SUPERVISED FEATURE LEARNING VIA SPARSE CODING FOR MUSIC INFORMATION RETRIEVAL

A Thesis
Presented to
The Academic Faculty

by

Cian O'Brien

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Music Technology in the
School of Music

Georgia Institute of Technology
May 2015

SUPERVISED FEATURE LEARNING VIA SPARSE CODING FOR MUSIC INFORMATION RETRIEVAL

Approved by:

Dr. Alexander Lerch, Committee Chair
School of Music
Georgia Institute of Technology

Dr. George Tzanetakis
Department of Computer Science
University of Victoria

Dr. Jason Freeman
School of Music
Georgia Institute of Technology

Date Approved: 24 April 2015

Dedicated to W

ACKNOWLEDGEMENTS

First I would like to thank my advisor Dr.Alexander Lerch for his valuable help, advice and patience throughout the last two years. With his support I have gained an incredible amount from my time at Georgia Tech. Additionally, Dr.Jason Freeman and Dr.George Tzanetakis for their feedback on this thesis.

My gratitude to my friends and family for putting up with me during this last semester and especially my parents, without whom none of this would be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis explores the ideas of feature learning and sparse coding for Music Information Retrieval (MIR). Sparse coding is an algorithm which aims to learn new feature representations from data automatically. In contrast to previous work which uses sparse coding in an MIR context, the concept of *supervised* sparse coding is also investigated, which makes use of the ground-truth labels explicitly during the learning process. Here sparse coding and supervised coding are applied to two MIR problems: classification of musical genre and recognition of the emotional content of music. A variation of Label Consistent K-SVD is used to add supervision during the dictionary learning process.

In the case of Music Genre Recognition (MGR) an additional discriminative term is added to encourage tracks from the same genre to have similar sparse codes. For Music Emotion Recognition (MER) a linear regression term is added to learn an optimal linear regression model and dictionary pair jointly. These results indicate that while sparse coding performs well for MGR, the additional supervision fails to improve the performance. In the case of MER, supervised coding significantly outperforms both standard sparse coding and commonly used designed features, namely MFCC and pitch chroma.

CHAPTER I

INTRODUCTION

At its core, Music Information Retrieval (MIR) is concerned with learning and understanding music. Combining aspects from engineering, computer science and musicology, MIR seeks automatic methods of processing musical data in order to extract from it meaningful information. MIR employs a variety of low level signal processing steps in order to extract elements like pitch, rhythm, instrumentation and emotion. A key element in such analysis is the idea of a 'feature'. Given some music, we often extract some summary information while discarding the rest – much of the data might not be relevant to our task. A good feature then is one which compresses the input track while preserving some higher level aspect of the music we care about.

The concept of a feature computed from some input data is used widely in Machine Learning. For example, in the field of computer vision a clear trend is to take a hierarchal approach to features where higher level concepts are represented as the amalgamation of a series of lower-level atomic elements; a bicycle can be recognized by the presence of a spoked wheel, which itself can be represented as a particular configuration of edges and curves. Similarly for music the observed time domain signal can be represented in the frequency domain as a superposition of sinusoids using FFT analysis. The FFT is at the heart of many commonly used MIR features such as Mel Frequency Cepstrum Coefficients, Pitch Chroma, Spectral Flux etc. By carefully designing various stages of processing (such as filtering, dimension reduction and FFT analysis) algorithms for extracting information of interest from music signals can be implemented.

An alternative approach – which is explored in this thesis – is to instead *learn good features from music automatically.* Feature learning has shown promising results across a range of tasks in Machine Learning. Standard MIR approaches require careful design of task-specific features involving expert knowledge. In contrast, feature learning offers a data-driven method, where useful features can be learned automatically. With a good enough representation of the data, a wide range of problems may be tackled efficiently. The focus of this work is on a technique called *sparse coding* which aims to learn a collection of atomic elements which can be used to reconstruct unseen data. Similar to the bicycle or FFT example, the core idea is that new data can be represented as a superposition of these atomic elements and we can train a system to recognize certain configurations of these learned features.

Much of the theory behind sparse coding is outlined in the field of Compressed Sensing (CS). The well known Shannon-Nyquist sampling gives the conditions under which a continuous signal can be reconstructed given a limited number of measurements – to reconstruct the signal we need to sample it a rate greater than twice its highest frequency component. This however is a *sufficient* but not *necessary* condition and under certain conditions we can recover the signal using much fewer measurements than is dictated by Shannon-Nyquist. When viewed from this angle, sparse coding can be seen as a method of cleverly sampling from a distribution. By dictating that any reconstruction in the learned atomic elements (samples) is sparse, each learned element needs to be as informative as possible.

In the context of a classification problem, sparse coding can be used by taking the activation of each data point in the collection of learned atomic elements (i.e. by taking the magnitude of contribution of each element in the reconstruction) and using these activations as a new feature representation. Recent developments have introduced approaches to *supervised* coding, where we require that the learned atomic elements should be good at distinguishing between items of different classes by incorporating the ground truth labels into the learning process.

The goal of this thesis is to evaluate *unsupervised* sparse coding and *supervised* sparse coding in the context of MIR problems. These methods were applied to two tasks: Music Genre Recognition (MGR) and Music Emotion Recognition (MER). In MGR the aim is to automatically assign a piece of music as belonging to one musical genre using only properties of the signal. The problem of MER is to predict the emotional content of music as perceived by human listeners. These tasks were chosen because (i) they are widely studied and difficult problems in general which can potentially demonstrate the power of learned features over designed ones and (ii) they represent different kinds of tasks (one classification and one regression).

The main contribution of this thesis is the exploration of sparse coding for MIR and a comparison between standard and supervised sprase coding in this context. While there exist works which explore feature learning and sparse coding for MIR, none have explored sparse coding formulations which make explicit use of the ground-truth labels *during the learning process*. The problem of designing good features is time consuming and difficult, requiring expert knowledge and experience to do well. Feature learning offers a data-driven approach which can potentially surpass designed features by leveraging large amounts of (potentially unlabeled) data.

The thesis is structured as follows: Chapter 1 introduces the idea of feature and dictionary learning in general and sparse coding in particular, including some technical background. Chapter 2 gives a review of feature learning approaches applied to MIR problems, of which sparse coding is a subset. Chapter 3 describes the application of sparse coding to Music Genre Recognition while Chapter 4 deals with Music Emotion Recognition. The final chapter consists of a conclusion and possible future directions.

CHAPTER II

SPARSE DICTIONARY METHODS AND FEATURE LEARNING

Sparse methods have seen much use in a variety of signal processing problems where we wish to reconstruct a signal using as few measurements as possible. Sparse approaches are concerned with finding or recovering a compressed representation of a given input, with applications ranging from denoising [10, 23], computer vision [30, 24, 14, 61] and music transcription [1]. In general, a sparse model assumes that the given input (which may be of very high dimension) has much lower *intrinsic* dimension which can be recovered. In such situations the observed data can be reconstructed from a linear combination of *basis* or *atomic* elements while still retaining much of the important structure of the data. For example, images of size $m \times n$ can be represented as a vector living in $\mathbb{R}^{mn}$. However, the affective dimension of natural images is vastly smaller since neighboring pixel values are highly correlated and the majority of such images can be well-approximated by a combination of edges – the intuition behind the use of Haar wavelet basis for computer vision. Another example of a sparse model is the use of the Discrete Fourier Transform (DFT). Audio waveforms are in general extremely high-dimensional and non-sparse. By taking the DFT of a periodic signal, we can represent it as a sparse combination of coefficients in the frequency domain.

Sparse dictionary methods attempt to find a robust encoding of the input over some fixed dictionary such that the original input can be recovered (up to some error) by representing it as a linear combination of items from the dictionary. A sparse dictionary model therefore consists of two parts: selecting a suitable dictionary, and then finding the encoding of each input in that dictionary. Often the dictionary can be designed or chosen – for example using an FFT or wavelet basis. Alternatively, a suitable dictionary can be *learned* from the data. This approach is attractive because it provides a means of solving a large range of problems by taking a data-driven approach to learn a dictionary which is adapted to the data.

2.1 Sparse Coding

Sparse coding is a technique for learning a dictionary and appropriate activation coefficients which was initially proposed as a model for the activation of neurons in the brain [35]. Sparse coding leverages the concept of *sparsity* in order to learn a robust representation of the underlying data. We say a given vector is sparse if all but a few of its entries are zero.

Given an $f \times n$ matrix X with each column corresponding to one input, we assume that each input can be represented as a linear combination of *dictionary elements* and *activations* i.e. we have $X \approx D\alpha$. Here D is the $f \times m$ matrix whose columns correspond to dictionary elements (where m is the number of dictionary elements) and α is the $m \times n$ matrix whose columns give the activation of each dictionary element for the corresponding column in X. Such a factorization is in general ill-posed since infinitely many solutions for D and α exist, so in practice additional constraints are placed on D and α — for example Non-Negative Matrix Factorization assumes that each of these matrices are positive. In sparse coding it is required that the derived α should be as sparse as possible given X and D. The full formulation is given by the following optimization problem:

$$\begin{gathered} \min\nolimits_{D,\alpha} \|X - D\alpha\|_2^2 + \lambda\|\alpha\|_1 \\ \text{such that } \|D^{(i)}\|^2 \leq 1 \ \forall i \end{gathered} \tag{1}$$

where λ is a regularization term to control the sparsity of the results. Equivalently this can be written as

$$\begin{gathered} \min\nolimits_{d_i,\alpha_i^{(j)}} \sum_{j=1}^{n} \|x^{(j)} - \sum_{i=1}^{m} d^{(i)}\alpha_i^{(j)}\|_2^2 + \lambda \sum_{i=1}^{m} |\alpha_i^{(j)}| \\ \text{such that } \|d^{(i)}\|^2 \leq 1 \ \forall i \end{gathered} \tag{2}$$

High values of λ will drive more of the activations in α to zero while lower values result in less zero-coefficients. Hence this formulation balances two competing objectives: the learned dictionary/activations should reconstruct the data well (we should observe small ℓ_2 error between the original data and its sparse representation) and the activation matrix should be sparse (each input should be represented by as few dictionary elements as possible).

In practice, the success of sparse coding often depends crucially on good selection of the sparsity parameter λ. Note that we constrain the norm of each dictionary element d, since we can always decrease the ℓ_1 penalty while leaving the reconstruction penalty unchanged by increasing d without bound while decreasing α.

2.1.1 Sparsity and the ℓ_1-norm

A notable feature of the objective function given above is the use of the ℓ_1-norm penalty. Since the goal is to recover the *sparsest* solution, a natural approach would be to penalize each $\alpha^{(i)}$ using the ℓ_0 pseudo-norm which counts the number of non-zero elements. This quantity is difficult to work with in practice however since it is discontinuous and undifferentiable. The ℓ_1-norm makes solving the optimization problem much easier while still leading to sparse solutions.

To understand intuitively why the ℓ_1-norm induces sparsity, consider the nature of the objective function: it is composed of a sum of a quadratic ℓ_2 reconstruction penalty and an ℓ_1 sparsity penalty. The solution of such a system must lie on a tangent of both level-sets (since otherwise we could decrease the objective function by following one of the level-sets in the direction which decreases the other). Since the level-set of the ℓ_1-norm is a polytope with corners on each axis and the level-set of the ℓ_2-norm is an ellipsoid, when the size of the ℓ_2 error gets suitably big the tangent point will tend to lay at one the corners of the polytope i.e. the points where the solution is sparse.

The objective function can be derived from a probabilistic interpretation [1]: we assume a model where observations X are composed of a latent variable plus additive zero-mean Gaussian noise:

$$X = D\alpha + \epsilon \tag{3}$$
$$\epsilon \sim \mathcal{N}(0, \sigma)$$

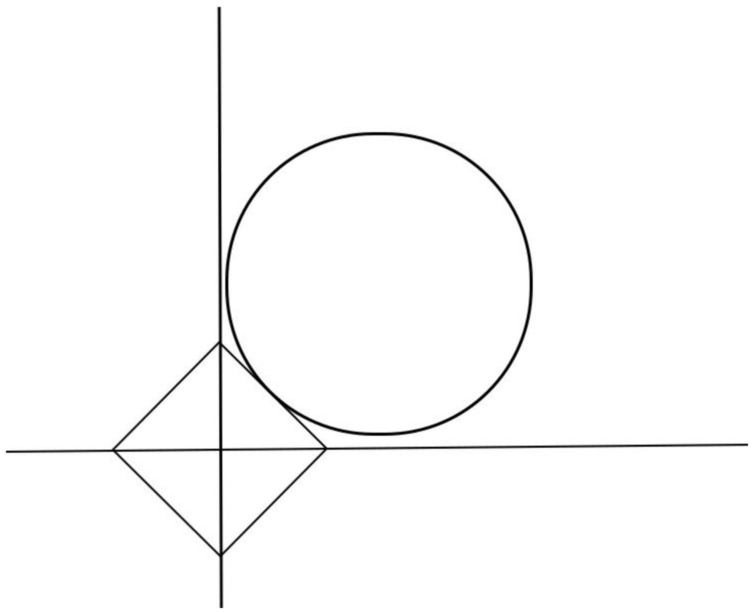

Figure 1: Level sets of the ℓ_1 norm (polytope) and ℓ_2 norm (circle). Solutions of the objective function lie on the tangent at their intersection.

