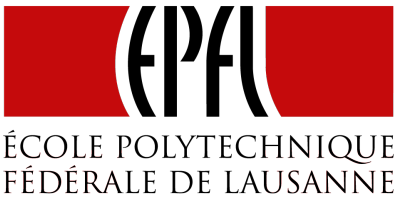


MASTER THESIS

Electrical and Electronic Section
Major in Information Technologies

Structured Auto-Encoder

with application to Music Genre Recognition

Student
Michaël Defferrard

Professor
Pierre Vandergheynst

Supervisors
Xavier Bresson
Johan Paratte

EPFL LTS2 Laboratory

June 19, 2015

Abstract

In this work, we present a technique that learns discriminative audio features for Music Information Retrieval (MIR). The novelty of the proposed technique is to design auto-encoders that make use of data structures to learn enhanced sparse data representations. The data structure is borrowed from the Manifold Learning field, that is data are supposed to be sampled from smooth manifolds, which are here represented by graphs of proximities of the input data. As a consequence, the proposed auto-encoders finds sparse data representations that are quite robust w.r.t. perturbations. The model is formulated as a non-convex optimization problem. However, it can be decomposed into iterative sub-optimization problems that are convex and for which well-posed iterative schemes are provided in the context of the Fast Iterative Shrinkage-Thresholding (FISTA) framework. Our numerical experiments show two main results. Firstly, our graph-based auto-encoders improve the classification accuracy by 2% over the auto-encoders without graph structure for the popular GTZAN music dataset. Secondly, our model is significantly more robust as it is 8% more accurate than the standard model in the presence of 10% of perturbations.

Acknowledgements

I would first like to thank Dr. Xavier Bresson for his dedicated day-to-day supervision, which included a weekly meeting during which we discussed the achieved results and debated new ideas. He further provided me with many inputs, intuitions and references, which were very helpful to clarify some concepts. Finally, I want to thank him for his review of parts of this manuscript.

I would also like to thank Johan Paratte for his helpful intuitions at the beginning of the project and his advices on how to structure the thesis.

Finally, I would like to thank Prof. Pierre Vandergheynst which has made this very interesting project possible. Thanks to him and this project, I've learned much more that the content of this thesis.

Contents

Introduction

This thesis introduces a *structured auto-encoder*, an auto-encoder variant which preserves the structure of the data while transforming it in a sparse representation. It learns sparse representations that explicitly take into account the local manifold structure of the data. The primary goal of the proposed algorithm is unsupervised representation learning toward the goal of feature extraction, while being robust to noisy data and fast at feature extraction (after the training phase). As the discriminative power of learned representations cannot be directly evaluated, the proposed algorithm shall be evaluated through a classification task. We propose an application in Music Information Retrieval (MIR) which consists of retrieving the genre of unknown musical clips.

This thesis report is divided in two parts. Through Part I, some background informations are given in Chapter 1 and the proposed model is constructed step by step in Chapter 2, which is the core of this work. Well-posed iterative schemes to solve the resulting convex sub-optimization problems are then proposed in Chapter 4.As a testbed of our algorithm's performance, an application to Music Genre Recognition (MGR) is proposed in Part II with a presentation of the application in Chapter 5 and the proposed system in Chapter 6. Chapter 7 is dedicated to the presentation of the experimental results.

This work was accomplished as a Master thesis, which is an integral part of the major in Information Technologies curriculum from the Electrical and Electronic Section of the École Polytechnique Fédérale de Lausanne (EPFL). It was conducted at the LTS2 laboratory[1].

While the project was ongoing, I continuously published my thoughts, observations, findings, experiments, results and plans as well as a summary of the weekly meetings with my supervisors on an online blog[2] in the format of an open laboratory notebook. In the spirit of open research, the code as well as all the results, this report and the ongoing paper are versioned with git and available online through GitHub[3]. Some continuation of this work shall be submitted to the 41st IEEE International Conference on Acoustics, Speech

[1]The laboratory homepage is accessible at `http://lts2www.epfl.ch/`.

[2]My research blog is available at `https://lts2research.epfl.ch/blog/mdeff/`.

[3]My GitHub account can be found at `https://github.com/mdeff`.

and Signal Processing (ICASSP).

Part I

Algorithm

Chapter 1

Background

1.1 Neural networks

Artificial Neural Networks (ANNs) are a family of statistical learning models inspired by biological neural networks and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. It is presented as a network of interconnected neurons whose connections have numeric weights that can be tuned based on experience. It makes the neural nets adaptive to inputs and capable of learning.

Such a network is composed by an input layer, a number of hidden layers and an output layer. The activation of the neurons in the input layer corresponds to the input vector, e.g. an image for computer vision, a song for MIR, a word vector for machine translation and sentiment analysis. A weight matrix, followed by a non-linear activation function, then transforms the vector in another representation. The output vector of the first layer is the input vector of the second, and so on until the output layer is reached. The activation of the neurons in the output layer may represent classes, probability distributions, or the estimated value of an unknown function to be learned.

In a feed-forward network, the connections go from one layer to the next, i.e. information only goes in one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. Such networks are known to be able to approximate any function. The perceptron, the Multi-Layer Perceptron (MLP) and the Convolutional Neural Network (CNN) are examples of this class of networks. By introducing backward connections, i.e. the connections between units form a directed cycle, we obtain a so called Recursive Neural Network (RNN). This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feed-forward neural networks, an RNN can use its internal memory to process arbitrary sequences of inputs. It is known to be able to approximate any program. Such networks have proven very successful for machine translation.

In a supervised learning setting, the network is trained by back-propagating the error, gradient based learning method, from the output layer to the input through all the hidden layers. The vanishing gradient problem, where errors shrink exponentially with the number of layers as they propagate from layer to layer, is a major issue of the algorithm [38]. Various methods, like unsupervised pre-training or Long Short Term Memory (LSTM) [37], were developed to work around this problem.

However, in an unsupervised learning setting, there is no desired output, which implies that there is no error to back-propagate. The training algorithm should thus optimize for another objective, which represent desired properties about the output. We will introduce next such an algorithm, called an auto-encoder.

1.2 Auto-encoders

An auto-encoder, auto-associator or Diabolo network is an artificial neural network composed of n input and output units and m hidden units. It is used for learning efficient codings [11, 36]. The aim of an auto-encoder is to learn a distributed representation (encoding) for a set of data. An auto-encoder is trained to encode the input $\mathbf{x} \in \mathbb{R}^n$ into some representation $\mathbf{z} \in \mathbb{R}^m$ so that the input can be reconstructed from that representation. It is thus a generative model. Hence the target output of the auto-encoder is the auto-encoder input itself. Auto-encoders may further be stacked to form a Deep Belief Network (DBN), while each layer can be trained separately [9, 70].

If there is one linear hidden layer and the mean squared error criterion is used to train the network, then the k hidden units learn to project the input in the span of the first k principal components of the data [11]. If the hidden layer is non-linear, the auto-encoder behaves differently from Principal Component Analysis (PCA), with the ability to capture multi-modal aspects of the input distribution [42].

The hope is that the code $\mathbf{z}$ is a distributed representation that captures the main factors of variation in the data: because $\mathbf{z}$ is viewed as a lossy representation of $\mathbf{x}$, it cannot be a good representation (with small loss) for all $\mathbf{x}$. So learning drives it to be one that is a good representation in particular for training examples, and hopefully for others as well (and that is the sense in which an auto-encoder generalizes), but not for arbitrary inputs.

It can typically be used for dimensionality reduction by learning a compressed ($m < n$) representation of the data. Another application is feature extraction before classification, for which we want an higher dimensionality ($m > n$) for easier separability. One serious issue with this approach is that if there is no other constraint, then an auto-encoder with n-dimensional input and an encoding of dimension $m \geq n$ could potentially just learn the

identity function. There are different ways that an auto-encoder with more hidden units than inputs could be prevented from learning the identity, and still capture something useful about the input in its hidden representation $\mathbf{z}$.

Sparse auto-encoders. One strategy, based on the concept of sparse coding, is to add a sparsity constraint on the code. While an ordinary auto-encoder or an Restricted Boltzmann Machine (RBM) has an encoder part which computes $P(\mathbf{z}|\mathbf{x})$ and a decoder part which computes $P(\mathbf{x}|\mathbf{z})$, sparse coding systems only parametrize the decoder: the encoder is implicitly defined as the solution of an optimization. A middle ground between ordinary auto-encoders and sparse coding was proposed in [69, 70] and applied to pattern recognition and machine vision tasks. They propose to let the codes $\mathbf{z}$ be free (as in sparse coding algorithms), but include a parametric encoder (as in an ordinary auto-encoder or RBM) and a penalty for the difference between the free non-parametric codes $\mathbf{z}$ and the outputs of the parametric encoder. In this way, the optimized codes $\mathbf{z}$ try to satisfy two objectives: reconstruct well the input (like in sparse coding), while not being too far from the output of the encoder (which is stable by construction, because of the simple parametrization of the encoder). See Section 2.6 for the definition of our encoder.

Denoising auto-encoders. Another strategy is to add noise in the encoding. The denoising auto-encoder thus minimizes the error in reconstructing the input from a stochastically corrupted transformation of the input [83]. Intuitively, a denoising auto-encoder does two things: try to encode the input (preserve the information about the input), and try to undo the effect of a corruption process stochastically applied to the input of the auto-encoder. This is essentially what a RBM does [35].

Chapter 2

Model

This chapter presents the proposed structured auto-encoder. Built on linear regression, the model increases in complexity as desired properties about its internal representation are progressively integrated in the form of regularizations.

