Extracting musical features using Restricted Boltzmann Machines

Andreas Hjortgaard Danielsen
Exam number: 9

November 4, 2011

Abstract

An approach for unsupervised learning of musical features using Restricted Boltzmann Machines (RBMs) is investigated and measured against two other popular features for music. Specifically, we train four RBMs to represent a piece of music as a series of 10-, 30- or 50-dimensional vectors which can the be used for genre classification or other musical information retrieval tasks. Visualisations of the RBM features extracted from a genre classification dataset show that the genres are less clearly grouped compared to Mel-Frequency Cepstral Coefficients (MFCCs) features. A preliminary classification test also shows that using magnitude spectrum features give the smallest classification error 40.2% using an LDA classifier. A test error of 76.1% using the learned RBM features indicates that the RBM has "unlearned" the characteristics of the genres.

1 Introduction

Genres in music are human-defined labels defining certain characteristics of individual music pieces or entire artist corpora. It is a way to group music pieces such that a piece in one group has many characteristics in common with the other pieces in the group. An example of such characteristics could be the use of instruments; a symphonic piece played with woodwinds, brass instruments and percussion instruments is different from a piece played by a four-piece band playing electrically amplified instruments. The genre labeling is inherently hierarchical so that avantgarde jazz is a subgenre of the more general jazz genre and similarly baroque is subgenre of the far more general classical genre. On top of that the labeling is highly subjective and therefore not unique, and so a given song, say The Beatles' Day Tripper, can be categorised as both rock and pop. Because of these ambiguities genres may not be the best way to represent similarity in, say, an automated recommendation system. But nonetheless, genres are broadly accepted labels for music characterisation.

The first step in any clustering or classification problem is to find a representation of the data to use. Ideally, the representation could simply be the raw audio samples but due to the curse of dimensionality [5] this is often intractable. Consider a 10-second audio clip sampled at 44,100 Hz. The vector that represents this raw signal is a huge 441,000 dimensional vector. Clearly, representing songs in such a high-dimensional space incurs computational problems. Instead, before feeding data to the classification algorithm, a step called dimensionality reduction is performed either by *feature selection* where a subset of the original features is selected or by *feature extraction* where the data is transformed from the high dimensional representation to a representation of lower dimensions for example using Principal Component Analysis.

Instead of relying on automated approaches for feature extraction it may be possible to exploit the prior knowledge you have about the data to manually construct a proper feature representation. From speech recognition a very popular feature representation called Mel-Frequency Cepstral Coefficients (MFCCs) has been developed and has later gained popularity in musical modeling as well [12]. The technique is based on the Short-Time Fourier Transform (STFT) where the signal is divided into frames of low duration (e.g. 23 ms) and transformed into the Mel scale where the most significant entries are the low frequency components.

In recent years, deep learning techniques have received a lot of attention especially in the computer vision community. The principle of deep learning is to exploit deep layered architecture to learn feature representations. Deep Belief Networks is an example of such an architecture: Each layer takes the activations of the previous layer as input and generates a new representation. Thus, low level features fed into the bottom input layer generates more abstract representations in the higher layers. It turns out that these architectures can be trained in a greedy, unsupervised manner that makes training tractable. In this report we will investigate the use of Restricted Boltzmann Machines for unsupervised feature learning of musical features.

The report is structured as follows: Section 2 describes classical methods of musical feature extraction used for genre classification and recommendation systems as well as methods using Deep Belief Networks for unsupervised musical feature learning. Restricted Boltzmann Machines are introduced in Section 3 as well as the use of Contrastive Divergence for tractable training. In Section 4 we describe how the RBM is set up and how the learned features perform on the Tzanetakis dataset for genre classification and in Section 5 the results are discussed. Finally, Section 6 concludes on the findings of the experiments and discussions.

2 Related work

Feature extraction for audio has its roots in speech recognition and many of the features proposed in the literature for genre classification comes from this area [17]. These features are the result of many years of research but in the recent years focus has been turned to automatic feature learning. This section describes the traditional approaches to feature extraction along with recent approaches to unsupervised feature learning.

2.1 Audio feature extraction

McKay and Fujinaja [13] discuss the general problem of musical genre classification from a musicologist point of view and present the issues of genre categorisation. The main problem in automatic genre classification is that genres are not well-defined and often ambiguous. Besides the ambiguities, several cultural factors not embedded in the audio data have a deciding impact in the human genre classification. The authors suggest that several layers of features should be used: from low-level audio features to music theoretic and cultural information about the specific piece of music.

Aucouturier and Pachet [1] discuss three approaches to genre classification: manual classification and two kinds of automatic classification, namely the prescriptive approach and classifications from similarity relations. The prescriptive approaches extract features from the audio directly and then classifies the individual audio signals according to some given labeling usually via supervised learning algorithms. This requires a training set with already labeled music, possibly labeled manually by professionals or by users. The similarity-based genre classification is either performed on features from the audio or external features such as metadata embedded in the audio files or textual descriptions mined from the Internet and a similarity measure is used to group musical pieces together.

One example of such a similarity-based approach for classification and a popular method for music recommendation (where the exact genre is not as important as the more general notion of similarity) is collaborative filtering [2, 16]. This technique is used in the iTunes Genius recommendation system by Apple Inc. for creating top-10 lists and for making playlists given a seed song. In user-based collaborative filtering, such as used in iTunes Genius, recommendations are based on users with similar purchasing and listening records as well as meta data information such as ID3 tags. The actual audio content is not used in such systems. Barrington et al. [2] show that despite this, Genius actually captures acoustic similarities which shows that such similarities can be derived from user-based purchase and listening statistics. One problem with this scheme is that, given a song of an unknown artist or with no tags, collaborative filtering is unable to gather information from other users and thus is not able to recommend songs.