For fixed D and X the maximum a posteriori estimate $\hat{\alpha}$ for the activations is given by:

$$\hat{\alpha} = \text{argmax}_\alpha \quad p(\alpha|X;D) = \frac{p(X|D;\alpha)p(\alpha)}{p(X;D)} \tag{4}$$

$$= \text{argmax}_\alpha - \log\left(\frac{p(X|D;\alpha)p(\alpha)}{p(X;D)}\right) \tag{5}$$

$$= \text{argmax}_\alpha - \log\Big(p(X|D;\alpha)\Big) - \log\Big(p(\alpha)\Big) \tag{6}$$

Using Equation 3 and assuming the α_i are independent we have

$$\hat{\alpha} = \text{argmax}_\alpha - \log\Big(p(X|D;\alpha)\Big) - \sum_{i=1}^{n} \log\Big(p(\alpha_i)\Big) \tag{7}$$

$$= \text{argmax}_\alpha \quad \frac{1}{2}\|X - D\alpha\|_2^2 - \sum_{i=1}^{n} \log\Big(p(\alpha_i)\Big) \tag{8}$$

To ensure sparse solutions we need to choose $p(\alpha)$ such that its probability mass is highly peaked around zero. A common choice is the zero-mean Laplace distribution given by

$$p(\alpha|\lambda) = \frac{1}{2\lambda}\exp\Big(-\frac{|\alpha|}{\lambda'}\Big) \tag{9}$$

and substituting this into equation 8 gives the final sparse coding objective function

$$\hat{\alpha} = \text{argminx}_\alpha \quad \|X - D\alpha\|_2^2 + \lambda \sum_{i=1}^{n} |\alpha_i| \tag{10}$$

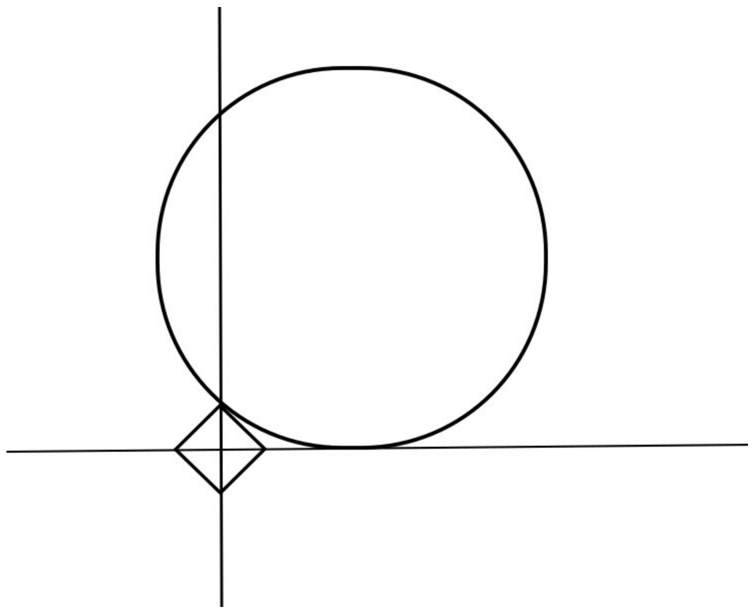

Figure 2: Due to the shape of the ℓ_1 level set, points of intersection will tend to lie at a 'corner' of the polytope where the solutions are sparse. Here the sparsity penalty corresponds to the size of the polytope.

Hence we find that the use of the ℓ_1-norm can be explained as being derived from the choice of prior on the activations. Note that we could use other distributions for $p(\alpha)$ – for example a Normal distribution which results in an ℓ_2 sparsity penalty (and non-sparse solutions). In general the solutions will be sparse assuming $-\log \ p(\alpha)$ is concave [1].

2.2 Feature Learning via Sparse Coding

Often we want to use the learned dictionary and activations in some kind of classification or regression task. Normally the first step in such problems is to extract appropriate *features* from the input. A feature can be thought of as a reduced or compressed representation of the input data which preserves the important qualitative information we want to use for the classification problem. Some commonly used audio features are Mel Frequency Cepstrum Coefficients (MFCC), Zero-crossing Rate (ZCR), Spectral Centroid (SC) etc. Each of these features capture some important quality of the input audio excerpt while drastically reducing its dimension. Vectorized features can then be used as input to general Machine Learning algorithms for model learning and classification.

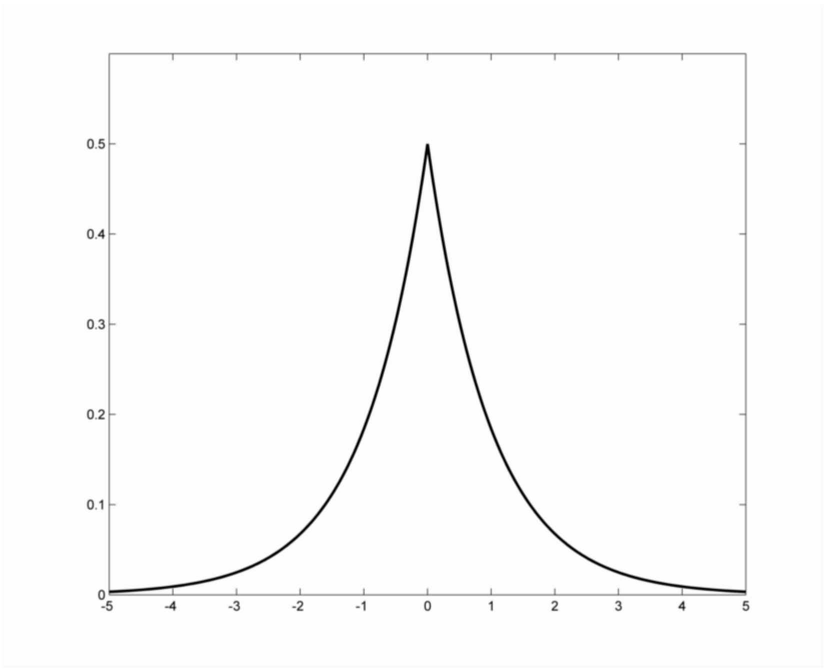

Figure 3: Probability Density Function for a Laplace Distribution. The pdf is highly peaked around 0 which encourages many zeroed coefficients. The scaling parameter λ controls the peak and spread of the distribution.

To achieve acceptable performance, good features are crucial. Much time is spent designing features using expert domain-knowledge and while such features can be effective they are extremely difficult to design well. An alternative paradigm is to *learn* features from data automatically, which can potentially lead to increased performance since the learned features will be specifically adapted to the data at hand. Sparse coding and similar methods have seen widespread use as feature learning approaches. Features can be learned from data as follows: given input matrix X, solve the above optimization problem to find D. Having learned the dictionary, new features for the input are generated by solving the optimization problem for α using fixed D and taking the learned α as the new feature representation, so that the learned representation for each input column of X is given by the corresponding column in α. Intuitively this can be thought of as a change of basis and by choosing a large dictionary size we effectively project the data into a higher dimensional space where it is more likely to be linearly separable.

A strength of sparse coding under this paradigm is that it can learn meaningful representations using *unlabeled* data and the learned dictionary will be useful in classifying future, unseen data. This is because the algorithm tends to learn a representative set of atomic elements which, when summed, can be used to reconstruct data with similar structure as the collection on which they were learned – for example when trained on natural images sparse coding will learn a dictionary of edges with varying degrees of rotation and contrast. This ability to leverage unlabeled data (which is much more easy to come by than labeled data) motivates its use a *self-taught* learner [33].

2.3 Solving the Objective

The sparse coding objective function is non-convex in D and α simultaneously, however it *is* convex for each one when the other is fixed. The strategy for solving the objective therefore alternates between solving for α with fixed D, and D with fixed α. Both of these sub-problems have been well studied and efficient methods for solving them exist. For learning α we have Orthogonal Matching Pursuit (OMP) [38], Least Angle Regression and Shrinkage (LARS) [8], Basis Pursuit (BP) [7] etc. Algorithms for finding D include K-SVD [2], Online Dictionary Learning (ODL) [29] and Efficient Sparse Coding [26]. Here we give special attention to ODL and LARS since they are the methods used for all experiments in this work.

2.3.1 Least Angle Regression and Shrinkage

Least Angle Regression and Shrinkage (LARS) [9] is an algorithm for solving the so-called LASSO, which stands for Least Absolute Shrinkage and Selection Operator. This problem belongs to a family of regularized regression problems where we want to learn a mapping from input to observed variables which minimizes the ℓ_2 error i.e. we want to solve

$$\min_W \|Y - XW\|_2^2 + \lambda\psi(W) \tag{11}$$

where ψ is a regularization function. Choosing $\psi = \ell_2$ results in a Ridge Regression model [20] while $\psi = \ell_1$ gives us the objective for LASSO regression (and the activation update step in sparse coding):

$$\min_W \|Y - XW\|_2^2 + \lambda\|W\|_1 \tag{12}$$

Because the ℓ_1 induces sparsity, this algorithm performs a kind of *variable selection* – it will attempt to model the responses using as few predictors as possible and this property can be desirable when the relationship between input and output is important since it can learn which variables are important for prediction.

Returning to sparse coding, the activation update sub-problem for fixed D is

$$\min_\alpha \|X - D\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{13}$$

For a given input x_i the LARS algorithm starts with no variables (it has zeros everywhere). From a fixed dictionary it then finds the dictionary item most correlated with x_i by taking the dot product of its residue with each dictionary item. Upon finding the most correlated dictionary item d_1, it proceeds in the direction of that dictionary item until it finds another dictionary item d_2 which is equally correlated with the residue of x_i. This dictionary item then gets added to the model and we proceed in such a way as to keep the residue of x_i equally correlated with d_1 and d_2. It iterates in this manner, gradually adding dictionary items. By requiring that once an added variable goes to zero it remains zero throughout the rest of the process, LARS will result in sparse solutions. A desirable feature of LARS is that it provides us with the full regularization path i.e. it gives the solutions for all values of regularization penalties up to and including λ.

2.3.2 Online Dictionary Learning

Online Dictionary Learning (ODL) [29] is a scalable method for sparse coding. By using stochastic gradient descent with warm restart, a dictionary can be learned in an online manner and can thus incorporate new training samples. At each step of the algorithm, the activations α are updated via LARS using the dictionary D_{t-1} learned at the previous step. The updated dictionary D_t is then given by

$$f(D) = \frac{1}{t}\sum_{i=1}^{t} \|x_i - D\alpha_t\|_2^2 + \lambda\|\alpha_i\|_1 \tag{14}$$

The advantage of this approach is that D_t can be learned efficiently using gradient descent since it will be close to the previous dictionary D_{t-1}. Another key point is that the function f keeps track of the past through α_i and the authors show that under mild assumptions f converges to the true *expected* cost in the limit. To update the dictionary, at each iteration ODL sets

$$\begin{aligned} A_t &\leftarrow A_{t-1} + \alpha_t \alpha_t^T \\ B_t &\leftarrow B_{t-1} + x_t \alpha_t^T \end{aligned} \tag{15}$$

and then the update for the j^{th} dictionary item of D_t is given by

$$\begin{aligned} u_j &\leftarrow \frac{1}{A_{jj}}(b_j - D\alpha_j) + d_j \\ d_j &\leftarrow \frac{1}{\max(\|u_j\|_2, 1)} u_j \end{aligned} \tag{16}$$

which is the solution to

$$\begin{aligned} &\arg\min_D \frac{1}{t} \sum_{i=1}^{t} \|x_i - D\alpha_t\|_2^2 + \lambda \|\alpha_i\|_1 \\ &= \arg\min_D \frac{1}{t} (\frac{1}{2} \text{tr}(D^T D A_t) - \text{tr}(D^T B)) \end{aligned} \tag{17}$$

2.4 Extensions of Sparse Coding

Under the standard sparse coding formulation given above the only terms which appear in the minimization problem are the reconstruction error term $\|X - D\alpha\|_2^2$ and the sparsity penalty $\|\alpha\|_1$. In particular we find that any label information we might have for the input is not included in the learning process. The ability for sparse coding to learn meaningfully from un-labeled data is a desirable property, however recent works have investigated supervised approaches where we make explicit use of the labels to learn a dictionary adapted for a specific task.

2.4.1 Supervised Discriminative and Local Coding

Jiang et al. [24] a modified version of sparse coding with two additional error terms: a "discriminative" term given by $\|Q - AX\|_2^2$ and a classifier term $\|H - WX\|_2^2$. Here Q is a binary matrix which encodes the class information of each column of the data matrix X and A is a linear transformation which maps the original sparse codes to be more discriminative in the new feature space. Unlike standard sparse coding, each dictionary item has a class label so that for each a given input and dictionary item, the corresponding term in Q will be 1 if they belong to the same class and 0 otherwise, which means examples which share a class label will be mapped to be 'close together' by A. The matrix H corresponds to the true class information for each input and W is a linear classifier. Each column of H corresponds to an input and each row corresponds to a class. For a given input which is a member of class c the c^{th} row of the column correspondong to the input in H will be 1 and the others 0. The full problem is given by

$$\min_{D,W,A,\alpha} \|X - D\alpha\|_2^2 + \gamma\|Q - A\alpha\|_2^2 + \beta\|H - W\alpha\|_2^2 + \lambda|\alpha| \tag{18}$$
$$\text{such that } \|D_i\|_0 \leq T \; \forall i$$

where T is a "sparsity constraint factor (each signal has fewer than T items in its decomposition)"[24]. An advantage to this approach is that it learns a dictionary and classifier jointly. In order to classify new points they are encoded in the learned discriminative dictionary and use the linear predictive classifier W for classification, using the index of the maximum value of $W\alpha_i$ as the label for the activation vector α_i. The system was tested on face and object recognition tasks. A similar approach was was used in [61] without the discriminative term and applied to face recognition.

