MODIFYING SPARSE CODING TO MODEL IMBALANCED DATASETS

A Dissertation
Presented to
The Academic Faculty

By

Bradley M. Whitaker

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2018

MODIFYING SPARSE CODING TO MODEL IMBALANCED DATASETS

Approved by:

Dr. David V. Anderson, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Christopher J. Rozell
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Justin K. Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Wing Li
School of Mathematics
Georgia Institute of Technology

Dr. Gari D. Clifford
Department of Biomedical Informatics
Emory University

Date Approved: April 6, 2018

What appears today to be a sacrifice will prove instead
to be the greatest investment that you will ever make.

Gordon B. Hinckley

For Cody and Brooke, who were with me from day one,

and for Robyn and Adam, who joined me along the way.

ACKNOWLEDGEMENTS

I must admit that while I am grateful for the monetary and computational support offered by the National Science Foundation[1] and Georgia Tech's PACE computing cluster[2], this research was truly made possibly by people.

My advisor, Dr. David Anderson, has been an excellent mentor and tutor throughout my graduate education. As a token of my appreciation, I give you the great distinction of being the only individual who gets his own paragraph in this Acknowledgements section.

I appreciate the contributions of my committee members: Drs. Chris Rozell, Justin Romberg, Gari Clifford, and Wing Li. Their comments and insights helped point my thesis in the right direction and made it a stronger final product.

In my time at Georgia Tech, I have had the opportunity to collaborate with many other researchers: Brandon Carroll, Pradyumna Suresha, Dr. Chengyu Liu, Dr. Muhammed Rizwan, Burak Aydemir, Lee Richert, Dr. Kaitlin Fair, and Nathan Parrish. I am grateful to them and other members of the ESP lab for emotional and academic support.

An unexpected part of my Ph.D. program involved moving to France to teach at Georgia Tech—Lorraine. I appreciate the work done by GTL faculty and staff—most notably Dr. Bertrand Boussert, Dr. Paul Voss, Alina Opreanu, and Nicolas Jacquet—to make that a successful experience.

Last, but certainly not least, I express appreciation to my family for supporting me on this adventure. My children, Brooke, Robyn, and Adam, have provided constant joy and motivation throughout my graduate studies. And I thank my wife, Cody, for her willingness to temporarily survive on a student stipend so I could grow up to be a professor. We did it!

[1]This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1650044.

[2]This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this research is to explore the use of sparse coding as a tool for unsupervised feature learning to more effectively model imbalanced datasets. Traditional sparse coding dictionaries are learned by minimizing the average approximation error between a vector and its sparse decomposition. As such, these dictionaries may overlook important features that occur infrequently in the data. Without these features, it may be difficult to accurately classify between classes if one or more classes are not well-represented in the training data. To overcome this problem, this work explores novel modifications to the sparse coding dictionary learning framework that encourage dictionaries to learn anomalous features. Sparse coding also inherently assumes that a vector can be represented as a sparse linear combination of a feature set. This work addresses the ability of sparse coding to learn a representative dictionary when the underlying data has a nonlinear sparse structure. Finally, this work illustrates one benefit of improved signal modeling by utilizing sparse coding in three imbalanced classification tasks.

CHAPTER 1
INTRODUCTION

1.1 Motivation

While humans may always play a critical role making the ultimate decision, machine learning tools can help us overcome our natural limitations by analyzing large datasets and extracting relevant information. In order for machine learning algorithms to be successful, they must be trained using all types of data they expect to see [1]. This can be a problem when attempting to diagnose rare diseases, detect fraudulent financial activities, or identify defects in factory-produced items. These applications are highly imbalanced by nature, and it is difficult for machine learning to model the rare events. For example, fraudulent transactions at an online banking website occur much less frequently than legitimate activity, so fewer examples are available to train the algorithms.

Machine learning is not the panacea for decision making; it comes with its own host of problems. Supervised machine learning algorithms require labeled training data that is often not readily available [2]. When the data is available, it may have an extremely large dimension that must be reduced in order to provide a practical solution [3]. Other preprocessing steps, such as noise reduction or feature extraction, may dramatically affect the performance of a given algorithm. Finally, traditional machine learning algorithms are most effective when trained on and applied to balanced classes [4], but as mentioned previously, many classification problems are imbalanced. If the goal of an algorithm is to reduce the mean squared error, the optimal approach may be to simply ignore the contributions of a few anomalous data points.

Recent research has looked at how to learn accurate models from imbalanced datasets [5, 6, 7]. This dissertation adds to this area of research, focusing on the use and modifi-

cation of a method called sparse coding. In this work, I present sparse coding dictionary learning as a form of unsupervised feature learning. I then introduce intuitive modifications to dictionary learning and analyze the ability of sparse coding (with and without these changes) to accurately model imbalanced datasets. Finally, I use sparse coding as a feature extraction tool for use in imbalanced classification tasks.

1.2 Contributions

In my analysis of sparse coding, I address the following three questions.

1. How well can sparse coding model known features of a dataset, especially features that occur infrequently?

2. To what extent can sparse coding model features when they are combined nonlinearly?

3. Can the model learned from sparse coding be used to successfully classify imbalanced datasets?

The first two questions explore the ability of sparse coding to learn the structure of a dataset, while the third investigates the potential to use sparse coding for feature extraction in practical machine learning problems.

Chapter 4 addresses the first question: learning sparse coding dictionaries from imbalanced datasets. Sparse coding has previously been used as a feature extraction tool to detect anomalies in image and video datasets [8, 9, 10, 11, 12]. Current methods focus on detection based on the abnormally large reconstruction errors associated with the minority class. Rather than attempting to model the abnormalities, they rely on the algorithm *failing* to model them. Yet one of the benefits of sparse coding is that it can learn features from a dataset that have intuitive meanings in the original signal domain [13]. The work presented in this document discusses how modifications to sparse coding allow the algorithm to learn a dictionary that directly models the minority class.

Chapter 5 investigates the second question by discussing the ability of sparse coding to model a dataset after performing nonlinear processing on the data. Sparse coding is inherently a linear decomposition, yet it is often used on nonlinearly-processed data. For example, in many audio applications sparse coding is performed in the magnitude spectrum or cepstral domains [14, 15, 16]. Even though the linear assumption of the models is not met, empirical results using sparse coding (and similar linear matrix decomposition techniques such as nonnegative matrix factorization) show that these techniques can successfully be applied after nonlinear preprocessing [17, 18]. The purpose of this work is to determine the importance of the linearity assumption when applying sparse coding to a nonlinear dataset.

The third question is more practical in nature and is addressed in Chapter 6. As mentioned previously, sparse coding has already been used in a wide variety of classification applications. In this work, I apply sparse coding to three different problems: detecting respiratory diseases from chicken vocalizations, detecting the presence of cardiac pathologies from human phonocardiograms (PCGs), and detecting atrial fibrillation from human electrocardiograms (ECGs). These problems are all active research areas, and finding effective solutions has the potential to improve welfare and save money [19, 20, 21]. The focus of my work in this area is to illustrate that learning an appropriate model of the data through sparse coding can be beneficial in an imbalanced classification setting.

CHAPTER 2

BACKGROUND

2.1 Imbalanced Data

Machine learning algorithms make two implicit assumptions that make it difficult to deal with imbalanced datasets [22]. The first is that the end goal is to maximize accuracy. The second is that the training data and the unseen test data come from the same distribution. When the training data is very imbalanced, the algorithms often converge to a simple, but uninformative solution: classify everything as coming from the majority class. Many common problems are inherently imbalanced, and for these datasets the first assumption is often not true [4]. The goal may not be to maximize accuracy, but to achieve a desired balance between missed detections and false alarms.

One way to address imbalance is to modify the dataset directly [23]. Dataset imbalance can be removed artificially by excluding examples from the majority class or augmenting the dataset with synthetic examples from the minority class. Changing the distribution, however, can often result in a model that overfits the training data and does not generalize to unseen test data. In [24], the authors use a sparse dictionary in order to create such synthetic examples in a way that attempts to reduce over-fitting.

Another way to deal with class imbalances involves adjusting the classifier. Traditional machine learning algorithms, such as Support Vector Machines (SVMs), decision trees, and neural networks, have modifications that allow them to take class imbalance into account [25, 26, 27, 28]. Other methods involve using ensembles of classifiers, each trained on more-balanced subsets of the data [29].

Neither of these solutions, however, addresses feature extraction. Feature extraction is an important preprocessing step in many machine learning applications. If done properly,

it can improve classification without requiring changes to the distribution of the data or the hyperparameters of a classifier [30]. This approach addresses the problem of dataset imbalance through the lens of modeling.

2.2 Feature Extraction

The effectiveness of many machine learning algorithms varies dramatically based on what features of the training data are used [1]. Features may need to be extracted in order to reduce the dimensionality of the data [3] or to remove some irrelevant information [31]. The chosen features can be application-specific [31]. For example, Mel-frequency cepstral coefficients (MFCCs) are often used in speech recognition or other audio classification tasks [32, 33, 34].

Features can also be learned from the training data [35]. One common method of feature extraction is Principal Component Analysis (PCA) [36, 37], which finds a subspace that best explains the variance of the data. While PCA has been around for over a century, researchers are still using it (and novel variations) as a foundational tool for data modeling [38, 39, 40]. Autoencoders are another method for extracting features in which a neural network is trained with the input layer matching the output layer [41, 42]. When the hidden layer is smaller than the input layer, the autoencoder learns efficient representations of the training data.

In the context of machine learning, sparse coding can also be thought of as an unsupervised feature extraction tool. Sparse coding attempts to model a vector as a sparse linear combination of a 'dictionary,' or set of features. One of the benefits of using sparse coding is that it has been shown to extract biologically interpretable features from audio and visual data [13, 43]. Transforming data to a sparse domain prior to performing classification has been explored in many different applications, including deep belief networks [44], computer vision [45], object recognition [46], face recognition [47], and acoustic scene classification [48]. It has also been used effectively in anomaly detection, a problem

closely related to imbalanced data classification [49]. Feature extraction via sparse coding has been used to detect anomalies in images [8, 9] and video [10, 11, 12].

2.3 Sparse Coding

In 1996, Olshausen and Field discovered that learning a sparse coding dictionary on natural images resulted in features similar to those found in the mammalian primary visual cortex [13]. Expanding on this work, they described sparse coding in a probabilistic framework and outlined an iterative algorithm for learning overcomplete dictionaries [50]. Their work, relating sparse coding to neurobiology, sparked an interest in sparse representations. Since then, researchers have applied sparse coding to many different domains of signal processing, including regression [51], image processing [52, 53, 54], classification [55, 56], and object recognition [57, 46].

The basic premise of sparse coding is that a vector can be written in the form

$$x = \mathbf{D}\alpha, \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the original vector, $\mathbf{D}$ is a dictionary matrix whose columns are feature vectors, and $\alpha \in \mathbb{R}^m$ is a *sparse* coefficient vector. The value of m is a user-defined parameter that represents the size of the dictionary. When $m > n$ the dictionary is overcomplete. A large value for m will give the dictionary more flexibility to model the input data, with the sparsity requirement providing the needed structure. When $m < n$, the dictionary reduces the dimensionality of the problem in addition to modeling sparse behavior.

Finding the sparse coefficient vector amounts to solving the following problem:

$$\arg\min_{\alpha} \|\alpha\|_0 \qquad \text{subject to} \qquad x = \mathbf{D}\alpha. \tag{2.2}$$

Despite its simple formulation, the fact that the $\|\cdot\|_0$ counting operator is not a mathematical norm causes this problem to be NP-hard [58]. There are two general methods used to

circumvent this difficulty. Incredibly, either method can be used to obtain *exactly* the same solution that would result from solving the NP-hard problem (under certain conditions) [59, 60, 61].

2.3.1 Greedy (ℓ_0) Algorithms

The first way to avoid the NP-hardness of Equation (2.2) is to solve it using greedy methods. In 1993, Mallat introduced a technique called Matching Pursuit [62]. In this iterative algorithm, coefficients are selected one at a time in order to reduce the reconstruction error as much as possible. A common variant, Orthogonal Matching Pursuit (OMP), adds a new coefficient and updates all previously-chosen coefficients in each iteration [63]. Stagewise OMP allows for multiple coefficients to introduced in a single iteration [64]. One important characteristic of these greedy algorithms is that they are very fast.

In addition to being fast, these methods are accurate at solving the sparse recovery problem presented in Equation (2.2). In 2004, Tropp presented conditions under which OMP exactly solves this NP-hard problem. Additionally, he proved that for *any* input signal, the solution to OMP has an approximation error that is not much higher than the minimal approximation error (when using the same number of non-zero terms). His conclusion: "greed is good" [59].

2.3.2 Relaxed (ℓ_1) Algorithms

Another workaround to the NP-hard sparse recovery problem is to replace the $\|\cdot\|_0$ operator with the ℓ_1-norm. Doing so transforms the problem into the domain of convex optimization. The updated problem, now a linear program, is called Basis Pursuit [65]:

$$\arg\min_{\alpha} \|\alpha\|_1 \qquad \text{subject to} \qquad x = \mathbf{D}\alpha. \tag{2.3}$$

Basis Pursuit Denoising (BPDN), takes into account both measurement noise (in x) and model noise (in $\mathbf{D}$) [65]:

$$\arg\min_{\alpha} \|x - \mathbf{D}\alpha\|_2^2 + \lambda \|\alpha\|_1 \tag{2.4}$$

Another formulation of BPDN, sometimes called Quadratically Constrained Basis Pursuit (QCBP), and LASSO are also worth mentioning. QCBP is formulated as

$$\arg\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad \|x - \mathbf{D}\alpha\|_2^2 \leq \delta \tag{2.5}$$

and LASSO is formulated as

$$\arg\min_{\alpha} \|x - \mathbf{D}\alpha\|_2^2 \quad \text{subject to} \quad \|\alpha\|_1 \leq \tau. \tag{2.6}$$

It is important to note that Equations (2.4)–(2.6) are equivalent up to a tradeoff design parameter (λ, δ, and τ, respectively), although the exact values of the parameters that determine equivalence cannot be known a priori [66]. Donoho proved that for sufficiently sparse α, the solution to Basis Pursuit is unique and equal to the solution of Equation (2.2) for almost all $\mathbf{D}$ [60]. He also showed that the noise-aware variants result in robust solutions [61].

Solving large ℓ_1-based problems cannot be done as quickly as the greedy ℓ_0-based methods. However, even problems with millions of features and examples can be solved in a few minutes on a PC [67]. The main benefit of the convex relaxation of sparse recovery the intuition and flexibility that come with convex programs.

