2004.ismir.net/genre\\_contest/](http://ismir2004.ismir.net/genre_contest/)

fellow, and Courville 2015; Ufdl.stanford.edu 2015) is a type of MLP that uses convolution operation instead of matrix multiplication for processing data that has grid-like topology. CDNNs are designed to recognize visual patterns directly from pixel images. In general, the architecture of a CDNN is based on:

- **Sparse connectivity:** the inputs of hidden units in an upper layer  $m$  are from a subset of units in a lower layer  $m-1$ .
- **Shared weights:** each hidden unit in a layer share the same weight vector and bias. The layer with this parametrisation form a *feature map*.
- **Convolutional layers:** A feature map is obtained by convolution of the input image with a linear filter, adding a bias term and then applying a non-linear function. Each convolutional (hidden) layer is composed of multiple feature maps.
- **Max-pooling:** is a non-linear down-sampling to divide the input image into a set of non-overlapping rectangles and, for each rectangle, the maximum value is returned.

LeNet-5 (LeCun 2015) is one model of CDNN designed for recognition of handwritten and machine-printed characters. In Figure 2.7, the LeNet model is illustrated. The lower-layers are composed of convolution and max-pooling layers and the upper-layer is a fully-connected MLP. The input to MLP is the set of all features maps at the layer below.

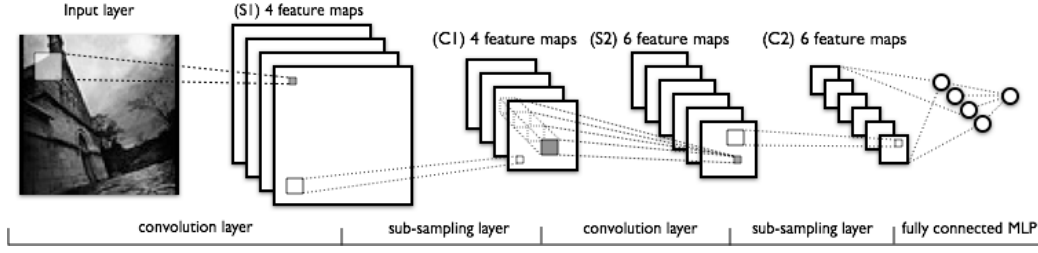


Figure 2.7: Convolutional deep neural network LeNet model (Deeplearning.net 2015)

### Deep content-based music recommendation

Oord, Dieleman, and Schrauwen (2013) proposed to use a latent factor model for CB recommendation and the implementation of a CDNN to predict the latent factors from music audio. To obtain 50-dimension latent vectors, they used a weighted matrix factorisation (WMF) algorithm on the Taste Profile Subset. Also, they retrieved audio clips for over 99% of the songs in the dataset from 7digital.com.

To train the CDNN, the latent vectors obtained through the WMF are used as ground truth. The input of the CDNN is a log-compressed mel-spectrogram with 128 components computed from windows of 1,024 samples and a hop size of 512 samples (sampling rate of 22,050 Hz) for each audio clip. The duration of audio clips is limited to 3 seconds.

They used 10-fold cross validation and obtained an average area under the ROC curve (AUC) of 0.86703 for prediction based on the latent factor vectors, outperforming the bag-of-timbres approach.

## 2.6 Estimation of Distribution Algorithms

Inspired in natural selection of species, an estimation of distribution algorithm (EDA) (Pelikan, Hauschild, and Lobo 2015; Ding, Ding, and Peng 2015; Santana et al. 2010) is an optimisation technique that estimates a probabilistic model from a sample of promising individuals, which is used to generate a new population and leading to an optimal solution of an objective function, called the *fitness* function, until a termination criteria, e.g., maximisation, minimisation, maximum number of generations, is satisfied. These algorithms were applied to solve complex problems such as load balancing for mobile networks (Hejazi and Stapleton 2015) or software reliability prediction (Jin and Jin 2014). In Figure 2.8 we show the general flowchart of an EDA.

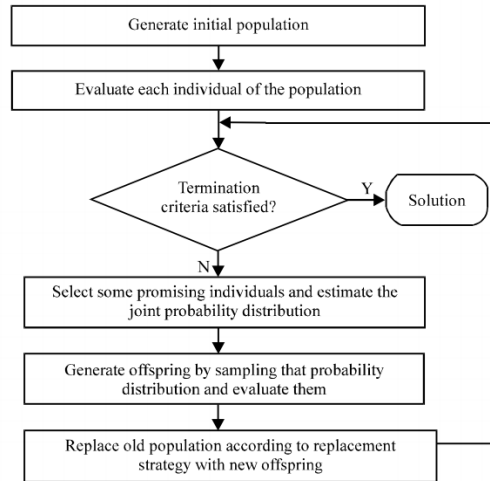


Figure 2.8: Flowchart of estimation of distribution algorithm (Ding, Ding, and Peng 2015)

According to Pelikan, Hauschild, and Lobo (2015), the main components of an EDA are a selection operator, a class of probabilistic models for mod-



elling and sampling, and a replacement operator for combining the old population with the offspring. Also, regarding the types of distributions that an EDA are able to capture (Pelikan, Hauschild, and Lobo 2015) can be categorised in four broad groups:

- **Discrete variables EDAs**, where candidate solutions are represented by fixed-length strings of a finite cardinality.
- **Permutation EDAs**, where candidate solutions are represented by permutations over a given set of elements.
- **Real-valued vectors (continuous) EDAs**, where candidate solutions are mapped from real-valued variables into a discrete domain or the probabilistic model defined on real-valued variables are considered.
- **Genetic programming EDAs**.

The advantages of using EDAs include the discovery of problem-specific features or reducing the memory requirements. However, it is time consuming to build explicit probabilistic models. (ibid.)

In our project, we investigate permutation EDAs and continuous EDAs for user profile modelling.

### 2.6.1 A Hybrid Recommendation Model Based on EDA

Liang et al. (2014) exploited a permutation EDA to model user profiles in an hybrid model for movie recommendation using the MovieLens 1M dataset<sup>11</sup>.

---

<sup>11</sup><http://grouplens.org/datasets/movielens/>

A movie,  $i$ , is described using a vector,  $t_i = \{(k_1, w_1), \dots, (k_n, w_n)\}$ , where the keywords  $k_n$  and weights  $w_n$  are calculated with term frequency-inverse document frequency (TF-IDF) technique. A user is initially represented by a set,  $S_u = \{(t_1, r_{u,1}), \dots, (t_i, r_{u,i}) | r_{u,i} > \bar{r}_u\}$ , where,  $r_{u,i}$  is the rating of the movie  $i$  given by user  $u$ , and  $\bar{r}_u$  is a threshold. The keywords in every  $S_u$  set are embedded in a new set,  $D_u$ .

The goal is to learn the user profile,  $profile_u = \{(k_1, w_1), \dots, (k_n, w_n)\}$ , by minimisation of the fitness function, defined by Equation (2.5)

$$fitness(profile_u) = \sum_{i \in S_u} \log(r_{u,i} \times sim(profile_u, t_i)) \quad (2.5)$$

where  $sim(profile_u, t_i)$  is computed by the cosine similarity coefficient, defined by Equation (2.6)

$$sim(profile_u, t_i) = cos(profile_u, t_i) = \frac{profile_u \cdot t_i}{\|profile_u\| \times \|t_i\|} \quad (2.6)$$

The pseudocode of EDA implemented by Liang et al. (2014) is delineated by Algorithm 1, where MAXGEN is the maximum number of generations.