The classical paper by Tzanetakis and Cook [17] describes three different feature representations of audio for use in a statistical genre classification system. The three representations correspond to timbral texture, rhythmic content and pitch content of audio. Timbral texture is computed using the STFT and MFCCs where six distinct features determine the texture: Spectral centroid (center of gravity of the STFT magnitude spectrum), spectral rolloff (the frequency at which 85% of the magnitude spectrum lies below), spectral flux (the square difference between successive, normalised magnitudes) and time-domain zero crossings (a measure of noise). The remaining two features are the first five coefficients of the MFCCs as well as a low-energy feature which model the level of silence in the music. This results in a 19-dimensional feature vector that describes timbral texture.

Besides texture, the rhythmic content is computed using the Discrete Wavelet Transform to construct a beat histogram that describes the beats-per-minute as well as other rhythmic content resulting in a six-dimensional vector. Finally, pitch content is computed and stored in a five-dimensional vector. Thus, the feature vector containing all features has $19 + 6 + 5 = 30$ dimensions. Using these features in a range of Gaussian mixture models with different number of Gaussians as well as different k-Nearest Neighbour classifiers with different values of k, Tzanetakis and Cook report a classification accuracy of about 60%.

2.2 Unsupervised feature learning

Until now we have mainly focused on manually created features which have either been designed through years of research, like MFCCs or which must be collected manually, like ID3 tags. If instead relevant features could be extracted automatically by some unsupervised method before performing the classification, a music recommendation system would be able to give recommendations given completely new songs based only on the audio data itself or some simple feature such as the frequency spectrum.

To learn high-level abstractions of data it has been proposed by Bengio [3] to use deep architectures in which abstract representations of low-level data can be generated in a layer-wise fashion. Each layer performs some non-linear operation on the input and sends the result to the next layer. Thus, each layer can be thought of as a more abstract representation of the previous layer. Deep Belief Networks is an example of such a deep architecture. The next section will go into the details of Restricted Boltzmann Machines which are the building blocks of DBNs and will describe the unsupervised training algorithm.

Lee et al. proposed a generative model for feature learning called Convolutional Deep Belief Networks (CDBNs) [10] and showed the results of performing unsupervised feature learning on various vision tasks. Specifically, it was shown that the CDBN was able to learn hierarchical representations of high-dimensional data (images). Using these ideas the CDBN was used for musical feature extraction by Lee et al. [11] and tested on the speech recognition task as well as the genre classification task. For both tasks the features obtained by the CDBN layers seem to outperform the more traditional MFCC features and specifically, the CDBN gives a classification accuracy of 73.1% against 68.3% using MFCC. Hamel and Eck [7] perform a similar experiment using traditional DBNs on the magnitude spectrum of the audio for genre classification. Again, this shows that the feature extraction gives good classification results with an accuracy of 77% against 63% using MFCC. This report will explore the method used by Hamel and Eck since it seem to obtain the best results. Even using only a 1-layer DBN they achieve a classification accuracy of 73.5% which justifies exploring features learned by a Restricted Boltzmann Machine.

3 Restricted Boltzmann Machines

Deep Belief Networks have been used for unsupervised feature extraction in both computer vision and for musical information retrieval tasks [3, 7, 8, 10, 11]. In this section we give a theoretical introduction to Restricted Boltzmann Machines which are the building blocks of DBNs and we discuss the Contrastive Divergence algorithm for performing unsupervised training. Deep Belief Networks are briefly touched upon towards the end of the section.

3.1 Structure and properties

A Boltzmann Machine is a Markov Random Field (MRF) with two layers of nodes, a visible layer x and a hidden layer h. The joint distribution of such a MRF is given by the Gibbs distribution

$$p(x,h) = \frac{1}{Z} e^{-E(x,h)} \tag{1}$$

where $E(x,h)$ is the energy function and Z is a normalisation constant called the partition function given by

$$Z = \sum_{x,h} e^{-E(x,h)}. \tag{2}$$

The energy function depends on the interlayer connections as well as the intralayer connections. Thus, the energy function for the Boltzmann Machine is given by the quadratic polynomial

$$E(x,h) = -b^T x - c^T h - h^T W x - x^T U x - h^T V h \tag{3}$$

where b and c are the biases of the visible and hidden units, respectively, and where W is a matrix of weights on the interlayer connections, U is a matrix of weights on the intralayer connections in the visible layer and V is a matrix of weights on the connections in the hidden layer.