2.4 Dictionary Learning

The algorithms discussed previously assume that the dictionary is known and attempt to find the sparsest linear combination of dictionary elements to represent a non-sparse data

vector. In some situations, however, the dictionary is not known a priori. The problem of finding a representative dictionary along with the sparsifying coefficients is called dictionary learning [68, 69].

Many dictionary learning algorithms have been presented, including the Method of Optimal Directions (MOD) [70], the k-SVD algorithm [71, 72], and the Alternating Minimization[1] algorithm [50, 54]. The general strategy of all methods is to alternate between (1) fixing the dictionary and learning the coefficients, and (2) holding the coefficients constant while updating the dictionary. The work discussed in this document makes adjustments within the convex optimization framework provided by the Alternating Minimization algorithm. As such, the greedy k-SVD and MOD algorithms will not be addressed in depth.

2.4.1 Alternating Minimization

Olshausen and Field viewed dictionary learning through the lens of maximum likelihood probabilities and introduced two key assumptions to simplify the learning algorithm: they assumed that each coefficient in a sparse vector is independent of the others and they assumed that the reconstruction error could be modeled as zero-mean Gaussian noise [50, 69]. Under these assumptions and after simplifications, they formulated the dictionary learning problem as follows:

$$\underset{\mathbf{D}\in\mathcal{C},\{\alpha_m\}_{m=1\ldots\mathrm{M}}}{\arg\min} \frac{1}{\mathrm{M}} \sum_{m=1}^{\mathrm{M}} \frac{1}{2} \left\|x_m - \mathbf{D}\alpha_m\right\|_2^2 + \lambda \left\|\alpha_m\right\|_1 , \tag{2.7}$$

where each x_m is a different training sample and $\mathcal{C}$ is the set of all matrices with column-norms less than 1. (The constraint $\mathbf{D} \in \mathcal{C}$ prevents the dictionary from growing arbitrarily large. This would allow the coefficients to be arbitrarily small, effectively removing the ℓ_1 term from the problem.)

[1] Olshausen and Field used the term "Sparse Coding" to refer to the dictionary-learning process as well as to the process of learning the sparse coefficient vectors given a fixed dictionary [50]. Possibly because of this ambiguity, their method does not appear to have a canonical name in the literature. In this document, I will refer to their method as the "Alternating Minimization" algorithm. It is outlined by Mairal in Algorithm 10 of [54].

While Equation (2.7) is formulated very nicely, it is still non-convex because of the joint optimization over $\mathbf{D}$ and the α's. The Alternating Minimization algorithm introduces convexity by alternating between learning the sparse coefficients and updating the dictionary while holding the other fixed. The coefficient learning step can be viewed as solving many instances of Quadratically Constrained Basis Pursuit, and can be done in parallel. Since the ℓ_1-term is not dependent on $\mathbf{D}$, the dictionary update step is simply a matrix-norm minimization.

2.4.2 Dictionary Learning Modifications

Many modifications have been made to sparse coding and dictionary learning over the past 20 years, including non-negative sparse coding and dictionary learning [73, 74], online dictionary learning [75, 76, 77], and supervised dictionary learning [78]. In addition, Nguyen explores kernel dictionary learning [79], where the input data undergoes a nonlinear transformation, and the dictionary is multiplied by a kernel matrix in both the sparse coding step and the dictionary update step of dictionary learning.

Recent work has also explored nonlinear sparse coding. Xie asserts that conventional linear sparse coding is a Euclidean special case of a more general nonlinear framework based on Riemannian manifolds [80]. Bornschein explores one particular nonlinear modification to sparse coding [81]. In his model, he uses a (nonlinear) pointwise max function to combine a sparse number of dictionary elements to recreate the original vector. He shows that minimizing the reconstruction error while combining the dictionary elements in this manner results in a higher percentage of globular basis functions, consistent with the globular receptive fields found experimentally in the visual cortex [82]. Shelton extends this work and explains that this nonlinear reconstruction is good for modeling occlusions and learns interpretable dictionary elements, particularly in image-based applications [83]. Others have also looked at this nonlinear 'spike-and-slab' model of sparse coding [84, 85, 86].

2.5 Contributions

The research presented in this dissertation extends the field of sparse coding in the direction of dictionary learning modifications. The modifications I introduce are designed to encourage sparse coding to model highly imbalanced datasets. In particular, I modify dictionary learning by adjusting the dictionary update step to include a frozen component or to utilize mixed-matrix norms [87, 88, 89, 90]. I also explore the ability of linear sparse coding to model nonlinearly preprocessed data, and show that in some instances linear sparse coding is very effective [91]. Finally, I apply sparse coding to classification problems related to disease detection in both humans and chickens [88, 92, 93, 94, 95]

CHAPTER 3

SPARSE CODING MODIFICATIONS

This chapter describes the Alternating Minimization algorithm for learning sparse coding dictionaries. I choose to discuss this particular sparse coding dictionary learning algorithm in order to take advantage of the convex nature of the dictionary update. In addition, I present modifications of the Alternating Minimization Algorithm that will be used throughout this thesis. These modifications are intended to encourage the dictionary to learn features that occur infrequently in the training data, allowing the dictionary to learn a better model of an imbalanced dataset.

3.1 Alternating Minimization

The Alternating Minimization algorithm is described in Algorithm 1 [50, 54]. (Recall that $\mathcal{C}$ is the set of matrices with column-norms less than 1.)

Algorithm 1 Alternating Minimization

Input: Signals $\{x_m \in \mathbb{R}^{\mathrm{N}}\}_{m=1,...,\mathrm{M}}$, initial dictionary $\mathbf{D}_0 \in \mathcal{C}$, regularization term λ, number of iterations K.

Initialize $\mathbf{D} \leftarrow \mathbf{D}_0$
for $k = 1, ..., \mathrm{K}$ **do**
  for $m \in \{1, ..., \mathrm{M}\}$ (*in parallel*) **do**
    Calculate coefficient vectors:
    $$\alpha_m = \arg\min_{\alpha} \frac{1}{2} \|x_m - \mathbf{D}\alpha\|_2^2 + \lambda \|\alpha\|_1$$
  end for
  Update dictionary:
  $$\mathbf{D} = \arg\min_{\mathbf{D} \in \mathcal{C}} \frac{1}{\mathrm{M}} \sum_{m=1}^{\mathrm{M}} \frac{1}{2} \|x_m - \mathbf{D}\alpha_m\|_2^2$$
end for
return $\mathbf{D}$

The algorithm can be implemented to run in batch mode. In batch mode, the `for` loop

in line 3 will iterate over a random subset of the M training vectors and only the corresponding coefficient vectors will be used in the dictionary update step. Another implementation option is to run a few iterations of gradient descent on the minimization problem in line 6 rather than solving the optimization program [74]. Both of these implementations make each iteration less computationally expensive, but cause more iterations to be required before convergence.

3.2 Frozen Alternating Minimization

In [88] I helped develop the frozen dictionary approach as a method to help the learned dictionary model the difference between normal and abnormal data. My contributions to the work were in developing and implementing the Frozen Alternating Minimization algorithm. My colleague, B. T. Carroll, focused his efforts on implementing the frozen K-SVD algorithm. The approach was inspired by Smaragdis' work in non-negative matrix factorization [96, 97], and follows three main steps:

1. Learn a dictionary using the Alternating Minimization algorithm on data with no anomalies present.

2. Freeze this dictionary and augment the dictionary with new, non-frozen elements.

3. Run the algorithm again on data with anomalies present, while holding the frozen elements constant.

The Frozen Alternating Minimization algorithm itself is defined in Algorithm 2. Note that this algorithm requires a knowledge about whether or not anomalies are present in different datasets. This is known in the literature as 'weakly labeled data' [98]. The easily-obtained 'weak' labels provide information about the presence or absence of anomalies, but do not specify any more details.

Algorithm 2 Frozen Alternating Minimization

Input: Signals $\{x_m \in \mathbb{R}^{\mathrm{N}}\}_{m=1,...,\mathrm{M}}$, initial frozen dictionary $\mathbf{D}_f$, initial unfrozen dictionary $\mathbf{D}_u$, regularization term λ, number of iterations K

Initialize $\mathbf{D} \leftarrow [\mathbf{D}_f|\mathbf{D}_u]$
for $k = 1, ..., \mathrm{K}$ **do**
  for $m = 1, ..., \mathrm{M}$ (*in parallel*) **do**
    Calculate coefficient vectors:
    $\alpha_m = \arg\min_{\alpha} \frac{1}{2} \|x_m - \mathbf{D}\alpha\|_2^2 + \lambda \|\alpha\|_1$
  end for
  Update dictionary (unfrozen elements only):
  $\mathbf{D}_u = \arg\min_{\mathbf{D}_u \in \mathcal{C}} \frac{1}{\mathrm{M}} \sum_{m=1}^{\mathrm{M}} \frac{1}{2} \|x_m - [\mathbf{D}_f|\mathbf{D}_u]\, \alpha_m\|_2^2$
  $\mathbf{D} = [\mathbf{D}_f|\mathbf{D}_u]$
end for
return $\mathbf{D}$

3.3 Matrix Norm Modification

One problem when applying the Alternating Minimization algorithm to anomaly detection is that during the dictionary update, the algorithm chooses a new dictionary that minimizes the average reconstruction error of each data vector. Since anomalies occur infrequently in the data, they may not carry enough weight to influence the average reconstruction error. Therefore, the anomalous structure of these points may be lost in sparse coding.

In considering this problem, I discovered a way to modify the Alternate Minimization algorithm in such a way that anomalies are not lost [87]. The discovery makes use of the mixed-matrix $\ell_{1,\infty}$ norm[1]. For an arbitrary matrix, $\mathbf{A} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$, this norm is defined as follows:

$$\|\mathbf{A}\|_{1,\infty} = \max_{1\le n\le \mathrm{N}} \sum_{m=1}^{\mathrm{M}} |\mathbf{A}_{m,n}| \tag{3.1}$$

In words, the $\ell_{1,\infty}$ norm finds the largest absolute column sum of the matrix.

[1]Note that in [87], this norm is called the matrix 1-norm and the mathematical notation is presented as $\|\cdot\|_1$. This notation could confuse it with the *vector* ℓ_1 norm. To avoid this ambiguity, and to more fully emphasize the fact that the matrix norm calculates the maximum absolute column sum, I refer to the norm as the mixed-matrix $\ell_{1,\infty}$ norm throughout this dissertation.

The Alternating Minimization algorithm can be modified to find anomalous basis vectors by replacing the dictionary update in Step 6 of Algorithm 1 with the following equation:

$$\mathbf{D} = \arg\min_{\mathbf{D}\in\mathcal{C}} \|\mathbf{X} - \mathbf{DA}\|_{1,\infty}, \tag{3.2}$$

where each sample point, x_m, forms a column of $\mathbf{X}$ and each coefficient vector, α_m, forms a column of $\mathbf{A}$. Each column of the minimized matrix, $\mathcal{E} = \mathbf{X} - \mathbf{DA}$, represents the reconstruction error associated with one data vector. Therefore, in minimizing the $\ell_{1,\infty}$ norm we are encouraging the dictionary to explain every data point, regardless of how infrequently it occurs.

There is one important consequence to minimizing the maximum error in this manner. The $\ell_{1,\infty}$ method cannot distinguish between data points that come from the minority class, data points that are noisy, or data points that are outliers. Thus, the dictionary may learn to model noise and outliers in addition to modeling anomalous features associated with the minority class. Future research will investigate algorithms that can learn to model anomalies while still being robust to noisy data samples.

3.4 Combining the Euclidean Norm and the Matrix Norm

When I first introduced the $\ell_{1,\infty}$ method, I showed that it performs better at recovering very rare anomalies [87]. However, the standard Alternating Minimization algorithm was faster and more robust to noise. With this in mind, I developed two combinations of the Euclidean norm method and the matrix norm method to make the overall algorithm more efficient, robust, and effective.

3.4.1 Cascading

When learning a dictionary using the cascading method, a dictionary is first learned using the standard Alternating Minimization algorithm. This dictionary is used as the initial dictionary for the $\ell_{1,\infty}$ algorithm, which can be run for fewer iterations since the non-anomalous basis vectors have already been uncovered. However, this method has the potential of losing some already-learned basis vectors in the new dictionary update, especially in the presence of noise. Work published in [94] shows the effectiveness of using the cascading approach.

3.4.2 Freezing

I also incorporated the mixed-matrix $\ell_{1,\infty}$ norm into the frozen dictionary algorithm. As with the cascaded approach, the initial dictionary is learned using the standard Euclidean norm. Rather than updating this dictionary, it is frozen and augmented with additional elements. These non-frozen dictionary elements are then trained using the $\ell_{1,\infty}$ norm.

3.5 Algorithm Summary

At this point I have introduced five different dictionary-learning algorithms: the Alternating Minimization algorithm and four intuitive modifications. For brevity throughout the remainder of this document, I will refer to the Alternating Minimization algorithm and the Frozen Alternating Minimization algorithm as the 'ℓ_2' and 'frozen ℓ_2' methods, respectively, since they update the dictionary by minimizing the average ℓ_2-norms of the columns of the reconstruction error matrix. I will refer to the matrix norm algorithm as the '$\ell_{1,\infty}$' method. I will refer to the combined norm algorithms as the '$\ell_2 \rightarrow \ell_{1,\infty}$' and 'frozen $\ell_{1,\infty}$' methods.

3.6 Implementation Details

In my implementations of the algorithms, I use `l1_ls` to update the coefficients [67]. For the ℓ_2 dictionary update, I use several iterations of gradient descent to update the dictionary. While the $\ell_{1,\infty}$ norm is convex, it is not smooth, and therefore does not have a well-defined gradient. However, there is a similar method called 'block coordinate descent' that is based on the subgradient of the $\ell_{1,\infty}$ norm [99, 100]. In block coordinate descent, the columns of the error matrix with the largest ℓ_1 norms are identified and the dictionary is updated to reduce the ℓ_1 norms of those columns. Because block coordinate descent only requires simple operations, the bulk of the computational time is spent calculating the sparse coefficients rather than updating the dictionary. Learning an ℓ_2 dictionary using gradient descent and an $\ell_{1,\infty}$ dictionary using block coordinate descent takes about the same amount of computational time.

The work published in [87] and [94], as well as the work reported in Section 6.3 of this dissertation, was performed prior to my discovery of block coordinate descent. In these situations, the dictionary was updated using CVX, and each iteration of the dictionary learning algorithm took approximately ten times longer [101, 87].