To recommend a new movie,  $j$ , to a user, the similarity between the user profile,  $u_i$ , and the movie vector,  $t_j$ , is calculated using Pearson correlation coefficient, defined by Equation (2.7):

$$sim(u_i, t_j) = \frac{\sum_{c \in I_i \cap I_j} (w_{i,c} - \bar{w}_i)(w_{j,c} - \bar{w}_j)}{\sqrt{\sum_{c \in I_i \cap I_j} (w_{i,c} - \bar{w}_i)^2} \sqrt{\sum_{c \in I_i \cap I_j} (w_{j,c} - \bar{w}_j)^2}} \quad (2.7)$$

where,  $c \in I_i \cap I_j$  are the keywords in common between the user profile and

---

**Algorithm 1** Calculate  $profile_u$ 

---

**Require:** set  $D_u$ , weights  $w_{n,i}$

**Require:** population size  $N$ , MAXGEN

Random selection of keywords  $k_n$  from  $D_u$

Assign a weight  $w_{n,i}$  to each  $k_n$  to build a set  $K_u$  of size  $N$

Assign a probability  $c_{n,i} = 1/N$  to each  $(k_n, w_{n,i})$

Generate initial population of  $profile_u$  by Monte Carlo method

**while**  $generation < \text{MAXGEN}$  **do**

    Compute each  $fitness(profile_u)$

    Rank individuals by their fitness value

    Select top  $M < N$  individuals

    Update  $c_{n,i}$  by counting the occurrences of  $(k_n, w_{n,i})$  in the  $M$  individuals profiles

    Generate  $profile_u$  by random sampling according to updated  $c_{n,i}$

**end while**

**return**  $profile_u$

---

the new movie vector,  $w_{i,c}$  and  $w_{j,c}$  are the weights of keyword  $c$  in the user profile and movie vector,  $\bar{w}_i$  is the mean weight of user profile and  $\bar{w}_j$  is the mean weight of movie vector.

In our approach, we use the algorithm proposed by Liang et al. (2014) to model user profiles but considering probability values of music genres instead of weight values of keywords. The adapted algorithm is explained in subsection 3.3.2.

## 2.6.2 Continuous Univariate Marginal Distribution Algorithm

Gallagher et al. (2007) presented the continuous univariate marginal distribution algorithm ( $UMDA_c^G$ ) as an extension of a discrete variable EDA. The general pseudocode of the  $UMDA_c^G$  is delineated in Algorithm 2, where

$x_i \in \mathbf{x}$  represent the  $i$  parameter of  $\mathbf{x}$  individual solution.

---

**Algorithm 2** Framework for  $UMDA_c^G$

---

**Require:** population size  $M$

**Require:** selection parameter  $\tau$

$t \leftarrow 0$

Generate  $M$  individuals at random

**while**  $t < \text{stopping criteria}$  **do**

$M_{sel} \leftarrow M \cdot \tau$

    Select  $M_{sel}$  individuals

$\mu_{i,t} \leftarrow \frac{1}{M_{sel}} \sum_{j=1}^{M_{sel}} x_i^j$

$\sigma_{i,t}^2 \leftarrow \frac{1}{M_{sel}-1} \sum_{j=1}^{M_{sel}} (x_i^j - \mu_{i,t})^2$

$p_t(x_i | \mu_{i,t}, \sigma_{i,t}^2) \leftarrow \frac{1}{\sqrt{2\pi}\sigma_{i,t}} \exp(-\frac{1}{2}(\frac{x_i - \mu_{i,t}}{\sigma_{i,t}})^2)$

    Sample  $M$  individuals from  $p_t(x_i | \mu_{i,t}, \sigma_{i,t}^2)$

$t \leftarrow t + 1$

**end while**

---

To our knowledge, our hybrid recommender design is the first work to consider a continuous EDA for user profile modelling in a recommender system. The implementation of the continuous EDA is explained in subsection 3.3.2.

## 2.7 Summary

In this chapter, previous work on recommender systems has been reviewed and novelty techniques to representing acoustical features and to model user profiles has been presented. The next steps are to collect the dataset by crawling online social information, to extract the acoustical features of a collection of songs to represent them as n-dimensional vectors, to model the user profiles by using EDAs, and therefore, to return a list of song recommendations.

# Chapter 3

## Methodology

The methodology used to develop our hybrid music recommender consists of four main stages. First, the collection of real world user-item data corresponding to the play counts of specific songs and the fetching of audio clips of the unique identified songs in the dataset. Secondly, the implementation of the CDNN to represent the audio clips in terms of music genre probabilities as n-dimensional vectors. Next, permutation EDA and a continuous EDA are investigated to model user profiles based on the rated songs above a threshold. Finally, the process of top-N recommendation for the baseline and the hybrid recommender is described.

Every stage of our hybrid recommender is entirely developed in Python 2.7<sup>1</sup>, although, they are implemented in different platforms, e.g., OS X (v10.10.4) for the most part of the implementation, Ubuntu (14.04 LTS installed on VirtualBox 5.0.0) for intermediate time-frequency representation and CentOS (Linux release 7.1.1503) for the data preprocessing and CDNN

---

<sup>1</sup><https://www.python.org/download/releases/2.7/>

implementation.

## 3.1 Data collection

The Million Song Dataset (Bertin-Mahieux et al. 2011) is a collection of audio features and metadata for a million contemporary popular music tracks which provides ground truth for evaluation research in MIR. This collection is also complemented by the Taste Profile subset which provides 48,373,586 triplets, each of them consist of anonymised user ID, Echo Nest song ID and play count. We choose this dataset because it is publicly available data and it contains enough data for user modelling and recommender evaluation.

### 3.1.1 Taste Profile subset cleaning

Due to potential mismatches<sup>2</sup> between song ID and track ID on the Echo Nest database, it is required to filter out the wrong matches in the Taste Profile subset. The cleaning process is illustrated in Figure 3.1

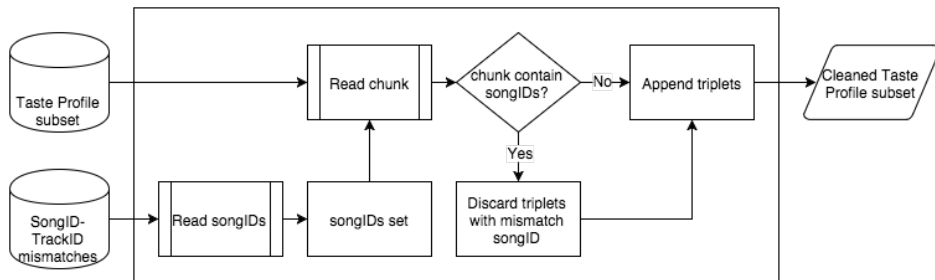


Figure 3.1: Diagram of the cleaning process of the Taste Profile subset

A script is implemented to discard the triplets that contain the song identifiers from the mismatches text file. First, we load the file to read each

<sup>2</sup><http://labrosa.ee.columbia.edu/millionsong/blog/12-2-12-fixing-matching-errors>

line of it to obtain song identifier. The identifiers are stored as elements of a set object to construct a collection of unique elements. Next, due to the size of the Taste Profile subset (about 3 GB, uncompressed), we load the dataset by chunks of 20,000 triplets in a *pandas*<sup>3</sup> dataframe to clean each chunk by discarding the triplets that contains the song identifiers in the set object of the previous step. The cleaning process takes around 2.47 minutes and we obtain 45,795,100 triplets.