A problem with standard sparse coding is that similar input data may end up getting dissimilar sparse codes which degrades the performance of the system. In some situations then it can be desirable to preserve the 'local' information of each input so that similar items have similar sparse codes. This is a similar idea to discriminative methods except that the idea of similarity is given by distance in the original space. An example of such a 'local coding' method is Laplacian Sparse Coding [14]. A Laplacian matrix L is often used in studying graphs comprised of nodes and edges and is given by $L = D - A$ where D is the degree matrix (which contains the degree of each node) and A is the adjacency matrix (which contains the edge information). Using a Laplacian L constructed from training data using local similarity information the Laplacian Sparse Coding problem is given by

$$\min_{D,\alpha} \|X - D\alpha\|_2^2 + \lambda\|\alpha\|_1 + \alpha tr(\alpha L \alpha^T) \tag{19}$$
$$\text{such that } \|D^{(i)}\|^2 \leq 1 \; \forall i$$

This approach has shown good performance in image classification where local spatial information is important.

Mairal et al. used a logistic loss penalty to incorporate the labels in the learning process [32]. Under this approach a classifier is learned jointly with the dictionary during training which makes the optimization procedure more difficult. To deal with this their approach consists of two stages: a supervised sparse coding stage (where the activations α_i are updated) and a supervised dictionary update step (where the dictionary D and model parameters θ are updated for fixed α_i). Formulations such as this where the dictionary and classifier model are updated separately can suffer from poor convergence and local minima.

2.5 Conclusion

In this chapter, the concept of sparse dictionary learning for feature learning has been introduced. In particular the details of the sparse coding algorithm and its application to feature learning has been discussed. An iterative method for solving the accompanying optimization problem was given using the LARS and ODL algorithms to solve the activation and dictionary learning sub-problems. Additionally, extensions of sparse coding which incorporate label ground-truth as well as other information were described.

CHAPTER III

FEATURE LEARNING FOR MUSIC INFORMATION RETRIEVAL

Music Information Retrieval (MIR) is concerned with extracting meaningful information out of musical signals. Some common problems in this field are chord detection, genre recognition, mood estimation, song similarity and automatic transcription. Given the high dimensionality of musical signals, a wide variety of features have been designed to capture important information while reducing the dimension to a reasonable size. Timbral information can be captured using Mel Frequency Cepstrum Coefficients (MFCC) for example, which have seen widespread use in genre and instrument recognition where the timbral characteristics of the signals are a distinguishing factor. Pitch Chroma are a 12-dimensional feature derived from the Fourier Transform, which group the magnitudes into bins corresponding to each note. This feature is therefore mostly independent of the timbre and instead captures tonal and pitch information.

As discussed previously, designing good features is extremely important and not an easy task. Several works have investigated approaches to learning good features automatically using Machine Learning techniques. Here several examples are reviewed.

3.1 Feature Learning for Music Information Retrieval

Humphrey et al. argue for the usefulness of feature learning and Deep learning in the context of MIR [21]. They argue that current MIR approaches are hitting a "glass-ceiling" in terms of performance across a range of tasks. They present a two-stage conception of how most MIR research is carried out – the first stage consists of transforming the data such that "its defining characteristics are made invariant across multiple realizations" which amounts to feature-encoding of the input. The next stage is semantic organization where "semantic meaning can subsequently be inferred and used to assign labels or concepts to it", which corresponds to designing some model to meaningfully interpret it (for example a classification or regression algorithm). The authors observe a trend in MIR to design more sophisticated classifiers (stage two) to the same set of features, which they evidence by the fact that 19 out of 20 submission to the International Society for Music Information Retrieval Conference (ISMIR) dealing with chord recognition used chroma features.

Hamel et al. trained a Deep Belief Network (DBN) to recognize music genre [17]. A Deep Belief Network is a Neural Network consisting of an input layer, a series of "hidden" layers and a final output layer. An interesting aspect of their approach is that instead of using the output of the DBN directly for prediction as would normally be the case, they experimented with using the output of the intermediate hidden layers as features. The output of each layer can be taken as higher level representation of the input data. The features learned in this way were used as input to a non-linear Support Vector Machine (SVM) classifier. They compared several systems: using the output of the DBN directly, using the outputs of the intermediate layers and finally using the mean and standard deviation of MFFCs. It was found that using the activations of each layer lead to better performance than taking the final output activations alone. They showed that learned features on Discrete Fourier Transforms of patches outperformed traditional MFCC features, achieving an accuracy of 84.42% on the genre classification task using the GTZAN genre dataset. One issue is that their system was evaluated without cross-validation: for their experiments the data set was divided in to a 50/20/30 train/test/validation split and as such direct comparison with other systems is not possible. This was due to the extremely large training time of 104 hours for the DBN.

Wulfing et al. used Constant-Q transforms as input features convolutional k-means for feature learning [54]. First, extracted patches were normalized to have zero mean and unit variance before being processed using Principal Component Analysis (PCA). Then k-means was used to learn a series of clusters in feature space. Clusters were learned using a bootstrapping technique whereby clusters are first learned on the space spanned by the first p principal components, with the entries corresponding to the other dimensions set to zero. These centroids were then used as a warm-start for the full learning process. This collection of centroids plays an equivalent role to the dictionary used in sparse coding – unseen items are encoded in this "dictionary" by soft thresholding based on distance to each centroid. They also noted the importance of feature pooling before the classification stage, whereby adjacent short-time frames are pooled over several seconds. Final classification was performed using a linear SVM resulting in an accuracy of 85.25% ± 3.5% on the GTZAN data set.

Henaff et al. used *Predictive Sparse Decomposition* (PSD), also for genre recognition [18]. PSD overcomes one of the challenges in using sparse coding where fast encoding is required, for example with extremely large data sets. Given a new input, sparse coding requires the optimization of the objective function using the given dictionary making it computationally expensive. PSD adds an additional term to the objective function, which learns a linear classifier to predict the encoding of each input in the learned dictionary. Items can then be encoded very quickly using matrix multiplication with the learned linear transformation.

They found a significant increase in performance by learning separate octave-specific dictionaries. Instead of learning one dictionary on the full spectrum, the input is split into four octaves and separate encoders were trained on each. Learned feature were then given by concatenating the activations in each dictionary. An accuracy of 83.4% ± 3.1% was reported using the GTZAN dataset. They note that their system is a "tranductive learner" since the dictionary is learned on all the input items. Performing 10-fold cross-validation when strictly separating the dictionary learning and testing resulted in an accuracy of 80%.

Traditional sparse coding is an unsupervised algorithm. Yeh et al. introduced a supervised sparse coding scheme for genre recognition [60]. Given a training split, ten different dictionaries were learned on each genre and the final dictionary was given by combining them into a single unified dictionary. They found that this approach significantly outperformed k-means for feature learning and report 84.7% classification accuracy on the GTZAN dataset using 10-fold cross-validation. This however represented only a small increase over traditional sparse coding, which they report at 83.4%.

Humphrey et al. demonstrated the potential of feature learning in the context of cover-song detection [21]. Given a large music corpus, the goal was to retrieve likely covers of a query track. This problem is most commonly tackled as ranking problem, where each potential track in the library is ranked according to its perceived similarity to the query. First, pitch chromagrams were taken and pooled and aligned over each beat. Then a 2D Fast Fourier Transform was calculated to summarize each song and after some pre-processing k-means clustering was used to learn clusters which could be used as a feature dictionary. Next they describe a process to "semantically organize" the space. For good performance, song/cover-song pairs should be close together (in the Euclidean distance sense) and song/non-cover pairs should be far apart. Supervised dimensionality reduction was performed using large scale Linear Discriminant Analysis, where each "clique" of cover songs was taken as one class. Evaluation was performed using the SecondHandSongs subset of the Million Song Dataset [5] resulting in a Mean Average Precision score of 13.4% $-$ a significant improvement in the MAP score of 2.95% reported in [11] which used the same raw features without any feature learning.

Features for sound effect ranking, mood classification and Classical composer classification where obtained by Ness and Lyon [34] using k-means clustering on Stabilized Auditory Images (SAI). SAI are features inspired by the auditory processing of the human ear. They found that features learned in this way perform similarly or better than established MFCC features, which were also coded using k-means. Unlike sparse coding, their algorithm used hard assignment of inputs to each learned cluster – a process known as vector quantization. The resulting feature vector then contains all zeros except at the index corresponding to the cluster to which it was assigned.

Unsupervised sparse coding using convolution was explored by Grosse et al., along with an efficient method for solving the optimization problem [16]. Convolutional methods may be more suitable when dealing with music since it is highly temporally dependent, where each state in the signal is depends on the on previous states. Under standard schemes basis elements can potentially be 'wasted' learning redundant features (for example multiple dictionary items learning the same feature shifted in time). Convolutional approaches are robust against such errors since they find common activation patterns in the whole input at once.

The algorithm was tested on speaker identification and music genre detection, finding that the learned features performed on-par with traditionally designed features. A similar idea appears in [6] where shift invariant sparse coding is used for blind source-separation and identifying individual notes in polyphonic mixtures.

The use of sparse coding as a *self-taught* learning algorithm is investigated by Markov et al. in the context of music genre detection [33]. Here the idea was to examine the sparse coding algorithm's ability to leverage large amounts of unlabeled data to learn good features for genre recognition. Often large amounts of good training data can be difficult to come by so having the option to make use of both labeled and unlabeled data can be very useful. The GTZAN data set was split into a series of smaller sets consisting of 50, 100, 250 and 500 examples per genre. Additional, several sizes of dictionary were tested:100, 200, 300 and 500. Evaluation was performed using a SVM trained on labeled examples taken from another data set. In order to isolate the effect of additionally unlabeled data, a baseline system learned purely on a labeled set was learned. They concluded that in situations where small amount of labeled data are available, sparse coding on unlabeled data can be used to learn good features for genre detection.

In [1] sparse coding is used to learn a dictionary of spectral characteristics of short-term Fourier spectra. They note "the ability to learn a dictionary of atomic spectra (most of which converge to harmonic spectral profiles associated with specific notes) from polyphonic examples alone−no separate training on monophonic examples is required" [1] as a key feature in their implementation. This ability of sparse coding to learn robust atomic elements from mixtures and in the presence of noise is one of its main strengths.

Scmidt et al. learned features for music mood recognition using a Deep Belief Network (DBN). Predicting the mood of a piece of music amounts to a regression task; songs are represented as a vector in 2-D "Arousal-Valence" space [40]. Arousal corresponds to the energy in a song, while valence measures the emotional content in terms of positive or negative emotions. As the authors note, many mood identification systems rely on a large amount of designed features involving loudness, rhythm, timbre and harmony together with dimensionality reduction techniques. Similar to the work of Hamel et al. they tested the performance of the features learned on each layer as well as the final output. They also tested the addition of a linear regressor layer, to learn a feature set and classifier jointly. They found that the best performing learned features outperformed traditionally used features such as MFCC, Spectral Contrast and chroma in terms of Average Mean Distance and Average KL Divergence.

The highest published results for genre recognition on the GTZAN data set were presented by Panagakis et al. [37]. Their approach consisted of two stages: a biology inspired acoustic feature and a spare-representation based classifier. They calculate 2D auditory representations for each input which model the auditory processing of the primary auditory cortex. First auditory spectrograms are calculated, which emulate the early auditory system. This stage consists of the calculate of Constant-Q tranforms using a bank of 96 filters. Next, "derivatives with respect to the logarithmic frequency are taken", followed by half wave rectification. The modeling of the later stage auditory processing consists a "multi-resolution wavelet analysis" using a bank of Gabor filters.

To perform classification, a dictionary and projection matrix are learned using sparse coding. The projection matrix here serves the purpose of dimensionality reduction, effectively projecting the coded data into a smaller subspace. Dictionaries are learned on each class separately and the final classification is given by the dictionary which minimizes the ℓ_2 reconstruction error. They report a significant improvement over other methods: 91% and 93.56% on the GTZAN and Ismir2004Genre data sets respectively.

An attempt at replicating the results of Panagakis et al. was made by Bob Sturm [49]. Using the same methodology outlined by Panagakis [37] he obtained an accuracy of 68% on GTZAN, noting that the only way he found comparable performance to Panagakis et al. was to limit the amount of genres to five. Elsewhere, of the results of Panagakis et al. the author say the are due to "... accidentally using the true labels in classification (private correspondence with Y. Panagakis)"[46].