When learning dictionaries on datasets from real applications (Chapter 6), I implemented the dictionary learning algorithm in batch mode, using only a random subset of the data during each iteration of the learning algorithm. When using synthetic datasets (Chapter 4 and Chapter 5), I used the entire set of training data during each iteration.

3.7 Discussion

This chapter introduced four modifications to standard sparse coding: the frozen-ℓ_2, $\ell_{1,\infty}$, $\ell_2 \rightarrow \ell_{1,\infty}$, and frozen-$\ell_{1,\infty}$ algorithms. Prior research has used modified sparse coding to work more effectively with non-negative data [73], streaming data [75], and labeled data [78]. The modifications presented in this chapter are intended to allow sparse coding to

better model imbalanced data. The ability of these algorithms to model imbalanced data will be discussed in Chapter 4.

CHAPTER 4
FEATURE RECOVERY

One intuition behind using sparse coding as a feature extraction tool in machine learning is that while the input data lives in a very rich space, each data point can be represented as a combination of a few sources. For example, an audio track of a piano concerto is a very complicated signal. However, at any given moment only a few of the 88 possible piano keys are being played. Likewise, an image may contain a rich mixture of lines, curves, and other basic shapes. However, a small patch of the image is likely to contain only a few individual elements. In both of these examples, dictionary learning can be used to uncover the underlying structure of the sources in an unsupervised manner.

This chapter explores the ability of sparse coding to model imbalanced datasets under a variety of circumstances. The work in this chapter was submitted for publication in the IEEE Transactions on Signal Processing [90] and builds on work previously published in [87].

4.1 Synthetic Dataset

While using measured data for applied signal processing tasks is ideal, the experiments performed in this chapter rely on having synthetic datasets. In order to test how well the different algorithms model the structure of a dataset, that structure must be known. Additionally, working with synthetic data provides the opportunity to adjust the structure of the dataset in order to form meaningful conclusions that can be extended to other situations.

The premise behind sparse coding is that natural signals can be represented as a sparse superposition of a set of vectors. This set of vectors represents the structure of the dataset. For example, in a music analysis setting, the underlying structure could be vectors that represent the spectral content of different instruments and notes played in isolation. A

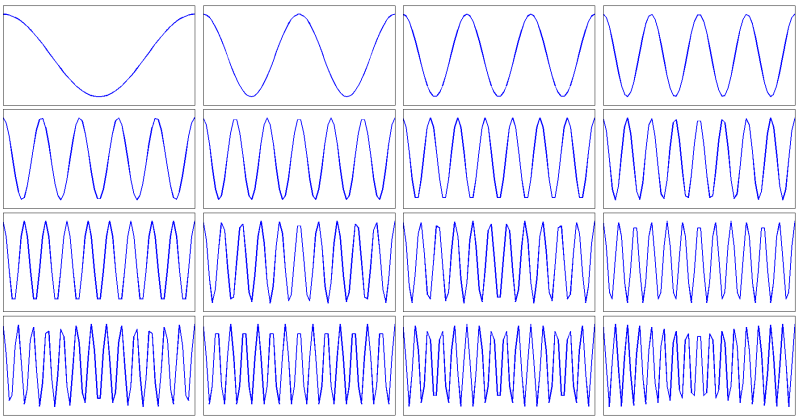

Figure 4.1: Simple basis vectors for anomaly recovery.

sample from the dataset might consist of a few instruments playing one note each. An instrument or a note that is not often played would still be represented in the structure, but its infrequent presence in the data would cause the dataset to be imbalanced. The synthetic datasets described here seek to model this type of behavior.

Each synthetic dataset discussed in this chapter consists of 400 data vectors. Each data vector is a sparse linear combination of what I call 'generating vectors.' The generating vectors used to create a data point are selected randomly. The 'normal' generating vectors are chosen with equal probability. The 'anomalous' generating vectors are selected less frequently, and all data points made using an anomalous generating vector are designated as anomalies.

For this chapter, each generating vector is a sinusoid of the form $g_k(t) = A\cos(2\pi kt)$, where A is a normalizing constant and t is a time vector of 128 linearly-spaced points between 0 and 1. Each data vector has the form $\sum_{i=1}^{K} a_i g_i(t)$, where K is the number of generating vectors. To create datasets with sparse structure, most of the a_i coefficient terms must be zero. Up to S coefficients may be nonzero, where S is the sparsity of the dataset. The nonzero coefficients are independently and uniformly sampled from $[-1.5, -0.5] \cup [0.5, 1.5]$. Figure 4.1 shows the set of generating vectors for $K = 16$. The goal of the experiments in this chapter is to demonstrate how well the various sparse coding algorithms can recover this set from the training data as various parameters are adjusted.

4.2 Dataset and Dictionary Parameters

There are several parameters of interest when using sparse coding to learn the structure of a dataset. Most of the controlled parameters are dataset-specific: the dataset balance (i.e. how frequently the anomalous generating vectors appear in the dataset), the SNR, the underlying sparsity, and the number of generating vectors. The one dictionary-dependent parameter studied is the number of elements in the learned dictionary. The remainder of Section 4.2 will discuss each parameter in more detail.

4.2.1 Number of Dictionary Elements

Dictionaries are learned with 16, 32, or 64 elements. For the frozen dictionaries of each size, and additional 2, 4, or 8 elements (12.5%) are included as non-frozen augmented elements, respectively.

4.2.2 Dataset Balance

The dataset balance parameter determines how often an 'anomalous' generating vector appears in the dataset. A dataset is randomly assigned to have one or two anomalous generating vectors. In creating the dataset, these vectors are used less frequently than the others to simulate dataset imbalance.

In addition to creating datasets with anomalous generating vectors that occur exactly once in the set of training data, datasets are created where the anomalous generating vectors occur 7%, 10%, 13%, 17%, 20%, 50% and 100% as often as the regular generating vectors. This percentage is denoted as the dataset balance. When an anomalous generating vector appears only once in the dataset, it is constrained to be in a linear combination with at least one normal generating vector. A dataset balance of 100% is a perfectly balanced dataset, but it is included as a control for this parameter.

4.2.3 SNR

Datasets are created with average SNRs of ∞dB (no noise), 30dB, 20dB, and 10dB. The noise is introduced by adding a random Gaussian vector with the appropriate expected energy to each training vector in the dataset.

4.2.4 Sparsity

As mentioned previously, each data vector is a sparse linear combination of generating vectors. For an S-sparse dataset, up to S generating vectors can be used to create a data sample. The datasets considered in this chapter are 3-, 4-, or 5-sparse.

4.2.5 Number of Generating Vectors

The generating vectors have the form $g_k(t) = A\cos(2\pi kt)$ for $k = 1...K$, where K is the number of generating vectors. The tested values of K in this chapter are 16 and 32.

4.2.6 Parameter Combinations

In order to analyze the sparse coding algorithms, I created one dataset for each possible combination of parameters. This yielded 576 datasets. In order to accommodate the frozen dictionary algorithms, I also created a 0%-balanced version of each dataset where the one or two generating vectors designated as 'anomalous' were not used to form any data point.

4.3 Experiment

4.3.1 Dictionary Learning

Figure 4.2 shows some examples of training data vectors. The examples shown in the figure are 4-sparse combinations of 16 generating vectors with an average SNR of 20 dB. The goal of the dictionary learning algorithms is to use a set of training vectors like these to learn the underlying structure of the dataset (i.e. the true generating vectors). I learned

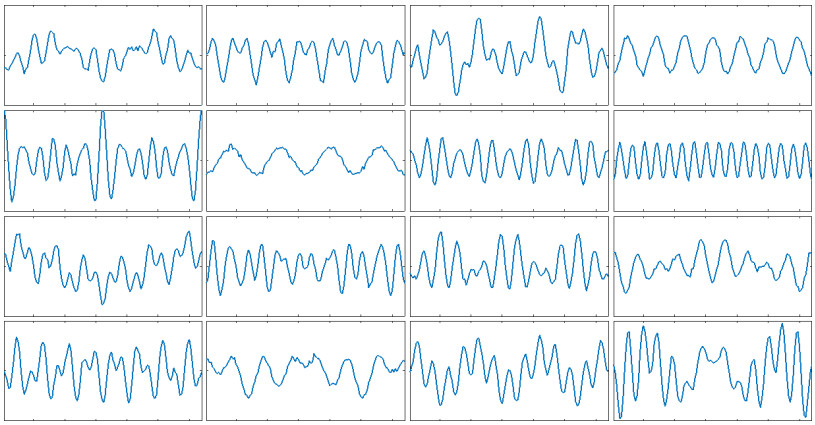

Figure 4.2: Examples of some training vectors. These particular vectors are 4-sparse combinations of 16 generating vectors with an SNR of 20 dB.

dictionaries using each of the sparse coding algorithms discussed in Chapter 3 on each of the 576 datasets.

4.3.2 Metrics

In analyzing the data, two different metrics are considered. The first is the normalized reconstruction error. For each data point, x, this is calculated as $100\% \times \|x - \mathbf{D}\alpha\|_2 \,/\, \|x\|_2$. This metric measures how well the dictionary models the *content* of the dataset.

The second metric is the angle (in degrees) between a generating vector and the closest learned dictionary element. This metric measures how well the dictionary represents the *structure* of the dataset:

$$\theta_k = \min_{d \in \{\mathbf{D}, -\mathbf{D}\}} \arccos\left(\frac{< g_k, d >}{\|g_k\|_2 \cdot \|d\|_2}\right) \tag{4.1}$$

An angle of 0° indicates that at least one dictionary element is an exact recovery of the generating vector. Higher angles (up to 90°) indicate that the learned dictionary does not have any elements that correspond to a particular generating vector. This metric gives valuable insight into how the dictionary learning algorithms perform as the dataset parameters are adjusted, because it indicates whether or not the algorithm has learned the true underlying

structure of a dataset, as opposed to a different representation of the data.

4.4 Recovery Analysis

Because of the large number of dictionaries learned using the various parameters, it is infeasible to discuss all of them. Instead, I focus on the individual parameters and comment on patterns associated with each. Important results associated with each parameter are summarized in Tables 4.1–4.5. At the end of this section, I discuss some observations with respect to general trends and relationships between parameters.

I wish to comment here on some notation that is common to all of the tables reported in this section. The top half of each table reports the average reconstruction error of training vectors that contain anomalous generating vectors. The bottom half reports the average angle between the true anomalous generating vectors and the closest learned dictionary element. For both metrics, smaller numbers are better.

The goal of this analysis is to determine whether or not the modified algorithms are more effective than standard sparse coding at modeling imbalanced datasets. For each of the non-ℓ_2 dictionaries, I compare the results with the respective ℓ_2 dictionary results using a paired-sample t-test. The asterisks denote whether the results are statistically significant with p-values less than 0.10 (*), 0.05 (**), or 0.01 (***). Additionally, some of the results in the tables are presented using bold numbers. The non-ℓ_2 results are bolded when they are significantly better than the standard ℓ_2 method. If no other method reports a significantly lower reconstruction error or angle deviation, the ℓ_2 results are shown in bold. In this case, if other methods are not significantly worse, they are also emphasized in bold.

4.4.1 Number of Dictionary Elements

The results associated with varying the number of dictionary elements in the learned dictionary are reported in Table 4.1. The results are averaged over two trials and over all tested dataset balance levels, SNR values, and numbers of generating vectors.

Table 4.1: Average recovery error (top) and average minimum angle (bottom) of anomalous vectors as a function of the number of learned dictionary elements

Dictionary Elements	ℓ_2	$\ell_{1,\infty}$	$\ell_2 \to \ell_{1,\infty}$	frozen-ℓ_2	frozen-$\ell_{1,\infty}$
16	19.2%	**15.8**% ***	**16.1**% ***	**11.8**% ***	23.2% ***
32	16.2%	15.4%	16.0%	**11.1**% ***	20.6% ***
64	**10.4**%	13.6% ***	13.6% ***	**9.9**%	18.2% ***
16	18.4°	**14.8**° ***	**15.1**° ***	**12.2**° ***	43.4° ***
32	19.0°	**17.4**° **	18.3°	**16.1**° ***	46.5° ***
64	**17.6**°	**17.7**°	19.3° ***	**17.5**°	45.5° ***

When the number of dictionary elements is strictly less than the number of generating vectors, all methods fail to model the dataset. The reconstruction error for all methods is about 75% and the minimum angles are around 80°. To avoid skewing the data, the results reported in Table 4.1 include only the average values when the number of dictionary elements is greater than or equal to the number of generating vectors. Likewise, the values reported in all subsequent tables do not take into account the dictionaries with fewer elements than generating vectors.

One interesting trend seen in Table 4.1 is that as the number of dictionary elements increases, the average reconstruction error decreases. This behavior has been previously documented for low-noise settings in [102]. It is also to be expected: a larger dictionary has more degrees of freedom in modeling the data, and can use this freedom to reduce the error. However, we see the opposite behavior with respect to the angle metric. As the dictionary size increases, the average angle between the true anomalous generating vectors and the closest learned dictionary element also increases. The large dictionary begins to model the *content* of the dataset rather than its *structure*. For some applications, such as in classification settings or source separation, it may be more important to have a dictionary that models the discriminative structure of the dataset rather than just picking the dictionary

with the smallest reconstruction error.

As mentioned previously, it is easy to tell from the reconstruction error if a dictionary is too small. The data in Table 4.1 suggests that in some cases a dictionary can possibly be too big, in the sense that it fails to learn the underlying structure of a dataset. However, this behavior cannot be determined by only considering the reconstruction error. Future work will test if this behavior affects classification performance.