2.1 Assumptions

Sparse representation. We make the hypothesis that a set of sample signals drawn from the same distribution can be sparsely represented in some frame[1]. Each signal should be approximately reconstructed by a linear combinations of a few atoms from a suitable dictionary. As we shall see, this dictionary may be adaptive and learned directly from the data. Many approaches have been developed to achieve sparse representations, e.g. sparse PCA [22], sparse NMF [39], K-SVD [2]. Sparse coding [62, 54] is however the most popular one. Sparsity has become a concept of great interest recently, not only in machine learning but also in statistics and signal processing, in particular with the work on Compressed Sensing (CS) [14, 25].

Structured data. Our second hypothesis is that the dataset holds some structure, in the sense that related samples are close to each other (with respect to some metric). Given a large enough training set, the structure of the data distribution should be able to be captured; i.e. a new valid sample (e.g. from the testing set) should be close to the seen examples. It suggests that the data is drawn from sampling a probability distribution that has support on or near to a submanifold of the ambient space. This manifold is however often unknown and must be learned. Self-Organizing Map (SOM) [45], Locally-Linear Embedding (LLE) [72], Laplacian Eigenmaps [6] and ISOMAP [79] are

[1] A frame of a vector space is a set of vectors which may be linearly dependent. It is a generalization of a basis.

some popular techniques which learn manifolds in a Non-Linear Dimensionality Reduction (NLDR) framework. All these algorithms use the so-called locally invariant idea [32], i.e. the nearby points are likely to have similar embeddings. Auto-encoders used for dimensionality reduction are able to learn a map from high to low-dimensional space with fewer hidden units than inputs. They are trained to learn to optimally encode the input vectors into a small number of dimensions and decode them back into the original space with minimal error [11].

Encoder. We further make the assumption that a simple encoder can be trained to avoid the need of an optimization process that extracts the features during testing, i.e. after the training phase. The encoder shall be more efficient, in the computational sense, than the optimization process while not degrading too much the quality of the extracted features. Note that even if this hypothesis is not verified, the algorithm is still an auto-encoder; although with an implicit encoder.

2.2 Linear regression

Model. Given a set $\{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^n$ of N signals, the subspace $\mathcal{X} = \text{span}\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^n$ is defined as the subspace spanned by the data. Then, given a signal $\mathbf{x} \in \mathcal{X}$ and a frame $\mathbf{D} \in \mathbb{R}^{n \times m}$, we want to find a representation $\mathbf{z} \in \mathbb{R}^m$ which satisfies the linear regression model

$$\mathbf{x} = \mathbf{D}\mathbf{z} + \epsilon, \tag{2.1}$$

where $\epsilon \in \mathbb{R}^n$ is the reconstruction error, which is not negligible as long as the frame $\mathbf{D}$ is not complete on $\mathcal{X}$.

Capacity. The hyper-parameter m defines the learning capacity of the auto-encoder. A capacity $m < n$ is good for dimensionality reduction as it exploits the statistical regularities present in the training set while being more compact. A capacity $m > n$ is good for classification as it allows for an easier (linear) separability enabled by the higher dimensional space.

Completeness. A frame is complete if it can represent any vector $\mathbf{x} \in \mathcal{X}$. It is overcomplete if the removal of a vector from the frame results in a complete frame. A set of $n < m$ linearly independent vectors would indeed be overcomplete on the whole space $\mathbb{R}^n$ and a set of $m = n$ linearly independent vectors, like the Fourier transform, would form a basis[3] of $\mathbb{R}^n$, which is

[2]Figure from Wikipedia.

[3]A basis is a set of linearly independent vectors who span the entire space, i.e. it is a linearly independent spanning set.

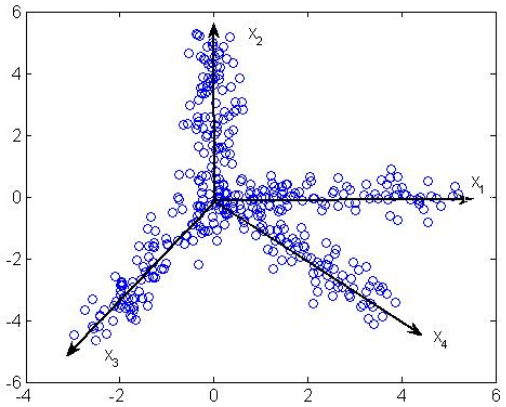


Figure 2.1: An example of overcomplete frame. While the data lies in a two-dimensional space, a four-dimensional space supported by an overcomplete frame allows a better representation.[2]

obviously complete.

A complete frame which is not overcomplete allows bidirectional lossless transformations. Such problems are well-posed as there exists a unique solution to (2.1) with $\epsilon = 0$.

If the frame is not complete, there exist no solution to (2.1) with $\epsilon = 0$ as the system is overdetermined. An error measure, like (2.2) for the Ordinary Least Squares (OLS) method, should instead be minimized.

In different research, such as signal processing and function approximation, overcomplete representations have been advocated because they have greater robustness in the presence of noise, can be sparser, and can have greater flexibility in matching structure in the data. However, because of this redundancy, a signal can have multiple expressions under an overcomplete frame [50]. See Figure 2.1 for an example of the flexibility of an overcomplete frame to represent a dataset. As there is then an infinite number of solutions to (2.1) with $\epsilon = 0$, i.e. the problem is ill-posed, a regularization over $\mathbf{z}$ shall be introduced. Optimization techniques are then used to find the optimal solution which minimizes the sum of the error measure and the regularization term, controlled by an hyper-parameter.

Ordinary least squares. The method of least squares is a standard approach in regression analysis to the approximate solution of overdetermined systems. "Least squares" means that the overall solution minimizes the sum of the squares of the errors made in the results of every single equation, i.e. it finds

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2, \tag{2.2}$$

where $\|\cdot\|_2^2$ denotes the squared ℓ_2, or Euclidean, norm. This problem has the closed-form solution

$$\mathbf{z}^* = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{x}, \tag{2.3}$$

where T denotes the matrix transpose. The primary assumption of OLS is that there are zero or negligible errors in the independent variable $\mathbf{D}$, since this method only attempts to minimize the mean squared error in the dependent variable $\mathbf{x}$. That is not an issue in an auto-encoder setting where $\mathbf{D}$ is either hand-crafted or learned.

Regularization. Tikhonov regularization [81], or ridge regression, is the most commonly used method of regularization of ill-posed problems. It adds a prior of the form $\lambda\|\mathbf{\Gamma z}\|_2^2$ to the minimization problem (2.2) as follows:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{Dz}\|_2^2 + \lambda\|\mathbf{\Gamma z}\|_2^2, \tag{2.4}$$

where $\mathbf{\Gamma}$ is the Tikhonov matrix. This matrix is often chosen to be a multiple of the identity matrix, i.e. $\mathbf{\Gamma} = \alpha\mathbf{I}$, giving preference to solutions with smaller norms. In a Bayesian context, this is equivalent to placing a zero-mean normally distributed prior on $\mathbf{z}$ [84]. In other cases, lowpass operators, e.g. a difference operator or a weighted Fourier operator, may be used to enforce smoothness if the underlying vector is believed to be mostly continuous. A regularization of this kind will be introduced in our model in Section 2.5. An explicit solution is given by

$$\mathbf{z}^* = (\mathbf{D}^T\mathbf{D} + \mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{D}^T\mathbf{x}. \tag{2.5}$$

Another commonly used regularization is the Least Absolute Shrinkage and Selection Operator (LASSO) [80], which adds the prior $\lambda\|\mathbf{z}\|_1$ to the minimization problem (2.2) as follows:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{Dz}\|_2^2 + \lambda\|\mathbf{z}\|_1, \tag{2.6}$$

where $\|\mathbf{z}\|_1 = \sum_{i=1}^m |z_i|$ is the ℓ_1 norm of $\mathbf{z}$, also called the Taxicab or Manhattan norm. In a Bayesian context, this is equivalent to placing a zero-mean Laplace prior distribution on $\mathbf{z}$ [66]. The advantage of the LASSO is that it promotes the simplest solutions, i.e. the solutions with many zeros. Driving parameters to zero effectively deselects the features from the regression. LASSO thus automatically selects the most relevant features, whereas ridge regression never fully discards any. For this reason, the LASSO and its variants are fundamental to the field of CS. A regularization of this kind will be introduced in our model in Section 2.3.

An extension of this approach is the elastic net regularization [91] which linearly combines the ℓ_1 and ℓ_2 penalties of the LASSO and ridge methods as follows:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda_2\|\mathbf{z}\|_2^2 + \lambda_1\|\mathbf{z}\|_1. \tag{2.7}$$

This regularization overcomes some limitations of the ℓ_1 penalty, e.g. the saturation which happens for high-dimensional data with few examples, or the fact that the LASSO tends to select only one variable and ignore the others if there is a group of highly correlated variables.

2.3 Sparse coding

The main idea behind sparse coding [62, 54] is to express the signal $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ as a sparse linear combination of basis functions $\{\mathbf{d}_i\}_{i=1}^m \in \mathbb{R}^n$, or atoms, from an overcomplete dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$. The sparse code $\mathbf{z}^* \in \mathbb{R}^m$ is given by

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \frac{\lambda_d}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda_z\|\mathbf{z}\|_0, \tag{2.8}$$

where $\|\mathbf{z}\|_0$ denotes the number of non-zero elements in $\mathbf{z}$. λ_d and λ_z are the (redundant) hyper-parameters setting the trade-off between the data term, an accurate reconstruction, and the prior, a sparse solution. Overcomplete sparse representations tend to be good features for classification systems as they provide a succinct representation of the signal, are robust to noise and are more likely to be linearly separable due to their high dimensionality.