In addition to the cleaning process, we reduce significantly the size of the dataset for experimental purposes. We only consider users with more than 1,000 played songs and select the identifiers of 1,500 most played songs. This additional process takes around 3.23 minutes and we obtain 65,327 triplets. The triplets are stored in a cPickle<sup>4</sup> data stream (2.8 MB).

### 3.1.2 Fetching audio data

First, for each element of the list of 1,500 songs identifiers obtained in the previous step is used to retrieve the associated Echo Nest track ID through a script using the *get\_tracks* method from the *Pyechnest*<sup>5</sup> package which allow us to acquire track ID and preview URL for each song ID through Echo Nest API. The reason behind this is 7digital API uses Echo Nest track ID instead of song ID to retrieve any data from its catalogue. If the track information of a song is not available, the script skips to retrieve the Echo Nest information of the next song ID. At this point, it is useful to check if the provided 7digital API keys, a preview URL, and the country parameter, e.g., 'GB' to access to

---

<sup>3</sup><http://pandas.pydata.org/>

<sup>4</sup><https://docs.python.org/2/library/pickle.html#module-cPickle>

<sup>5</sup><http://echonest.github.io/pyechonest/>

UK catalogue, work in the *OAuth 1.0 Signature Reference Implementation*<sup>6</sup>.

Next, for each preview URL, we can create a GET request using *python-oauth2*<sup>7</sup> package, because it allows us to assign the nonce, the timestamp, the signature method and the country parameters. The request is converted to a URL to be opened with *urlopen* function from the *urllib2*<sup>8</sup> module, to download a MP3 file (44.1 kHz, 128 kbps, stereo) of 30 to 60 seconds of duration in a song repository.

Considering the Echo Nest API and 7digital API limited number of requests (see Section 2.2), the process of fetching data from 1,500 song IDs takes at least 8 hours, resulting in a total of 640 MP3 files.

Additionally, the script accumulates the Echo Nest song identifier, track ID, artist name, song title and the 7digital preview audio URL for each downloaded track in a text file only if the audio clip is available for download. The generated text file is used for the preprocessing of the cleaned taste profile dataset in subsection 3.2.1. The flowchart of the script is shown in Figure 3.2

### 3.1.3 Intermediate time-frequency representation for audio signals

Intermediate audio representation instead of waveform (time-domain) representation is required to feed a CDNN according to Oord, Dieleman, and Schrauwen (2013). The flowchart to obtain the time-frequency representa-

---

<sup>6</sup><http://7digital.github.io/oauth-reference-page/>

<sup>7</sup>[https://github.com/jasonrubenstein/python\\_oauth2](https://github.com/jasonrubenstein/python_oauth2)

<sup>8</sup><https://docs.python.org/2/library/urllib2.html>



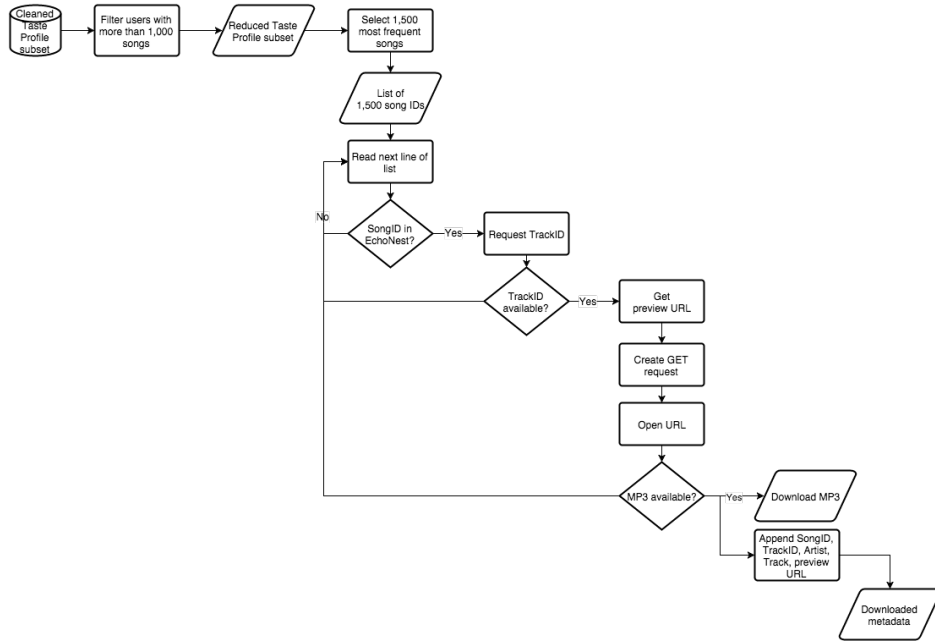


Figure 3.2: Flowchart of the fetching audio process

tion from raw audio content of the song repository assembled in the previous section is shown in Figure 3.3.

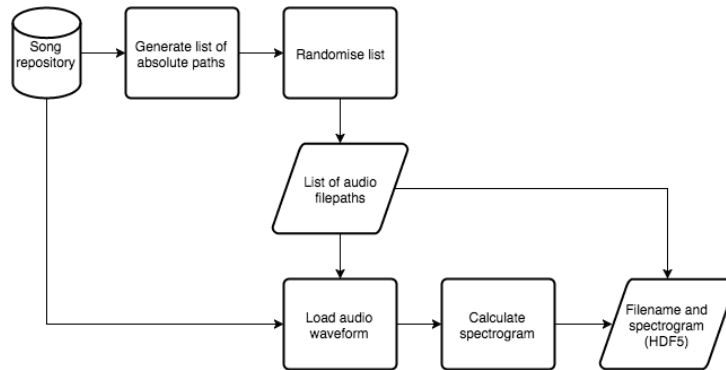


Figure 3.3: Flowchart for time-frequency representation process

First, a list of absolute paths corresponding to the songs in the repository is generated. The sequence of paths in the list is modified by random shuffling. This new sequence of absolute paths is saved in a text file.

Second, for every path in the text file of randomised absolute paths, a fragment equivalent to 3 seconds of the associated audio clip is loaded at a sampling rate of 22,050 Hz and converted to mono channel. For every fragment, a mel-scaled power spectrogram with 128 bands is computed from windows of 1,024 samples with a hop size of 512 samples, resulting in a spectrogram of 130 frames with 128 components. Hence, the spectrogram is converted to logarithmic scale in dB using peak power as reference. The functions *load*, *feature.melspectrogram* and *logamplitude*, correspondingly to load an audio clip, spectrogram computation and logarithmic conversion, from the LibROSA<sup>9</sup> package are used.

To handle audio with LibROSA functions, it is recommended to use the Samplerate<sup>10</sup> package for efficient resampling. In our project, we considered to use the SoX<sup>11</sup> cross-platform without success due to operating system restrictions. Alternatively, we use the FFmpeg<sup>12</sup> cross-platform and *libmp3lame0*<sup>13</sup> packages for efficient resampling.