4.4.2 Dataset Balance

Table 4.2: Average recovery error (top) and average minimum angle (bottom) of anomalous vectors as a function of dataset balance

Dataset Balance	ℓ_2	$\ell_{1,\infty}$	$\ell_2 \to \ell_{1,\infty}$	frozen-ℓ_2	frozen-$\ell_{1,\infty}$
1 Anomaly	40.3%	**19.4**% ***	**18.7**% ***	**6.7**% ***	**18.8**% ***
7%	23.4%	**16.9**% *	**18.0**% *	**10.8**% **	22.7%
10%	23.1%	**16.2**% *	20.1%	**11.3**% ***	21.1%
13%	18.1%	15.2%	16.9%	**11.5**% **	21.7%
17%	13.8%	15.8% **	15.6% **	**12.1**% *	23.3% ***
20%	**12.7**%	15.1% ***	15.0% ***	**12.4**%	23.5% ***
50%	**12.7**%	14.6% ***	14.6% ***	12.8% **	23.0% ***
100%	**12.7**%	14.2% ***	14.2% ***	12.9% ***	22.2% ***
1 Anomaly	78.4°	**48.8**° ***	**51.2**° ***	**51.7**° ***	**59.7**° ***
7%	32.6°	**25.8**° *	**27.9**° *	**23.8**° **	49.4° ***
10%	25.5°	**18.5**° *	21.9°	**16.4**° **	45.0° ***
13%	18.2°	13.9°	16.3°	**9.9**° **	43.3° ***
17%	**9.2**°	**10.8**°	**10.7**°	**9.5**°	43.4° ***
20%	**5.9**°	9.1° ***	9.2° ***	7.7° **	43.8° ***
50%	**2.9**°	6.0° ***	5.8° ***	3.0° ***	41.2° ***
100%	**1.9**°	4.9° ***	4.2° ***	2.7° ***	39.3° ***

The results associated with varying the anomaly frequency are reported in Table 4.2. These results are averaged over two trials and over all tested SNRs, sparsities, and numbers of generating vectors. For the purposes of testing this parameter, we constrain the

dictionary size to match the number of generating vectors. As we saw when discussing the dictionary size parameter, a smaller dictionary resulted in poor performance and a larger dictionary skewed both the error and angle metrics.

The table indicates that the $\ell_{1,\infty}$ and frozen-ℓ_2 methods outperform standard ℓ_2 sparse coding using both metrics when the dataset balance is less than 13-17%. The frozen-ℓ_2 method is better than the $\ell_{1,\infty}$ method, but also requires more a priori knowledge: the training data must be split into two groups with only one containing anomalies in order to learn a frozen dictionary.

The ℓ_2 method outperforms all others when the dataset balance is at least 20%. At this point, the anomalous vectors may be prevalent enough to affect the average reconstruction error, and there are enough examples in the training data for standard sparse coding to model them. A dataset with a balance of 20% can be considered to be 'balanced enough' for the purposes of sparse coding, in the sense that the ℓ_2 method is able to model the underlying structure of the dataset.

The cascaded $\ell_2 \rightarrow \ell_{1,\infty}$ and the $\ell_{1,\infty}$ have similar performance using both metrics. In addition, when using block corrdinate descent for the $\ell_{1,\infty}$ update, both methods have similar computational complexity.

One final comment is that the frozen-ℓ_2 method reports a very low reconstruction error (6.7%) when there is only one anomaly present in the dataset. However, the average angle between the anomalous generating vector and the learned dictionary element is still large (51.7°). These results indicate that the dictionary is learning the content of the dataset without learning its structure. Upon further investigation, I discovered that dictionaries in this situation were often modeling the entire training vector. This is to be expected under the optimization criteria; since there is only one training vector that contains an anomaly, the augmented portion can reduce both the error and the sparsity by exactly replicating that vector. The negative consequence of this is that the dictionary also models the noise associated with those vectors and fails to learn useful structure from the dataset.

4.4.3 SNR

Table 4.3: Average recovery error (top) and average minimum angle (bottom) of anomalous vectors as a function of signal-to-noise ratio

SNR	ℓ_2	$\ell_{1,\infty}$	$\ell_2 \rightarrow \ell_{1,\infty}$	frozen-ℓ_2	frozen-$\ell_{1,\infty}$
∞ dB	21.4%	**9.3**% ***	**11.6**% ***	**4.5**% ***	15.5%
30 dB	24.4%	**9.8**% ***	**12.4**% ***	**5.0**% ***	16.5% **
20 dB	21.2%	**14.3**% ***	**14.1**% ***	**8.1**% ***	18.7%
10 dB	28.0%	33.2% ***	33.1% ***	**24.2**% **	35.1% ***
∞ dB	33.4°	**20.5**° ***	**23.2**° ***	**20.3**° ***	46.6° ***
30 dB	36.5°	**20.4**° ***	**24.6**° ***	**17.4**° ***	44.3°
20 dB	31.6°	**21.5**° ***	**22.9**° ***	**25.5**° **	48.7° ***
10 dB	29.7°	31.8°	31.7°	**25.8**° ***	53.2° ***

The results associated with varying the SNR of the dataset are reported in Table 4.3. These results are averaged over two trials and over all tested sparsity values. For the purposes of testing this parameter, the dictionary size is constrained to match the number of generating vectors. The dataset balance level is constrained to be less than or equal to 17%. As seen in the previous section, datasets more balanced than this may be considered to be balanced enough for the purposes of sparse coding.

The results indicate that the frozen-ℓ_2 method performs the best in all noise conditions. The $\ell_{1,\infty}$ and $\ell_2 \rightarrow \ell_{1,\infty}$ methods also perform better than the standard ℓ_2 method under low-noise situations when the SNR is 20 dB or greater. However, the ℓ_2 and frozen-ℓ_2 methods are more effective in the high-noise situation (10 dB).

I hypothesize that with an SNR of 10 dB, some training vectors may have noise components with a higher ℓ_1-norm than the individual generating vector components. Since the mixed-matrix norm minimizes the maximum ℓ_1 error, the dictionaries may choose to model the noise rather than the signal in these training vectors. This would explain the large jump in reconstruction error for the two $\ell_{1,\infty}$ methods between 20 dB and 10 dB.

Table 4.4: Average recovery error (top) and average minimum angle (bottom) of anomalous vectors as a function of sparsity

Sparsity	ℓ_2	$\ell_{1,\infty}$	$\ell_2 \to \ell_{1,\infty}$	frozen-ℓ_2	frozen-$\ell_{1,\infty}$
3	29.8%	**17.1**% ***	**19.4**% ***	**10.0**% ***	**22.1**% ***
4	22.8%	**17.2**% **	**18.4**% **	**11.1**% ***	21.2%
5	18.6%	**15.7**% *	**15.6**% *	**10.4**% ***	21.2%
3	40.1°	**23.0**° ***	**27.6**° ***	**21.0**° ***	45.6°
4	29.7°	**22.4**° ***	**25.2**° **	**21.5**° ***	47.0° ***
5	28.5°	**25.2**° *	**24.1**° **	**24.3**° **	52.0° ***

4.4.4 Sparsity

The results associated with varying the underlying sparsity of the dataset are reported in Table 4.4. The results are averaged over two trials and over all tested SNR values and numbers of generating vectors. The dataset balance is constrained to be 17% or smaller and the dictionary size is constrained to match the number of generating vectors.

The values in the table indicate that there is less variance in dictionary performance as the sparsity is changed than when other parameters are changed. Also note that the ℓ_2 method performs more poorly, in both metrics, when the sparsity is 3. Apart from these instances, the results for both metrics are fairly uniform across all sparsities. The frozen-ℓ_2, $\ell_{1,\infty}$, and $\ell_2 \to \ell_{1,\infty}$ methods report a lower reconstruction error and smaller angle deviation than standard sparse coding. This is because the table considers only the most highly imbalanced datasets, and these methods are tailored to model such dataset behavior.

4.4.5 Number of Generating Vectors

The results associated with varying the number of generating vectors of the dataset are reported in Table 4.5. The results are averaged over two trials and over all tested SNR and sparsity values. Again, the dictionary size is constrained to match the number of generating

Table 4.5: Average recovery error (top) and average minimum angle (bottom) of anomalous vectors as a function of the number of generating vectors

Generating Vectors	ℓ_2	$\ell_{1,\infty}$	$\ell_2 \to \ell_{1,\infty}$	frozen-ℓ_2	frozen-$\ell_{1,\infty}$
16	23.0%	**16.4%** ***	**16.8%** ***	**11.1%** ***	23.1%
32	24.5%	**17.0%** ***	**18.9%** ***	**9.9%** ***	**19.9%** **
16	27.9°	**19.9°** ***	**20.7°** ***	**17.7°** ***	46.0° ***
32	37.7°	**27.2°** ***	**30.5°** ***	**26.8°** ***	50.3° ***

vectors and the anomaly frequency is restricted to be below 20%.

Again, the frozen-ℓ_2 method performs the best in both metrics, and the two $\ell_{1,\infty}$ methods outperform the standard ℓ_2 method. The reconstruction errors are fairly consistent as the number of generating vectors is changed from 16 to 32. However, all methods are more successful with respect to the angle deviation metric when there are 16 generating vectors. I hypothesize that this has more to do with the nature of large-dimensional spaces than with the sparse coding algorithms, but it is something that should be taken into account when using dictionary learning to model a high-dimensional dataset.

4.4.6 Results

At this point, I wish to discuss some general trends across all tests. First, the frozen-$\ell_{1,\infty}$ method has a poor average performance in all circumstances. Second, the $\ell_2 \to \ell_{1,\infty}$ method and the $\ell_{1,\infty}$ method have comparable performance in all tests. Since the iterations in the learning process for ℓ_2 dictionaries (using gradient descent) and $\ell_{1,\infty}$ dictionaries (using block coordinate descent) take roughly the same amount of time, I have no concrete conclusion about the tradeoffs between the two methods.

The frozen-ℓ_2 method consistently outperforms all of the other metrics. However, as has been mentioned before, the frozen dictionary method requires a dataset that is known to be free of anomalies. If such a dataset is not available, the $\ell_{1,\infty}$ method is quite successful

when there are infrequent anomalies (occuring about 17% as often as regular generating vectors) in a low-noise setting (with an SNR of at least 20 dB). Otherwise, it may be better to use standard ℓ_2 sparse coding.

The results obtained from investigating the sparsity and number of generating vectors did not seem to indicate that these parameters directly affect the dictionary learning process. However, the number of generating vectors will affect the choice of dictionary size when performing sparse coding. It is also possible that the underlying sparsity may affect the choice of the sparsity-fidelity tradeoff parameter (λ), but this was not tested.

It is important to note that one property of sinusoids is their mutual orthogonality: $\langle \cos(2\pi kt), \cos(2\pi lt) \rangle = 0$ for $k \neq l$. Since the generating vectors are uncorrelated, it is difficult for sparse coding to accurately reconstruct a training vector without a good model of every generating vector. The modified dictionary learning methods may be less successful in recovering rare generating vectors in situations where the generating vectors are correlated with each other. This is another property that has not yet been tested.

4.5 Discussion

At this point, I wish to reiterate three important conclusions. First, while a larger dictionary size tends to reduce the reconstruction error, the analysis has shown that larger dictionaries may overfit the data and begin modeling noise. Therefore, care should be taken when determining this parameter. Second, it appears that the point at which a dataset becomes truly imbalanced, with respect to sparse coding, is when the anomalous vectors occur about 17% as often as non-anomalous vectors. Third, while the mixed-norm methods are successful at recovering the structure of anomalies in a dataset, they perform poorly when the noise in the dataset is too large.

While the insights about dictionary size, dataset imbalance, and noise levels are enlightening, the main contribution of this chapter relates to the ability of the modified sparse coding algorithms to model the original structure of the imbalanced training vectors. Pre-

viously, sparse coding was used to identify anomalies by investigating vectors with a large reconstruction error [8, 9, 10, 11, 12]. The frozen-ℓ_2 and $\ell_{1,\infty}$ algorithms presented here are new tools that can be used to model imbalanced datasets. The effectiveness of these tools in a classification setting will be discussed later, in Chapter 6. Prior to that, in Chapter 5, I will investigate how well sparse coding is able to model nonlinearities.

CHAPTER 5
NONLINEARITY IN SPARSE CODING

An important assumption made by sparse coding is that a vector in a dataset can be written as a sparse *linear* combination of some underlying feature vectors. However, sparse coding is often performed in a *nonlinear* feature domain. For example, in many audio applications, data is transformed from the time domain to a magnitude spectra or cepstral domain prior to further processing. Even though the linear assumption of the models is not perfectly met, empirical results show that these techniques can be successfully applied to speech or music signals after the audio data undergoes nonlinear preprocessing [97]. This behavior may not extend to other, less harmonic audio environments.

In some situations, parts of a signal may fit the linear model assumed by sparse coding, even after nonlinear processing. The truly nonlinear portions of the training vectors could then be modeled as noise, and sparse coding could extract a portion of the original structure. For example, consider a dataset with training vectors consisting of a sparse linear combination of generating vectors, squared. In this case, each training vector consists of a linear combination of the squares of the generating vectors plus some cross-product terms. Sparse coding may recover the squares of the generating vectors and treat the cross-product terms as noise. Since each cross-product term occurs relatively infrequently, they may not affect the average reconstruction error enough to be modeled by the learned dictionaries.

The purpose of this chapter is to investigate the importance of the linearity assumption when applying sparse coding to a nonlinear dataset. In particular, this chapter explores the ability of linear sparse coding to recover a known, but hidden, dictionary when the training vectors are sparse *nonlinear* combinations of the hidden vectors. The work in this chapter has been submitted for publication in the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018) [91].

5.1 Experiment

To test the ability of linear models to recover nonlinear dictionaries, two sets of generating vectors (hidden dictionaries) are used to create several nonlinear datasets. The vectors in the datasets are formed by taking a sparse linear combination of generating vectors and then applying a nonlinearity. Linear versions of each dataset are also created by applying the nonlinearity to each generating vector prior to adding them together. Sparse coding is applied to each dataset and the learned dictionary is compared to the hidden dictionary.

5.1.1 Hidden Dictionary Descriptions

In designing this experiment, I considered two hidden dictionaries. The first consists of $\mathrm{B} = 16$ vectors randomly generated from $[0, 1]^{64}$. The second mimics a set of harmonic signals. It consists of $\mathrm{B} = 8$ vectors with different fundamental frequencies (from 100 to 800 Hz) and harmonics of decreasing amplitude. The spike of the fundamental frequency has a magnitude of 1 and a half-amplitude bandwidth of 40 Hz. The hidden dictionary is a set of generating vectors denoted as $\{\boldsymbol{b}_i\}_{i=1}^{\mathrm{B}}$. Plots of these vectors can be seen in Figure 5.1.

The datasets associated with the random dictionary consist of 500 training vectors, each a linear or nonlinear combination of three hidden vectors. The dataset associated with the harmonic dictionary consists of 200 training vectors which are combinations of two hidden vectors.