Finding the sparse code $\mathbf{z}^*$ however requires a combinatorial search which is an NP-hard problem [59], intractable in high dimensional spaces. Various approximations have thus been proposed. Matching Pursuit (MP) [56] offers a greedy approximation to the solution while Basis Pursuit (BP) [17] is the popular convex approximation

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \frac{\lambda_d}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda_z\|\mathbf{z}\|_1, \tag{2.9}$$

which is the LASSO regularized least square problem introduced in (2.6). As is now well understood [14, 25], the ℓ_1 norm is a very good proxy for the ℓ_0 pseudo-norm and naturally induces sparse results. It can even be shown to recover exactly the true sparse code, i.e. the solution of (2.8) (if there is one), under mild conditions [26]. A number of algorithms have been proposed to efficiently solve this problem [17, 5, 48, 51]. They however still rely on computationally expensive iterative procedures which limit the system's scalability and real-time applications. While a direct method will always be preferred for feature extraction, iterative methods will still be necessary during training. Distributed computing with Graphical Processing Unit (GPU) or via cloud computing will hopefully accelerate the process.

2.4 Dictionary learning

Model. In classical sparse coding, the dictionary is composed of known functions such as sinusoids [12], wavelets [55], Gabors [28], curvelets [13] or contourlets [24]; i.e. hand-crafted features. One may also want to learn a dictionary that is adaptive to the type of data at hand. This approach may allow an even more compact representation and may lead to the discovery of previously unknown discriminative features.

To use the dictionary $\mathbf{D}$ as an unknown variable, all the training data shall be part of the objective function as the dictionary depends on all of them. The energy function, composed by an ℓ_2 fidelity term and an ℓ_1 penalty, becomes

$$E_1(\mathbf{Z}, \mathbf{D}) = \frac{\lambda_d}{2}\|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_{\mathrm{F}}^2 + \lambda_z\|\mathbf{Z}\|_1, \tag{2.10}$$

where $\|\cdot\|_{\mathrm{F}}^2$ denotes the squared Frobenius norm, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^{n\times N}$ is the set of training vectors and $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N \in \mathbb{R}^{m\times N}$ their associated sparse codes. N is naturally the number of training vectors, which should be much greater than the size m of the dictionary to avoid the trivial solution where examples are copied in the dictionary. The problem to solve is then

$$\underset{\mathbf{Z},\mathbf{D}}{\text{minimize}}\ E_1(\mathbf{Z}, \mathbf{D}) \text{ s.t. } \|\mathbf{d}_i\|_2 \leq 1,\ i = 1, \ldots, m, \tag{2.11}$$

where the ℓ_2 ball constraint (usually implemented by rescaling the columns $\mathbf{d}_i$ of $\mathbf{D}$ at each iteration) prevents the trivial solution where the code coefficients go to zero while the bases are scaled up. While this problem is not convex, a good approximate solution can be found by iteratively minimizing for $\mathbf{Z}$ and $\mathbf{D}$ [62].

Completeness. The learned dictionary may be seen as an overcomplete frame of the subspace $\mathcal{X}$ spanned by the training data. The overcompleteness of the learned dictionary could indeed be tested: the frame should be able to perfectly represent any training sample, i.e. in the absence of the ℓ_1 regularization, the reconstruction error ϵ of (2.1) should be zero.

Biological motivation. There is evidence that sparse coding may be a strategy employed by the brain in the early stages of visual and auditory processing [62, 64, 75]. Basis functions learned on natural images have been shown to resemble the receptive fields of neurons in the visual cortex [62, 64]. Basis functions learned on natural sounds were found to be highly similar to gammatone functions [75] which have been used to model the action of the basilar membrane in the inner ear. Moreover, learning on natural time-varying stimuli such as speech or video has been shown to produce localized bases [49, 63].

2.5 Manifold learning

Motivation. Most of the existing approaches to sparse coding do not consider the geometrical structure of the data space. The data is however more likely to reside on a low-dimensional submanifold embedded in the high-dimensional ambient space. It has been shown that the learning performance of a sparse coding scheme can be significantly enhanced if the geometrical structure is exploited and the local invariance is considered [88].

Similarity graphs. Graphs are generic data representation forms which are useful for describing the geometric structures of data domains in numerous applications, including social, energy, transportation, sensor, and neuronal networks [74]. The connectivity and weight associated with each edge in the graph is either dictated by the physics of the problem at hand or inferred from the data. Weighted graphs are commonly used to represent similarities between data points in statistical learning problems for applications such as machine vision [52] and automatic text classification [3].

From the set of training vectors $\mathbf{X}$, we can construct an undirected, connected and weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ which consists of a finite set of vertices $\mathcal{V}$ with $|\mathcal{V}| = N$, a set of edges $\mathcal{E}$, and a weighted adjacency matrix $\mathbf{W} = (w_{ij}) \in \mathbb{R}^{N \times N}$. Each vertex $i \in \mathcal{V}$ represents a training vector $\mathbf{x}_i$. If there is an edge $e = (i, j)$ connecting vertices i and j, the entry w_{ij} represents the weight of the edge; otherwise, $w_{ij} = 0$. The set of sparse codes $\mathbf{Z}$ is a signal which resides on the graph, i.e. a signal with one sample $\mathbf{z}_i$ at each vertex i of the graph.

While there exist several ways to define the edge weights when they are not naturally defined by the application, they often represent the similarity between the two vertices they connect [74]. For instance, the edge weight may be inversely proportional to the Euclidean distance between the vectors:

$$w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \in [0, 1], \tag{2.12}$$

where the Gaussian kernel width σ controls the width of the neighborhoods and $\langle \cdot, \cdot \rangle$ denotes the scalar product. While the Euclidean distance is a good choice for low-dimensional data, its discriminative power vanishes in higher dimensional space [1, 23]. An option is then to use the cosine similarity as the edge weight:

$$w_{ij} = \frac{1}{2} + \frac{1}{2}\cos(\theta) = \frac{1}{2}\left(1 + \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}\right) \in [0, 1], \tag{2.13}$$

where θ is the angle between the two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. Another reason for the popularity of the cosine similarity is that it is very efficient to evaluate, espe-

cially for sparse vectors, as only the non-zero dimensions need to be considered. See [30] for other graph construction methods.

A fully connected graph is usually not wanted: the number of edges are often artificially limited in order to reduce the storage and computational cost associated with the graph manipulation, effectively sparsifying the weight matrix $\mathbf{W}$. The ϵ-neighborhood graph is the popular approach which sets to 0 any weigh $w_{ij} < \epsilon$ for some threshold ϵ. A second common method is to connect each vertex to its k-nearest neighbors only and drop the smallest weights; in which case [87] suggests to set the Gaussian kernel scale σ to the mean of the k^{st} distances.

Graph Laplacian. The unnormalized graph Laplacian, also called the combinatorial graph Laplacian, is defined as

$$\mathbf{L} = \mathbf{A} - \mathbf{W}, \tag{2.14}$$

where the degree matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{N \times N}$ is a diagonal matrix whose ith diagonal element a_{ii} is equal to the sum of the weights of all the edges incident to vertex i:

$$a_{ii} = \sum_{j=1}^{N} w_{ij}. \tag{2.15}$$

The graph Laplacian is a difference operator as it satisfies

$$\mathbf{L}\mathbf{z}_i = \sum_{j=1}^{N} w_{ij}(\mathbf{z}_i - \mathbf{z}_j). \tag{2.16}$$

Dirichlet energy. The Dirichlet energy is a measure of the smoothness of a graph signal given by

$$\operatorname{tr}(\mathbf{Z}\mathbf{L}\mathbf{Z}^T) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \geq 0, \tag{2.17}$$

which is a suitable candidate for regularization [7]. The assumption that the representations $\mathbf{Z}$ should be smooth on the similarity graph $\mathcal{G}$ constructed by the training vectors $\mathbf{X}$, usually referred to as the manifold assumption[4], plays an essential role in various kinds of algorithms including dimensionality reduction algorithms [6], clustering algorithms [61] and semi-supervised learning algorithms [7, 89]. Note that the Euler-Lagrange of the Dirichlet energy is precisely the graph Laplacian.

[4]Because the graph is used as a proxy for the manifold.

Consistency. Two normalized graph Laplacians are found in the literature [18]:

$$\mathbf{L}_{sym} = \mathbf{A}^{-1/2}\mathbf{L}\mathbf{A}^{-1/2} = \mathbf{I} - \mathbf{A}^{-1/2}\mathbf{W}\mathbf{A}^{-1/2}, \tag{2.18}$$

and

$$\mathbf{L}_{rw} = \mathbf{A}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{A}^{-1}\mathbf{W}, \tag{2.19}$$

where $\mathbf{I}$ denotes the identity matrix. While there is no convergence guarantee for the unnormalized graph Laplacian, these two normalized Laplacian can be shown to converge to the continuous Laplace-Beltrami operator as the number of samples increase [85]. The similarity graph is indeed a good approximation of the unknown manifold.

Model. The geometrical information about the data is encoded in a similarity graph constructed by the training vectors $\mathbf{X}$ and the graph Laplacian is used as a smooth operator to preserve the local manifold structure. Introducing the Dirichlet energy into the objective function as an additional ℓ_2 regularization, similar to the Tikhonov regularization presented in Section 2.2, gives

$$E_2(\mathbf{Z}, \mathbf{D}) = \frac{\lambda_d}{2}\|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_{\mathrm{F}}^2 + \lambda_z\|\mathbf{Z}\|_1 + \frac{\lambda_g}{2}\operatorname{tr}(\mathbf{Z}^T\mathbf{L}\mathbf{Z}). \tag{2.20}$$

This regularization promotes a smooth variations of the representations along the geodesics of the data manifold.