3.2 Discussion

As we can see, learned features can outperform traditional features like MFCC and chroma in a variety of MIR tasks. In reviewing these approaches we find many similarities:

- Approaches tend to use features derived from the Discrete Fourier Transform such Constant-Q transforms or raw spectra as input [21] [54] [18] [18]
- These input features are computed over short time frames such as 50ms *Predictive Sparse Decomposition* [60] [33]
- Often a pre-processing step such as whitening, Principal Component Analysis or feature-standardization/normalization is used [54] [18]
- A suitable dictionary is learned on these short time features, often using k-means or sparse coding dictionary learning [18] [60] [33] [37] [54]
- Items are encoded in this dictionary using vector-quantization (hard assignment), k-means with soft-assignment, LASSO regression (sparse coding) or others [54] [60] [33] [18]
- Encoded data are pooled over a perceptually-relevant timescale (typically 1 to 5 seconds) [18] [60] [33]
- These pooled features are used to train a classifier [60] [33]

The approach outlined in this thesis builds from the ideas discussed so far, in particular sparse coding and dictionary learning approaches ([60] [33] [18] [37]). Similar works can be divided into two categories: unsupervised ([33], [18]) and supervised ([60], [37]) sparse coding. Of the supervised works, we note that in none of these examples are the ground truth labels *explicitly* used during learning. When supervision is employed the labels are used only weakly or implicitly by learning separate dictionaries on each class independently before combining them into one final dictionary, with the hope that examples from a given class will show more activation in their exemplar dictionary. The goal of this work is to explore methods of using the ground-truth information explicitly during the dictionary learning stage, in order to learn a dictionary with both reconstructive and discriminative power.

3.3 Conclusion

In this chapter, on overview of the use of feature learning in MIR was given. Several different approaches for feature learning such as soft k-means, vector-quantization and sparse coding were discussed. Examining the similarities between these approaches, we find a general pipeline for feature learning in a musical context which includes pre-processing, dictionary learning, encoding and feature pooling. Table 1 summarizes the approaches outlined above, including information on whether they are supervised or unsupervised during the feature learning stage.

Table 1: Some examples of feature learning in Music Information Retrieval.

Reference	**Feature Learning**	**Task**	**Supervised**
[17]	Deep Belief Network	Genre Recognition	Yes
[54]	Convolutional k-means	Genre Recognition	No
[18]	Predictive Sparse Decomposition	Genre Recognition	No
[60]	Sparse Coding	Genre Recognition	Yes (multiple dictionaries)
[22]	k-means (centroid distance)+LDA	Cover-Song Detection	Yes (LDA dim. reduction)
[34]	k-means (vector quantization)	Mood, Composer, sound-ranking	No
[16]	Convolutional Sparse Coding	Speaker ID, Genre Recognition	No
[6]	Shift Invariant Sparse Coding	Blind Source Separation, Note ID	Yes (musical priors)
[33]	Sparse Coding	Genre Detection	No
[1]	Sparse Coding	Note Feature Learning from Polyphonic Audio	No
[42]	Deep Belief Network	Music Emotion Recognition	Yes
[34]	Sparse Coding on Auditory Features	Genre Recognition	Yes

CHAPTER IV

SPARSE CODING FOR MUSIC GENRE RECOGNITION

In this chapter the use of sparse coding for the problem of Music Genre Recognition (MGR) is investigated. While the concept of genre is in general not well-defined, most popular music can be said to belong to a specific genre such as pop, rock, jazz, classical etc. The goal of a MGR system is to assign a genre label automatically to an input audio signal. MGR systems are an extremely popular area in Music Information Retrieval (as is revealed in Table 1). For an extensive overview of MGR systems and their evaluation, see the work of Sturm [48].

The goal of this chapter is to explore the use of features learned using standard sparse coding and supervised sparse coding for MGR. For supervised feature learning, an approach based on Label Consistent K-SVD [24] was used to learn a discriminative dictionary. Sparse coding and Label Consistent K-SVD have primarily been used in the context of computer vision, and their application to music signals presents many challenges related to pre-processing and feature representation. First an overview of some related work in the area of MGR followed by an outline of the implementation and methodology used for evaluating both approaches.

4.1 Related Work

One of the first works which explored MGR was done by Tzanetakis and Cook [51]. Here several features were used such as Spectral Centroid, Spectal Rolloff, Spectral Flux, Zero Crossing Rate and Mel Frequency Cepstrum Coefficients. Additionally, beat features were calculated from a beat histogram. An important consideration introduced by Tzanetakis is the concept of a *texture window.* Many of the above mentioned features are derived from short-time Fourier Transforms divided into so called *analysis windows.* As the authors note, these analysis windows need be short so that the “the frequency characteristics of the magnitude spectrum are relatively stable”. However, much of the perceptual content of music is found over long time scales which motivates the use of a *texture window* – a longer time block in which short term features are aggregated. In their work, Tzanetakis and Cook used a texture window of 1 second over which means and variances of the short time features were calculated. For classification, they used Gaussian Mixture models and k-Nearest Neighbor to predict the genre label using features calculated both over the whole file and in sub-segments.

Tzanetakis et al. introduced the idea of a pitch histogram for MGR using audio and symbolic (e.g. MIDI) data [53]. Pitch Histograms aim to capture the melodic information contained in a piece of music consisting of a list of 128 numbers (indexed by MIDI note) which give the number of occurances of that note in the piece. In the case of MIDI information, this information is easily calculated using note-on and note-off events. In the case of audio data, the authors used a polyphonic pitch detection scheme to estimate the pitch content. The inevitable error introduced in the pitch detection algorithm is estimated by comparing the performance of MIDI to audio-from-MIDI. They concluded that the pitch content contained valuable genre-related information. Using audio derived from MIDI and a selection of timbral features [51] they achieved $75 \pm 6\%$ accuracy on a dataset consisting of five genres; Electronica, Classical, Jazz, Irish Folk and Rock.

Bergstra et al. used a large number of features and an Adaptive Boosting framework for MGR, achieving the highest reported accuracy at the MIREX 2005 MGR task [3]. Given a collection of features, the AdaBoost algorithm [13] iteratively builds a strong classifier out of a series of weaker classifiers. At each step of the algorithm, training examples are weighted according to the error of their classification with the current classifier. At each stage an optimal weak classifier is learned and added to the ensemble with an appropriate weighting. In this way, one can construct a classifier which exhibits very strong performance by systematically combining the output of a group of simple classifiers. The authors used a simple decision stump as the weak learner at each stage, which is basically an axis-aligned separating hyperplane. As input features they took many common timbral features such as FFT coefficients, MFCCs, zero-crossing rate, spectral spread, spectral centroid, spectral roll-off and auto regression coefficients.

They compared three different strategies for classification: using the derived short time features, using features aggregated over the whole song and features aggregated over segments. They found that taking features over segments lead to the best performance, similar to the findings of Tzanetakis with respect to texture windows.

Feature learning based approaches for MGR have been discussed in the previous chapter, using unsupervised ([16, 33, 18]) or some manner of supervised ([60, 37]) learning where separate dictionaries are learned on each genre and then combined. However, the examples listed here do not incorporate the ground truth in the learning process explicitly in the optimization problem as is the goal of this work.

4.2 Implementation and Methodology

The data set used for experiments was the GTZAN collection consisting of 1000 clips of music. Each clip is 30 in duration and belongs to one of 10 genres: blues, country, classical, rock, reggae, jazz, disco, metal, pop and hip-hop. Each genre has 100 examples. As verification, first unsupervised sparse coding was implemented with the dictionary learned on the full data set. Admittedly this dataset has some problems (see for example the analyses by Sturm [45][47]) however its use here is justified by the fact that (i) it is widely used throughout the literature and (ii) we are more interested in the comparative performance of standard versus supervised sparse coding. After achieving satisfactory results unsupervised and discriminative coding were implemented with proper cross-validation (i.e. the dictionary was learned only on the test set).

4.2.1 Initial Implementation: Unsupervised Dictionary Learning

The full sparse coding process can be very slow when the dataset is large. As an initial implementation unsupervised sparse coding was used, with the dictionary being learned on the full data set. Classification was performed using a Linear Support Vector Machine and stratified 10-fold cross validation: first the dataset was split into 10 subsets such that each set had the same number of tracks from each genre. Each set was used as a testing set once, with the remaining data being used to train the SVM. The final accuracy was taken as the average across each set.

For input to the sparse coding algorithm, we first extract short-time features from each input file. Both standard DFT and Constant-Q transform (24 bands per octave) were tested, with the DFT performing slightly better. DFT were extracted over from audio files downsampled to 10 kHz using a window size of 1024 frames and an overlap of 512 frames, resulting in a feature vector of size 512 for each window.

4.2.1.1 Normalization

In general, dictionary learning algorithms do not require data to be standardizing via mean subtraction and variance normalization (though it can help in some situations). However, badly scaled features can degrade the performance of the final classifier – especially Support Vector Machines. Experiments found that proper normalization and pre-processing had a large effect on the final performance of the system. Looking at a typical magnitude spectrogram, we find that most of the energy of the signal is contained in the lower frequencies with the result that the relative contribution of high and low frequency content may be uneven during the learning. Additionally, it may be helpful to accentuate higher frequency content which contains timbral information which could be important for recognizing genre since instrumentation is often strongly discriminative.

As an initial step, each column of the input spectrogram was normalized to have unit ℓ_2 norm. In order to obtain a more balanced spectrogram, Local Contrast Normalization (LCN) was performed. LCN is a commonly used process in the field of computer vision, where it is used to balance different lighting conditions. To apply LCN, each point in the spectrogram is standardized by subtracting its local mean and dividing by its local standard deviation. This "local region" is defined by the parameters ω_1 and ω_2 which set the window sizes for the mean and standard deviation respectively. For a given window f its local normalization is given by

$$\frac{f(x,y) - m_f(x,y)}{\sigma_f(x,y)} \tag{20}$$

where $m_f(x,y)$ is the local mean of f and $\sigma_f(x,y)$ is the local standard deviation. The effect of LCN on an audio spectrograms is to attenuate the higher-magnitude low frequencies while increasing the relative magnitude of upper harmonic information, leading to a more balanced spectrogram. An example of a spectrogram before and after this processes is given in Figure 4.

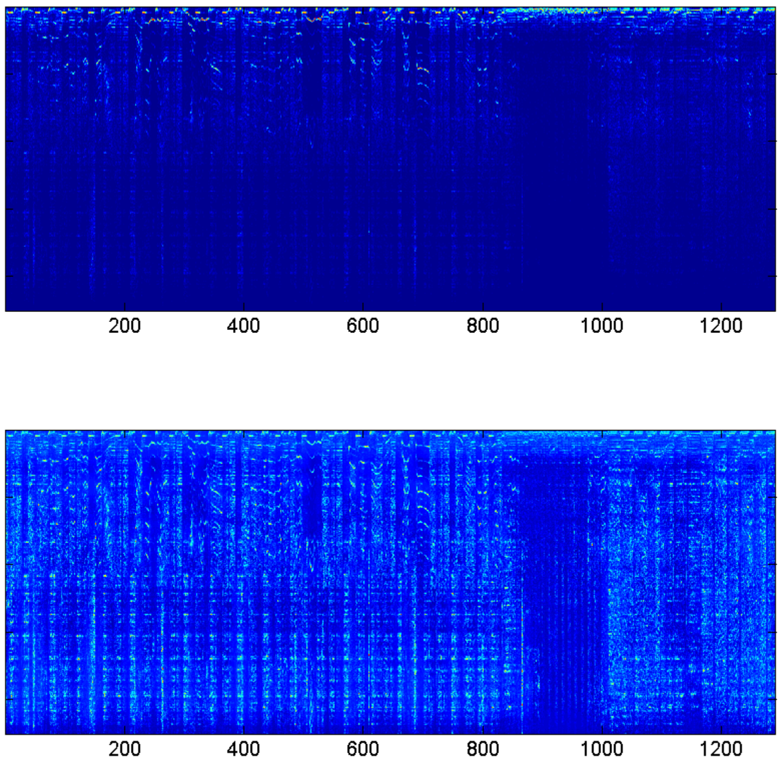

Figure 4: Spectrogram before (top) and after (bottom) local contrast normalization.

4.2.1.2 Feature Pooling

Tzanetakis et al. introduced the concept of a *texture window* for MGR [51]. As was shown in Chapter 2, many systems start by extracting features over short time windows of around 50–100ms. Since this is not a perceptually meaningful amount of time, a common strategy is to aggregate features over a longer time period. The importance of pooling was again highlighted by Bergstra et al. [3] who found improved MGR performance by using features extracted from segments of audio instead of the full track. For MGR two pooling strategies were tested: max pooling and mean pooling. After the encoding stage, each input song is represented by a matrix where each column corresponds to approximately 50ms of audio.

Given a window or pooling size (typically anywhere from 1 to 5 seconds), max pooling is performed by taking the maximum value in each row, reducing the 512-by-n matrix into a 512-by-1 column vector. Mean pooling is similarly achieved by taking the mean over each row. Essentially, max pooling selects the highest activation of each dictionary item in that window and summarizes it using a histogram-like representation. Mean pooling can be seen as a low pass filter. Max pooling was found to be superior to mean pooling in this context.

4.2.1.3 Classification

First, the full system pipeline is briefly outlined: spectrum are extracted from input audio files using a window length of 1024 frames with an overlap of 512 frames. The extracted spectrum are then normalized using Local Contrast Normalization. Dictionary learning using sparse coding is performed on normalized magnitude spectra and the full dataset is encoded in the learned dictionary. Encoded data is pooled using max-pooling over 1-second. Pooled codes from the training set were used to train a Support Vector Machine, which was used to classify the testing set. This process is outlined in Figure 11.

After pooling, each track is represented by a 512-by-m array with m depending on the size of the pooling window. The SVM is trained on such windows taken from the training set with labels given by their parent song. LibLinear [12] was used due to its increased speed when compared with LibSVM with linear kernel.