5.1.2 Nonlinearity Descriptions

The first nonlinearity I tested was the square operator. The linear dataset associated with this nonlinearity consists of training vectors of the form $\boldsymbol{x_l} = (a_i\boldsymbol{b}_i)^2 + (a_j\boldsymbol{b}_j)^2 + (a_k\boldsymbol{b}_k)^2$. The scalars a_i, a_j, and a_k are randomly selected from $\{[-1.5, 0.5] \cup [0.5, 1.5]\}$. Each training vector is truly a sparse linear combination of vectors, where the hidden dictio-

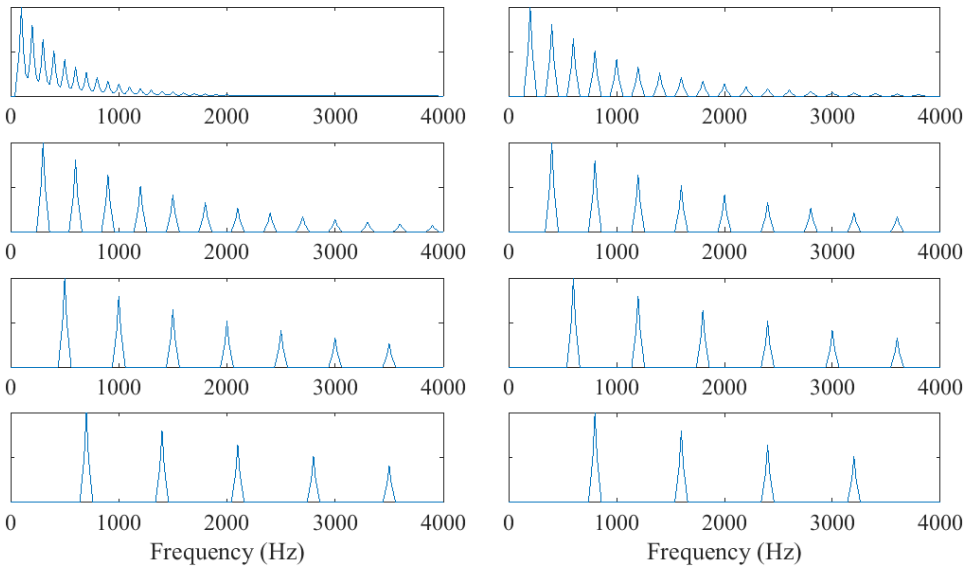

Figure 5.1: Basis vectors simulating harmonic structure in the frequency domain.

nary is $\{\boldsymbol{b}_i^2\}_{i=1}^{\mathrm{B}}$. The nonlinear dataset consists of training vectors of the form $\boldsymbol{x_{nl}} = (a_i\boldsymbol{b}_i + a_j\boldsymbol{b}_j + a_k\boldsymbol{b}_k)^2$. In this dataset, the training vectors are sparse *nonlinear* combinations of hidden vectors.

The definition of the scalar coefficients (a_i, a_j, and a_k) remains constant for all databases. In addition, to clarify the notation in this section, note that the nonlinearities are applied to the vectors element-wise.

The second nonlinearity I tested was the absolute value function. The training vectors in the linear version of the dataset have the form $\boldsymbol{x_l} = |a_i\boldsymbol{b}_i| + |a_j\boldsymbol{b}_j| + |a_k\boldsymbol{b}_k|$. In this case, each training vector is a 2- or 3-sparse linear combination of vectors taken from $\{|\boldsymbol{b}_i|\}_{i=1}^{\mathrm{B}}$. The training vectors in the nonlinear dataset have the form $\boldsymbol{x_{nl}} = |a_i\boldsymbol{b}_i + a_j\boldsymbol{b}_j + a_k\boldsymbol{b}_k|$.

The third nonlinearity I explored was the magnitude operator. When combining the hidden vectors to form the training vectors, each hidden vector is first multiplied (element-wise) by a random phase vector. The random phase is insignificant in the linear dataset as $\boldsymbol{x_l} = |a_i\boldsymbol{b}_i e^{j\boldsymbol{\theta}_i}| + |a_j\boldsymbol{b}_j e^{j\boldsymbol{\theta}_j}| + |a_k\boldsymbol{b}_k e^{j\boldsymbol{\theta}_k}| = |a_i\boldsymbol{b}_i| + |a_j\boldsymbol{b}_j| + |a_k\boldsymbol{b}_k|$. However, the random phase component makes a difference when applying the magnitude operator after combining the hidden vectors. The nonlinear dataset contains vectors of the form $\boldsymbol{x_{nl}} =$

$|a_i\boldsymbol{b}_i e^{j\boldsymbol{\theta}_i} + a_j\boldsymbol{b}_j e^{j\boldsymbol{\theta}_j} + a_k\boldsymbol{b}_k e^{j\boldsymbol{\theta}_k}|$.

The final nonlinearity was the magnitude-squared function. The linear and nonlinear datasets used for this nonlinearity were identical to those used by the magnitude operator, but the elements of the training vectors were each squared after taking the magnitude. The vectors in the linear dataset have the form $\boldsymbol{x_l} = |a_i\boldsymbol{b}_i|^2 + |a_j\boldsymbol{b}_j|^2 + |a_k\boldsymbol{b}_k|^2$. The training vectors in the nonlinear dataset have the form $\boldsymbol{x_{nl}} = |a_i\boldsymbol{b}_i e^{j\boldsymbol{\theta}_i} + a_j\boldsymbol{b}_j e^{j\boldsymbol{\theta}_j} + a_k\boldsymbol{b}_k e^{j\boldsymbol{\theta}_k}|^2$.

Dictionaries are also learned on datasets made up of nonnegative linear combinations of the hidden vectors with added Gaussian noise. These are used as reference databases to illustrate how much 'noise' the nonlinearities introduce into their respective databases. The training vectors in the reference databases have the form $\boldsymbol{x_l} = |a_i\boldsymbol{b}_i| + |a_j\boldsymbol{b}_j| + |a_k\boldsymbol{b}_k| + \boldsymbol{\epsilon}$, where the expected norm of $\boldsymbol{\epsilon}$ determines the SNR of the database. The noise term can be positive or negative to reflect the fact that nonlinearities can introduce positive or negative interference when combining vectors. However, after introducing (possibly negative) noise, each vector in the database is constrained to be nonnegative.

Table 5.1 summarizes the format of the training vectors associated with each database.

Table 5.1: Format of Training Vectors in Each Dataset

Nonlinearity	Linear (l) and Nonlinear (nl) Vectors
Square	$\boldsymbol{x_l} = (a_i\boldsymbol{b}_i)^2 + (a_j\boldsymbol{b}_j)^2 + (a_k\boldsymbol{b}_k)^2$
	$\boldsymbol{x_{nl}} = (a_i\boldsymbol{b}_i + a_j\boldsymbol{b}_j + a_k\boldsymbol{b}_k)^2$
Abs. Value	$\boldsymbol{x_l} = \|a_i\boldsymbol{b}_i\| + \|a_j\boldsymbol{b}_j\| + \|a_k\boldsymbol{b}_k\|$
	$\boldsymbol{x_{nl}} = \|a_i\boldsymbol{b}_i + a_j\boldsymbol{b}_j + a_k\boldsymbol{b}_k\|$
Magnitude	$\boldsymbol{x_l} = \|a_i\boldsymbol{b}_i\| + \|a_j\boldsymbol{b}_j\| + \|a_k\boldsymbol{b}_k\|$
	$\boldsymbol{x_{nl}} = \|a_i\boldsymbol{b}_i e^{j\boldsymbol{\theta}_i} + a_j\boldsymbol{b}_j e^{j\boldsymbol{\theta}_j} + a_k\boldsymbol{b}_k e^{j\boldsymbol{\theta}_k}\|$
Mag. Square	$\boldsymbol{x_l} = \|a_i\boldsymbol{b}_i\|^2 + \|a_j\boldsymbol{b}_j\|^2 + \|a_k\boldsymbol{b}_k\|^2$
	$\boldsymbol{x_{nl}} = \|a_i\boldsymbol{b}_i e^{j\boldsymbol{\theta}_i} + a_j\boldsymbol{b}_j e^{j\boldsymbol{\theta}_j} + a_k\boldsymbol{b}_k e^{j\boldsymbol{\theta}_k}\|^2$
Noisy Linear	$\boldsymbol{x_l} = \|a_i\boldsymbol{b}_i\| + \|a_j\boldsymbol{b}_j\| + \|a_k\boldsymbol{b}_k\| + \boldsymbol{\epsilon}$

5.1.3 Experiment Details

After creating the datasets, linear sparse coding dictionaries are learned using the ℓ_2 sparse coding algorithm. Since the nonlinearities discussed are nonnegative, the algorithm is initialized with a random nonnegative dictionary and a nonnegative constraint is imposed on both the dictionary elements and the sparse coefficient vectors. The number of learned dictionary elements matches the size of the hidden dictionary (16 or 8).

5.2 Results

Table 5.2 outlines the results from learning dictionaries using the different datasets. The value reported in the angle column is the average Euclidean angle between the dictionary elements and the closest nonlinearized hidden vector ($\boldsymbol{b}_i^2$ or $|\boldsymbol{b}_i|$). The reconstruction error reports the average reconstruction error, normalized by the norm of the training vector and reported as a percentage.

Table 5.2 reports that dictionaries recovered the entire hidden dictionary for each linear dataset. The feature vectors align well (within 2°) with the hidden vectors. Figure 5.2 shows the angle between the sparse coding dictionary elements and the nonlinearized hidden vector for the four linear databases using the random hidden dictionary. The entries on the diagonal are each below 2°, showing that the learned dictionary recovers the structure of the hidden dictionary very well.

The reconstruction error is also small (under 1% for the random dataset and under 4% for the harmonic dataset).

The small angles and reconstruction errors associated with these databases is to be expected. Since the nonlinearity was applied to the hidden vectors prior to their being added together, the training vectors in the dataset were truly sparse linear combinations of a set of vectors. What merits further discussion is where the algorithms were able to recover the original vectors when trained on the nonlinear datasets.

Table 5.2: Summary of results

Hidden Dictionary	Random		Harmonics	
Database	Angle	Error	Angle	Error
Square				
Linear	$0.59°$	0.75%	$0.25°$	3.19%
Nonlinear	$9.30°$	32.83%	$11.66°$	18.78%
Absolute Value				
Linear	$1.53°$	0.49%	$0.23°$	2.10%
Nonlinear	$15.56°$	21.14%	$6.38°$	15.62%
Magnitude				
Linear	$1.49°$	0.48%	$0.21°$	2.15%
Nonlinear	$29.21°$	32.64%	$4.27°$	14.80%
Magnitude Square				
Linear	$0.60°$	0.72%	$0.29°$	3.53%
Nonlinear	$41.33°$	49.44%	$4.94°$	19.69%
Reference Linear				
30 dB SNR	$3.09°$	3.13%	$0.71°$	4.10%
20 dB SNR	$7.97°$	9.78%	$2.24°$	11.18%
10 dB SNR	$21.24°$	28.18%	$8.59°$	33.08%
0 dB SNR	$51.50°$	61.12%	$23.57°$	73.57%

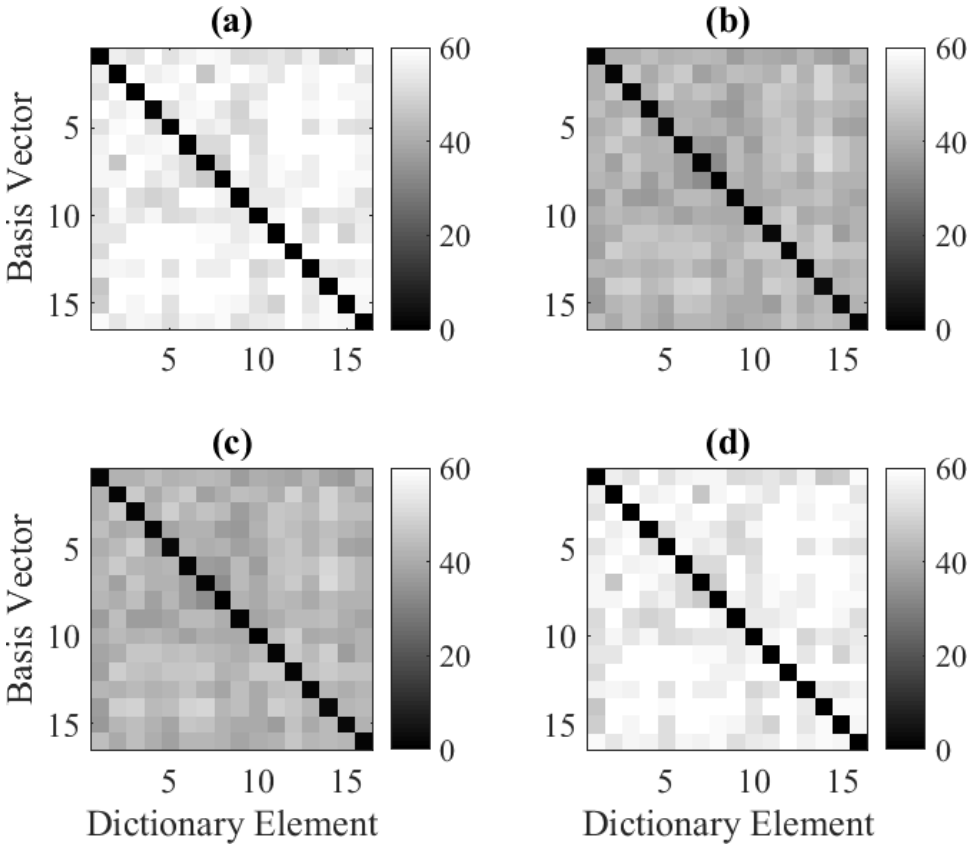

Figure 5.2: Angle between the sparse coding dictionary elements and the nonlinearized hidden vectors for the four linear databases using the random hidden dictionary. Part (a) corresponds to the square nonlinearity, (b) corresponds to the absolute value nonlinearity, (c) corresponds to the magnitude nonlinearity, and (d) corresponds to the magnitude-squared nonlinearity. All plots show a clear 1-1 correspondence between the learned elements and the original vectors, indicating that the learned dictionary was able to model each hidden dictionary element.

5.2.1 Random Hidden Dictionary

The dictionaries learned on the nonlinear square, absolute value, and magnitude databases were able to somewhat recover the original (nonlinearized) vectors ($|\boldsymbol{b}_i|$ or $\boldsymbol{b}_i^2$). While the average angles between the dictionary and the hidden vectors were substantial (9.30°, 15.56°, and 29.21°, respectively), there is a clear one-to-one correspondence that suggests complete recovery. Figure 5.3(a-c) shows this correspondence by plotting the correlations between the hidden dictionary and the sparse coding dictionary. Figure 5.3(c) illustrates this for the magnitude nonlinearity: while the average angle along the diagonal is 29.21°, the average angle on the off-diagonal is 54.10°. Figure 5.3(d) shows that the dictionary learned on the magnitude-squared database was unable to completely recover all of the original vectors. In this case, the average angle is 41.33°.