2.6 Encoder

Motivation. In order to avoid the iterative procedure typically required to infer the sparse code, we aim at an explicit encoder which can quickly map inputs to approximations of their sparse code. Several works [44, 31, 71] have been done in this direction. The addition of an explicit encoder to the sparse coding scheme bridges the gap between auto-encoders and sparse coding and is often referred to as sparse auto-encoders [8].

Moreover, adding structure to the problem should enhance the behavior of the loss function and help sparse recovery [46, 4, 40, 43].

Model. Introducing a trainable encoder $\mathbf{E} \in \mathbb{R}^{m\times n}$, designed to predict sparse codes from input vectors with minimum error, into our model gives the energy function

$$E_3(\mathbf{Z}, \mathbf{D}, \mathbf{E}) = \frac{\lambda_d}{2}\|\mathbf{X}-\mathbf{D}\mathbf{Z}\|_{\mathrm{F}}^2+\lambda_z\|\mathbf{Z}\|_1+\frac{\lambda_g}{2}\operatorname{tr}(\mathbf{Z}^T\mathbf{L}\mathbf{Z})+\frac{\lambda_e}{2}\|\mathbf{Z}-\mathbf{E}\mathbf{X}\|_{\mathrm{F}}^2, \tag{2.21}$$

where λ_e is an additional hyper-parameter which controls the relative weight of the prediction error. The problem is then defined by

$$\underset{\mathbf{Z},\mathbf{D},\mathbf{E}}{\text{minimize}} \; E_3(\mathbf{Z},\mathbf{D},\mathbf{E}) \text{ s.t. } \|\mathbf{d}_i\|_2 \leq 1, \|\mathbf{e}_k\|_2 \leq 1, \; i = 1,\ldots,m, \; k = 1,\ldots,n, \tag{2.22}$$

where $\mathbf{e}_k$ are the columns of $\mathbf{E}$.

Energy. While it is often a good idea to control the energy, the constraint on the columns of $\mathbf{E}$ is not needed in practice. While the columns of $\mathbf{D}$ are constrained to a norm smaller than one, they are in practice normalized because of the $\|\mathbf{Z}\|_1$ objective. The energy of a vector transformed by $\mathbf{D}$ does thus not change, which means that the inferred sparse code $\mathbf{z}_i$ as the same energy as its corresponding vector $\mathbf{x}_i$. The encoder does then not need to add energy and will have a column norm smaller than one, even without the constraint. It has been verified empirically.

2.7 Auto-encoder

Energy formulation. The energy function (2.21), which defines our model, may be rewritten as a sum of functions of the variables $\mathbf{Z}$, $\mathbf{D}$ and $\mathbf{E}$ as follows:

$$E(\mathbf{Z},\mathbf{D},\mathbf{E}) = \underbrace{\frac{\lambda_d}{2}\|\mathbf{X}-\mathbf{D}\mathbf{Z}\|_{\mathrm{F}}^2}_{f_d(\mathbf{Z},\mathbf{D})} + \underbrace{\lambda_z\|\mathbf{Z}\|_1}_{f_z(\mathbf{Z})} + \underbrace{\frac{\lambda_g}{2}\operatorname{tr}(\mathbf{Z}^T\mathbf{L}\mathbf{Z})}_{f_g(\mathbf{Z})} + \underbrace{\frac{\lambda_e}{2}\|\mathbf{Z}-\mathbf{E}\mathbf{X}\|_{\mathrm{F}}^2}_{f_e(\mathbf{Z},\mathbf{E})}. \tag{2.23}$$

An advantage of energetic formulations is that it is easy to control the relative importance of the sub-objectives, whether they are fidelity or prior terms. As seen through this chapter, the model can be easily constructed by sequentially adding terms to the objective. They are as easily removed, or muted, when experimentations require it. Other advantages include good understanding, robustness, existence of solutions, design and analysis of optimization algorithms.

Auto-encoder model. Training the model is akin to minimize the objective function (2.23) over the variables, or model parameters $\mathbf{Z}$, $\mathbf{D}$ and $\mathbf{E}$:

$$\begin{aligned}\underset{\mathbf{Z},\mathbf{D},\mathbf{E}}{\text{minimize}} \; & f_d(\mathbf{Z},\mathbf{D}) + f_z(\mathbf{Z}) + f_g(\mathbf{Z}) + f_e(\mathbf{Z},\mathbf{E}) \\ & \text{s.t. } \|\mathbf{d}_i\|_2 \leq 1, \; \|\mathbf{e}_k\|_2 \leq 1, \; i = 1,\ldots,m, \; k = 1,\ldots,n. \end{aligned} \tag{2.24}$$

Posing the problem as the minimization of an objective function is a sound expression of the model. Indeed, many laws of nature are nothing but optimality conditions, often expressed in terms of a minimum energy principle.

Note that the regularizations $f_z(\mathbf{Z})$ and $f_g(\mathbf{Z})$ on the internal representation are preventing the auto-encoder to learn the identity $\mathbf{D} = \mathbf{E} = \mathbf{I}$.

Encoder. The proposed model is an auto-encoder. Given its hyper-parameters $\lambda_d, \lambda_z, \lambda_g, \lambda_e \geq 0$ and a training set $\mathbf{X}$, model (2.24) learns the auto-encoder variables, i.e. the dictionary $\mathbf{D}$ and the encoder $\mathbf{E}$. Then, given $\mathbf{D}$ and $\mathbf{E}$, the sparse and structured internal representation $\mathbf{z}^*$ of an unseen sample $\mathbf{x}$ is given by

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \frac{\lambda_d}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda_z\|\mathbf{z}\|_1 + \frac{\lambda_g}{2}\langle \mathbf{z}, \mathbf{L}\mathbf{z}\rangle + \frac{\lambda_e}{2}\|\mathbf{z} - \mathbf{E}\mathbf{x}\|_2^2, \quad (2.25)$$

where the graph Laplacian $\mathbf{L}$ is constructed from the data.

Decoder. Similarly, the mapping of a representation $\mathbf{z}$ back to the input domain is given by

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{\lambda_d}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \frac{\lambda_e}{2}\|\mathbf{z} - \mathbf{E}\mathbf{x}\|_2^2. \quad (2.26)$$

2.8 Approximate schemes

Motivation. While encoder (2.25) extracts the exact representation given the auto-encoder model (2.24), it is computationally heavy. We aim at a faster encoder model to infer an approximate representation $\tilde{\mathbf{z}} \approx \mathbf{z}^*$.

Direct encoder. Neglecting some of the terms in (2.25) because of our model's third assumption given in Section 2.1, the approximation:

$$\tilde{\mathbf{z}} = \arg\min_{\mathbf{z}} \frac{\lambda_e}{2}\|\mathbf{z} - \mathbf{E}\mathbf{x}\|_{\mathrm{F}}^2 + \lambda_z\|\mathbf{z}\|_1 \quad (2.27)$$

is able to infer good enough representations. Problem (2.27) holds a closed-form solution:

$$\tilde{\mathbf{z}} = h_{\lambda_z/\lambda_e}(\mathbf{E}\mathbf{x}), \quad (2.28)$$

where h_λ is the shrinkage function defined here by

$$h_\lambda(\mathbf{x})_k = \mathrm{sign}(x_k)\left(|x_k| - \lambda\right)_+, \quad (2.29)$$

where $(\cdot)_+ = \max(\cdot, 0)$. Therefore, the explicit encoder formulation introduced in Section 2.6 allows a direct inference of the representation.

Note that although the dictionary $\mathbf{D}$ and the graph Laplacian $\mathbf{L}$ are not used in the approximate scheme (2.28), they however add structure to the problem because $\mathbf{E}$ is learned simultaneously with $\mathbf{D}$ and a regularization which makes use of $\mathbf{L}$. Additional structure has been shown to enhance the behavior of the loss function and help sparse recovery [46, 4, 40, 43].

Direct decoder. Although we care less about decoder (2.26), a similar approach may be used to approximate $\mathbf{x}^*$:

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{\lambda_d}{2} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 = \mathbf{D}\mathbf{z}. \tag{2.30}$$

Chapter 3

Related works

Standard auto-encoders. Auto-encoders, as introduced by [11, 36], are defined by one linear hidden layer and the mean squared error criterion is used to train the network, i.e. their model is (2.24) without any regularization. Because of the lack of constraint, an auto-encoder with n-dimensional input and an encoding of dimension at least n could potentially just learn the identity function. However, it as been shown that Stochastic Gradient Descent (SGD) with early stopping is similar to an ℓ_2 regularization of the parameters [90].

Sparse auto-encoders. Direct decoder (2.30) is the definition used for sparse auto-encoders [69, 70]. While the encoder definition may vary, it includes at least the first two terms of encoder (2.25), related to sparse coding.

Predictive sparse decomposition. A technique introduced in [44] which, similarly to us, adds an explicit encoder to the sparse coding scheme. Their encoder architecture is very close to direct encoder (2.28). Using sparse coding, their decoder is defined by direct decoder (2.30).

Denoising auto-encoders. Denoising auto-encoders share the same model as the standard auto-encoders, but are trained with stochastically corrupted data [83]. They thus learn to undo the effect of the corruption. In model (2.24), the Dirichlet energy term promotes smooth variations along the data manifold. It has the effect of pushing noisy samples toward the manifold, effectively denoising them.

Chapter 4

Optimization

The whole process of training the auto-encoder is to solve the non-convex optimization problem which defines model (2.24). Because of its non-convexity, there is no guarantee to find its global minimum. The minimization will end up in a local minimum depending on the initialization. That is, given the same training data, different random initializations of the variables will lead to different solutions.