Finally, we store the absolute path and the log-mel-spectrogram values of the 640 songs in a HDF5<sup>14</sup> data file.

In the particular case for the time-frequency representation of each audio clip in the GTZAN dataset, we generate a list of the genre associated to each audio fragment that represent the target values (ground truth). This procedure for the GTZAN dataset is repeated for 9 times, considering the

---

<sup>9</sup><https://bmcfee.github.io/librosa/index.html>

<sup>10</sup><https://pypi.python.org/pypi/scikits.samplerate/>

<sup>11</sup><http://sox.sourceforge.net/>

<sup>12</sup><https://www.ffmpeg.org/>

<sup>13</sup><http://packages.ubuntu.com/precise/libmp3lame0>

<sup>14</sup><https://www.hdfgroup.org/HDF5/>

rest of 3-seconds fragments in each audio clip of the dataset for training, validation and testing of the CDNN (see Section 3.3.1)

The time elapsed to obtain the time-frequency representation of the clips in the GTZAN dataset with the procedure described above is about 55 seconds, generating a HDF5 file (66.9 MB). Because of the number of MP3 files in the song repository is less than the number of files of the GTZAN dataset, the process is faster and the size of the HDF5 file is smaller (42.8 MB).

## **3.2 Data preprocessing**

In order to obtain suitable representations for users' interest in the taste profile dataset and for songs' spectrograms, it is necessary an additional process of the data.

### **3.2.1 Rating from implicit user feedback**

First, the text file of the downloaded MP3 metadata (see subsection 3.1.3) is used to retain the triplets, from the cleaned taste profile subset, that contain the song IDs of the available audio clips. A reduced taste profile dataset with 4,685 triplets is obtained, corresponding to information of 53 users.

The reduced taste profile dataset represent the user listening habits as implicit feedback, i.e., play counts of songs, it is necessary to normalise the listening habits as explicit feedback, i.e., range of values  $[1 \dots 5]$  that indicate how much a user likes a song. Normalisation of play counts is computed with the complementary cumulative distribution of play counts of a user, following the procedure given by Celma (2008). Songs in the top 80 - 100% of the

distribution get a rating of 5, songs in the 60 - 80% range get a 4, songs in the 40 - 60% range get a 3, songs in the 20 - 40% get a 2 and songs in the 0 - 20% range get a rating of 1. An exception for this allocation of ratings comes out when the coefficient of variation, given by Equation (3.1):

$$CV = \frac{\sigma}{\mu} \quad (3.1)$$

where,  $\sigma$  is the standard deviation and  $\mu$  is the mean of the play counts of a user, is less or equal than 0.5. In that case, every song gets a rating of 3.

### 3.2.2 Standardise time-frequency representation

The logarithmic mel-scaled power spectrograms obtained in subsection 3.1.3 are normalised to have zero mean and unit variance in each frequency band, using the *fit* and *transform* methods of the *StandardScaler* class from the Scikit-learn (Pedregosa et al. 2011) package, as a common requirement of several machine learning classifiers.

Additionally, the GTZAN normalised spectrograms dataset is split in 3 subsets: 500 spectrograms for training, 250 spectrograms for validation and 250 spectrograms for testing. Each spectrogram is saved as a tuple (*spectrogram*, *tag*) in a cPickle file, where tag is the number of the music genre: 0 for blues, 1 for classical, 2 for country, 3 for disco, 4 for hiphop, 5 for jazz, 6 for metal, 7 for pop, 8 for reggae and 9 for rock.

### 3.3 Algorithms

The hybrid music recommender approach in this project can be considered as implementation of feature augmentation method and a meta-level method presented in subsection 2.3.5. First, user profiles are generated using the rating matrix and the song vector representation. Next, the model generated is the input of a CB recommender to produce *top-N* song recommendations. The general model of our hybrid recommender is shown in Figure 3.4

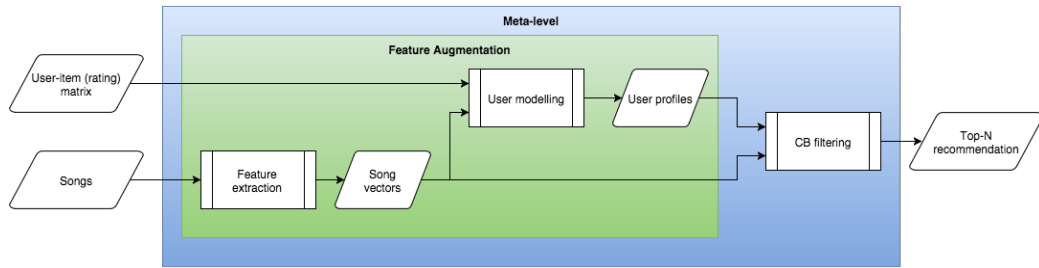


Figure 3.4: Diagram of the hybrid music recommender

#### 3.3.1 Probability of music genre representation

To represent an audio file in a 10-dimensional vector, whose dimensions correspond to the 10 music genres specified in the GTZAN dataset, a CDNN is implemented using Theano library. For intensive computation processes, such as convolution, the implementation on equipment with Graphical Processing Unit (GPU) acceleration is recommended. In this project, a CentOS (Linux release 7.1.1503) server with a Tesla K40c<sup>15</sup> GPU is exploited.

The scripts for logistic regression, multilayer perceptron and deep convolutional network designed for character recognition of MNIST<sup>16</sup> dataset,

<sup>15</sup><http://www.nvidia.com/object/tesla-servers.html>

<sup>16</sup><http://www.iro.umontreal.ca/~lisa/deep/data/mnist/mnist.pkl.gz>

available on Deeplearning.net (2015) is adapted to our purpose of music genre classification. ReLU and dropout functions are defined in the deep convolutional network script.

### CDNN architecture

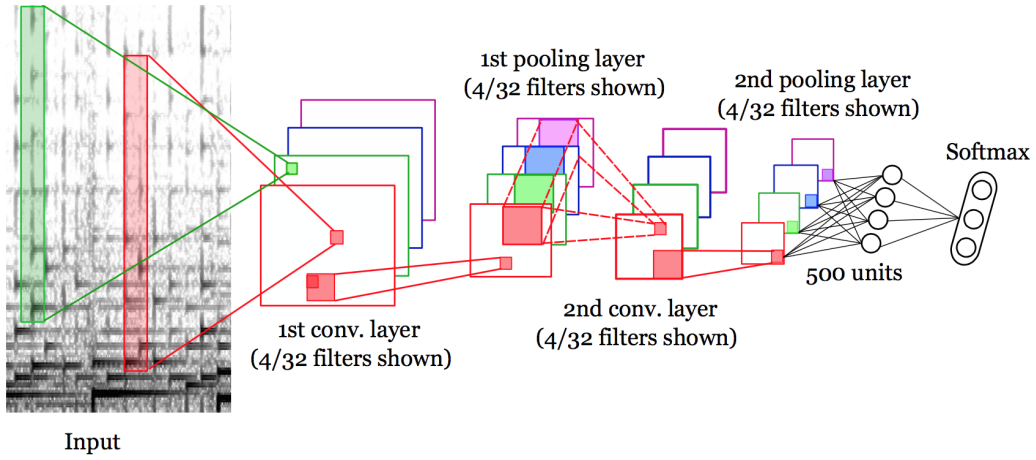


Figure 3.5: Diagram of CDNN for music genre classification (Kereliuk, Sturm, and Larsen 2015)

A similar architecture of a CDNN for music genre classification (Kereliuk, Sturm, and Larsen 2015) is recreated in our project. A batch size of 20 and a dropout rate of 0.20 for the convolutional layer units are considered.