Figure 5.4 shows some pairs of training vectors with their reconstructed vectors for various reconstruction errors. One pair is taken from each database. The reconstruction error of the specific vector is presented in the figure, and these particular vectors are included in the figure because their reconstruction error is close to the average error of the respective database. This figure illustrates what a certain reconstruction error 'looks like' in the context of the problem. Note that while errors in the 20 to 30 percent range may seem large, the reconstructed vectors have much in common with the original.

The square nonlinearity reports the most successful recovery with respect to the angle metric. Recall that $(a+b+c)^2 = a^2+b^2+c^2+2ab+2ac+2bc$. In the linear framework, this expansion can be represented as $a^2+b^2+c^2+\epsilon$. For example, for the random databases each training vector has 3 square terms and 3 cross-product terms. There are only 16 different square terms while there are $\binom{16}{2} = 120$ different cross-product terms. Therefore, while a cross-product term contributes significantly to an individual training vector, it does not contribute to the overall structure of the database.

Despite this recovery, the reconstructed training vectors using these dictionaries have average reconstruction errors around 33%. While the hidden vectors were somewhat re-

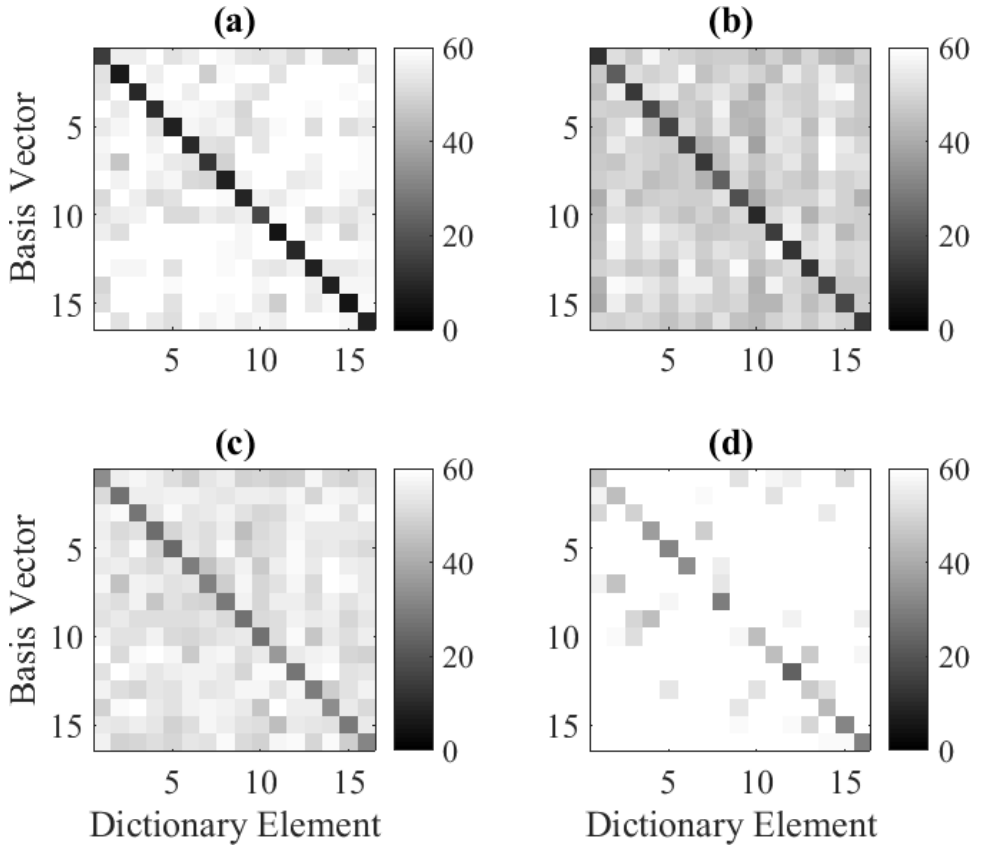

Figure 5.3: Angle between the sparse coding dictionary elements and the nonlinearized hidden vectors for the four nonlinear databases using the random hidden dictionary. Part (a) corresponds to the square nonlinearity, (b) corresponds to the absolute value nonlinearity, (c) corresponds to the magnitude nonlinearity, and (d) corresponds to the magnitude-squared nonlinearity. Plots (a-c) show a clear 1-1 correspondence between the learned elements and the original vectors.

covered, they are recombined only in a linear manner. This prevents reconstructed vectors in the square nonlinearity case from containing the cross-products terms that make up the original signal. For the absolute value nonlinearity, the training vectors can be recovered with a linear combination of dictionary elements if the signs of the coefficients are all the same; additional difference terms are needed in reconstruction if the coefficient signs are not the same.

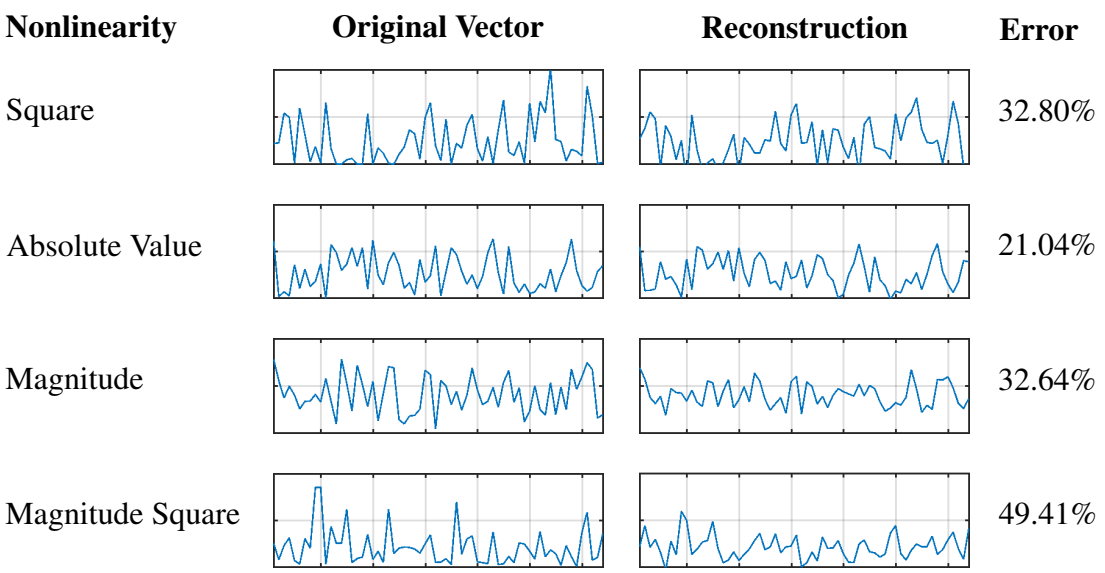

Figure 5.4: Visualization of reconstruction errors associated with each nonlinear database.

Nonlinear Reconstruction: Method and Results

The reconstruction error for the square nonlinearity can be improved by inferring cross-product terms from the learned dictionary elements. After calculating the sparse coefficients corresponding to each training vector, a 'nonlinear' dictionary was created. This dictionary consisted of the learned dictionary elements corresponding to the three largest (in magnitude) coefficients, as well as the three cross-product terms inferred from these elements:

$$\mathbf{D}_{\text{nonlinear}} = \left[\, d_1 \mid d_2 \mid d_3 \mid \sqrt{d_1}\sqrt{d_2} \mid \sqrt{d_1}\sqrt{d_3} \mid \sqrt{d_2}\sqrt{d_3} \, \right] \tag{5.1}$$

The reconstructed vector that takes into account the cross-product terms can be formulated as follows:

$$\hat{x}_{\text{nonlinear}} = \mathbf{D}_{\text{nonlinear}} \mathbf{D}^{\dagger}_{\text{nonlinear}} x, \tag{5.2}$$

where $\mathbf{D}^{\dagger}_{\text{nonlinear}}$ represents the pseudo-inverse of $\mathbf{D}_{\text{nonlinear}}$. When taking the cross-product terms into account in the reconstruction, the percent error decreases from 32.83% to 23.50%.

While this does not completely solve the nonlinear sparse coding problem, it does show that information about known nonlinearities can be used to improve the performance of sparse coding. This cross-product reconstruction algorithm is tailored to the square nonlinearity and does not improve performance when applied to the other nonlinearites considered in this chapter. When performed on the other three nonlinearities, the percent error actually increases.

5.2.2 Harmonic Hidden Dictionary

There is a dramatic difference in the performance of sparse coding when applied to the databases made from the harmonic hidden dictionary. The learned dictionaries more fully recover the hidden dictionaries and more accurately reconstruct the training vectors. The average recovery angle ranges from 4.27° to 11.66°, and the reconstruction error ranges from 14.80% to 19.69%.

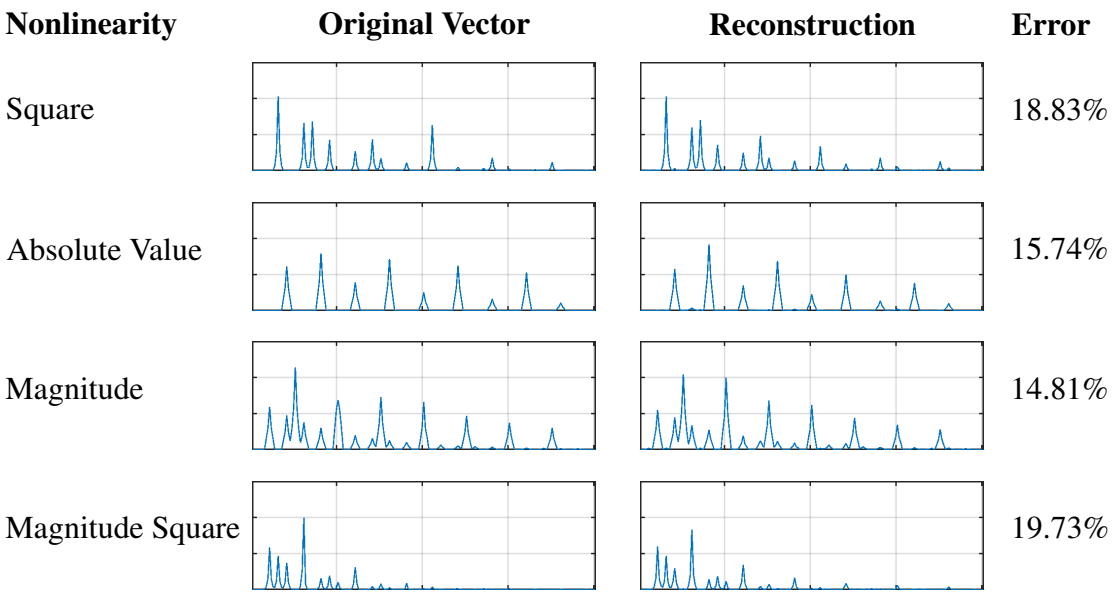

Figure 5.5: Visualization of reconstruction errors associated with each harmonic database.

Figure 5.5 shows examples of some original vectors with their reconstructions using the different nonlinearities and illustrates that even with reconstruction errors between 14% and 20%, the reconstructed vectors are still able to represent much of the content of the original vectors. This validates the empirical evidence of others who show that sparse coding works on harmonic audio signals after nonlinear preprocessing [103].

5.3 Discussion

In general, sparse coding does a poor job at recovering a nonlinear hidden dictionary with random vectors. This is relevant, for example, in the analysis of audio signals with broadband components. However, when the hidden dictionary had additional (harmonic) structure, as is common in speech and music audio applications, the sparse coding was more successful in recovering the original vectors. This explains why previous work has been able to use sparse coding on speech and music audio data after nonlinear processing [14, 15, 16].

While sparse coding does perform poorly in general after nonlinear processing, reconstruction can be improved by taking into account information about the specific nonlinearity. I demonstrated that including cross-product terms in the reconstruction after applying the square nonlinearity lowered the reconstruction error. The idea of tailoring reconstruction using nonlinear functions that match the properties of the data has been explored by others as well [82]. This idea can possibly be extended to a more general nonlinear sparse coding framework.

CHAPTER 6
CLASSIFICATION

The main intellectual contributions of this dissertation are related to signal modeling. I have discussed using sparse coding and some modifications to model imbalanced data, and I have addressed the ability of linear sparse coding to model nonlinear data. The purpose of this chapter is to illustrate that learning a good model of an imbalanced dataset can result in improved classification performance..

This chapter discusses the application of sparse coding to three imbalanced datasets: chicken vocalizations, human phonocardiograms (PCGs), and human electrocardiograms (ECGs). The chicken vocalization dataset includes recordings from healthy chickens and chickens infected with infectious bronchitis (IB), a respiratory disease with symptoms that include raspy breathing (known as rales). The PCG dataset includes audio recordings of heartbeats from healthy subjects and heartbeats from subjects who have been clinically diagnosed with one of a variety of cardiac diseases. The ECG dataset includes measurements of electrical signals produced by the hearts of healthy subjects and subjects who suffer from atrial fibrillation (AF), as well as other unusual ECG signals. All datasets have a different level of data imbalance, but they each have more examples of healthy recordings than sick recordings.

6.1 Chicken Audio

The work of applying sparse coding to chicken audio that is reported in this section was previously published in [88]. This is joint work with B. T. Carroll. As mentioned previously, Carroll implemented the frozen K-SVD algorithm and applied it to classify the chicken audio signals while I implemented the frozen-ℓ_2 algorithm. We worked in collaboration to design the experiment and analyze results.

The chicken audio dataset consists of continuous recordings of small flocks of chickens (six birds each) in isolation chambers for 25 days. During the course of the recordings, the experimental flock is exposed to the IB virus and contracts the respiratory disease. After exposure, the chickens begin to exhibit audible symptoms of IB (rales). These sounds occur occasionally during the period where the flock is sick, and are present in addition to other, normal chicken sounds. The control flock is never exposed to IB and the recordings of this group do not contain rales.

In the dataset, the chickens can be broadly labeled as having symptoms of disease or not. However, it is very difficult to individually label each rale. Despite this challenge, the rales from approximately 20 minutes of audio (taken from the experimental group) have have been hand-labeled. In addition to classifying entire audio segments as coming from a healthy group or a sick group of chickens, individual rales within these labeled audio segments can also be classified. In both cases, the classes are extremely imbalanced and the imbalance comes from additional sounds present in the sick group.