Convex sub-problems. As problem (2.24) is non-convex only when all the variables are taken together, a natural way is to decompose it into three convex sub-problems:

$$\underset{\mathbf{Z}}{\text{minimize}}\ f_d(\mathbf{Z}, \mathbf{D}) + f_z(\mathbf{Z}) + f_g(\mathbf{Z}) + f_e(\mathbf{Z}, \mathbf{E}), \tag{4.1}$$

$$\underset{\mathbf{D}}{\text{minimize}}\ f_d(\mathbf{Z}, \mathbf{D}) \text{ s.t. } \|\mathbf{d}_i\|_2 \leq 1,\ i = 1, \ldots, m, \tag{4.2}$$

$$\underset{\mathbf{E}}{\text{minimize}}\ f_e(\mathbf{Z}, \mathbf{E}) \text{ s.t. } \|\mathbf{e}_k\|_2 \leq 1,\ k = 1, \ldots, n. \tag{4.3}$$

Sub-problem (4.1) is an ℓ_1 and ℓ_2 regularized least squares problem while sub-problems (4.2) and (4.2) are constrained least squares problems. Both of which can efficiently be solved by several convex optimization methods [19, 5, 15].

Iterative scheme. The idea is to iteratively solve (4.1), (4.2) and (4.3), i.e. to minimize the objective for one variable at a time while fixing the two others. While we have no convergence guarantee, the overall loss function is guaranteed to decrease monotonically if each convex sub-problem is optimally solved, which is the case if we let each sub-minimization converge.

Proximal splitting. Proximal splitting methods are a class of algorithms designed to solve convex problems of the form

$$\underset{\mathbf{x}}{\text{minimize}}\ f_1(\mathbf{x}) + f_2(\mathbf{x}), \tag{4.4}$$

where f_1 is a convex but non-smooth function and f_2 is convex and differentiable with a β-Lipschitz continuous gradient ∇f_2, i.e.,

$$\forall(\mathbf{x},\mathbf{y}) \quad \|\nabla f_2(\mathbf{x}) - \nabla f_2(\mathbf{y})\|_2 \leq \beta\|\mathbf{x}-\mathbf{y}\|_2, \tag{4.5}$$

where $\beta \in]0,+\infty[$.

It can be shown [20] that problem (4.4) admits at least one solution and that, for any $\gamma \in]0,+\infty[$, its solutions are characterized by the fixed point equation

$$x = \text{prox}_{\gamma f_1}\left(\mathbf{x} - \gamma\nabla f_2(\mathbf{x})\right), \tag{4.6}$$

where prox_f denotes the proximity operator of f, defined as the minimization problem

$$\text{prox}_f\mathbf{x} = \underset{\mathbf{y}}{\text{minimize}} \;\; f(\mathbf{y}) + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2, \tag{4.7}$$

which is an extension of the notion of a projection operator [57]. The fixed point equation (4.6) suggests the possibility of iterating

$$\mathbf{x}^{t+1} = \underbrace{\text{prox}_{\gamma_t f_1}}_{\text{backward step}} \underbrace{\left(\mathbf{x}^t - \gamma\nabla f_2(\mathbf{x}^t)\right)}_{\text{forward step}} \tag{4.8}$$

for values of the step-size parameter γ_t in a suitable bounded interval which depends on β. This type of scheme is known as a *forward-backward* splitting algorithm. It can be broken up into a forward (explicit) gradient step using the function f_2, and a backward (implicit) step using the function f_1 [19].

The Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [5] is an efficient forward-backward scheme which exploits variable time steps and multiple points. It achieves an optimal [60] $O(1/t^2)$ rate of convergence of the objective function.

Sub-problems casting. To be solved via FISTA, each of the minimization sub-problems (4.1), (4.2) and (4.3) has to be cast to the form of (4.4) and provide the gradient ∇f_2, its Lipschitz constant β and the proximity operator of f_1.

Sub-problem (4.1) is split into a smooth and a non-smooth part:

$$\underset{\mathbf{Z}}{\text{minimize}} \;\; \underbrace{f_d(\mathbf{Z},\mathbf{D}) + f_g(\mathbf{Z}) + f_e(\mathbf{Z},\mathbf{E})}_{f_2(\mathbf{Z})} + \underbrace{f_z(\mathbf{Z})}_{f_1(\mathbf{Z})}. \tag{4.9}$$

The gradient ∇f_2, its Lipschitz constant β and the proximity operator of f_1 are as follows:

$$\nabla f_2(\mathbf{Z}) = \lambda_d\mathbf{D}^T(\mathbf{X}-\mathbf{D}\mathbf{Z}) + \lambda_e(\mathbf{Z}-\mathbf{E}\mathbf{X}) + \lambda_g\mathbf{L}\mathbf{Z} \tag{4.10}$$

$$\beta \geq \lambda_e + \lambda_d\|\mathbf{D}^T\mathbf{D}\|_2 + \lambda_g\|\mathbf{L}\|_2 \tag{4.11}$$

$$\text{prox}_{\beta^{-1}f_1}(\mathbf{D}) = h_{\lambda_z/\beta}(\mathbf{Z}) \tag{4.12}$$

where h_λ is the shrinkage function (2.29).

The constraint of (4.2) can be integrated in the loss function via the Lagrange multiplier method:

$$\underset{\mathbf{D}}{\text{minimize}} \underbrace{\frac{\lambda_d}{2}\|\mathbf{X}^T - \mathbf{Z}^T\mathbf{D}^T\|_{\mathrm{F}}^2}_{f_2(\mathbf{D})} + \underbrace{\iota_C(\mathbf{D})}_{f_1(\mathbf{D})}, \tag{4.13}$$

where ι_C is the indicator function of the subset $C \in \mathbb{R}^{n\times m}$ defined as

$$\iota_C(\mathbf{D}) = \begin{cases} 0, & \text{if } \|\mathbf{d}_i\|_2 \leq 1,\ i = 1, \ldots, m; \\ +\infty, & \text{otherwise.} \end{cases} \tag{4.14}$$

The gradient ∇f_2, its Lipschitz constant β and the proximity operator of f_1 are as follows:

$$\nabla f_2(\mathbf{D}) = \lambda_d \mathbf{Z}(\mathbf{X}^T - \mathbf{Z}^T\mathbf{D}^T) \tag{4.15}$$

$$\beta \geq \lambda_d \|\mathbf{Z}\mathbf{Z}^T\|_2 \tag{4.16}$$

$$\mathrm{prox}_{\beta^{-1}f_1}(\mathbf{D}) = \left\{ \frac{\mathbf{d}_i}{\max(1, \|\mathbf{d}_i\|_2)} \right\}_{i=1}^{m}. \tag{4.17}$$

Similarly, for (4.3):

$$\nabla f_2(\mathbf{E}) = \lambda_e \mathbf{X}(\mathbf{Z}^T - \mathbf{X}^T\mathbf{E}^T) \tag{4.18}$$

$$\beta \geq \lambda_e \|\mathbf{X}\mathbf{X}^T\|_2 \tag{4.19}$$

$$\mathrm{prox}_{\beta^{-1}f_1}(\mathbf{E}) = \left\{ \frac{\mathbf{e}_k}{\max(1, \|\mathbf{e}_k\|_2)} \right\}_{k=1}^{n}. \tag{4.20}$$

Part II

Application

Chapter 5

Music genre recognition

5.1 Problem formulation

Music genre recognition (MGR) is a common task for MIR and is the usual task for the yearly Music Information Retrieval Evaluation eXchange (MIREX)[1]. The MGR problem is the task of automatically recognizing the musical genre of an unknown audio clip given a set of labeled clips. A clip is unknown in the sense that only the raw audio is available, and there is no access to any meta-data.

The accuracy of the classified clips is used as a proxy for the discriminative power of the learned representations.

5.2 Dataset

The system's performance is evaluated on GTZAN[2], the most-used public dataset for evaluation in machine listening research for MGR [77] which was created by Tzanetakis and Cook for their work in the domain in 2002 [82]. It consists of 1000 30-second audio clips (all truncated to $660,000$ samples to not bias the classifier in any ways) with 100 examples in each of 10 different categories: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae and rock. All clips are sampled at 22,050 Hz.

The composition and integrity of the dataset has been analyzed in [76]. It is known that the dataset contains recurrent faults: repetitions, mislabelings, and distortions. These faults obviously challenge the interpretability of any result derived using it. The work [78] goes even further and disprove the claims that all MGR systems are affected in the same ways by these faults, and that the performances of MGR systems in GTZAN, working with the same data

[1] `http://www.music-ir.org/mirex/wiki/MIREX_HOME`

[2] Available at `http://marsyasweb.appspot.com/download/data_sets/`.

and faults, are still meaningfully comparable.

Chapter 6

System

The design of the genre recognition system is inspired by the work [34], which uses a layer of sparse auto-encoder to learn sparse representations of audio spectrograms targeted at MGR. Although they call their encoder technique Predictive Sparse Decomposition (PSD) [44], it is effectively a sparse auto-encoder [8]. The structured auto-encoder proposed and developed in Part I generalizes the loss function of [34] as an energy function and introduces a graph-based structuring term (Section 2.5).

The system mainly consists of three independent building blocks:

1. **Preprocessing**: the goal is to transform the raw audio signal into a time-frequency representation. It is essentially a first feature extraction pass, built on prior knowledge about signal processing and audio signals in particular. The spectrograms are further sliced in short time frames to provide time invariance.

2. **Feature extraction**: the set of spectrogram slices is given as input vectors to the proposed structured auto-encoder which transforms them into a sparse and structured representations.

3. **Classification**: the extracted features are used for genre classification after a post-processing step analog to feature pooling. The accuracy is assessed by a cross-validation scheme.

6.1 Preprocessing

Frames. Each clip is divided into short frames of $n_a = 1024$ samples, which roughly corresponds to 46 ms of raw audio, in order to provide translation invariance. A 50% overlap between consecutive frames introduces redundancy in the data. The GTZAN dataset is thus decomposed in $N = 1,288,000$ frames of dimensionality $n_a = 1024$.