Initially, the reshape of the 2-dimension normalised spectrograms (130 frames $\times$ 128 frequency bands) obtained in subsection 3.2.2 to a 4-dimension tensor, compatible with the input of the first convolutional layer (batch size $\times$ 1 $\times$ 130 $\times$ 128), is required.

The first convolutional layer consists of 32 filters, each one with a size of 8 frames, with a max-pooling downsampling of 4, to reduce the size of the spectrogram along the time axis. The size of the resulting spectrogram

is  $30 \times 128$  and the output of this first convolutional layer is a 4-dimension tensor with a size of  $20 \times 32 \times 30 \times 128$ .

The second convolutional layer consists of 32 filters, each one with a size of 8 frames, with a max-pooling downsampling of 4, to reduce the size of the spectrogram obtained in the first layer. The size of the new spectrogram is  $5 \times 128$  and the output of this second convolutional layer is a 4-dimension tensor with a size of  $20 \times 32 \times 5 \times 128$ .

Following the convolution process, the reshape of the 4-dimensional tensor of the output of the second convolutional layer is required to feed the fully connected MLP. The MLP consists of 500 ReLUs.

Finally, the classification of music genre is accomplished with logistic regression layer of the 500 output values from the MLP. This output layer consists of 10 units with softmax activation function (see Equation (2.4)).

### **Learning parameters**

The weights and biases of the units of the CDNN are the parameters to be modelled by SGD to minimise a cost function. The cost function is the negative log likelihood of the prediction in the output layer given the target values, i.e., music genre ground truth.

The CDNN for training, validation and testing is run for 200 epochs, each epoch equivalent to 50 iterations. The number of iterations corresponds to the ratio between the number of spectrograms (1,000 for GTZAN dataset) and the batch size.

According to Bengio (2012), the patience value is the minimum number of training examples. In our project, the patience value is set at 1,000.

In our testing, after 9 trials in the CDNN, we obtain a best classification error of 38.8 % using the spectrograms corresponding to the GTZAN dataset (see Table 5.1). The weights and biases for this best classification error are saved in a cPickle file to be applied as initial parameters of the CDNN for vector representation.

### Vector representation

The script of CDNN is adapted to produce the vector representation of the spectrograms. This CDNN uses the weights and biases learnt in genre classification process as initial parameters.

A 10-dimension vector is produced by the softmax output layer. Each dimension corresponds to a music genre and each value represents the probability of a song to belong to a specific music genre, given the normalised spectrogram at the input layer.

### 3.3.2 User profile modelling

To model user profiles from the triplets in the normalised taste profile dataset, we adapt the permutation EDA (see algorithm 1 on page 26) and the continuous EDA (see algorithm 2 on page 27). For both EDAs, we consider the following:

- User representation  $S_u = \{(t_1, r_{u,1}), \dots, (t_i, r_{u,i}) | r_{u,i} > \bar{r}_u\}$ .
- Rating threshold  $\bar{r}_u = 2$ , assuming that a user does not like songs with ratings of 1 and 2 out of 5.



- The stopping criteria is the maximum number of generations limited to 250.

### Modelling with Permutation EDA

In the case of permutation EDA, the genre tags (0 for blues, 1 for classical, 2 for country, 3 for disco, 4 for hiphop, 5 for jazz, 6 for metal, 7 for pop, 8 for reggae and 9 for rock) are considered as the keywords  $k_n$  in the set  $D_u$  and the weights  $w_{n,i}$  are 50 evenly spaced samples over the interval  $[0.1, 0.9]$ , thus, the size of the set  $K_u$  is  $N = 500$  and the initial probability is  $c_{n,i} = 1/500$ .

The population size is equal to  $u = 53$ , that is the number of users in the normalised taste profile dataset. Instead of using the Monte Carlo method to generate the initial population of  $profile_u$ , 10 tuples  $(k_n, w_{n,i})$  from  $K_u$  are random sampled for each user. The number of top individuals  $M$  is a half of the total of users. The process of sampling new individuals is preserved. The adapted permutation EDA for user modelling is illustrated in Algorithm 3:

The time elapsed for modelling user profiles with the permutation EDA is approximately 7.82 seconds.

### Modelling with $UMDA_c^G$

The  $UMDA_c^G$  algorithm is adapted to select the top  $M_{sel}$  individuals by using the fitness function (Equation (2.5) on page 25) exploited by the permutation EDA. The population size is  $M = 53$  users, the selection parameter is  $\tau = 0.5$ .  $x_i$  represent the probability value of the music genre dimension,  $i$ , in the  $profile_u$  vector.

In each generation,  $t$ , the mean value  $\mu_{i,t}$  and the variance  $\sigma_{i,t}^2$  is com-

---

**Algorithm 3** Calculate  $profile_u$  for users in taste profile

---

**Require:** set  $D_u$ , weights  $w_{n,i}$

**Require:** population size  $u$ , MAXGEN

**Require:**  $M = Round(u/2)$

Assign a weight  $w_{n,i}$  to each  $k_n$  to build a set  $K_u$  of size  $N$

Assign a probability  $c_{n,i} = 1/N$  to each  $(k_n, w_{n,i})$

Generate initial population of  $profile_u$

**while**  $generation < MAXGEN$  **do**

    Compute each  $fitness(profile_u)$

    Rank individuals by their fitness value

    Select top  $M < N$  individuals

    Update  $c_{n,i}$  by counting the occurrences of  $(k_n, w_{n,i})$  in the  $M$  individuals profiles

    Generate  $profile_u$  by random sampling according to updated  $c_{n,i}$

$generation \leftarrow generation + 1$

**end while**

**return**  $profile_u$

---

puted for every dimension,  $i$ , along the  $M_{sel}$  individuals vectors. For each dimension, i.e., music genre, the normal distribution is calculated with its corresponding mean value and variance, to estimate the individuals vectors of the next generation. The time elapsed for modelling user profiles with the continuous EDA is approximately 4.20 seconds.

### 3.3.3 Top-N songs recommendation

The final stage of the recommender systems implemented is to generate a list of song recommendations according to the similarity values computed with Equation (2.7) (see page 25).

---

**Algorithm 4** Framework for  $UMDA_c^G$  to model users

---

**Require:** population size  $M$

**Require:** selection parameter  $\tau$

Generate  $M$  individuals at random

$M_{sel} \leftarrow M \cdot \tau$

$t \leftarrow 0$

**while**  $t < \text{MAXGEN}$  **do**

    Compute each  $\text{fitness}(\text{profile}_u)$

    Rank individuals by their fitness value

    Select top  $M_{sel}$  individuals

$\mu_{i,t} \leftarrow \frac{1}{M_{sel}} \sum_{j=1}^{M_{sel}} x_i^j$

$\sigma_{i,t}^2 \leftarrow \frac{1}{M_{sel}-1} \sum_{j=1}^{M_{sel}} (x_i^j - \mu_{i,t})^2$

$p_t(x_i | \mu_{i,t}, \sigma_{i,t}^2) \leftarrow \frac{1}{\sqrt{2\pi}\sigma_{i,t}} \exp(-\frac{1}{2}(\frac{x_i - \mu_{i,t}}{\sigma_{i,t}})^2)$

    Sample  $M$  individuals from  $p_t(x_i | \mu_{i,t}, \sigma_{i,t}^2)$

$t \leftarrow t + 1$

**end while**

---

### Top-N recommendations in CB baseline

The list of recommendations in a CB recommender is given by the similarities between the items that a user has already rated and the new items. It is assumed the user has not seen before the new items.