6.1.1 Method

The audio features are calculated as follows:

1. Apply a pre-emphasis filter (having a zero at 0.97) to the waveform.
2. Calculate the short-time Fourier transform (STFT) with a window width of 50ms and 50% overlap.
3. Take the magnitude of the result from the STFT.
4. Apply a triangular filter bank with 13 Mel-scaled frequency bands between 100Hz and 8kHz to the magnitude spectrum.

This feature calculation is similar to that of Mel-frequency cepstral coefficients (MFCCs) [104], but without the log scaling or the discrete cosine transform at the end. (Leaving off

these two steps preserves much of the additivity of the magnitude spectrum, which matches the assumption of linearity inherent in sparse coding.) In addition, an explicit energy term is not included, as is often done with MFCCs.

After extracting the audio features and applying a zero-mean preprocessing step, a dictionary is learned using the frozen-ℓ_2 method. A full whitening by normalizing to unit variance was not performed. Normalizing the variance would increase the influence of noise, particularly at night when the chicken noises themselves are relatively quiet and the sound scene is dominated by background noise.

The dictionary has 36 elements that are trained on the normal, or healthy, training data. Those 36 elements are frozen and the dictionary is augmented with 4 additional elements. The augmented elements are trained using the unlabeled anomalous, or sick, training data. These dictionaries are used to calculate sparse coefficient vectors for the training and test data in each dataset. While the features are all non-negative, non-negativity is not enforced when learning the dictionaries or calculating the coefficients.

The number of frozen elements and augmented elements was not chosen through a rigorous parameter search. These numbers were chosen somewhat arbitrarily, and future work could explore how sensitive the frozen dictionary approach is to these parameters. The number of augmented elements was chosen to be small because there is little variance in the audio characteristics of chicken rales.

The learned coefficients are used as features in two different analyses. The first analysis is applied to the hand-labeled portion of the IB dataset and requires that individual anomalies in the audio files be labeled. The second analysis is performed on the entire IB dataset and only requires the same high-level labels needed to learn the frozen dictionary. The analyses are summarized as follows:

1. Use the anomaly labels to train a support vector machine (SVM) classifying whether or not each 50ms audio window contains a chicken rale.

2. Average the coefficients calculated across an entire one-minute audio file and classify

whether or not each audio file came from the healthy flock or the sick flock.

In order to compare sparse coding to another data modeling method, I repeated the rale detection analysis after substituting the dictionary learning process with PCA. I used six PCA coefficients, which represents 90% of the variance in the dataset. I also performed the classification using the MFCC-like audio features described earlier.

6.1.2 Results

A signal could be classified simply based on the augmented dictionary elements that are likely used to represent those anomalies. However, a more effective way to classify is by using common machine learning algorithms, such as SVMs. This requires that the signals of interest are labeled in the SVM training data. The results reported here use `LIBSVM` to train and test the SVMs [105].

For the first classification task, the coefficient vectors calculated from the dictionaries are used to train SVMs to determine whether or not a given 50ms time window contains a chicken rale. For the second problem, the coefficient vectors from each one-minute audio file are averaged. The 'average coefficient vectors' are then used to train an SVM that can identify if a test file comes from a healthy flock or a sick flock. The following sections report the results for each of the two classification problems.

In addition to presenting the results of the frozen-ℓ_2 method, this chapter includes results associated with learning dictionaries using the frozen K-SVD algorithm [88]. Both dictionary learning algorithms report similar performances and outperform the classifier previously reported in [106].

Rale Detection

For the first problem of rale detection, a 10-fold cross-validated SVM is trained on the sparse coefficients to classify whether or not individual time windows contain a rale. These results are outlined in Table 6.1.

The top half of the table reports the accuracy, precision, and recall of the classification task with respect to labeling individual 50ms time windows[1]. The numbers represent the mean (μ) and standard deviation (σ) of scores after learning ten dictionaries and their respective classifiers.

While all four methods have similar accuracy, precision, and recall values, we can see some differences. First, while the classifier that uses the baseline audio features achieves the highest accuracy, the frozen k-SVD method performs the best in terms of precision and recall. Second, when PCA is applied to the audio features, the precision and recall both tend to drop with respect to the classifier using just the audio features. When the AM dictionary method is applied to those features, the precision is higher and recall is lower. This means that the resulting classifier is has fewer false alarms.

In addition, these values are significantly higher than prior results of 73.4% precision and 51.4% recall reported in [106]. The increase in performance can likely be attributed to a number of factors. The clustering approach used in the prior work cannot effectively

[1]Precision = TP / (TP+FP), Recall = TP / (TP+FN), and Accuracy = (TP+TN) / (TP+TN+FP+FN) with TP, FP, TN, and FN being the number of true positives, false positives, true negatives, and false negatives, respectively.

Table 6.1: Individual Rale Detection Results

	Accuracy ($\mu \pm \sigma$)	**Precision** ($\mu \pm \sigma$)	**Recall** ($\mu \pm \sigma$)
Unfiltered			
Audio Features	**0.969** $\pm$ 0.000	0.848 $\pm$ 0.000	0.787 $\pm$ 0.000
PCA	0.968 $\pm$ 0.000	0.845 $\pm$ 0.005	0.778 $\pm$ 0.004
AM	**0.969** $\pm$ 0.000	0.862 $\pm$ 0.005	0.763 $\pm$ 0.004
K-SVD	0.968 $\pm$ 0.001	**0.875** $\pm$ 0.021	**0.791** $\pm$ 0.018
3-pt. Med. Filter			
Audio Features	**0.964** $\pm$ 0.000	0.889 $\pm$ 0.000	0.672 $\pm$ 0.000
PCA	0.963 $\pm$ 0.000	0.885 $\pm$ 0.005	0.661 $\pm$ 0.006
AM	0.963 $\pm$ 0.000	0.911 $\pm$ 0.005	0.632 $\pm$ 0.006
K-SVD	0.963 $\pm$ 0.001	**0.912** $\pm$ 0.018	**0.700** $\pm$ 0.029

represent multiple sounds happening at once without creating new clusters to represent each different possible combination of sounds. The dictionary approach used here can efficiently represent combinations of sounds as the sum of a few underlying elements. The previous work also had fewer files with labeled rales to use for training, and the decision tree algorithm used for its classifier is likely not as robust as an SVM.

The bottom half of the table reports the results after filtering the labels with a median filter. The intuition behind the median filter is that rales are not temporally isolated events, but are usually continuous noises over 50 to 400ms. In all four cases, applying the median filter increases the precision of the classifier but reduces the recall. If the goal of the classifier is to reduce false alarms, it would be beneficial to include the median filter.

Disease Detection

Table 6.2: Minute-Average IB Detection Results

	Accuracy ($\mu \pm \sigma$)	**Precision** ($\mu \pm \sigma$)	**Recall** ($\mu \pm \sigma$)
K-SVD	0.997 ± 0.002	0.990 ± 0.004	0.994 ± 0.008
AM	0.993 ± 0.003	0.977 ± 0.014	0.983 ± 0.011

Manually labeling individual rales in audio data requires experts and is difficult and time-consuming. However, labeling whether a one-minute audio file comes from a healthy flock or a sick flock is simple. The sparse coefficients from the audio windows are averaged across each one-minute file, and this averaged vector is labeled as healthy or sick. An SVM is trained on the averaged coefficients, and the classifier results are reported in Table 6.2. Both the frozen-ℓ_2 and frozen K-SVD dictionaries result in classifiers that have a higher accuracy than previous work. The previous work uses standard ℓ_2 sparse coding on linearly-spaced spectral data and reports an accuracy of 97.85% [92], but does not report precision or recall.

6.1.3 Discussion

The frozen dictionary approach to anomaly detection can be used to detect the presence of subtle events in acoustic sequences without the need to label those events specifically. Using this approach, it would be easy to create detectors for specific, natural events by first training on background sounds and then presenting samples that contain both the background and the desired sound. If labels are available, the frozen-ℓ_2 method can also be used to generate features to train a classifier that detects individual events.

In the particular example of individual event detection, a successful classifier can be trained using just the 13-dimensional audio features. PCA does not offer improvements in either precision or recall, and may not be needed because the dimensionality of the features is so small. However, it appears that learning an overcomplete frozen dictionary allows the classifier to improve precision at the expense of recall. Thus, using the sparse coding methods combined with the median filter results in the smallest number of false alarms.

6.2 Phonocardiograms

In addition to the chicken data available at Georgia Tech, there is an open-source PCG dataset available through Physionet. This dataset consists of 3,153 PCG recordings, ranging from 6 to 120 seconds long. Each PCG is labeled as coming from a healthy patient or a patient with a clinically-diagnosed cardiac disease. Most diseases are heart murmurs, and the audio symptoms of the murmurs occur during a usually-silent portion of the heartbeat. Approximately 80% of the samples come from healthy patients, making it a somewhat imbalanced dataset. The dataset is described in detail in [107].

Figure 6.1 shows a plot of a PCG recording selected from the dataset. The plot includes labels that show the two fundamental heart sounds (S1 and S2) which represent the noise that occurs when different heart valves close. The periods of silence between these two sounds are called ‘systole’ and ‘diastole.’

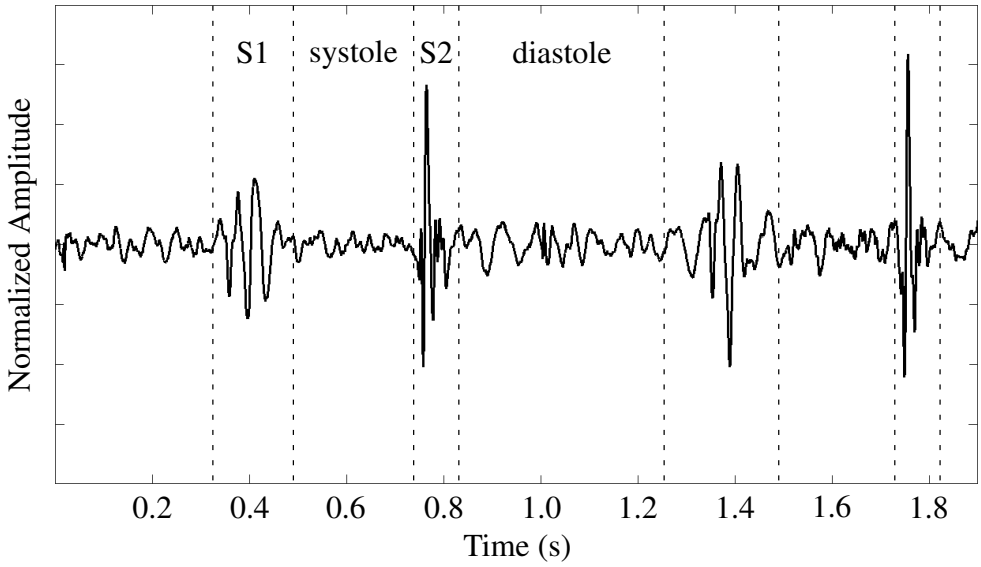

Figure 6.1: A segmented PCG signal with four heart sounds labeled.

In 2016, the Computing in Cardiology (CinC) Conference organized a challenge to use this dataset to create a classifier to identify whether the subject of a recording should be referred to an expert for further diagnosis. An accurate classifier could be used in a clinical setting to allow doctors to focus on higher-risk individuals and to reduce patient waiting times. Some of the work presented in this section was submitted as an entry to the 2016 CinC challenge in [93]. This section also includes additional research, some of which was published in [94].

6.2.1 Method

Audio Preprocessing

Prior to extracting features and learning a classifier, the audio data is preprocessed by segmenting the heart sounds and converting the data into the frequency domain. Figure 6.2 offers a visual representation of the preprocessing steps.

Segmenting a PCG into periods is fundamental in the automated analysis of heart sounds [108]. The first step of the algorithm utilizes Springer's state-of-the-art segmentation code [109] to separate each audio file into five arrays of smaller audio segments.

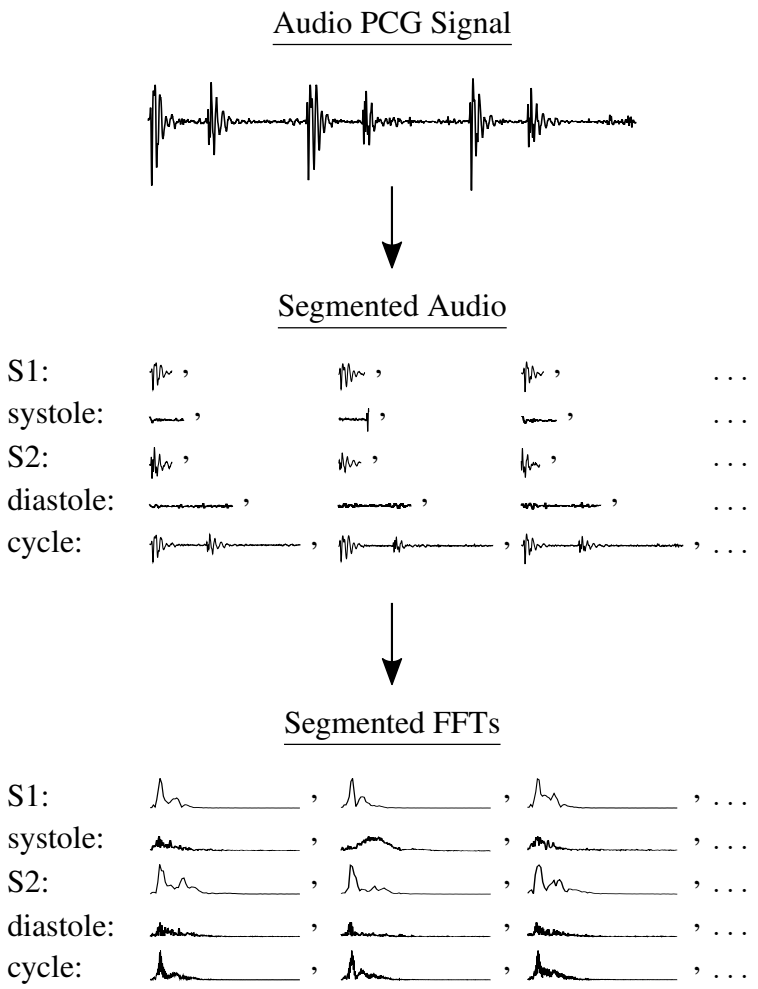

Figure 6.2: Visual representation of preprocessing applied to one PCG file. The preprocessing converts an audio file in the time domain into five arrays of frequency information, grouped by segmented heart sounds.