A Restricted Boltzmann Machine (RBM) is a Boltzmann Machine in which no intralayer connections exist. That is, U and V are zero-matrices giving the bilinear energy function

$$E(x,h) = -b^T x - c^T h - h^T W x. \tag{4}$$

Figure 1 shows the structure of the RBM. For training the RBM we must be able to sample from the model distribution. Since we only know the visible units we are interested in the marginal distribution $p(x)$ which can be factorised as

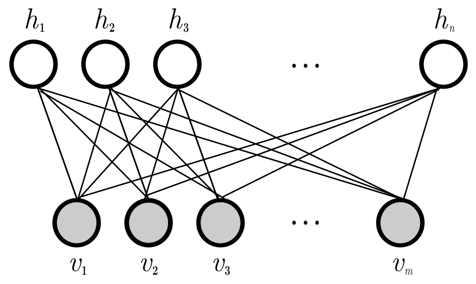

Figure 1: The Restricted Boltzmann Machine. Only connections between the visible and the hidden layer exist. In this figure the v_i's correspond to x_i's in the text.

follows [3]:

$$\begin{align}
p(x) &= \frac{1}{Z}\sum_h e^{-E(x,h)} \tag{5}\\
&= \frac{1}{Z}\sum_h e^{b^T x + c^T h + h^T W x} \tag{6}\\
&= \frac{1}{Z}e^{b^T x}\sum_h e^{c^T h + h^T W x} \tag{7}\\
&= \frac{1}{Z}e^{b^T x}\sum_h e^{\sum_i c_i h_i + h_i W_i x} \tag{8}\\
&= \frac{1}{Z}e^{b^T x}\sum_h \prod_i e^{c_i h_i + h_i W_i x} \tag{9}\\
&= \frac{1}{Z}e^{b^T x}\sum_{h_1} e^{c_1 h_1 + h_1 W_1 x}\sum_{h_2} e^{c_2 h_2 + h_2 W_2 x}\cdots\sum_{h_k} e^{c_k h_k + h_k W_k x} \tag{10}\\
&= \frac{1}{Z}e^{b^T x}\prod_i \sum_{h_i} e^{c_i h_i + h_i W_i x} \tag{11}
\end{align}$$

But even though the distribution factorises, we still need to compute the exponential normalisation term Z given by (2). The next subsection discusses this issue. We will also need to sample from the conditional distribution $p(h|x)$ to sample from the hidden layer. This can be done tractably because the conditional distribution factorises. First note that (as above) the joint probability can be written as

$$\begin{align}
p(x,h) &= \frac{1}{Z}e^{b^T x + c^T h + h^T W x} \tag{12}\\
&= \frac{1}{Z}e^{b^T x}e^{\sum_i c_i h_i + h_i W_i x} \tag{13}\\
&= \frac{1}{Z}e^{b^T x}\prod_i e^{c_i h_i + h_i W_i x}. \tag{14}
\end{align}$$

From this and (11) we get the factorisation of the conditional probability

$$\begin{aligned} p(h|x) &= \frac{p(x,h)}{p(x)} && (15) \\ &= \frac{\frac{1}{Z}e^{b^T x}\prod_i e^{c_i h_i + h_i W_i x}}{\frac{1}{Z}e^{b^T x}\prod_i \sum_{h_i} e^{c_i h_i + h_i W_i x}} && (16) \\ &= \frac{\prod_i e^{c_i h_i + h_i W_i x}}{\prod_i \sum_{h_i} e^{c_i h_i + h_i W_i x}} && (17) \\ &= \prod_i \frac{e^{c_i h_i + h_i W_i x}}{\sum_{h_i} e^{c_i h_i + h_i W_i x}} && (18) \\ &= \prod_i \frac{p(x,h_i)}{p(x)} && (19) \\ &= \prod_i p(h_i|x) && (20) \end{aligned}$$

Using corresponding arguments we can show that $p(x|h)$ factorises similarly and this makes it tractable to perform alternating Gibbs sampling as discussed below.

3.2 Contrastive Divergence

We see that maximising the joint probability of the model corresponds to choosing b, c and W(collectively denoted θ) such that $E(x,h)$ is minimised. This can be done by maximum likelihood using gradient ascend [6]:

$$\theta^{t+1} = \theta^t + \eta \frac{\partial \log L(\theta;x)}{\partial \theta} \quad (21)$$

where η is the learning rate and $\log L(\theta;x)$[1] is the data log-likelihood given by

$$\begin{aligned} \log L(\theta;x) &= \log p(x;\theta) && (22) \\ &= \log \frac{\sum_h e^{-E(x,h)}}{\sum_{x,h} e^{-E(x,h)}} && (23) \\ &= \log \sum_h e^{-E(x,h)} - \log \sum_{x,h} e^{-E(x,h)}. && (24) \end{aligned}$$

[1]We use the notation $L(\theta;x) = p(x;\theta)$ here to emphasize the dependence on the parameter vector θ.

Differentiating with respect to θ gives

$$\frac{\partial \log L(\theta;x)}{\partial\theta} = \frac{\partial \log \sum_h e^{-E(x,h)}}{\partial\theta} - \frac{\partial \log \sum_{x,h} e^{-E(x,h)}}{\partial\theta} \tag{25}$$

$$\begin{aligned} &= \frac{1}{\sum_h e^{-E(x,h)}} \sum_h \frac{\partial e^{-E(x,h)}}{\partial\theta} \\ &\quad - \frac{1}{\sum_{x,h} e^{-E(x,h)}} \sum_{x,h} \frac{\partial e^{-E(x,h)}}{\partial\theta} \end{aligned} \tag{26}$$

$$\begin{aligned} &= -\frac{1}{\sum_h e^{-E(x,h)}} \sum_h e^{-E(x,h)} \frac{\partial E(x,h)}{\partial\theta} \\ &\quad + \frac{1}{\sum_{x,h} e^{-E(x,h)}} \sum_{x,h} e^{-E(x,h)} \frac{\partial E(x,h)}{\partial\theta} \end{aligned} \tag{27}$$

$$= -\sum_h \frac{p(x,h)}{p(x)} \frac{\partial E(x,h)}{\partial\theta} + \sum_{x,h} p(x,h) \frac{\partial E(x,h)}{\partial\theta} \tag{28}$$

$$= -\sum_h p(h|x) \frac{\partial E(x,h)}{\partial\theta} + \sum_x p(x) \sum_h p(h|x) \frac{\partial E(x,h)}{\partial\theta}. \tag{29}$$