To predict the genre of a new song we can extract the magnitude spectra, normalize, encode in the learned dictionary and perform max pooling. We then use the trained SVM model to predict each pooled frame and use majority vote over all frames as the prediction for that song. The mean prediction accuracy was calculated for each fold and the final accuracy is the average across all folds.

4.2.1.4 Results: Unsupervised Coding

Table 2 summarizes the 10–fold cross validation results using sparse coding (SC) and sparse coding on normalized magnitude spectrum (N-SC) using different pooling window lengths. The results show that proper normalization is extremely important for good accuracy. These results verify that the normalization, dictionary learning and encoding implementations are

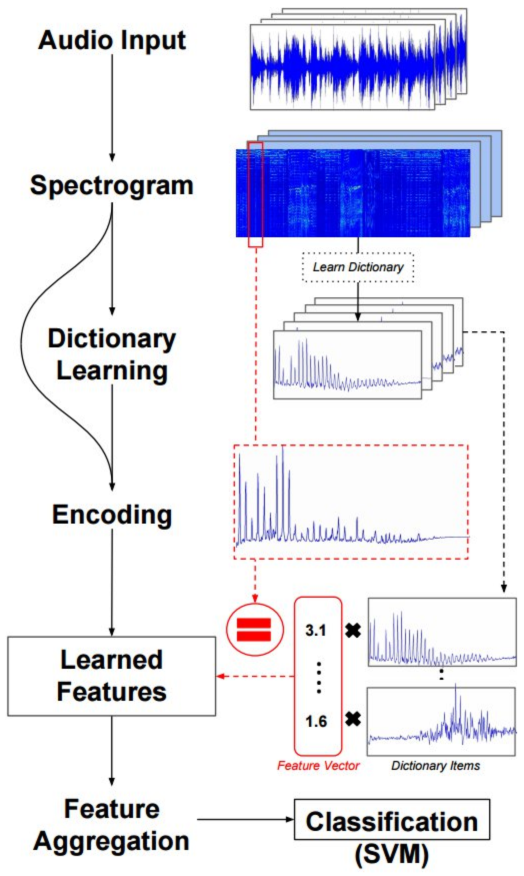

Figure 5: System flow-chart for unsupervised sparse coding.

Table 2: 10-fold CV results using max pooling at various window lengths.

	0.5 sec.	**1 sec.**	**2 sec.**	**5 sec.**
SC	0.7337	0.7448	0.7468	0.7362
N-SC	0.7837	**0.8128**	0.7888	0.7588

performing correctly. For the next section, the best performing parameters found in this section were used:

- $\lambda = 0.8$
- SVM hyperparameter C = 60
- LCN $\omega_1, \omega_2 = 120, 20$

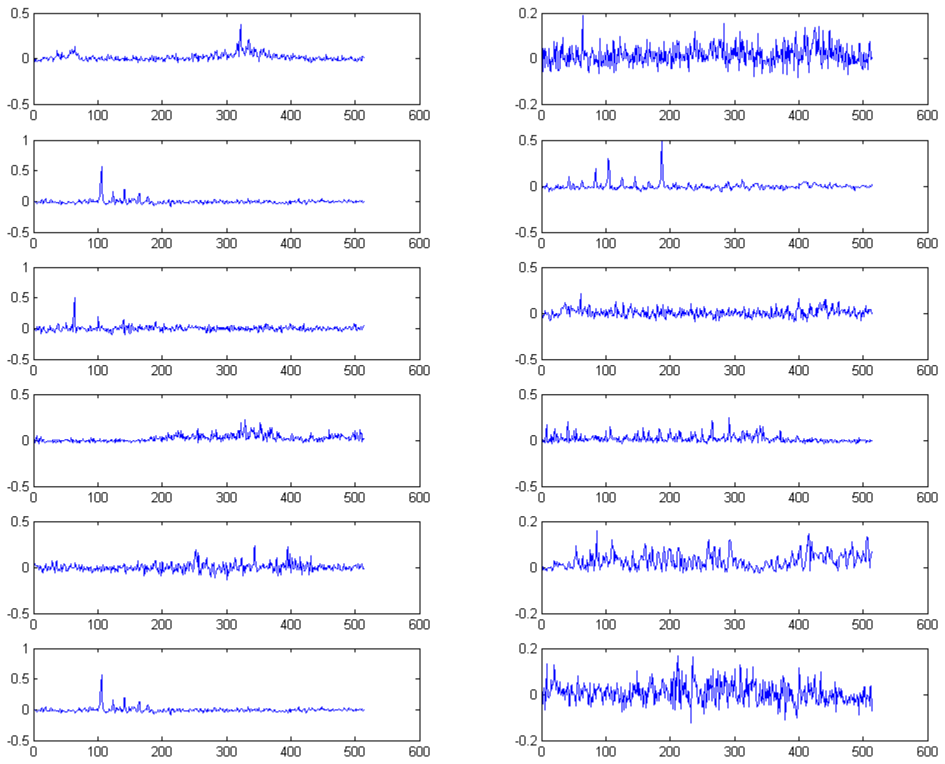

Figure 6: Examples of dictionary elements learned from music using sparse coding.

4.2.2 Discriminative Dictionary Learning

Supervised sparse coding was performed using a variation of Label Consistent K-SVD [24]. This allows us to make explicit use of the label information of the training set during dictionary learning. Given an input training set $X \in \mathbb{R}^{512 \times n}$ consisting of n 512-point spectra, we assign each column of X the same label as its parent song. The standard sparse coding objective is given by:

$$\min_{D,\alpha} \|X - D\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{21}$$
$$\text{such that } \|D^{(i)}\|^2 \leq 1 \; \forall i$$

To incorporate the labels into the dictionary learning process, we additionally assign a genre-label to each *dictionary item*. Given a new spectrum with label g, we can imagine its ideal sparse code: it would be a vector consisting mostly of 0, with 1 in the rows corresponding to dictionary items with the same class label. Define an "ideal sparse code" matrix Q whose columns consist of the ideal code for each column in X, so Q will be a block-diagonal binary matrix. Then the discriminative coding objective function is then given by

$$\min_{D,A,\alpha} \|X - D\alpha\|_2^2 + \gamma\|Q - A\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{22}$$
$$\text{such that } \|D^{(i)}\|^2 \leq 1 \ \forall i$$

Here A is a linear transformation, whose purpose is to make the sparse codes α discriminative. Note that A is not used in the actually classification process i.e. we do not calculate $A\alpha$ during the future classification step. It's purpose is to make the sparse codes "linearly discriminative" which helps classification. Similar formulations of a discriminative objective function have been used in image classification, usually by incorporating classifier learning into the training stage [32, 28, 31]. Strategies for solving the optimization problem can often suffer from local minima when the coding and classifier optimizations are distinct. An advantage of the approach of Jiang et al. [24] is that it learns the dictionary and classifier jointly and hence avoids this problem by concatenating $\{X, Q\}$ and $\{D, A\}$. It is also possible to add an additional classification error term to the objective which allows us to learn a linear predictive classifier at the same time. In practice however it was found that this learned classifier was outperformed by Linear SVM so it was omitted.

The key idea in the discriminative formulation of Jiang et al. is the concatenation step, which avoids the problem of local minima. The 2-norm of a matrix B can be written as $\|B\|_2^2 = \text{trace}(B^*B)$ where B^* is the conjugate transpose of B (and $B^* = B^T$ when B is Real). Therefore we can rewrite equation 22 as:

$$\begin{aligned} \{D, A, \alpha\} &= \text{argmin } \|X - D\alpha\|_2^2 + \gamma\|Q - A\alpha\|_2^2 + \lambda\|\alpha\|_1 \\ &= \text{argmin } \text{tr}((X - D\alpha)^T(X - D\alpha)) + \text{tr}((Q - A\alpha)^T\sqrt{\gamma}(Q - A\alpha)) + \lambda\|\alpha\|_1 \end{aligned} \tag{23}$$

Since $\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$ we have

$$\begin{aligned}
\{D, A, \alpha\} &= \text{argmin } \mathrm{tr}((X - D\alpha)^T(X - D\alpha) + (Q - A\alpha)^T\sqrt{\gamma}(Q - A\alpha)) + \lambda\|\alpha\|_1 \\
&= \text{argmin } \mathrm{tr}(X^TX + X^TD\alpha + \alpha^TD^TX + \alpha^TD^TD\alpha + \\
&\quad Q^T\sqrt{\gamma}Q + Q^T\sqrt{\gamma}A\alpha + \alpha^TA^T\sqrt{\gamma}Q + \alpha^TA^T\sqrt{\gamma}A\alpha) + \lambda\|\alpha\|_1 \qquad (24) \\
&= \text{argmin } \mathrm{tr}(X^TX + Q^T\sqrt{\gamma}Q) + \mathrm{tr}(X^TD\alpha + Q^T\sqrt{\gamma}A\alpha) + \mathrm{tr}(\alpha^TD^TX + \alpha^TQ^T\sqrt{\gamma}A) + \\
&\quad \mathrm{tr}(\alpha^TD^TD\alpha + \alpha^TQ^T\sqrt{\gamma}Q\alpha) + \lambda\|\alpha\|_1 \qquad (25) \\
&= \text{argmin } \mathrm{tr}((X')^TX') + \mathrm{tr}((X')^TD'\alpha) + \mathrm{tr}(\alpha^T(D')^TX') + \mathrm{tr}(\alpha^T(D')^TD'\alpha) + \lambda\|\alpha\|_1 \qquad (26) \\
&= \text{argmin } \ \|X' - D'\alpha\|_2^2 + \lambda\|\alpha\|_1 \qquad (27)
\end{aligned}$$

Where

$$D' = \begin{pmatrix} D \\ \sqrt{\gamma}A \end{pmatrix} \qquad\qquad X' = \begin{pmatrix} X \\ \sqrt{\gamma}Q \end{pmatrix}$$

This expression is exactly the problem solved by sparse coding, so this problem can be efficiently solved by suitable concatenations. To initialize D and assign a label to each dictionary item, sub-dictionaries are learned on each class using standard sparse coding and concatenated into one larger dictionary. Each dictionary item will then be associated with the genre on which it was learned. Jiang et al. used the K-SVD algorithm to learn the final dictionary based on this formulation, while here we use ODL.

The linear transformation A is initialized using Ridge Regression:

$$A_{init} = \arg\min_A \ \|Q - A\alpha\|_2^2 + \beta\|A\|_2^2 \qquad (28)$$

which has solution given by

$$A_{init} = (\alpha\alpha^T + \beta I)^{-1}\alpha Q^T \qquad (29)$$

where α are the sparse codes learned on the initialized dictionary.

Recall that the dictionary learned in this way is given by

$$D' = \begin{pmatrix} D \\ \sqrt{\gamma}A \end{pmatrix} \qquad (30)$$

During learning the columns of the dictionary are constrained to have unit 2-norm (in practice this constraint is always saturated). Hence, the dictionary D' learned during discriminative coding has columns of unit length. To recover the desired dictionary we first need to scale each entry in D' so that the desired D_{new} is column normalized. The final dictionary D_{new} is obained by

$$D_{new} = \Big(\frac{d^1}{\|d^1\|^2} \; \frac{d^2}{\|d^2\|^2} \cdots \frac{d^k}{\|d^k\|^2} \Big) \tag{31}$$

where d^n is the n^{th} column of the submatrix D.

To evaluate the discriminative coding approach, 5–fold cross validation was used. Since the ground-truth labels are used during the learning, the dictionary learning step was performed on the testing set only. Parameters such as the sparsity constraint λ, pooling time, normalization strategy and SVM parameters were set to the same values used in section 4.2.

4.2.2.1 Results: Discriminative Sparse Coding

Table 3: 5-fold cross-validaiton results or MGR on GTZAN.

SC	0.7970
D-SC	0.7945

Table 3 summarizes the results using sparse coding (SC) and discriminative sparse coding (D-SC). These results show that adding the discriminative penalty fails to increase the final accuracy. Looking at the accuracy per-fold, the discriminative coding scheme shows more variance than unsupervised coding; for some folds significantly outperforming it while for others badly under-performing. In order to verify the implementation, the results of Jiang et al. were successfully reproduced using the extended YaleB image dataset [15]. To examine the behavior of the learned discriminative dictionary we can plot the activation profiles for each dictionary item. After encoding the testing set, we sum the activation matrix α along each row to find the total activation of each dictionary item (recall that for a given column of α each row represents the activation of one dictionary element for that input spectrum). We perform this summation for each genre in the training set individually to find the activation of each item *in that specific genre.* During the training phase each dictionary element was assigned a class label, so we would therefore expect that each item would show more activation it its corresponding genre than in others. Figure 7 shows the activations in a discriminative dictionary for each genre in the test set. In the ideal case, these figures should show peaked activations in one class with low activations in the others. While this seems to be the case to some degree for certain genres, other dictionary items seem to be highly active for many training items – even those with different genre labels. Figure 8 shows the ratio of within-class activation to total activation for each item in a learned dictionary of size 1000 – again we find that some dictionary items are more discriminative than others.