First, the similarity matrix between every item in the training set is computed. Only the  $k = 30$  most similar items are kept for each item. Next, for each song that a user rated above the threshold (rating  $> 2$ ), the  $k$  neighbours are retrieved as a list of candidate items. The list is normalised to have a maximum value of 1. The lists of candidates are appended. For the repeated candidates, the similarity values are summed up. The  $N$  candidates with higher similarity values are recommended to a user.

### **Top-N recommendations in hybrid music recommender**

In our hybrid music model (see Figure 3.4 on page 36), the content based filtering computes the similarity between a user interest profile and a each song vector in the test set. The songs are ranked in descending order and the first  $N$  songs of this ranking are recommended.

In our project, we experiment with different values for  $N$ , obtaining the best results with the hybrid music recommender based on permutation EDA for all the experiments. Refer to Section 5.2 for detailed results of evaluation.

## **3.4 Summary**

In this chapter, we presented the collection and preprocessing of the taste profile subset to model the user profiles with EDAs. As well, we presented the procedure of time-frequency representation of the audio content to feed a CDNN in order to obtain a 10-dimension vector representation corresponding to the probability of a song to belong to a music genre. Also, we presented the adapted architecture of the CDNN and the EDAs for hybrid recommendation. In the following chapter, we introduce the evaluation method and experiments to evaluate our hybrid recommender approach.

# Chapter 4

## Experiments

In order to evaluate the performance of a recommender system, there are several scenarios to be considered depending on the structure of the dataset and the prediction accuracy. It is therefore necessary to determine a suitable experiment for the evaluation of our proposed hybrid music recommender that employs a rating matrix and vector representation of songs as inputs to produce *top-N* song recommendations.

In addition, the performance of our hybrid approaches is compared with a pure content-based recommender algorithm.

### 4.1 Evaluation for recommender systems

#### 4.1.1 Types of experiments

The scenarios for experiments requires to define an hypothesis, controlling variables and generalization of the results. Three types of experiments (Shani and Gunawardana 2009) can be used to compare and evaluate recommender

algorithms:

- **Offline experiments:** where recorded historic data of users' ratings are used to simulate online users behaviour. The aim of this type of experiment is to refine approaches before testing with real users. On the other hand, results may have biases due to distribution of users.
- **User studies:** where test subjects interact with the recommendation system and its behaviour is recorded giving a large sets of quantitative measurements. One disadvantage of this type of experiment is to recruit subjects that represent the population of the users of the real recommendation system.
- **Online evaluation:** where the designer of the recommender application expect to influence the users' behaviour. Usually, this type of evaluation are run after extensive offline studies.

#### 4.1.2 Evaluation strategies

On the other hand, evaluation of recommender systems can be classified (Celma 2008) in:

- **System-centric** process has been extensively exploited in CF systems. The accuracy of recommendations is based exclusively on users' dataset and is evaluated through predictive accuracy, decision based and rank based metrics.
- **Network-centric** process examines other components of the recommendation system, such as diversity of recommendations, and they are

measured as a complement of the metrics of system-centric evaluation.

- **User-centric:** The perceived quality and usefulness of recommendations for the users are measured via provided feedback.

### 4.1.3 Decision based metrics

Our hybrid recommender produces a list of songs for each user, hence, it is necessary to evaluate the recommendation with a metrics derived from *confusion matrix* that reflects the categorisation of test items as true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). In this project we consider the following metrics (Celma 2008):

- **Precision** is the ratio of correct positive predictions.

$$Precision = \frac{TP}{TP + FP} \quad (4.1)$$

- **Recall** is the ratio of positive instances predicted as positive.

$$Recall = \frac{TP}{TP + FN} \quad (4.2)$$

- **F1 measure**, is the harmonic relation of precision and recall.

$$Recall = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4.3)$$

- **Accuracy**, is the ratio of correct predictions.

$$Recall = \frac{TP + TN}{TP + FP + TN + FN} \quad (4.4)$$

## 4.2 Evaluation method

The hybrid music recommender system proposed in this project is evaluated through an offline experiment and the results are presented with decision based metrics described in the previous section.

### 4.2.1 Training set and test set

The normalised taste profile dataset (refer to subsection 3.2.1) is split in a training and a test set. For each user in the dataset, a random sample corresponding to 20 % of the total number of ratings is assigned to the test set, and the rest 80 % is assigned to the training set. The split process is iterated for 10 times, resulting in a total of 10 training and 10 test sets.

### 4.2.2 Top-N evaluation

For each song in the user test set, we look up if the song is included or not in the list of top-N recommendations.

If the test song is in the top-N recommendation and if the rating of the test song is above the threshold (rating  $> 2$ ), we count as a true positive, otherwise is counted as a false positive.

If the test song is not in the top-N recommendation and if the rating of the test song is above the threshold, we count as a false negative, otherwise is counted as a true positive.



# Chapter 5

## Results

### 5.1 Genre classification results

A total of 9 trials are executed for training, validating and testing the CDNN using the normalised spectrograms of GTZAN dataset (see subsection 3.2.2).

We obtained the following results showed in Table 5.1.

Table 5.1: Genre classification results

Trial	Validation error (%)	Test error (%)	Iterations	Time elapsed (min.)
1	58.0	65.2	650	7.00
2	37.6	46.0	2150	13.07
3	39.6	46.0	700	7.54
4	35.6	36.8	550	6.01
5	36.4	40.0	250	5.47
6	40.4	44.8	150	5.41
7	32.4	40.4	800	8.64
8	36.0	38.8	250	5.42
9	34.0	38.8	850	9.14

For the initial trial, the error is higher because the weight and bias values for each unit of the layers in the deep learning classifier are randomly

initialised.

## 5.2 Recommender evaluation results

In general, the results demonstrate the hybrid music recommender based on a permutation EDA presents a better performance compared with both the CB recommender and the hybrid approach based on a continuous EDA. Nevertheless, the recall values are lower in all cases. In Table 5.2, the results of top-5 recommendation are shown.

Table 5.2: Evaluation of recommender systems (N=5)

Recommender	Precision	Recall	F1	Accuracy
Content-based (baseline)	$0.275 \pm 0.087$	$0.010 \pm 0.003$	$0.020 \pm 0.007$	$0.681 \pm 0.008$
Hybrid (permutation EDA)	<b><math>0.391 \pm 0.182</math></b>	<b><math>0.013 \pm 0.007</math></b>	<b><math>0.025 \pm 0.013</math></b>	<b><math>0.685 \pm 0.009</math></b>
Hybrid (continuous UMDA)	$0.318 \pm 0.142$	$0.011 \pm 0.005$	$0.021 \pm 0.011$	$0.683 \pm 0.009$

In Table 5.3, the results of top-10 songs recommendation are shown. In this case, the precision value improve for the CB recommender. The accuracy values for all recommender systems tend to decrease.