From (4) we see that $\frac{\partial E(x,h)}{\partial\theta}$ is easily evaluated and from (20) we know that $p(h|x)$ factorises, such that sampling from this distribution is tractable. However, as seen in (11) and (2), evaluating $p(x)$ requires the evaluation of Z which is a sum over an exponential number of terms. Instead, we can estimate the log-likelihood gradient by using alternating Gibbs sampling [3] to estimate the expectation of $p(x)$. This is done by iteratively sampling from $p(h|x)$ and $p(x|h)$ starting with a single sample $x^{(0)}$ until an equilibrium distribution is reached. Thus, the alternating Gibbs sampling gives the following series of samples for k steps:

$$\begin{aligned} x^{(0)} &\sim p(x) \\ h^{(0)} &\sim p(h|x^{(0)}) \\ x^{(1)} &\sim p(x|h^{(0)}) \\ h^{(1)} &\sim p(h|x^{(1)}) \\ &\vdots \\ h^{(k-1)} &\sim p(h|x^{(k-1)}) \\ x^{(k)} &\sim \; p(x|h^{(k-1)}) \end{aligned} \tag{30}$$

Sampling from the unbiased distribution $p_\infty(x)$ requires many steps of the Markov chain before it ends in an equilibrium state but it turns out that it is possible to estimate the log-likelihood gradient efficiently at the cost of a small bias.

Maximum likelihood estimation corresponds to minimising the Kullback-Leibler divergence $KL(p_0||p_\infty)$ between the data distribution $p_0(x)$ and the equilibrium distribution $p_\infty(x)$ obtained by MCMC estimation [8]. Contrastive Divergence [3, 6] only takes k steps of the Markov chain starting with the data distribution to obtain $p_k(x)$ and then approximates the log-likelihood gradient by the difference between the two Kullback-Leibler divergences:

$$\mathrm{CD}_k = KL(p_0||p_\infty) - KL(p_k||p_\infty) \tag{31}$$

Since $p_k(x)$ is a biased estimate of $p(x)$, CD_k gives biased estimates of the log-likelihood gradient. But Caerreira-Perpiñán and Hinton [6] show that the bias is small in practice and can be eliminated by performing fine-tuning maximum-likelihood estimation after having pretrained the model using Contrastive Divergence. In fact, even for $k = 1$ the bias is small enough to give good results in practice.

3.3 Conditional densities

The RBM was originally introduced to have binary units in both the input and hidden layers [3, 8]. This means that $h_i \in \{0, 1\}$ and $x_j \in \{0, 1\}$ and from (20) we get

$$\begin{aligned} p(h_i = 1|x) &= \frac{e^{b^T x + c_i + W_i x}}{e^{b^T x} + e^{b^T x + c_i + W_i x}} &(32)\\ &= \frac{e^{b^T x} e^{c_i + W_i x}}{e^{b^T x} + e^{b^T x} e^{c_i + W_i x}} &(33)\\ &= \frac{e^{c_i + W_i x}}{1 + e^{c_i + W_i x}} &(34)\\ &= \sigma(c_i + W_i x) &(35)\end{aligned}$$

where σ is the sigmoid function. Similarly, we find the conditional probability of the input units to be

$$p(x_j = 1|h) = \sigma(b_j + W_{\cdot,j} h) \tag{36}$$

where $W_{\cdot,j}$ denotes the j'th column of W. Binary units, however, are not suitable for handling continuous inputs and therefore it is suggested to use units with a conditional density that is a *truncated exponential* [4]. For $x_j \in I$, where I is a closed interval the truncated exponential density is given by

$$p(x_j|h) = \frac{e^{b_j x_j + x_j W_{\cdot,j} h}}{\int_{v \in I} e^{b_j v + v W_{\cdot,j} h} dv} \tag{37}$$

We will use truncated exponentials for the input units and binary hidden units for our experiments. Specifically, we choose $I = [0, 1]$ so that the normalising integral always exists [4].

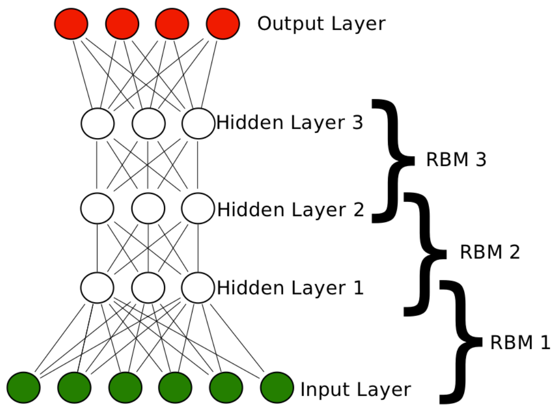

Figure 2: A Deep Belief Network constructed from layers of RBMs. The output layer is used for e.g. predicting the class of the input. The image is taken from [7].

3.4 Deep Belief Networks

Deep Belief Networks are stacked RBMs as shown in Figure 2. Training each RBM using Contrastive Divergence in a greedy manner has been shown to be an effective way for unsupervised training of such networks [4, 8]. This unsupervised training step is often used as an initialisation (or pre-training) step before performing supervised training (or fine-tuning) of the model. In [4] it is argued that exploiting their deep architecture, DBNs are able to learn abstract features that, in turn, gives higher classification accuracy compared to so-called shallow architectures such as Support Vector Machines. In this text we will only investigate RBMs (or 1-layer DBNs) and therefore we do not investigate the merits of utilising truly deep architectures.