For comparison, Figure 9 shows the total activations using a discriminative dictionary for 10 classes from the extended YaleB dataset. As we can see, the dictionary is much more discriminative in this case with sharp peaks specific to each class. Increasing the size of the discriminative term γ to increase the discriminative power of the dictionary badly degrades the performance. The best performance was found using $\gamma = 0.001$ which is relatively small compared to the sparsity parameter $\lambda = 0.8$ which was chosen using unsupervised coding (outlined in section 4.2). These results indicate that the music data is much less separable than the images in the YaleB dataset – at the scale of magnitude spectra all the genres 'look alike'.

4.2.3 Two Layer Coding

Applying discriminative coding to the magnitude spectrum fails to meaningfully improve the performance according to these results. It is possible that enforcing the label information on this scale is not appropriate since we can't expect any given 50ms spectrum to contain meaningful information about the genre. In the case of unsupervised coding, it was necessary to pool the learned codes over a perceptually meaningful amount of time (the so-called texture window discussed in Section 4.1).

With this in mind, a two-layer scheme was investigated. In the first layer, sparse coding

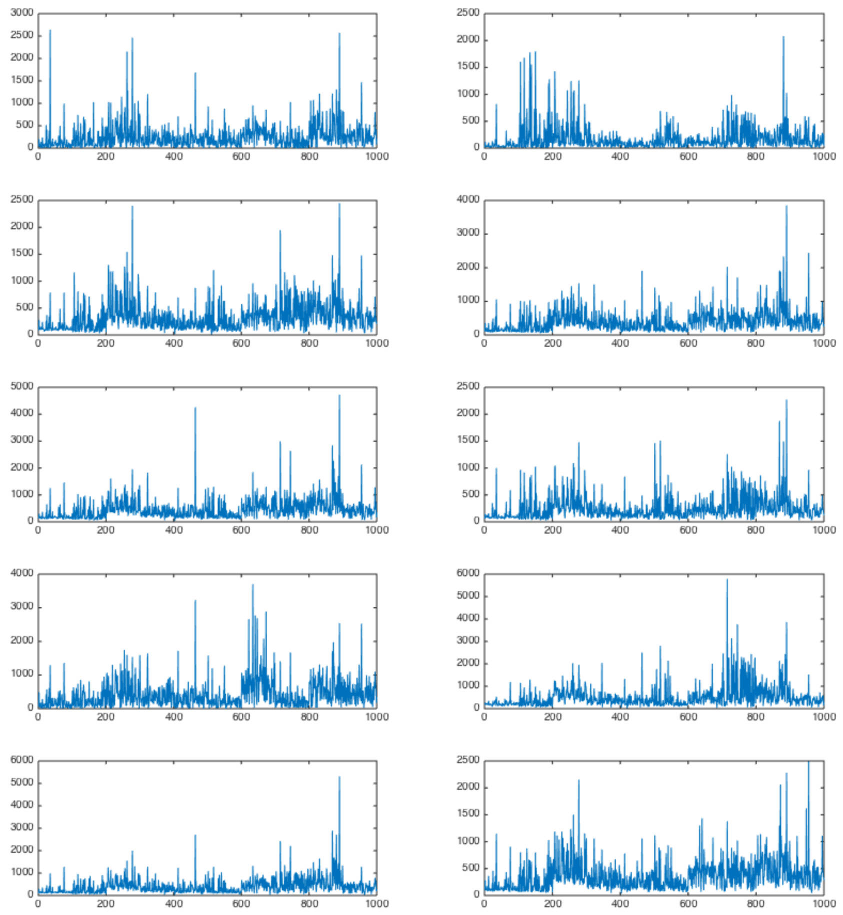

Figure 7: Activations for each genre in the testing set using a dictionary learned with discriminative coding.

was used to learn a suitable dictionary (see Section 4.2.1) and items from the test and training sets were pooled as discussed in Section 4.2.1.2. Then an additional dictionary was learned using discriminative sparse coding, using the pooled sparse codes as input. In this case, the label information is being enforced at a level of 1–second (the length of the window over which the sparse codes were pooled). This approach is similar in flavor to the idea behind Spatial Pyramid Matching used in computer vision, where short time features are pooled and taken as input for an additional feature learning step (for example, the work of Yang et al. [56]). Learning and encoding using this approach is very slow, since

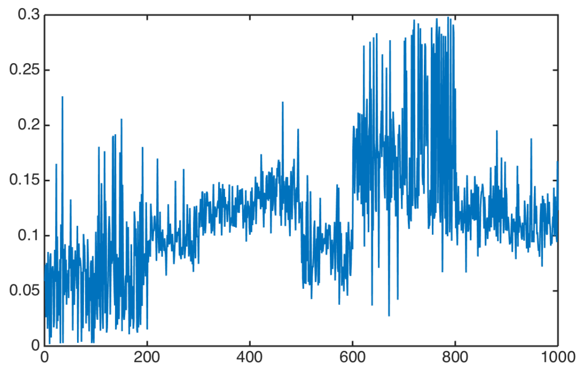

Figure 8: Ratio of activations within the correct genre versus all activations for each dictionary item (testing set).

we need to perform both of these steps on each layer. To deal with this, instead of using full cross-validation the data set was split into a training set of 500 songs, a testing set of 300 songs and a validation set of 200 songs. This meant that the process only needed to be carried out once (and not multiple times as would be the case for cross-validation). The settings used for the first layer were the same as found in Section 4.2. The discriminative term γ and second-layer sparsity parameter λ was tuned using 5-fold cross-validation on the validation set.

4.2.3.1 Results: Two Layer Coding

Table 4 shows the classification accuracy of sparse coding (SC) versus two-layer sparse coding (TL-SC). The results show that adding the second layer actually degrades the performance. The drop in accuracy may be due to the additional reconstruction error introduced in second layer coding. The accuracy using sparse coding is substantially lower than those reported in Section 4.2.2 which can be explained by the fact that in this case the training set is smaller and only one training set was used.

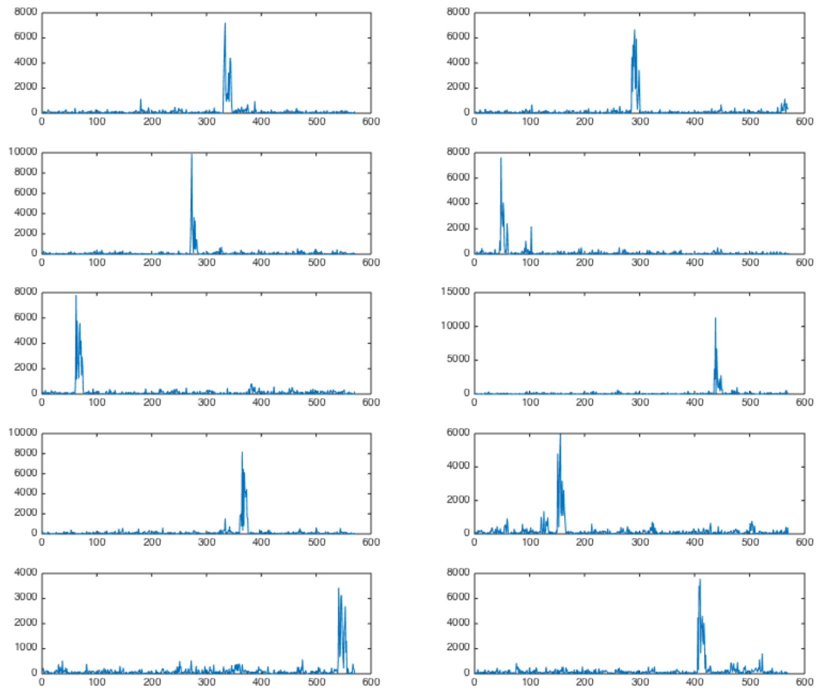

Figure 9: Activations for different 10 different classes using the extended YaleB image dataset.

Table 4: Accuracy on test set using sparse coding and sparse coding with a additional discriminative layer.

SC	0.7090
TL-SC	0.6689

4.3 Discussion

For the sake of completeness, Table 5 summarizes MGR accuracies of several other systems in the literature using the same dataset. The system outlined here compares comparably to similar systems. Note that the main aim of this section was to compare the performance of *supervised* feature learning to *unsupervised* feature learning and as such the implementation was kept relatively simple. Systems which outperform the ones described here tend to have additional steps to improve the raw accuracy — for example non-linear instead of linear

Table 5: MGR accuracies using GTZAN dataset.

Reference	**Accuracy**
[37]	0.92
[59]	0.857
[17]	0.843
[18]	0.834
[3]	0.83
SC	**0.8128**
D-SC	**0.7945**
[18]	0.794
[4]	0.77
[27]	0.71
[52]	0.61

SVM [60] or processing performed on separate octaves [18]. The results obtained using sparse coding with a dictionary learned on the full set (SC) and discriminative dictionary with cross-validation (D-SC) are highlighted in bold.

Overall the results indicate that the discriminative formulation does not improve the performance of sparse coding for MGR. Each processing step was verified separately — satisfactory results were achieved first using sparse coding with the dictionary learned on the full data set. The implementation of discriminative coding was verified by successfully reproducing the results on an image recognition task by Jiang et al. [24]. A range of parameters for the discriminative term γ was tested, from 0.0001 to 100.

Using the idea that the time frame over which the input magnitude spectrogram features were taken may be too short to meaningful enforce label information, the two layer coding scheme learns an additional dictionary using the pooled first-layer sparse codes. It should also be noted that two other approaches were also tested: in the first the magnitude spectrograms were aggregated over a time interval of 1-second (see Section 5.4 for details of a similar approach), with dictionary learning being performed on these pooled magnitude spectra. The other approach used the bio-inspired features made available by Bob Sturm in his work ([49]) which sought to replicate the findings of Panagakis ([37]). In both cases the performance was in the range of 60% which is consistent with Sturm's findings.

4.4 Conclusion

In this chapter, the performance of sparse coding for MGR was investigated. Initial results were obtained using sparse coding with dictionary learned on the full data set and 10-fold cross validation at the classification stage. A supervised sparse coding approach was outlined using the ideas of Label Consistent K-SVD and Online Dictionary Learning to incorporate the ground truth data during the dictionary learning step. To evaluate both methods, cross-validation was performed using the GTZAN dataset with dictionary learning on the training splits only, with final classification being preformed using linear SVM. It was found that the supervised formulation failed to outperform the standard sparse coding approach based on cross-validation accuracy.

CHAPTER V

SPARSE CODING FOR MUSIC EMOTION RECOGNITION

In this chapter we investigate the use of sparse coding and supervised sparse coding for the task of estimating the emotion or mood of music. Music often contains a high amount of emotional content, however unlike other MIR problems such as artist or chord identification the perceived mood of a piece is necessarily subjective. As Kim et al. note: "there may be considerable disagreement regarding the perception and interpretation of the emotions of a song or ambiguity with the piece itself" [25]. While this problem can be examined from a wide variety of approaches, here we are concerned with purely acoustic features derived from the signal, as opposed to proposed holistic approaches incorporating metadata such as lyrics or even social context.

5.1 Related Work: Quantifying and Predicting the Emotional Content of Music

As a first step toward evaluation, a suitable representation of the emotional content of a given song needs to be found. Heve conducted a study concerned with the "affective value" and "expressiveness" of music [19]. Interestingly, the author notes that the notion that music has consistent associated emotional content is taken as an assumption, noting that this point is somewhat controversial. A list of 66 adjectives, arranged into 8 groups were used by listeners to describe the music they heard. They found several correlations between the music and their adjectives: major modes were described as "happy", "merry" and "playful". Dissonant harmonies were found to be "exciting", "agitating" and "vigorous".

The Music Information Retrieval Evaluation eXchange (MIREX) ran a music emotion task in 2007 and 2010. Here the task was to label a song using one of 5 "mood clusters". Each cluster was associated with a set of adjectives derived from mood-labels of popular songs.

Many approaches for quantifying music emotion use discrete assignment of emotional

Table 6: Clusters and their associated mood adjectives used in the MIREX 2007 music emotion task.

Cluster 1	passionate, rousing, confident, boisterous, rowdy
Cluster 2	rollicking, cheerful, fun, sweet, amiable
Cluster 3	literate, poignant, wistful, bittersweet, autumnul, brooding
Cluster 4	humorous, silly, campy, quirky, whimsical, witty, wry
Cluster 5	aggressive, fiery, tense, intense, volatile, visceral

labels to songs. An alternative approach is to use a continuous measure, as introduced by Russell and Thayer in the *Valence-Arousal* (V-A) model [40]. Here the emotional content of music is represented on a 2-dimensional plane, with one axis corresponding to arousal (the energy or intensity of the music) and the other corresponding to valence (which is designed to capture the 'attractiveness' or 'averseness' of an emotion).

Yang et al. used the Valence-Arousal formulation of Music Emotion Recognition (MER) [57]. They extracted many standard acoustical features such as *Spectral Centroid*, *Loudness*, *Spectral Dissonance* and others producing features of dimension 114. Noting that there is often a dependency between the two dimensions in the Valence-Arousal model, they attempted to by first reducing the data correlation using Principal Component Analysis (PCA). Next a feature selection procedure [39] was used to find the most useful features. For prediction they used Support Vector Regression (SVR) and Multiple Linear Regression (MLR), finding best performance was achieved using PCA with feature selection and SVR.

In contrast to the continuous Valence-Arousal model, many works have used a discrete space for MER. In their approach Yang et al. divided the V-A space into four quadrants with each sample being given a fuzzy labeling corresponding to its membership in each class [58]. Trohidis also used a multi-label system, where pieces of music could belong to more than one class [50]. The 6 classes of emotions used were derived from the Tellegen-Watson-Clark model [55]. Prediction again consisted of a host of low level audio features (including Beats Per Minute and timbral features such as Spectral Centroid, Spectral Flux and Spectral Kurtosis) combined with Support Vector Regression.