Table 5.3: Evaluation of recommender systems (N=10)

Recommender	Precision	Recall	F1	Accuracy
Content-based (baseline)	$0.301 \pm 0.059$	$0.022 \pm 0.007$	$0.041 \pm 0.012$	$0.678 \pm 0.007$
Hybrid (permutation EDA)	<b><math>0.370 \pm 0.073</math></b>	<b><math>0.024 \pm 0.007</math></b>	<b><math>0.045 \pm 0.013</math></b>	<b><math>0.682 \pm 0.009</math></b>
Hybrid (continuous UMDA)	$0.309 \pm 0.100$	$0.019 \pm 0.007$	$0.036 \pm 0.013$	$0.679 \pm 0.009$

In Table 5.4, the results of top-20 songs recommendation are shown. In this case, the recall values rise for all the recommender systems, compared with the top-5 and top-10 recommendations, but the precision and accuracy tend to decrease. At this point, we can deduce that our hybrid recommender

approaches could improve the recall without losing reached precision if  $N$  is in a value between 10 and 20.

Table 5.4: Evaluation of recommender systems (N=20)

Recommender	Precision	Recall	F1	Accuracy
Content-based (baseline)	$0.281 \pm 0.052$	$0.041 \pm 0.006$	$0.071 \pm 0.010$	$0.666 \pm 0.006$
Hybrid (permutation EDA)	<b><math>0.363 \pm 0.041</math></b>	<b><math>0.047 \pm 0.008</math></b>	<b><math>0.084 \pm 0.014</math></b>	<b><math>0.676 \pm 0.007</math></b>
Hybrid (continuous UMDA)	$0.302 \pm 0.067$	$0.039 \pm 0.011$	$0.070 \pm 0.019$	$0.671 \pm 0.010$

# Chapter 6

## Conclusion

The whole aim of our project has been the design and the implementation of an hybrid music recommender in order to mitigate the cold-start problem in content-based recommender systems. We investigated several types of hybridisation in recommender systems to choose a suitable architecture (shown in 3.4) for the available datasets. To represent real world users and raw waveforms, we decided to investigate and implement state-of-the-art techniques.

Despite of the success in computer vision field, we found in our project that convolutional deep neural networks achieve similar results to long-established music genre classifier approaches in music information retrieval field.

Due to the natural selection concept associated to estimation of distribution algorithms, we investigated and considered these optimisation techniques for modelling users' listening behaviour in terms of probabilities of music genres from the songs in they listened.

On the other hand, we found that a limited number of genres for song representation lead us to coarse predictions according to decision-based met-

rics.

## **6.1 Future work**

For the future, we have the intention to enhance our hybrid music recommender considering a wide range of music genres or latent vectors for item representation. We shall work on investigating several configurations of convolutional deep neural networks and different types of deep learning techniques, particularly, unsupervised learning approaches, for a better high-level representation of audio waveforms. In addition, we will continue investigating the fascinating estimation of distribution algorithms, considering another fitness functions to optimise, to model user profiles in recommender systems. Finally, we also consider the evaluation of hybrid recommender with an online experiment.

# References

- Bastien, Frédéric et al. (2012). *Theano: new features and speed improvements*. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.
- Bengio, Yoshua (2012). “Practical recommendations for gradient-based training of deep architectures”. In: *Neural Networks: Tricks of the Trade*. Springer, pp. 437–478.
- Bengio, Yoshua, Ian J. Goodfellow, and Aaron Courville (2015). “Deep Learning”. Book in preparation for MIT Press. URL: <http://www.iro.umontreal.ca/~bengioy/dlbook>.
- Bergstra, James et al. (2010). “Theano: a CPU and GPU Math Expression Compiler”. In: *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation. Austin, TX.
- Bertin-Mahieux, Thierry et al. (2011). “The Million Song Dataset”. In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.
- Blog.seagatesoft.com (2015). *Belajar Sistem Perekomendasi Corat-coret di Halaman Web*. URL: <http://blog.seagatesoft.com/2013/07/14/belajar-sistem-perekomendasi/> (visited on 08/29/2015).

- Bouneffouf, Djallel (2013). “Towards User Profile Modelling in Recommender System”. In: *CoRR* abs/1305.1114. URL: <http://arxiv.org/abs/1305.1114>.
- boyd, danah m. and Nicole B. Ellison (2007). “Social Network Sites: Definition, History, and Scholarship”. In: *Journal of Computer-Mediated Communication* 13.1, pp. 210–230. ISSN: 1083-6101. DOI: 10.1111/j.1083-6101.2007.00393.x. URL: <http://dx.doi.org/10.1111/j.1083-6101.2007.00393.x>.
- Brown, Larry (2014). *Accelerate Machine Learning with the cuDNN Deep Neural Network Library*. URL: <http://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/> (visited on 08/30/2015).
- Burke, R. (2002). “Hybrid recommender systems: Survey and experiments”. In: *User Modelling and User-Adapted Interaction* 12.4, pp. 331–370. DOI: 10.1023/A:1021240730564. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0036959356&partnerID=40&md5=28885a102109be826507abc2435>
- Casey, M.A. et al. (2008). “Content-based music information retrieval: Current directions and future challenges”. In: *Proceedings of the IEEE* 96.4, pp. 668–696. DOI: 10.1109/JPROC.2008.916370. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-64649105397&partnerID=40&md5=2d8ec7231e10bc686566dd419ce47ae8>.
- Celma, Ò. (2008). “Music Recommendation and Discovery in the Long Tail”. PhD thesis. Barcelona: Universitat Pompeu Fabra. URL: [http://mtg.upf.edu/static/media/PhD\\_ocelma.pdf](http://mtg.upf.edu/static/media/PhD_ocelma.pdf).

- Celma, O., P. Herrera, and X. Serra (2006). “Bridging the music semantic gap”. In: vol. 187. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84884332226&partnerID=40&md5=d028cb2aca5d2d6d8f25a8a8b555edbf>.
- Dai, C. et al. (2014). “A personalized recommendation system for netease dating site”. In: vol. 7. 13, pp. 1760–1765. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84905828317&partnerID=40&md5=90fbd8b20ad39757895bdee3ca58f459>.
- Deeplearning.net (2015). *Convolutional Neural Networks (LeNet) DeepLearning 0.1 documentation*. URL: <http://deeplearning.net/tutorial/lenet.html> (visited on 08/28/2015).
- Ding, C., L. Ding, and W. Peng (2015). “Comparison of effects of different learning methods on estimation of distribution algorithms”. In: *Journal of Software Engineering* 9.3, pp. 451–468. DOI: 10.3923/jse.2015.451.468. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84924609049&partnerID=40&md5=e6419e97e218f8ef1600e3d21e6a9e36>.
- Gallagher, Marcus et al. (2007). “Bayesian inference in estimation of distribution algorithms”. In: *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, pp. 127–133.
- Hejazi, S. A. and S. P. Stapleton (2015). “A self-organized network for load balancing using intelligent distributed antenna system”. In: *Canadian Journal of Electrical and Computer Engineering* 38.2, pp. 89–99. URL: [www.scopus.com](http://www.scopus.com).
- Hinton, Geoffrey et al. (2012). “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *Signal Processing Magazine, IEEE* 29.6, pp. 82–97.