Constant-Q Transform (CQT). A spectral representation of each of those frames is computed via the CQT with $n_s = 96$ filters spanning four octaves from C_2 to C_6 at quarter-tone resolution, where C_4 is the middle C in the scientific pitch notation [86]. The A440 tuning standard sets $A_4 = 440$ Hz [41].

Apart from the constant quality factor, i.e. a constant frequency over bandwidth ratio, an important property of the CQT is that the center frequencies of the filters are logarithmically spaced, so that consecutive notes in the musical scale are linearly spaced. This transform is generally well suited to musical data: (i) the logarithm scale requires fewer frequency bins to cover a given range effectively, which proves useful when frequencies span several octaves, (ii) it mirrors the human auditory system, whereby at lower frequencies spectral resolution is better, whereas temporal resolution improves at higher frequencies, and (iii) the harmonics of musical notes form a pattern characteristic of the timbre of the instrument; as the fundamental frequency changes, the relative position of these harmonics remains constant.

This step is an efficient dimensionality reduction from $n_a = 1024$ to $n_s = 96$. It is efficient in the sense that it extracts useful features, as demonstrated by the baseline experiment. See Section 7.1.

The implementation uses *librosa*[1], a Python library which implements an efficient algorithm proposed by the authors of [73]. An efficient and accurate implementation of the necessary sample rate conversion is provided by *libsamplerate*[2].

Local Contrast Normalization (LCN). A subtractive and divisive LCN, described in [47], may then be applied to the spectrograms in order to enhance the contrast. Consider a point in the spectrogram and its neighborhood along both the time and frequency axes weighted by a Gaussian window. First, the average of the weighted neighborhood is subtracted from each point (the subtractive part). Then, each point is divided by the standard deviation of its new weighted neighborhood (the divisive part).

The technique enforces competition between neighboring points in the spectrogram, so that low-energy signals are amplified while high-energy ones are muted. The entire process can be seen as a simple form of Automatic Gain Control (AGC). It acts as an inverse heat diffusion operator, similarly to a shock filter [65]. The idea of contrast normalization is inspired by visual neuroscience models [53, 68].

Scaling. The range of the independent variables is finally rescaled in $[0, 1]$ to eliminate any bias toward features who have a broad range of values. Each feature then contributes approximately proportionally to the distance between

[1] Available at `https://github.com/bmcfee/librosa/`.

[2] Available at `http://www.mega-nerd.com/SRC/index.html`.

two features vectors. Another commonly used scaling method is standardization: the independent variables are rescaled to have zero-mean and unit-variance. Yet another method is to rescale each features vector to unit length, which usually means dividing each component by the Euclidean length of the vector.

6.2 Feature extraction

In a transductive learning paradigm [29], the auto-encoder model (Chapter 2) is trained on the entire dataset (which includes the training and test sets without labels). Under this paradigm, test samples are known in advance, and the system is simply asked to produce labels for them. This paradigm was chosen for speed and accuracy reasons: since training is computationally expensive, it is best done only once per experiment. The training phase is essentially the application of the algorithm presented in Chapter 4 to solve (2.24) over the whole dataset. A structured and sparse representation is readily available for each spectrogram of the dataset, they are a by-product of model fitting.

Note that while that paradigm is used for this application, the algorithm proposed in Part I is still purely unsupervised, i.e. it makes no use of the training labels. Note also that while the system was designed for transductive learning, it can also act in a supervised way. A predictive model was built during training, and, using the learned parameters, a representation for a previously unknown spectrogram may be inferred with (2.25) or (2.28).

The optimization scheme presented in Chapter 2 is implemented with the PyUNlocBoX[3], an open-source Python convex optimization toolbox developed and maintained by ourselves. The toolbox has been enhanced by the needs of this project, e.g. its memory footprint has been greatly reduced. The FLANN library [58] is used for a fast approximate k-nearest neighbors search, used to construct the graph as described in Section 2.5.

6.3 Classification

Aggregate features. Aggregate features are computed for each song by summing up the frame-level features over 5-second time windows overlapping by half, which has been found to substantially improve classification performance [10, 33]. Since each sparse code records which dictionary elements are present in a given CQT frame, these aggregate features vectors can be thought of as histograms recording the number of occurrences of each dictionary element in the time window.

[3] Available at `https://github.com/epfl-lts2/pyunlocbox`.

Support Vector Machine (SVM). Aggregate features are then classified by SVM [21], which is a non-probabilistic binary linear classifier. It was chosen because it is fast to train and scale well to large datasets (thanks to the few support vectors), which is an important consideration in MIR.

In a one-vs-the-rest strategy to multi-class classification, SVM basically constructs a set of maximum-margin hyperplanes which separate a class from all the others. The alternative approach to multi-class, one-vs-one, requires to train one classifier per pair of class.

The implementation uses the *scikit-learn* Python framework [67] which provides wrappers to *libsvm* [16] and *liblinear* [27], two independent implementations of the SVM. We observed no significant variations between the two.

Majority voting. The genre prediction for a clip is given by a majority vote between the 12 aggregate features. Instead of using it in a winner-take-all fashion, this information could be exploited as a confidence level about the chosen class.

Chapter 7

Results

Performance evaluation Following standard practice, the classification accuracy was measured by 10-fold cross-validation. For each fold, 10% of the aggregate features were randomly selected to serve as a test set, wile the remaining 90% served as training data. This procedure was repeated 20 times, and the results averaged to produce a final classification accuracy.

Furthermore, the classification accuracy was measured on aggregate features, i.e. before majority voting, rather than on whole clip. The rational is to capture the confidence about the class of a clip. Another positive impact is the reduction of the variance caused by the increased number of samples.

Note also that no contrast enhancement (the LCN described in Section 6.1) was applied to the spectrograms for these experiments.

Speed and memory considerations. The iterative optimization scheme presented in Chapter 4 is, by definition, computationally heavy. Despite having been sped up by an order of magnitude already, our implementation is still quite sub-optimal from a computational point-of-view. As the algorithm itself is still a prototype, we did not invest too much time to further optimize its implementation. For this reason, the following experiments where all conducted on a subset of the dataset.

Despite the fact that the memory consumption was divided by five since the first working implementation, it is still not enough to run a simulation with the whole dataset on a virtual served equipped with 30GB of RAM. Note that the current implementation retains everything in memory.

Reproducibility. The complete simulation reports of all the conducted experiments, whether they succeed or failed, may be viewed online[1]. The code

[1]The IPython notebooks are stored on GitHub and can be visualized at `http://nbviewer.ipython.org/github/mdeff/dlaudio_results/`.

to reproduce all the results may also be downloaded[2].

7.1 Spectrograms

On a classification experiment with $N_{genres} = 2$ genres, $N_{clips} = 100$ clips per genre and $N_{frames} = 644$ frames per clip, the system achieved a classification accuracy of 96 (+/- 4.7) using the CQT spectrograms, whereas classification with raw audio yielded an accuracy of 89 (+/- 5.0). It confirms that CQT spectrograms have more discriminative power than raw audio.

Classifications with CQT spectrograms will be our baseline for further experiments. Improvements in accuracy with the extracted features will be reported with respect to this baseline.

7.2 Figures of merit

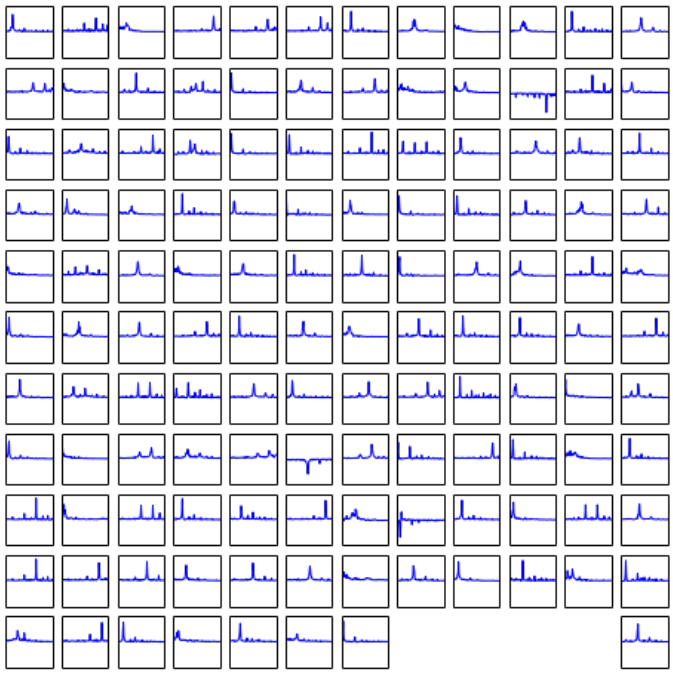


Figure 7.1: $m = 128$ atoms of a learned dictionary of spectrograms.

Learned dictionary. Figure 7.1 depicts a learned dictionary. Although the atoms are not easily interpretable, single notes seem to appear here and there.

[2] Available at `https://github.com/mdeff/dlaudio`.

[44] showed that dictionaries trained on individual octaves did discover harmonics, chords and harmonies without any prior about music theory.