4 Experiments

The previous section described the general theory of Restricted Boltzmann Machines and how Contrastive Divergence makes training tractable. In this section we will compare the learned RBM features with standard musical features such as the magnitude spectrum and the MFCCs for the genre classification problem.

4.1 The Tzanetakis dataset

The Tzanetakis dataset[2] is the dataset that was used by Tzanetakis and Cook in [17] for genre classification. It consists of 1000 song examples divided into

[2]The dataset can be downloaded from `http://marsyas.info/download/data_sets`.

10 genres with 100 examples in each. Each example is a 30 second audio clip encoded as 16-bit mono with a sampling rate of 22,050 Hz. Due to the time constraints of the project we limit the number of songs in each genre to 10 and use only 100 frames (see below) from each song. In Section 5 we will discuss the implications of this limitation. We create a training set by randomly selecting 7000 frames which is used for learning feature representations and for training a two classifiers. A test set of the remaining 3000 frames is used for computing classification test error only.

4.2 Standard features

To compare the learned features of our RBM we will investigate two musical features that are obtained by standard digital signal processing techniques, namely the magnitude spectrum and the MFCCs. Both are based on the Short-Time Fourier Transform (STFT) which is the Discrete Fourier Transform taken on a windowed signal. The window function is often chosen to be small enough to assume stationarity properties of the resulting signal. Both features are computed using the Marsyas audio processing framework[3].

Magnitude spectrum

The frames for each song are obtained by applying a window function and a Hamming filter to remove edge effects. The time domain signals $x(n)$ are 1024-dimensional vectors in the interval $[-1, 1]$. They represent sound waves of duration approximately 46 ms and are the most low-level of the musical features. Instead of using these raw signals for input to the RBM we use the frequency information given in the individual frames. This is done using the Discrete Fourier Transform to compute the magnitude spectrum [15] . Given a discrete signal $x(n)$ for $n = 0, 1, ..., N-1$, the DFT is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-2\pi i k n/N} \tag{38}$$

where $k = 0, 1, ..., N-1$ corresponds to the frequency component with frequency $f(k) = kf_s/N$. The magnitude spectrum is then given by $|X(k)|$ which gives

[3]`http://marsyas.info/`

information about the frequency content of the frame. From (38) we see that

$$\begin{aligned}
X(N-k) &= \sum_{n=0}^{N-1} x(n)e^{-2\pi i(N-k)n/N} && (39)\\
&= \sum_{n=0}^{N-1} x(n)e^{-2\pi i n}e^{2\pi i k n/N} && (40)\\
&= \sum_{n=0}^{N-1} x(n)e^{2\pi i k n/N} && (41)\\
&= X(-k) && (42)
\end{aligned}$$

where we have used

$$\begin{aligned}
e^{-2\pi i n} &= \cos(2\pi n) - i\sin(2\pi n) && (43)\\
&= 1. && (44)
\end{aligned}$$

Thus, the magnitude spectrum is symmetric and therefore it is sufficient to keep only the $N/2+1$ samples. For the frames with 1024 samples the magnitude spectrum can be represented as a 513-dimensional vector. Since the magnitude spectrum describes the amount of frequency content for each of the frequencies $f(k)$ this might give an indication of the type of sound present in the frames, e.g., whether it is a distorted guitar or a symphonic orchestra.

Mel-Frequency Cepstral Coefficients

In speech recognition a popular feature is the Mel-Frequency Cepstral Coefficients (MFCCs) which more compactly represents the important characteristics of speech signals. It has also been used in musical modeling and instrument recognition [1, 12] and therefore we present it here as well. Computing the MFCC features from a single sound wave frame is done using four steps:

1. Compute the DFT, $X(k)$.
2. Compute logarithm of the magnitude spectrum, $\log|X(k)|$.
3. Perform Mel-scaling, $X'(k) = m(\log|X(k)|)$.
4. Compute the Discrete Cosine Transform, $DCT(X'(k))$.

Steps 1-3 map the frequencies into a representation that mimics the human auditory system in such a way the the lower frequencies in this new representation are perceptually more important than higher frequencies. Step 4 is used to decorrelate the Mel-frequency components. We refer the readers to [12] and the references therein for the details. Fair classification results using MFCCs are reported by Hamel and Eck as well as Lee et al. [7, 11] outperformed only by their DBN representations. Neither state the number of coefficients used but based on [17] we will choose five coefficients which is noted to give the best classification performance.

	Input units	Hidden units	Trials	Iterations
RBM1	Spectrum (513)	50	10	1000
RBM2	Spectrum (513)	50	5	500
RBM3	Spectrum (513)	30	5	500
RBM4	Spectrum (513)	10	5	500

Learning rate	0.1
Markov chain steps (k)	1

Table 1: Parameters for the four RBMs.

4.3 RBM setup

Setting up the RBM we need to choose the features to feed into the input layer as well as the number of hidden units. As input features we choose the magnitude spectrum which gives 513 input units. For representing the learned features we investigate four different RBMs: two with 50 hidden units and two with 30 and 10 hidden units, respectively. The parameters for the tests are seen in Table 1. Note that these parameters have been chosen in an ad-hoc fashion and more time should be invested in finding optimal parameters. We have used the RBM implementation of the Shark machine learning library [9] to implement and train our RBMs.