- Hu, Y., C. Volinsky, and Y. Koren (2008). “Collaborative filtering for implicit feedback datasets”. In: pp. 263–272. DOI: 10.1109/ICDM.2008.22. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-67049164166&partnerID=40&md5=01238b08208962fd0fdcc7503fa3af99>.
- Hypebot.com (2015). *Streaming Music Discovery: It’s More Than Just Showing Album Credits - hypebot*. URL: <http://www.hypebot.com/hypebot/2015/07/streaming-music-discovery-its-more-than-just-showing-album-credits.html> (visited on 08/26/2015).
- Jin, C. and S.-W. Jin (2014). “Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms”. In: *Applied Soft Computing Journal* 15, pp. 113–120. DOI: 10.1016/j.asoc.2013.10.016. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84889065631&partnerID=40&md5=6ba595eee679fa8355329646504b3ae3>.
- Kereliuk, Corey, Bob L Sturm, and Jan Larsen (2015). “Deep Learning and Music Adversaries”. In: *arXiv preprint arXiv:1507.04761*.
- Larranaga, Pedro and Jose A Lozano (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Vol. 2. Springer Science & Business Media.
- LeCun, Yann (2015). *MNIST Demos on Yann LeCun’s website*. URL: <http://yann.lecun.com/exdb/lenet/> (visited on 08/30/2015).
- Liang, T. et al. (2014). “A hybrid recommendation model based on estimation of distribution algorithms”. In: *Journal of Computational Information Systems* 10.2, pp. 781–788. DOI: 10.12733/jcis9623. URL: <http://>

[www.scopus.com/inward/record.url?eid=2-s2.0-84892865461&partnerID=40&md5=a2927d36b493e8ef4d1cdab3055fa68b](http://www.scopus.com/inward/record.url?eid=2-s2.0-84892865461&partnerID=40&md5=a2927d36b493e8ef4d1cdab3055fa68b).

- Lops, Pasquale, Marco de Gemmis, and Giovanni Semeraro (2011). “Content-based Recommender Systems: State of the Art and Trends”. English. In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Springer US, pp. 73–105. ISBN: 978-0-387-85819-7. DOI: 10.1007/978-0-387-85820-3\_3. URL: [http://dx.doi.org/10.1007/978-0-387-85820-3\\_3](http://dx.doi.org/10.1007/978-0-387-85820-3_3).
- Melville, Prem and Vikas Sindhwani (2010). “Recommender systems”. In: *Encyclopedia of machine learning*. Springer, pp. 829–838.
- Oord, Aaron van den, Sander Dieleman, and Benjamin Schrauwen (2013). “Deep content-based music recommendation”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C.J.C. Burges et al. Curran Associates, Inc., pp. 2643–2651. URL: <http://papers.nips.cc/paper/5004-deep-content-based-music-recommendation.pdf>.
- Park, Y.-J. and A. Tuzhilin (2008). “The long tail of recommender systems and how to leverage it”. In: pp. 11–18. DOI: 10.1145/1454008.1454012. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-63449136183&partnerID=40&md5=648e50cac2d99764f891b5bc4b97bbfe>.
- Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Pelikan, Martin, Mark W Hauschild, and Fernando G Lobo (2015). “Estimation of Distribution Algorithms”. In: *Springer Handbook of Computational Intelligence*. Springer, pp. 899–928.

- Putzke, Johannes et al. (2014). “Cross-cultural gender differences in the adoption and usage of social media platforms An exploratory study of Last.FM”. In: *Computer Networks* 75, Part B. Special Issue on Online Social Networks The Connectedness, Pervasiveness and Ubiquity of Online Social Networks, pp. 519–530. ISSN: 1389-1286. DOI: <http://dx.doi.org/10.1016/j.comnet.2014.08.027>. URL: <http://www.sciencedirect.com/science/article/pii/S1389128614003302>.
- Recommendation Engine* (2013). URL: <https://spatnaik77.wordpress.com/2013/07/17/recommendation-engine/> (visited on 08/28/2015).
- Recommender Systems* (2012). [Accessed: 26th August 2015]. URL: <http://recommender-systems.org/>.
- Ringen, Jonathan (2015). *Spotify, Apple Music, And The Streaming Wars: 5 Things We’ve Learned*. URL: <http://www.fastcompany.com/3048653/innovation-agents/listen-up> (visited on 08/26/2015).
- Santana, Roberto et al. (2010). “Mateda-2.0: A MATLAB Package for the Implementation and Analysis of Estimation of Distribution Algorithms”. In: *Journal of Statistical Software* 35.7, pp. 1–30. ISSN: 1548-7660. URL: <http://www.jstatsoft.org/v35/i07>.
- Sarwar, Badrul et al. (2001). “Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th international conference on World Wide Web*. ACM, pp. 285–295.
- Shani, Guy and Asela Gunawardana (2009). *Evaluating Recommender Systems*. Tech. rep. MSR-TR-2009-159. URL: <http://research.microsoft.com/apps/pubs/default.aspx?id=115396>.

- Sigtia, S. and S. Dixon (2014). “Improved music feature learning with deep neural networks”. In: pp. 6959–6963. DOI: 10.1109/ICASSP.2014.6854949. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84905259152&partnerID=40&md5=3441dfa8c7998a8eb39f668d43efb8a1>.
- Smith, Tom (2009). “The social media revolution”. In: *International journal of market research* 51.4, pp. 559–561.
- Sturm, B.L. (2012). “An analysis of the GTZAN music genre dataset”. In: pp. 7–12. DOI: 10.1145/2390848.2390851. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84870497334&partnerID=40&md5=40a48c1c9d787308dd315694b54b64ec>.
- Tzanetakis, G. and P. Cook (2002). “Musical genre classification of audio signals”. In: *IEEE Transactions on Speech and Audio Processing* 10.5, pp. 293–302. DOI: 10.1109/TSA.2002.800560. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0036648502&partnerID=40&md5=72d2fee186b42c9998f13415cbb79eea>.
- Ufldl.stanford.edu (2015). *Unsupervised Feature Learning and Deep Learning Tutorial*. URL: <http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/> (visited on 08/30/2015).
- Weston, Jason et al. (2012). “Latent collaborative retrieval”. In: *arXiv preprint arXiv:1206.4603*.
- Yao, L. et al. (2015). “Unified collaborative and content-based web service recommendation”. In: *IEEE Transactions on Services Computing* 8.3, pp. 453–466. DOI: 10.1109/TSC.2014.2355842. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84932619562&partnerID=40&md5=4483a697e12fc53f620393586f85aebe>.

- Yin, H. et al. (2012). “Challenging the long tail recommendation”. In: vol. 5. 9, pp. 896–907. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84863735354&partnerID=40&md5=2bd887772ba832fbbb4631afd25514d9>.
- Yoshii, K. et al. (2008). “An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model”. In: *IEEE Transactions on Audio, Speech and Language Processing* 16.2, pp. 435–447. DOI: 10.1109/TASL.2007.911503. URL: <http://www.scopus.com/inward/record.url?eid=2-s2.0-39649112098&partnerID=40&md5=6827f82844ae1da58a6fa95caf5092d9>.