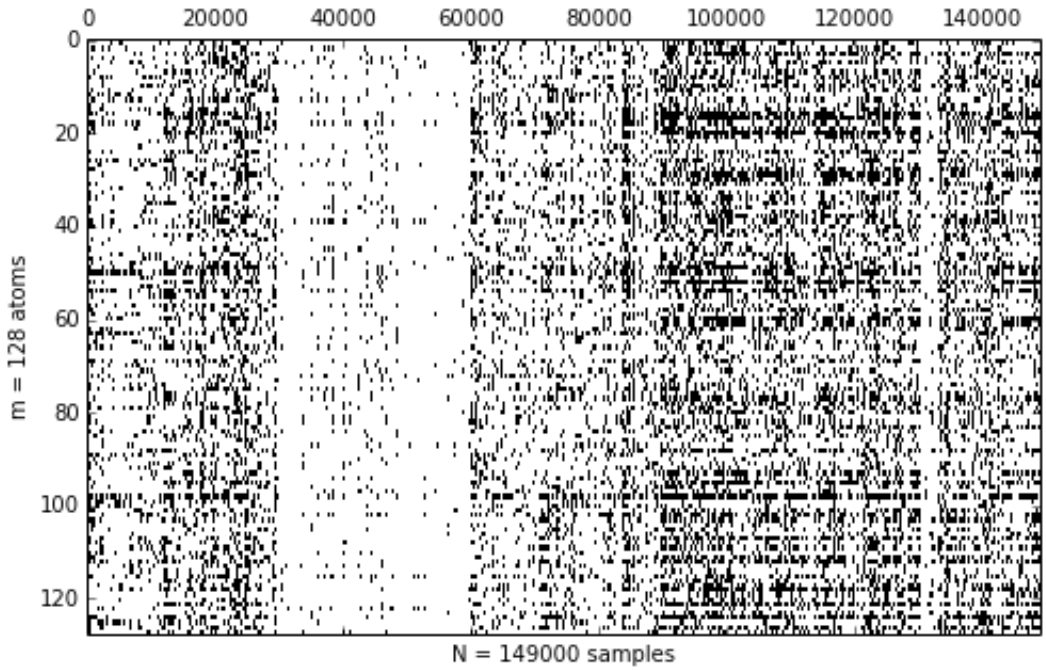


Figure 7.2: A learned sparse representation of spectrograms.

Sparse representations. Figure 7.2 shows the representations inferred by (2.24) after convergence. With only 19.8% of non-zero coefficients, the learned representations are indeed sparse.

Convergence. While the optimization always end in a different local minimum (Chapter 4), a pretty stable convergence behavior has been observed in all the experiments. Figure 7.3 shows the evolution of the various objectives in a typical convergence of the algorithm.

7.3 Classification performance

Table 7.1 shows the classification accuracy when using CQT spectrograms (the baseline), extracted features with $\lambda_g = 0$ (sparse auto-encoder) and $\lambda_g = 100$ (structured auto-encoder). The other hyper-parameters are set to $\lambda_z = 1$, $\lambda_d = 10$ and $\lambda_e = 0$.

The first conclusion to draw from this experiment is that the representation extracted from the spectrograms by the structured auto-encoder defined in

Part I is indeed more discriminative than the spectrograms themselves in a classification task.

The second conclusion we can draw is that the addition of the smooth prior on the data manifold (Section 2.5) always lead to an improved accuracy.

Finally, the sparse and structured representations are robust with respect to noisy data. While a sparse auto-encoder ($\lambda_g = 0$) performs even worse than the baseline, our structured auto-encoder ($\lambda_g = 100$) out-performs the baseline by 7%.

Noise level (standard deviation)	0.00	0.10	0.20
Accuracy (baseline) [%]	69.7	58.7	46.9
Accuracy ($\lambda_g = 0$) [%]	75.9	57.1	42.6
Accuracy ($\lambda_g = 100$) [%]	78.0	65.9	51.6

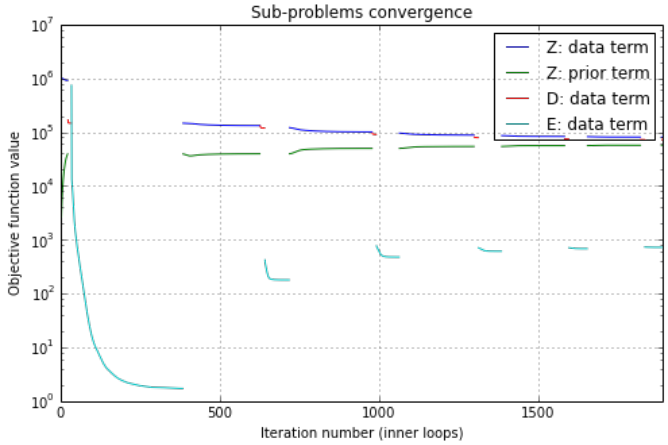
Table 7.1: Classification accuracies on a subset of GTZAN: $N_{genres} = 5$ genres, $N_{clips} = 100$ clips per genre and $N_{frames} = 149$ frames per clip.

7.4 Discussion

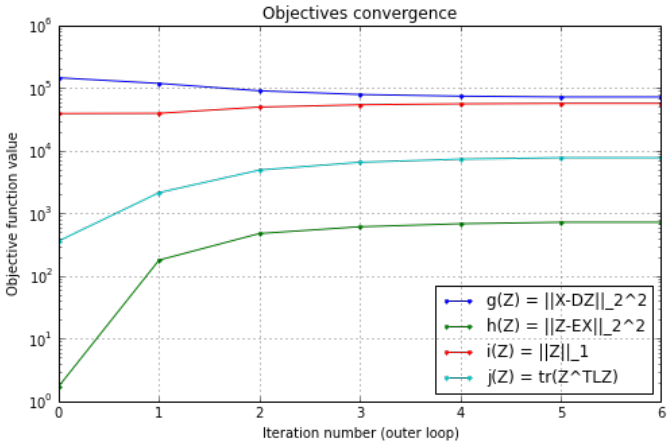
Our experiments demonstrated the usefulness of an important property of the proposed model: the conservation of the structure in the data via graph regularization. When compared to a sparse auto-encoder, it always leads to a better accuracy, especially in a noisy scenario, when the structure is the most helpful.

We experimentally confirmed two assumptions of our model (Section 2.1): the learned representations are sparse (Figure 7.2) and structured.

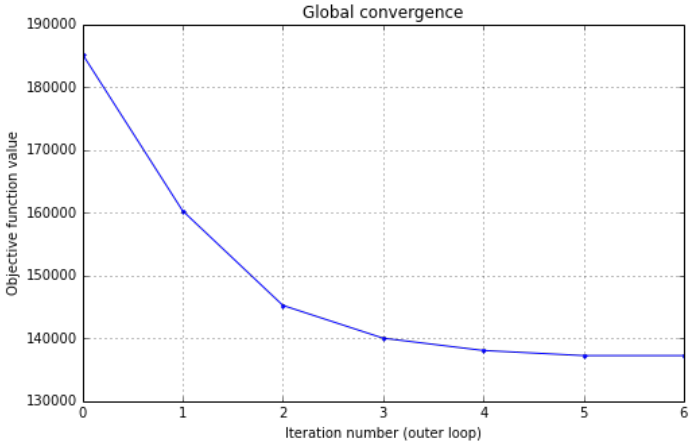
Higher classification accuracies are probably achievable by fine-tuning the hyper-parameters and introducing adding further complexity; e.g. by applying a LCN to the CQT spectrograms or working on individual octaves, two techniques used by [34]. We may even further improve the performance of our model by creating better graphs, i.e. graphs more suited to the problem at hand, for example by tuning the construction hyper-parameters.



(a) Sub-problem objectives $f_2(\mathbf{Z})$, $f_1(\mathbf{Z})$, $f_2(\mathbf{D})$ and $f_2(\mathbf{E})$ evaluated at each inner iteration.



(b) Sub-objectives $f_d(\mathbf{Z}, \mathbf{D})$, $f_e(\mathbf{Z}, \mathbf{E})$, $f_z(\mathbf{Z})$ and $f_g(\mathbf{Z})$ evaluated at each outer iteration.



(c) Global objective $f_d(\mathbf{Z}, \mathbf{D}) + f_e(\mathbf{Z}, \mathbf{E}) + f_z(\mathbf{Z}) + f_g(\mathbf{Z})$ evaluated at each outer iteration.

Figure 7.3: Example of a typical convergence of the algorithm.

Conclusion

In this work we introduced a novel auto-encoder architecture, which main characteristic is to learn sparse and structured representation of input data, borrowing ideas from sparse coding and manifold learning. Precisely, it learns sparse representations that explicitly take into account the local manifold structure of the data. The experimental results on music genre recognition showed that our model extracted more discriminant features than sparse auto-encoders.

Bibliography

[1] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. *On the surprising behavior of distance metrics in high dimensional space*. Springer, 2001.

[2] Michal Aharon, Michael Elad, and Alfred Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *Signal Processing, IEEE Transactions on* 54.11 (2006), pp. 4311–4322.

[3] Chidanand Apté, Fred Damerau, and Sholom M Weiss. "Automated learning of decision rules for text categorization". In: *ACM Transactions on Information Systems (TOIS)* 12.3 (1994), pp. 233–251.

[4] Richard G Baraniuk et al. "Model-based compressive sensing". In: *Information Theory, IEEE Transactions on* 56.4 (2010), pp. 1982–2001.

[5] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202.

[6] Mikhail Belkin and Partha Niyogi. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering." In: *NIPS*. Vol. 14. 2001, pp. 585–591.

[7] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples". In: *The Journal of Machine Learning Research* 7 (2006), pp. 2399–2434.

[8] Yoshua Bengio. "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[9] Yoshua Bengio et al. "Greedy layer-wise training of deep networks". In: *Advances in neural information processing systems* 19 (2007), p. 153.

[10] James Bergstra et al. "Aggregate features and AdaBoost for music classification". In: *Machine learning* 65.2-3 (2006), pp. 473–484.

[11] Hervé Bourlard and Yves Kamp. "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological cybernetics* 59.4-5 (1988), pp. 291–294.

[12] Ron Bracewell. "The Fourier Transform and IIS Applications". In: *New York* (1965).