4.4 Visualisation using t-SNE

The first experiment will try to justify whether the learned RBM features more clearly separates the different genres in the feature space. We will use the t-distributed Stochastic Neighbour Embedding (t-SNE) visualisation technique by van der Maaten and Hinton [18] to do this. It is a technique for mapping points in a high-dimensional space into a low-dimensional space by representing point similarities as joint probabilities and then minimising the Kullback-Leibler divergence between the distributions in the two spaces. This ensures that local as well as global structure in the high-dimensional space is preserved in the low-dimensional representation.

We have randomly selected 500 frames from the 7000 frames used in the training set. In Figure 3 we see a visualisation plot performed on the magnitude spectrum features. We see that there is little structure to the placement of the different frames. Only classical frames seem to be somewhat distinctly grouped. Figure 4 shows the visualisation plot using MFCC features. Here we clearly see three major groupings: In the top of the plot we see that classical and jazz group up, in the bottom left corner we have rock frames and in the bottom right corner we have metal frames. It is interesting to note that blues frames are placed between the rock and jazz frames and that jazz and blues frames are clearly related. Similarly, country frames are found in the area between rock and metal frames. Reggae, hiphop and disco frames are more spread out.

The feature representation obtained by RBM1, which uses 50 hidden units

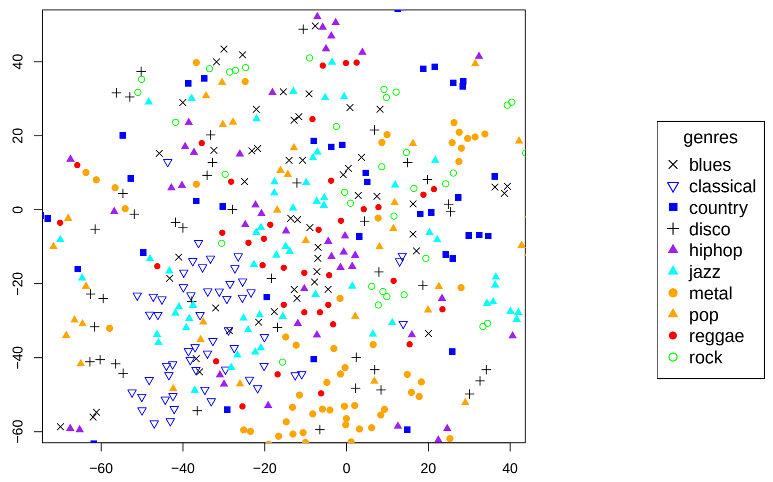

Figure 3: 2D visualisation of magnitude spectrum features.

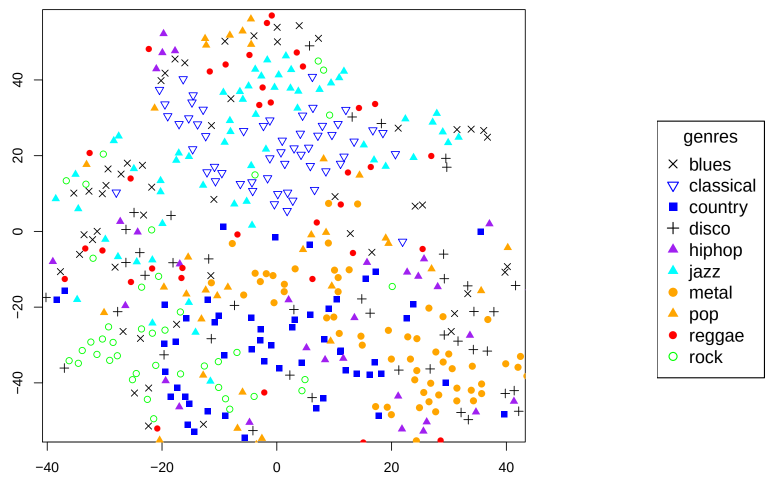

Figure 4: 2D visualisation of MFCC features.

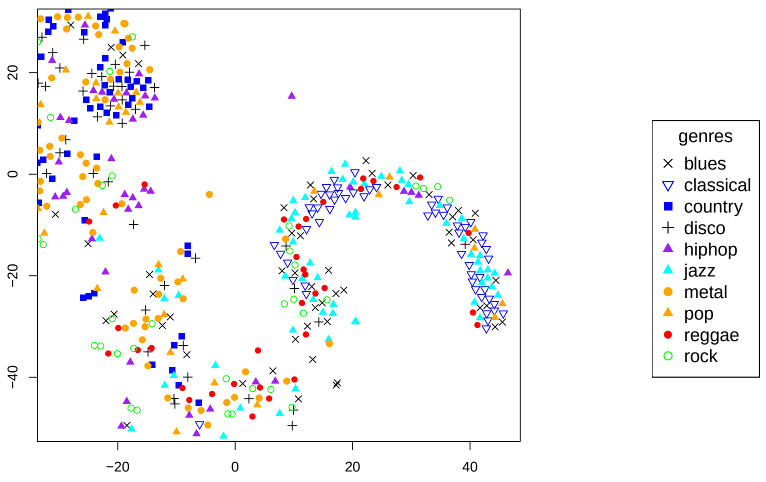

Figure 5: 2D visualisation of the RBM1 features. 50 hidden units, 1000 training iterations over 10 trials.

and has been trained for the longest time is seen in Figure 5. The features seem to lie in a strange snake-like structure compared the the magnitude spectrum and MFCC features. But classical and jazz are still grouped together in the rightmost end of the "snake" whereas metal and country dominate the left part of the plot. The connections between blues, jazz and rock are not as easily determined as for the MFCC features. Using fewer training iterations and fewer hidden units results in even more vague clusterings as seen in Figure 6. But still, classical and jazz are in general closely grouped together and almost completely separated from country. But none of the other genres show any logical grouping.