This approach of using a large number of standard features together with a feature reduction step to train a regression model is common in MER. In comparison the learning of features which are adapted to the problem of MER is a relatively unexplored topic. One such system was developed by Schmidt and Kim [42]. Their work was concerned with predicting Valence and Arousal labels using features learned via a Deep Belief Network (DBN), comparing it to standard acoustic features such as Spectral Contrast, Pitch Chroma and Mel Frequency Cepstrum Coefficients. Raw spectrograms were used as input features to a 3-layer DBN. Similar to a neural network, DBNs allow for much improved training procedure consisting of an unsupervised pre-training stage and a supervised 'fine tuning' stage. The outputs of each layer on the DBN were used as features, with the best performance being achieved using the hidden layer. They employed 5-fold cross validation a dataset consisting of 240 songs of length 15 seconds and reported the average mean distance as well as the average KL Divergence between the predicted and ground truth values. They also investigated the learned features by observing the activations, noting that the first layer learns a sparse basis which is heavily dependent on certain frequency bands.

Pachet and Zils introduce an algorithm to learn high level features for music analysis using Genetic Algorithms with application to predicting the perceived energy of songs [36]. In their systems, a high level representation is achieved by combining several atomic processing elements such as taking FFT, mathematical operations like mean/max, temporal operators like autocorrelation and more. The fitness of a given combination is evaluated and new functions are produced through mutation and crossover. They found that descriptors learned in this way out performed popular low level features. While their approach is based on learning features, they rely on combining low level feature descriptors and using evolutionary algorithms to produce the final high level feature. In contrast, sparse coding is a low-level feature learner which uses traditional convex optimization methods based on the reconstruction and sparsity errors.

5.2 Dataset

For this work, we deal with the Valence-Arousal model of music emotion. The V-A model was chosen because as a regression problem it differs from the Music Genre Recognition task

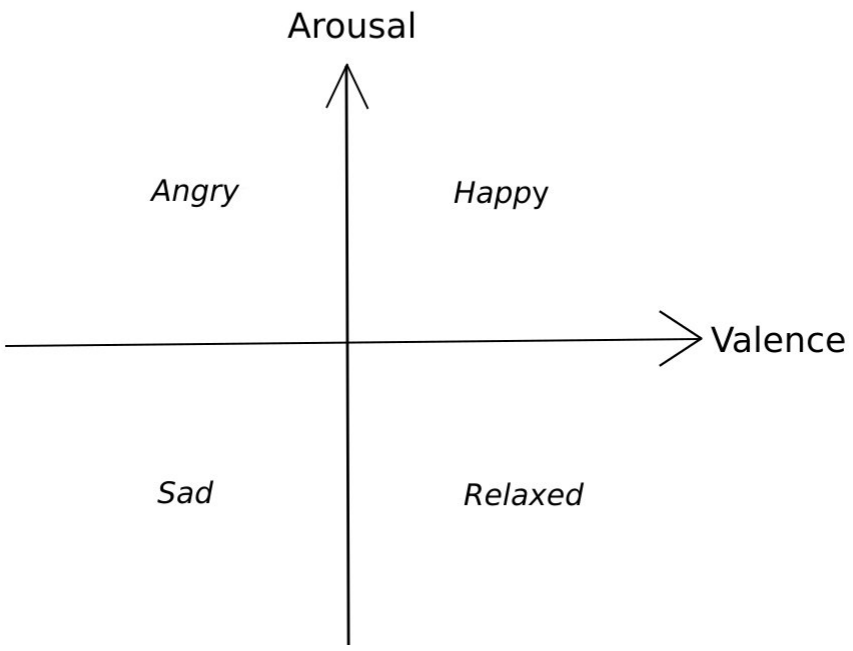

Figure 10: The Valence-Arousal plane. The emotional content of music is quantified by its position in the 2 dimensions.

which is a classification problem. Under the V-A model, the emotional content of music is given by its position in the 2-dimensional Valence-Arousal space and we are concerned with predicted these values. As ground-truth a data set consisting of Valence-Arousal values for 1000 songs [44] was used. Due to the subjective nature of music emotion, the creation of a good dataset requires human annotations. To achieve this the authors used Amazon Mechanical Turk and the idea of a "musical game" to collect continuous V-A values for 1000 creative commons songs found in the Free Music Archive. While listening to each song, subjects would move an on-screen ball whose position corresponded to either the arousal or the valence of the currently playing track, resulting in continuous annotations for each parameter.

Duplicates were later found in the collection, so a reduced data set consisting of 734 songs was also released. Since the collected V-A labels had sample rates which varied according to the user's browser and computer, the annotations were resampled at a rate of 2Hz. Each track consists of 45 seconds of music, with the annotations corresponding to the first 15 seconds of music excluded due to "instability of the annotations at the start of the clips". Therefore the final data set used for these experiments consisted of 734 excerpts of length 30 seconds with arousal and valence labels for every 500ms.

5.3 Baseline Features

Common methods for music emotion involve extracting a large amount of commonly used audio features, performing dimension reduction and a regression step to predict the final labels. As baseline comparison we use Mel-Frequency Cepstral Coefficients (MFCCs) and pitch chroma features, which have been shown to work well for the task of music emotion [43, 41]. Pitch chroma are a feature which measures the relative contribution of each pitch class (note) in a given signal, by summing the magnitudes of the DFT centered around bins corresponding to each note (and their octaves). These features were extracted for each annotated 500ms section of audio (no significant difference was found by taking features over shorter time scales and calculating statistical descriptors). The purpose of extracting these baseline features is to compare the performance of the feature learning approach to commonly-used MIR features.

5.4 Unsupervised Dictionary Learning

Unsupervised sparse coding was used to learn features from the audio data. Each input track was first downsampled to 11050kHz. Then the DFT was taken using a window size corresponding to 50ms with 25ms overlap. Each spectrum was then normalized using Local Contrast Normalization. One problem was that the labels were quite coarse with respect to the DFT window size. In the case of Music Genre Recognition this was irrelevant since each song had only one genre label, so each constituent spectrum could be assigned with that label and there was some freedom with regards to pooling. In this case however we are attempting to estimate the *dynamic* V-A annotation which changes at a rate of 2Hz. This means that if we were to proceed as before we would need to be extremely careful with how the pooling is performed after encoded each 50ms window – for example the previous strategy of 1-second windows with 50% overlap would not work since it will “spill over” into the next annotated block.

Therefore a new approach was used: rather than encode the short time spectra and pool the results using overlapping blocks, the input spectrum was pooled at different levels and concatenated. First the spectrum is max-pooled in each annotated 500ms block. Next we pool over each block *twice*: first over the initial 250ms and then over the remaining 250ms and concatenate the results into a single column vector. Finally we perform a similar process using successive blocks of 125ms.With an initial DFT size of 512, this process results in $512 \times (1 + 2 + 4) = 3584$ dimensional column vector for each annotated 500ms section.

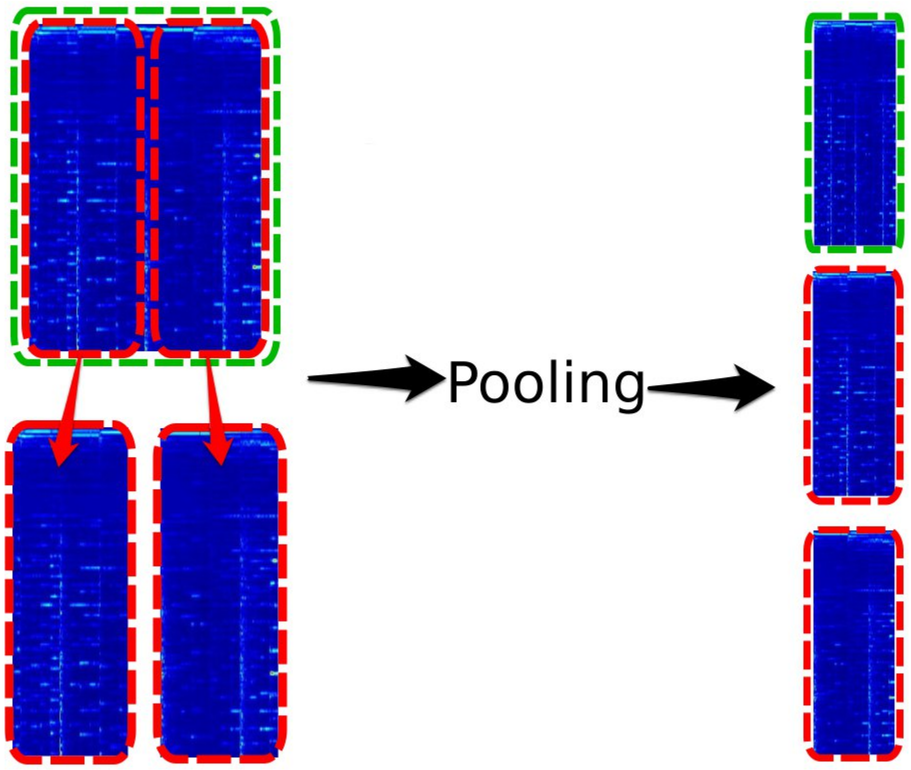

Figure 11: Feature aggregation for the music emotion task. Extracted spectrograms are divided and pooled, with the final feature vector formed by concatenating each segment.

5.5 Combined Regression and Dictionary Learning

Predicting the continuous V-A labels amounts to a regression problem: given the input features, find the relationship between those features and the observed labels. A usual approach to such a problem is to first extract features from the data and then use the training set to learn this relationship (using linear regression for example). So far 3 features have been discussed: MFCCs, pitch chroma and features learned via sparse coding. Here we investigate an approach to learn the linear regression parameters jointly during the dictionary learning step. The discriminative approach discussed in Chapter 3 was framed in terms of a classification problem: we attempted to learn a dictionary with more class-discriminative power by adding an additional discriminative error term in the objective function. Noting however that this was essentially a linear regression term (recall that the role of the linear transformation A was essentially to regress the sparse codes onto the "discriminative" vectors contained in Q) we apply the same idea here to jointly learn a dictionary and linear regression model. In fact this setting may be more natural for this approach since it is a pure regression problem (and not a classification problem being solved by a linear predictive map).

Let X be the matrix of input training features, D the dictionary matrix and α the matrix of sparse codes. Let L be the matrix of observed labels (in this case a 2-by-N matrix consisting of an arousal and valence labels for each of the N 500ms blocks). W is a linear transformation and we have the following objective function:

$$\min_{D,W,\alpha} \|X - D\alpha\|_2^2 + \gamma\|L - W\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{32}$$
$$\text{such that } \|D^{(i)}\|^2 \leq 1 \ \forall i$$

Proceeding as before this can be rewritten as

$$\|X' - D'\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{33}$$

Where

$$D' = \begin{pmatrix} D \\ \sqrt{\gamma}W \end{pmatrix} \qquad X' = \begin{pmatrix} X \\ \sqrt{\gamma}L \end{pmatrix}$$

D was initialized using an initial round unsupervised sparse coding on the training set. After encoding the training data using D_{init} the linear regressor W was initialized using ridge regression as in Chapter 3. The final D and W were learned using ODL to solve the optimization problem.

5.6 Evaluation

In order to evaluate the performance of each feature, 5-fold cross-validation was used. For MFCC and pitch chroma, given the annotated training feature set X with annotations Y, ridge regression was used to learn a predictor W':

$$W' = \text{argmin}_W \ \|Y - WX\|_2^2 + \beta\|W\|_2^2 \tag{34}$$

which has solution

$$W' = (XX^T + \beta I)^{-1}XY^T \tag{35}$$

For unsupervised sparse coding, a dictionary was learned on the training data and used for encoding and ridge regression was then used to learn the regression model. Additionally a supervised dictionary and linear regression model were learned jointly using the training data as outlined in Section 5.5. Parameters such as sparsity term λ and learning time were kept the same for both methods. Items in the training set were then encoded and their V-A labels predicted using the appropriate linear regression model.

Each feature was tested on the task of predicting the V-A annotation of each labeled 500ms block. To evaluate their performance Root Mean Square Error (RMSE) for both the arousal and valence labels separately as well as the mean distance from the predicted V-A vector to the actual was calculated. For a set of predictions x_i and observations y_i the RMSE is given by

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{36}$$

and for a set of V-A prediction vectors X_i and observed V-A labels Y_i the mean distance is given by

$$\frac{1}{n}\sum_{i=1}^{n}\|Y_i - X_i\|_2 \tag{37}$$

5.7 Results and Discussion

Table 7: Results for Music Emotion Recognition.

	Valence (RMSE)	**Arousal (RMSE)**	**Mean Dist.**
SC	0.2532	0.3087	0.3553
S-SC	**0.2032**	**0.2500**	**0.2713**
MFCC	0.2364	0.2502	0.3048
PC	0.2424	0.2857	0.3347

Table 7 shows the results using sparse coding (SC), supervised sparse coding (S-SC), Mel Frequency Cepstrum Coefficients (MFCC) and Pitch Chroma (PC). For Arousal and Valence the RMSE values are reported. Distance is given by the Euclidean distance in VA-space. As can be seen, supervised sparse coding significatnly outperforms both normal sparse coding and MFCC features. By combing the classifier and dictionary learning into one optimization problem, supervised coding learns a dictionary and optimal classifier jointly.