[13] Emmanuel J Candes and David L Donoho. "Recovering edges in ill-posed inverse problems: Optimality of curvelet frames". In: *Annals of statistics* (2002), pp. 784–842.

[14] Emmanuel J Candes and Terence Tao. "Decoding by linear programming". In: *Information Theory, IEEE Transactions on* 51.12 (2005), pp. 4203–4215.

[15] Antonin Chambolle and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145.

[16] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), p. 27.

[17] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. "Atomic decomposition by basis pursuit". In: *SIAM journal on scientific computing* 20.1 (1998), pp. 33–61.

[18] Fan RK Chung. *Spectral graph theory*. Vol. 92. American Mathematical Soc., 1997.

[19] Patrick L Combettes and Jean-Christophe Pesquet. "Proximal splitting methods in signal processing". In: *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[20] Patrick L Combettes and Valérie R Wajs. "Signal recovery by proximal forward-backward splitting". In: *Multiscale Modeling & Simulation* 4.4 (2005), pp. 1168–1200.

[21] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[22] Alexandre d'Aspremont et al. "A direct formulation for sparse PCA using semidefinite programming". In: *SIAM review* 49.3 (2007), pp. 434–448.

[23] Pedro Domingos. "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[24] Minh N Do and Martin Vetterli. "Framing pyramids". In: *Signal Processing, IEEE Transactions on* 51.9 (2003), pp. 2329–2342.

[25] David L Donoho. "Compressed sensing". In: *Information Theory, IEEE Transactions on* 52.4 (2006), pp. 1289–1306.

[26] David L Donoho and Michael Elad. "Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization". In: *Proceedings of the National Academy of Sciences* 100.5 (2003), pp. 2197–2202.

[27] Rong-En Fan et al. "LIBLINEAR: A library for large linear classification". In: *The Journal of Machine Learning Research* 9 (2008), pp. 1871–1874.

[28] Dennis Gabor. "Theory of communication. Part 1: The analysis of information". In: *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering* 93.26 (1946), pp. 429–441.

[29] Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. "Learning by transduction". In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1998, pp. 148–155.

[30] Leo J Grady and Jonathan Polimeni. *Discrete calculus: Applied analysis on graphs for computational science*. Springer Science & Business Media, 2010.

[31] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 399–406.

[32] Raia Hadsell, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping". In: *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[33] Philippe Hamel and Douglas Eck. "Learning Features from Music Audio with Deep Belief Networks." In: *ISMIR*. Utrecht, The Netherlands. 2010, pp. 339–344.

[34] Mikael Henaff et al. "Unsupervised learning of sparse features for scalable audio classification." In: *ISMIR*. Vol. 11. 2011, p. 276.

[35] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[36] Geoffrey E Hinton and Richard S Zemel. "Autoencoders, minimum description length, and Helmholtz free energy". In: *Advances in neural information processing systems* (1994), pp. 3–3.

[37] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[38] Sepp Hochreiter et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.

[39] Patrik O Hoyer. "Non-negative matrix factorization with sparseness constraints". In: *The Journal of Machine Learning Research* 5 (2004), pp. 1457–1469.

[40] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. "Learning with structured sparsity". In: *The Journal of Machine Learning Research* 12 (2011), pp. 3371–3412.

[41] ISO. *Acoustics – Standard tuning frequency (Standard musical pitch).* ISO 16:1975. Geneva, Switzerland: International Organization for Standardization, 1975.

[42] Nathalie Japkowicz, Stephen Jose Hanson, Mark Gluck, et al. "Nonlinear autoassociation is not equivalent to PCA". In: *Neural computation* 12.3 (2000), pp. 531–545.

[43] Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. "Structured variable selection with sparsity-inducing norms". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2777–2824.

[44] Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. "Fast inference in sparse coding algorithms with applications to object recognition". In: *arXiv preprint arXiv:1010.3467* (2010).

[45] Teuvo Kohonen. "Self-organized formation of topologically correct feature maps". In: *Biological cybernetics* 43.1 (1982), pp. 59–69.

[46] Matthieu Kowalski. "Sparse regression using mixed norms". In: *Applied and Computational Harmonic Analysis* 27.3 (2009), pp. 303–324.

[47] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. "Convolutional networks and applications in vision". In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on.* IEEE. 2010, pp. 253–256.

[48] Honglak Lee et al. "Efficient sparse coding algorithms". In: *Advances in neural information processing systems.* 2006, pp. 801–808.

[49] Michael Lewicki and Terrence Sejnowski. "Learning overcomplete representations". In: *Neural computation* 12.2 (2000), pp. 337–365.

[50] Michael S Lewicki and Terrence J Sejnowski. "Learning overcomplete representations". In: *Neural computation* 12.2 (2000), pp. 337–365.

[51] Yingying Li and Stanley Osher. "Coordinate descent optimization for l1 minimization with application to compressed sensing; a greedy algorithm". In: *Inverse Problems and Imaging* 3.3 (2009), pp. 487–503.

[52] David G Lowe. "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on.* Vol. 2. Ieee. 1999, pp. 1150–1157.

[53] Siwei Lyu and Eero P Simoncelli. "Nonlinear image representation using divisive normalization". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[54] Julien Mairal, Michael Elad, and Guillermo Sapiro. "Sparse representation for color image restoration". In: *Image Processing, IEEE Transactions on* 17.1 (2008), pp. 53–69.

[55] Stéphane Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

[56] Stéphane G Mallat and Zhifeng Zhang. "Matching pursuits with time-frequency dictionaries". In: *Signal Processing, IEEE Transactions on* 41.12 (1993), pp. 3397–3415.

[57] Jean-Jacques Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *CR Acad. Sci. Paris Sér. A Math* 255 (1962), pp. 2897–2899.

[58] Marius Muja and David G. Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration". In: *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[59] Balas Kausik Natarajan. "Sparse approximate solutions to linear systems". In: *SIAM journal on computing* 24.2 (1995), pp. 227–234.

[60] Arkadi Semenovich Nemirovsky and David Borisovich Yudin. "Problem complexity and method efficiency in optimization." In: (1983).

[61] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 2 (2002), pp. 849–856.

[62] Bruno A Olshausen et al. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), pp. 607–609.

[63] Bruno A Olshausen. "Sparse coding of time-varying natural images". In: *Proc. of the Int. Conf. on Independent Component Analysis and Blind Source Separation*. Citeseer. 2000, pp. 603–608.

[64] Bruno A Olshausen and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[65] Stanley Osher and Leonid I Rudin. "Feature-oriented image enhancement using shock filters". In: *SIAM Journal on Numerical Analysis* 27.4 (1990), pp. 919–940.

[66] Trevor Park and George Casella. "The bayesian lasso". In: *Journal of the American Statistical Association* 103.482 (2008), pp. 681–686.

[67] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[68] Nicolas Pinto, David D Cox, and James J DiCarlo. "Why is real-world visual object recognition hard?" In: *PLoS computational biology* 4.1 (2008), e27.

[69] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. "Efficient learning of sparse representations with an energy-based model". In: *Advances in neural information processing systems*. 2006, pp. 1137–1144.

[70] Marc Aurelio Ranzato et al. "Unsupervised learning of invariant feature hierarchies with applications to object recognition". In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[71] Jason Tyler Rolfe and Yann LeCun. "Discriminative recurrent sparse auto-encoders". In: *arXiv preprint arXiv:1301.3775* (2013).

[72] Sam T Roweis and Lawrence K Saul. "Nonlinear dimensionality reduction by locally linear embedding". In: *Science* 290.5500 (2000), pp. 2323–2326.

[73] Christian Schörkhuber and Anssi Klapuri. "Constant-Q transform toolbox for music processing". In: *7th Sound and Music Computing Conference, Barcelona, Spain*. 2010, pp. 3–64.

[74] David Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *Signal Processing Magazine, IEEE* 30.3 (2013), pp. 83–98.

[75] Evan C Smith and Michael S Lewicki. "Efficient auditory coding". In: *Nature* 439.7079 (2006), pp. 978–982.

[76] Bob L Sturm. "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*. ACM. 2012, pp. 7–12.

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

[78] Bob L Sturm. "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use". In: *arXiv preprint arXiv:1306.1461* (2013).

[79] Joshua B Tenenbaum, Vin De Silva, and John C Langford. "A global geometric framework for nonlinear dimensionality reduction". In: *Science* 290.5500 (2000), pp. 2319–2323.

[80] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[81] Andrey Tikhonov. "Solution of incorrectly formulated problems and the regularization method". In: *Soviet Math. Dokl.* Vol. 5. 1963, pp. 1035–1038.

[82] George Tzanetakis and Perry Cook. "Musical genre classification of audio signals". In: *Speech and Audio Processing, IEEE transactions on* 10.5 (2002), pp. 293–302.

[83] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103.

[84] Curtis R Vogel. *Computational methods for inverse problems*. Vol. 23. Siam, 2002.

[85] Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet. "Consistency of spectral clustering". In: *The Annals of Statistics* (2008), pp. 555–586.

[86] Robert W Young. "Terminology for logarithmic frequency units". In: *The Journal of the Acoustical Society of America* 11.1 (1939), pp. 134–139.

[87] Lihi Zelnik-Manor and Pietro Perona. "Self-tuning spectral clustering". In: *Advances in neural information processing systems*. 2004, pp. 1601–1608.

[88] Miao Zheng et al. "Graph regularized sparse coding for image representation". In: *Image Processing, IEEE Transactions on* 20.5 (2011), pp. 1327–1336.

[89] Dengyong Zhou et al. "Learning with local and global consistency". In: *Advances in neural information processing systems* 16.16 (2004), pp. 321–328.

[90] Martin Zinkevich. "Online convex programming and generalized infinitesimal gradient ascent". In: (2003).

[91] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.