4.5 Genre classification

This visual estimation of the classification performance of the features suggests that the MFCC features will give the best classification performance and that that the RBM features give poor classification performance because of the strange structures that seem to dominate the feature space. We have performed a preliminary genre classification test using two simple classifiers, namely Linear Discriminant Analysis (LDA) and k-NN. To avoid parameter estimation we use no regularisation which means that we will use a 1-NN classifier. The classification errors on the training and test sets are shown in Table 2. Quite surprisingly we find that the classification error is lower for the magnitude spectrum features than the MFCC features which seemed to give a better grouping of the genres in the features space as indicated in Figure 4.

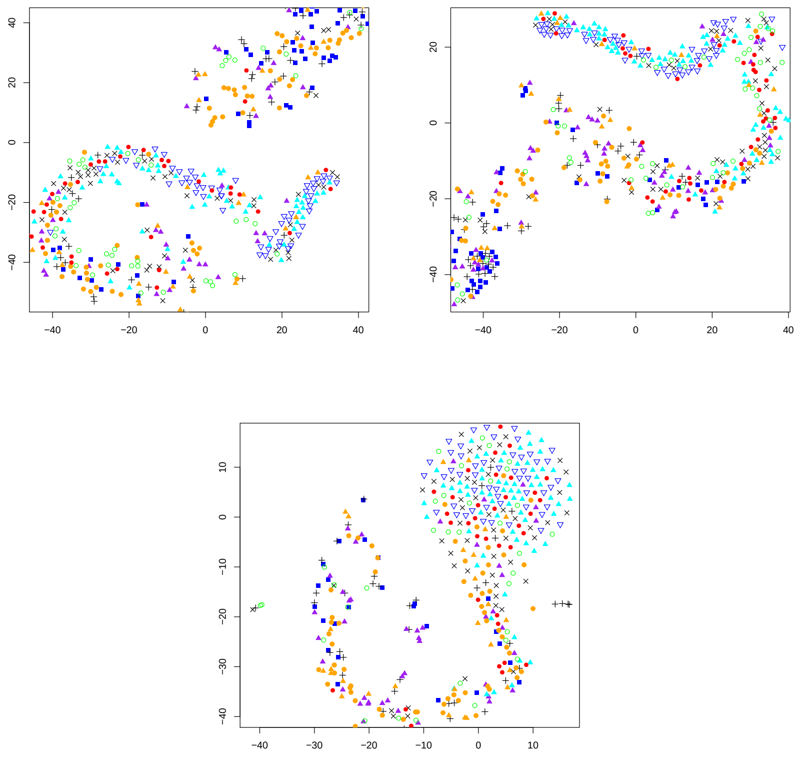

Figure 6: 2D visualisations of the RBM2 (top-left), RBM3 (top-right) and RBM4 (bottom) features with 50, 30 and 10 hidden units, respectively. 500 training iterations over 5 trials.

	LDA error	1-NN error
Spectrum	40.2% (29.9%)	47.5% (0.0%)
MFCC	66.7% (61.3%)	58.4% (0.0%)
RBM1	76.1% (70.9%)	75.5% (0.0%)
RBM2	77.3% (69.7%)	75.7% (0.0%)
RBM3	80.8% (73.9%)	77.1% (0.0%)
RBM4	86.6% (81.5%)	81.3% (24.6%)

Table 2: Classification test errors using LDA and 1-NN. The training errors are shown in parentheses.

The four RBM features give test errors that increase as the number of hidden units decrease. This is expected and it indicates that using 50 hidden units is reasonable for representing genres. We also note that the test error decreases slightly as the training iterations are increased. This could indicate that more iterations would result in a better representation. The high classification errors for the RBMs compared to the magnitude spectrum features (which were also the inputs to the RBMs) indicates that the RBMs actually "unlearn" some of the structure embedded in the spectrum features. This is unfortunate since the RBMs was supposed to learn better representations. The next section will discuss this issue in more detail.

A test using twice the number of frames in each song, i.e., 200 frames, and training the RBM1 on this set provided only a minor decrease in test error but a similar decrease was observed for spectrum and MFCC features. This indicates that it is not the number of training examples that is too small for the RBM to learn reasonable features.

5 Discussion

The experiments presented in the previous section showed that, contrary to the results of Hamel and Eck [7], our RBM features were not able to represent the music pieces in a way that would decrease classification error in the genre classification problem. In this section we discuss why this might be the case.

5.1 Supervised fine-tuning

The first thing to note is that our approach is slightly different from the approach utilised by Hamel and Eck. They perform unsupervised pretraining of the DBN using only 5 epochs and a very low learning rate and use this result as a starting point for the a supervised fine-tuning procedure. This fine-tuning is performed by adding a layer at the top of the DBN (as seen in Figure 2) which predicts one of the 10 genres. For this fine-tuning, 474 epochs are used with a learning rate of 0.1. This means that the features learned in the hidden layers of the DBN are already fitted for the classification problem. They do show, however, that the same features perform well on the auto-tagging problem.

In the approach presented in this report we train a Restricted Boltzmann Machine (i.e., only one hidden layer) in a purely unsupervised manner using Contrastive Divergence. Therefore, our features are not in any way weighted according the the genres which may explain why no clear clustering of the genres is shown when using the RBM features.