5.8 Conclusion

Here the use of sparse coding for Music Emotion Recognition (MER) was discussed. Many systems for MER consist of a feature extraction stage combined with linear regression. Ideas from Label Consistent K-SVD were adapted for the purpose of learning audio features and a linear classifier *jointly*. The results indicate that the combined dictionary/regression framework significantly outperforms regular sparse coding for this task, as well as standard features such as MFCC and pitch chroma together with linear regression.

CHAPTER VI

CONCLUSION AND FUTURE WORK

The goal of this work was two-fold: first investigate the use of sparse coding for Music Information Retrieval (MIR). Second: compare the performance of supervised and unsupervised sparse coding. To this end a feature learning framework using sparse coding and a supervised sparse encoding approach based on the ideas of Label Consistent K-SVD (LC K-SVD) was implemented and used to learn features for tasks of (i) Music Genre Recognition and (ii) Music Emotion Recognition. The described approach aims to learn a single dictionary with discriminative power, distinguishing it from previous work which used only weak supervision where separate dictionaries are trained on each genre.

For the MGR task several approaches were tested: sparse coding with dictionary learning on the whole data set, sparse coding with dictionary learning on training sets (with cross-validation) and a discriminative, supervised formulation which uses the labels during dictionary learning (unlike previous related work which does not include the ground-truth labels explicitly). Unsupervised sparse coding was found to perform well for MGR. However the discriminative formulation failed to improve the results, which is consistent with some other related works ([60]). One finding was that the normalization and preprocessing strategies are of great importance in sparse coding and it is possible that using a different strategy might show improved performance for the discriminative formulation. It is also possible that a better parameter search might help; parameters such as the sparsity penalty and SVM slack variable were tuned using an unsupervised sparse coding stage and the best performing values were used for the discriminative coding. This was done in order to make computation time feasible – performing full cross-validation (with dictionary learning on each test split) is already an extremely slow process and having to deal with additional parameter tuning at this stage would make the computation time unmanageable. However it is possible that the values obtained for the unsupervised approach are not optimal for supervised sparse coding. For future work, different input features and pooling methods could be looked at. Given the short time period over which FFTs are taken, it may not make sense to enforce label consistency at this level. To try and account for this, an approach similar to those used in computer vision consisting of a two-coding scheme was also implemented, which failed to show improvement over standard sparse coding.

For MER, the main contribution was the implementation of a system which incorporates learning of a linear regression model jointly with the dictionary. Since the Valence and Arousal labels are continuous the input spectrograms were processed by aggregating them over each labeled section. To evaluate supervised versus unsupervised coding, cross-validation was used using Root Mean Square Error and Mean Distance in Valence-Arousal space. The results show that combing the dictionary and classifier learning in one process leads to significantly increased performance over standard sparse coding as well as commonly-used designed feature like MFCC and pitch chroma

Overall, feature learning is a promising direction for MIR. However, the promise of a *purely* data-driven pipeline for feature learning and classification failed to materialize. Normalization and pre-processing proved to have a significant effect on the performance of sparse coding – while the features themselves are learned automatically, the choice of input is still left up to the user. This can clearly be seen by comparing the complicated features used by Panagakis [37] – whose accuracy was found to be around 60% by Sturm [49] – against the final system described in Chapter 4 – which achieved over 80%. Additionally the temporal nature of music presents some unique complications not present in for example computer vision: mainly the problem of how to summarize a piece of music before the sparse coding algorithm can even begin. An interesting topic of future work would be to examine this in more detail by comparing features learned from pooled magnitude spectra against pooled features learned on short-time magnitude spectra.

Supervised sparse coding for the Music Emotion task showed more promise. Instead of working on short-time FFT, here pooled magnitude spectrograms were used as input to the sparse coding stage. To the authors knowledge, this is the first work which applies sparse coding and supervised sparse coding ot the problem of MER.

REFERENCES

[1] Abdallah, S. A. and Plumbley, M. D., "Unsupervised Analysis of Polyphonic Music by Sparse Coding," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 179–196, 2006.

[2] Aharon, M., Elad, M., and Bruckstein, A. M., "K-SVD and its non-negative variant for dictionary design," in *Optics & Photonics 2005*, pp. 591411–591411, International Society for Optics and Photonics, 2005.

[3] Bergstra, J., Casagrande, N., Erhan, D., Eck, D., and Kgl, B., "Aggregate features and AdaBoost for music classification," *Machine learning*, vol. 65, no. 2-3, pp. 473–484, 2006.

[4] Bergstra, J., Mandel, M. I., and Eck, D., "Scalable Genre and Tag Prediction with Spectral Covariance.," in *ISMIR*, pp. 507–512, 2010.

[5] Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P., "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pp. 591–596, University of Miami, 2011.

[6] Blumensath, T. and Davies, M., "Sparse and shift-invariant representations of music," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 1, pp. 50–57, 2006.

[7] Chen, S. S., Donoho, D. L., and Saunders, M. A., "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1998.

[8] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., and others, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[9] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., and others, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[10] Elad, M. and Aharon, M., "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[11] Ellis, D. P. and Thierry, B.-M., "Large-scale cover song recognition using the 2d Fourier transform magnitude," in *The 13th International Society for Music Information Retrieval Conference*, pp. 241–246, 2012.

[12] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J., "LIBLINEAR: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[13] Freund, Y., Schapire, R., and Abe, N., "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[14] Gao, S., Tsang, I. W., Chia, L.-T., and Zhao, P., "Local features are not lonelyLaplacian sparse coding for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3555–3561, IEEE, 2010.

[15] Georghiades, A. S., Belhumeur, P. N., and Kriegman, D., "From few to many: Illumination cone models for face recognition under variable lighting and pose," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 6, pp. 643–660, 2001.

[16] Grosse, R., Raina, R., Kwong, H., and Ng, A. Y., "Shift-invariance sparse coding for audio classification," *arXiv preprint arXiv:1206.5241*, 2012.

[17] Hamel, P. and Eck, D., "Learning Features from Music Audio with Deep Belief Networks.," in *The 11th International Society for Music Information Retrieval Conference*, pp. 339–344, Utrecht, The Netherlands, 2010.

[18] Henaff, M., Jarrett, K., Kavukcuoglu, K., and LeCun, Y., "Unsupervised learning of sparse features for scalable audio classification.," in *The 12th International Society for Music Information Retrieval Conference*, pp. 681–686, 2011.

[19] Hevner, K., "Experimental studies of the elements of expression in music," *The American Journal of Psychology*, pp. 246–268, 1936.

[20] Hoerl, A. E. and Kennard, R. W., "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[21] Humphrey, E. J., Bello, J. P., and LeCun, Y., "Feature learning and deep architectures: new directions for music informatics," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 461–481, 2013.

[22] Humphrey, E. J., Nieto, O., and Bello, J. P., "Data Driven and Discriminative Projections for Large-Scale Cover Song Identification.," in *ISMIR*, pp. 149–154, 2013.

[23] Hyvrinen, A., Hoyer, P., and Oja, E., "Image denoising by sparse code shrinkage," in *Intelligent Signal Processing*, Citeseer, 1999.

[24] Jiang, Z., Lin, Z., and Davis, L. S., "Learning a discriminative dictionary for sparse coding via label consistent K-SVD," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1697–1704, IEEE, 2011.

[25] Kim, Y. E., Schmidt, E. M., Migneco, R., Morton, B. G., Richardson, P., Scott, J., Speck, J. A., and Turnbull, D., "Music emotion recognition: A state of the art review," in *Proc. ISMIR*, pp. 255–266, Citeseer, 2010.

[26] Lee, H., Battle, A., Raina, R., and Ng, A. Y., "Efficient sparse coding algorithms," in *Advances in neural information processing systems*, pp. 801–808, 2006.

[27] Li, T. and Tzanetakis, G., "Factors in automatic musical genre classification of audio signals," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pp. 143–146, IEEE, 2003.

[28] Mairal, J., Bach, F., and Ponce, J., "Task-driven dictionary learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 4, pp. 791–804, 2012.

[29] Mairal, J., Bach, F., Ponce, J., and Sapiro, G., "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 689–696, ACM, 2009.

[30] Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A., "Discriminative learned dictionaries for local image analysis," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[31] Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A., "Discriminative learned dictionaries for local image analysis," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[32] Mairal, J., Ponce, J., Sapiro, G., Zisserman, A., and Bach, F. R., "Supervised dictionary learning," in *Advances in neural information processing systems*, pp. 1033–1040, 2009.

[33] Markov, K. and Matsui, T., "Music genre classification using self-taught learning via sparse coding," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 1929–1932, IEEE, 2012.

[34] Ness, S., Walters, T., and Lyon, R., "Auditory Sparse Coding," in *Music Data Mining* (Li, T., Ogihara, M., and Tzanetakis, G., eds.), pp. 77–92, CRC Press, 2011.

[35] Olshausen, B. A. and Field, D. J., "Sparse coding with an overcomplete basis set: A strategy employed by V1?," *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[36] Pachet, F. and Zils, A., "Evolving automatically high-level music descriptors from acoustic signals," in *Computer Music Modeling and Retrieval*, pp. 42–53, Springer, 2004.

[37] Panagakis, Y. and Kotropoulos, C., "Music genre classification via topology preserving non-negative tensor factorization and sparse representations," in *Acoustics speech and signal processing (ICASSP), 2010 IEEE international conference on*, pp. 249–252, IEEE, 2010.

[38] Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S., "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pp. 40–44, IEEE, 1993.

[39] Robnik-ikonja, M. and Kononenko, I., "Theoretical and empirical analysis of ReliefF and RReliefF," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[40] Russell, J. A., "A circumplex model of affect.," *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.

[41] Schmidt, E. M. and Kim, Y. E., "Prediction of Time-varying Musical Mood Distributions from Audio.," in *ISMIR*, pp. 465–470, 2010.

[42] Schmidt, E. M. and Kim, Y. E., "Learning emotion-based acoustic features with deep belief networks," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*, pp. 65–68, IEEE, 2011.

[43] Schmidt, E. M., Turnbull, D., and Kim, Y. E., "Feature selection for content-based, time-varying musical emotion regression," in *Proceedings of the international conference on Multimedia information retrieval*, pp. 267–274, ACM, 2010.

[44] Soleymani, M., Caro, M. N., Schmidt, E. M., Sha, C.-Y., and Yang, Y.-H., "1000 songs for emotional analysis of music," in *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*, pp. 1–6, ACM, 2013.

[45] Sturm, B. L., "An analysis of the GTZAN music genre dataset," in *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, pp. 7–12, ACM, 2012.

[46] Sturm, B. L., "Classification accuracy is not enough," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 371–406, 2013.

[47] Sturm, B. L., "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use," *arXiv preprint arXiv:1306.1461*, 2013.

[48] Sturm, B. L., "A survey of evaluation in music genre recognition," in *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation*, pp. 29–66, Springer, 2014.

[49] Sturm, B. L. and Noorzad, P., "On automatic music genre recognition by sparse representation classification using auditory temporal modulations," *Computer music modeling and retrieval*, pp. 379–394, 2012.

[50] Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. P., "Multi-Label Classification of Music into Emotions.," in *ISMIR*, vol. 8, pp. 325–330, 2008.

[51] Tzanetakis, G. and Cook, P., "Musical genre classification of audio signals," *Speech and Audio Processing, IEEE transactions on*, vol. 10, no. 5, pp. 293–302, 2002.

[52] Tzanetakis, G. and Cook, P., "Musical genre classification of audio signals," *Speech and Audio Processing, IEEE transactions on*, vol. 10, no. 5, pp. 293–302, 2002.

[53] Tzanetakis, G., Ermolinskyi, A., and Cook, P., "Pitch histograms in audio and symbolic music information retrieval," *Journal of New Music Research*, vol. 32, no. 2, pp. 143–152, 2003.

[54] Wlfing, J. and Riedmiller, M., "Unsupervised Learning of Local Features for Music Classification.," in *The 13th International Society for Music Information Retrieval Conference*, pp. 139–144, 2012.

[55] Yang, D. and Lee, W.-S., "Disambiguating Music Emotion Using Software Agents.," in *ISMIR*, vol. 4, pp. 218–223, 2004.

[56] Yang, J., Yu, K., Gong, Y., and Huang, T., "Linear spatial pyramid matching using sparse coding for image classification," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1794–1801, IEEE, 2009.

[57] Yang, Y.-H., Lin, Y.-C., Su, Y.-F., and Chen, H. H., "A regression approach to music emotion recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 448–457, 2008.

[58] Yang, Y.-H., Liu, C.-C., and Chen, H. H., "Music emotion classification: a fuzzy approach," in *Proceedings of the 14th annual ACM international conference on Multimedia*, pp. 81–84, ACM, 2006.

[59] Yeh, C.-C. M., Su, L., and Yang, Y.-H., "Dual-layer bag-of-frames model for music genre classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 246–250, IEEE, 2013.

[60] Yeh, C.-C. M. and Yang, Y.-H., "Supervised dictionary learning for music genre classification," in *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, p. 55, ACM, 2012.

[61] Zhang, Q. and Li, B., "Discriminative K-SVD for dictionary learning in face recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2691–2698, IEEE, 2010.