5.2 Parameter estimation

Several parameters must be decided upon in several places in a project such as this. Specifically, the RBM model has the following parameters that must be chosen: the number of hidden units, the number of CD steps, the number of iterations for each training trial, the number of trials and the learning rate for the gradient descend algorithm. These parameters should ideally be investigated thoroughly in order to obtain an idea of how they influence the results. We have tested with 10, 30 and 50 hidden units and showed that, similar to the results in [7], 50 hidden units seem reasonable. In [7] the best set of parameters was chosen by trying several parameter combinations on a validation set and since the approach presented in this report differ from their approach it would have been sensible to have performed a similar parameter selection.

5.3 Training time

The Tzanetakis dataset consists 1000 songs of 30 seconds duration but as described in the previous section, we only utilised a very minor part of this dataset due to the very long training time. It was not possible to get the program to run on a dedicated system so the training set was limited so that the training time took approximately eight hours for one parameter setting. Increasing the number of training iterations and trials was indicated to have caused the RBM to learn more sensible features. This is a subject for further investigation. Increasing the number of training examples seemed to decrease the test error for all the tried features and therefore was no benefit to the RBM itself.

5.4 Other input features

Besides the practical considerations of parameter estimation and training time one could also consider whether there exists more reasonable input features. The classification results are often compared with the MFCCs in the literature. Using the MFCCs as input to the RBM or to a DBN might make it able to learn even more abstract features that contain relevant information about the genre. Besides MFCCs further work should also investigate the use of spectrograms and wavelets as input features.

Another approach is to preprocess the high-dimensional (low-level) features and perform some sort of feature selection for use as input features for the RBM. This could for example be by performing Principal Component Analysis on the raw sound wave data or the DFT and selecting a given number of principle components exhibiting the highest variance.

5.5 Aggregated features

Only considering a 46 ms frame for classification clearly seems to be too small to make an estimate about the genre. Therefore it is suggested to perform a vote among the classifications of the frames for each song such that the entire song is classified to the class of the majority of its frames. An additional step suggested by Hamel and Eck is to use aggregated features by which we compute the mean and variance of the short-time features over a period of 5 seconds which increases the classification accuracy. A discussion of feature aggregation (or feature integration) is found in Meng et al. [14] where different short-, medium- and long-time features are investigated and analysed on the genre classification problem. A medium time feature called Autoregressive Model achieves the best classification accuracy.

6 Conclusion

Musical feature extraction is an area that finds applications in different musical information retrieval tasks such as tagging, genre classification, style recognition and recommendation systems. In this text we have examined musical feature extraction using Restricted Boltzmann Machines. Specifically, a theoretic introduction to the properties and the training of RBMs was presented and the unsupervised learning of features was investigated for the purpose of genre classification.

We tested four different RBMs with different input features and different number of hidden nodes and compared the learned features with two classical musical features, namely, the magnitude spectrum and MFCCs. For the magnitude spectrum we achieved a test error of 40.2% and 47.5% using an LDA and a 1-NN classifier, respectively, and for MFCC we achieved test errors of 66.7% and 58.4%. The learned RBM features that performed best achieved a classification error of 76.1% and 75.5%, respectively. This is much worse than the standard features and we therefore conclude that our learned representation was unable to extract more useful information about the genre from the individual frames than the two simpler features.

The parameters for the test was chosen in an ad-hoc fashion and therefore, in future work, we will investigate the choice of parameters more thoroughly and allow the RBM more training iterations in the hopes of reaching a better classification error. Another test would be to add a supervised fine-tuning step after the unsupervised pre-training. This is what is done by Hamel and Eck [7] which inspired this text. The classification was done on frames of 46 ms duration, which are arguably too small to completely determine the genre. A majority vote of frames should be used to determine the class of an entire song. Aggregated features such as the mean and variance over 5 seconds of frames are also reported in the literature to give better results.

Acknowledgements

I would like to thank Christian Igel for great supervision and for helping me understand the theory of RBMs. Also, I would like the thank Christian, Asja Fischer, Oswin Krause and Tobias Glasmachers for assisting me with the installation and use of the Shark C++ machine learning library. Thanks should also go to George Tzanetakis for helping me with the Marsyas audio processing framework.

References

[1] J.-J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):83–93, 2003.

[2] L. Barrington, R. Oda, and G. Lanckriet. Smarter than Genius? Human evaluation of music recommender systems. In *10th International Society for Music Information Retrieval Conference*, 2009.

[3] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009.

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[6] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Artificial Intelligence and Statistics*, 2005.

[7] P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In *Proceedings for 11'th International Society for Music Information Retrieval Conference (ISMIR)*, pages 339–344, 2010.

[8] G. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[9] C. Igel, V. Heidrich-Meisner, and T. Glasmachers. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

[10] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[11] H. Lee, Y. Largman, P. Pham, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in Neural Information Processing Systems (NIPS)*, 22:1096–1104, 2009.

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

[13] C. McKay and I. Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved. In *7th International Conference on Music Information Retrieval (ISMIR)*, pages 101–106, 2006.

[14] A. Meng, P. Ahrendt, and J. Larsen. Improving music genre classification by short-time feature integration. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume V, pages 497–500, march 2005.

[15] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing - Principles, Algorithms and Applications.* Pearson Prentice Hall, fourth edition, 2007.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, New York, NY, USA, 2001. ACM.

[17] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[18] L. van der Maaten and G. E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.