The first array contains a list of all S1 sounds present in the audio. The second, third, and fourth arrays contain all of the systole, S2, and diastole portions of the segmented PCG, respectively. The fifth array contains copies of the full heart cycles, starting with S1.

The next step in preprocessing the audio is to convert each sound segment from the time domain to the frequency domain with an N-point FFT. The value of N was determined by looking at the maximum lengths of the segmented heart sounds. The segmentation algorithm used in our implementation quantizes the sound segments' lengths to certain values. With high probability, using a random subset would produce the same maximum lengths.

The value of N was selected to be 364, 1024, 324, 2048, and 3760 for S1, systole, S2, diastole, and the full cycle, respectively. At the $2\,\mathrm{kHz}$ sampling frequency of the provided PCGs, these N-values correspond to $182\,\mathrm{ms}$, $512\,\mathrm{ms}$, $162\,\mathrm{ms}$, $1.024\,\mathrm{s}$, and $1.88\,\mathrm{s}$, respectively.

Since the segmented audio files are purely real, the FFT is also symmetric, so the negative frequency portion of the spectrum is removed in order to reduce computational complexity. As the spectrum magnitude contains the audible information available from the single-channel signal, the phase is also ignored to reduce computational requirements.

Sparse Coding

After preprocessing the PCG data, 1,000 of the 3,153 files are randomly selected to train a dictionary. These training files are used to create five data matrices. The columns of the first data matrix were the preprocessed S1 segments. Likewise, the systole, S2, diastole, and full-cycle segments made up the columns for the other data matrices. Sparse coding is then applied on these data matrices as a form of unsupervised feature extraction.

Applying sparse coding on the data matrices results in five different dictionaries. Each dictionary represents commonly-occurring spectral features present in the preprocessed S1, systole, S2, diastole, and full-cycle segments. Using these dictionaries, sparse coefficient vectors are computed for each segment of each audio file, and these vectors are averaged across the file. Thus, each PCG signal is represented by five sparse coefficient vectors. This process is outlined in Figure 6.3. The ℓ_2 method and the cascaded $\ell_2 \to \ell_{1,\infty}$ methods are both used, and their performance is compared. Recall that in this experiment, the $\ell_{1,\infty}$ update was implemented using CVX rather than block coordinate descent [101].

PCA

Again, to test the effectiveness of modeling the data using sparse coding, I compare it to PCA. For each of the five heart sounds, I used the 1,000 magnitude spectrum training files

to learn a subspace using PCA. The subspaces had dimensions of 12, 13, 11, 18, and 61 for the respective heart sounds (S1, systole, S2, diastole, and full cycle). New data samples are projected on to these low-dimensional subspaces and used as inputs in the classification stage.

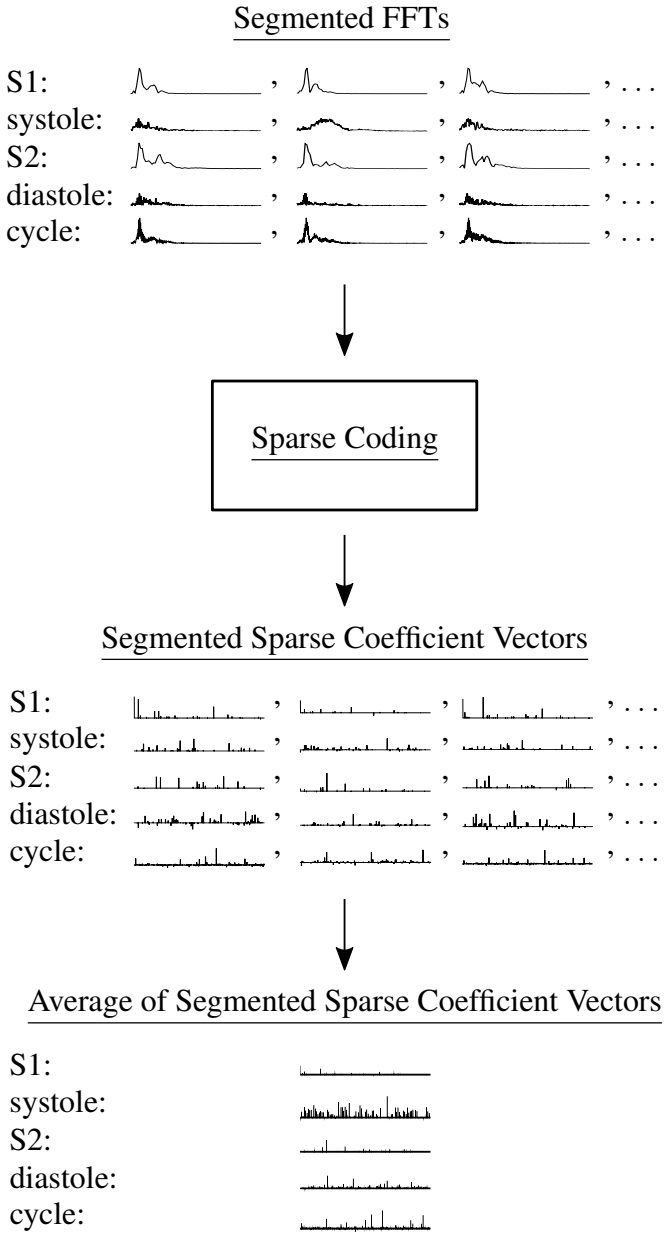

Figure 6.3: Visual representation of sparse coding applied to one PCG file. The sparse coding process converts each FFT from the preprocessing phase into a sparse vector representation. Each group of sparse vectors is then averaged across the PCG file, resulting in five sparse vectors—one for each heart sound segment.

Additional Time-Domain Features

In some experiments, 20 time-domain features are included in addition to the sparse coding features explained previously. These features are described in Table 6.3. Ten of the features are the average and standard deviation of the durations of the four heart sounds and the full cardiac cycle (RR-interval). Six features are the average and standard deviation of ratios of heart sound durations. The remaining four features are the average and standard deviation of ratios of heart sound amplitudes. All features are calculated during the segmentation preprocessing portion of the algorithm.

Table 6.3: Time-Domain Features

Feature	Description
μ_{RR}	Mean of RR-interval duration
σ_{RR}	Standard deviation of RR-interval duration
μ_{S1}	Mean of S1 duration
σ_{S1}	Standard deviation of S1 duration
μ_{sys}	Mean of systole duration
σ_{sys}	Standard deviation of systole duration
μ_{S2}	Mean of S2 duration
σ_{S2}	Standard deviation of S2 duration
μ_{dia}	Mean of diastole duration
σ_{dia}	Standard deviation of diastole duration
$\mu_{\text{sys/RR}}$	Mean ratio of single beat's systole and RR-interval durations
$\sigma_{\text{sys/RR}}$	Standard deviation ratio of single beat's systole and RR-interval durations
$\mu_{\text{dia/RR}}$	Mean ratio of single beat's diastole and RR-interval durations
$\sigma_{\text{dia/RR}}$	Standard deviation ratio of single beat's diastole and RR-interval durations
$\mu_{\text{sys/dia}}$	Mean ratio of single beat's systole and diastole durations
$\sigma_{\text{sys/dia}}$	Standard deviation ratio of single beat's systole and diastole durations
$\mu_{\text{sys/S1}}$	Mean ratio of single beat's systole and S1 amplitudes
$\sigma_{\text{sys/S1}}$	Standard deviation ratio of single beat's systole and S1 amplitudes
$\mu_{\text{dia/S2}}$	Mean ratio of single beat's diastole and S2 amplitudes
$\sigma_{\text{dia/S2}}$	Standard deviation ratio of single beat's diastole and S2 amplitudes

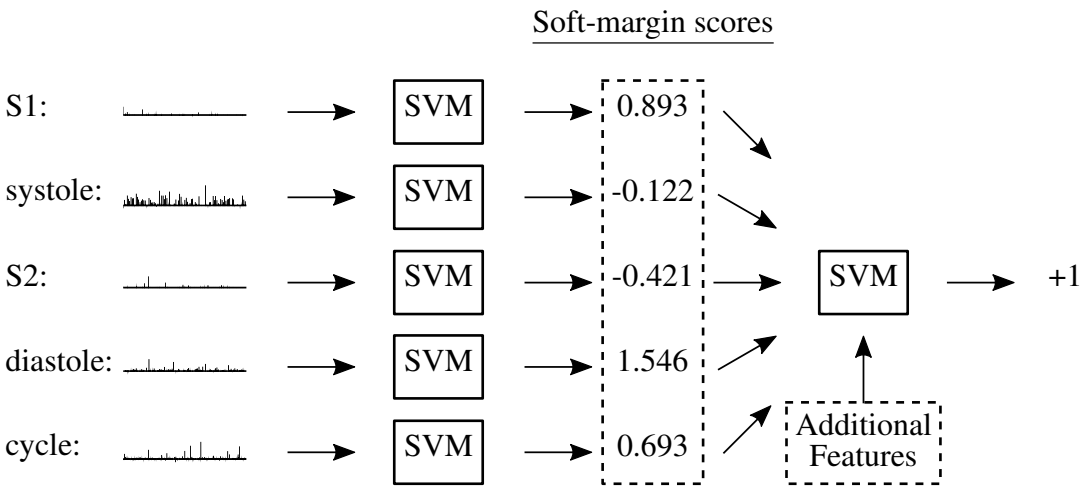

Figure 6.4: Visual representation of the classification process applied to one PCG file. After preprocessing and sparse coding, each file is represented by five different sparse coefficient vectors. An SVM is trained for each heart sound, and the respective sparse coefficient vector is classified with a soft margin score. These five scores are then combined using a final SVM to determine the label of a PCG file. When used, the twenty time-domain features are inputs to the final SVM as well.

Classification

The coefficient vectors from the unused 2,153 files are used to learn five cross-validated SVM classifiers, one for each segment type (S1, systole, S2, diastole, and full cycle). The SVMs are trained using the `libsvm` software package [105]. A first-order polynomial kernel is used because nonlinear kernels can cause extreme overfitting when classifying in a sparse domain. Other SVM parameters are tuned using the modified cuckoo search algorithm [110]. The soft-margin scores from each segment-specific SVM are used to train a final SVM that classifies a PCG file. Figure 6.4 displays a visual representation of the classification process.

The classified label produced by an SVM simply reports on which side of the learned decision boundary a data point lies. The SVM soft-margin score, however, is related to how close the data point is to the decision boundary. The magnitude of the soft-margin score can be interpreted as a measure of an SVM's 'confidence' when assigning the label and the sign of the soft-margin score corresponds to the label it would receive.

All SVMs are learned to maximize the mean accuracy score (MAcc) while including a parameter to encourage the sensitivity and specificity to be equal[2]. Learning the five sparse coding dictionaries and the six SVMs can be done offline. When a new PCG file is received, it undergoes the same preprocessing procedure (which involves segmenting the audio and calculating the FFTs), and then its sparse coefficient vectors are calculated using the five dictionaries corresponding to each heart sound. After averaging the sparse vectors across the file, five soft-margin scores are generated using the segment-specific SVMs. These five scores, and optionally the twenty time-domain features, are combined into a single label using the sixth SVM, resulting in a final classification of the new file.

6.2.2 Results

Recall that the sparse coding dictionaries are learned using only frequency information from the PCG signals. As a result, the learned features may not encode any time-domain information. Training a classifier using both the sparse coding (frequency-domain) features and the time-domain features results in higher accuracy than using the sparse codes alone.

As explained previously, several different methods of heart sound classification are included in this analysis. The first method is identical to the method presented in [93] at the 2016 Computing in Cardiology Conference, which used standard sparse coding features to classify the PCG audio files. The second method is similar, but involves the $\ell_2 \rightarrow \ell_{1,\infty}$ dictionary update. The mixed-matrix norm encourages the sparse coding dictionary to learn influential features by minimizing the maximum reconstruction error. Intuitively, this allows the dictionary to learn high-energy spectral features associated with abnormal heart sounds. In standard sparse coding, that information could be lost as the average reconstruction error is minimized. The third method is used to compare sparse coding against PCA, a more widely-used feature extraction algorithm.

The fourth method uses the ℓ_2 method, but introduces 20 time-domain features as inputs

[2]MAcc = (Sp+Se) / 2, where Sp = TN / (TN+FP) and Se = TP / (TP+FN). Se is sensitivity and Sp is specificity. TP, TN, FP, and FN are true positive, true negative, false positive, and false negative, respectively.

into the final classifying SVM. Similarly, the fifth and sixth methods combine the $\ell_2 \rightarrow \ell_{1,\infty}$ method and PCA, respectively, with the time-domain features. Finally, the last method learns a classifier using only the 20 time-domain features.

The procedure described previously is repeated twice using each method in order to provide a measure of statistical reliability using the two-sample t-test. Table 6.4 presents the ten-fold cross-validated sensitivity, specificity, and mean accuracy resulting from the two trials of each method. It also reports the p-values calculated from the two-sample t-test with respect to the standard sparse coding method. These values are calculated after classifying the 2,153 files from the SVM training set and exclude the 1,000 files used to learn the sparse coding dictionaries. As can be seen in the table, the matrix norm sparse coding features obtain higher sensitivity and specificity than standard sparse coding, but these improvements are not statistically significant. The table also shows that sparse coding features can be combined with other features for improved classification, as the two methods that use the twenty time-domain features outperform the other two methods. When adding the additional features to the $\ell_{1,\infty}$ sparse coding method, the improved overall scores are significant with a p-value of 0.08.

The method that reports the best cross-validated score is the matrix norm sparse coding combined with time-domain features, resulting in Se=0.887, Sp=0.882, and MAcc=0.884. Both sparse coding methods outperform PCA. In addition, we see the worst performance when the classifier is trained only on the time-domain features. This indicates that all three feature extraction methods (performed in the frequency domain) provide discriminative information that is not present in the selected time-domain features.

Table 1 in [21] itemizes state-of-the-art performance in this classification task. The work done by Potes reports Se=0.942, Sp=0.778, and MAcc=0.860 [111] . However, these values are calculated on hidden test data, while the results presented in Table 6.4 indicate cross-validated scores on the training data.

In [112], Zabihi achieves an average score nearly as good as Potes on the hidden test set

Table 6.4: Cross-Validated PCG Results

	Se	Sp	MAcc
ℓ_2 Sparse Coding	0.845	0.854	0.852
$\ell_{1,\infty}$ Sparse Coding	0.858	0.865	0.862
p-value (w.r.t. ℓ_2 Sparse Coding)	0.580	0.068	0.312
PCA	0.812	0.821	0.816
p-value (w.r.t. ℓ_2 Sparse Coding)	0.021	0.006	0.013
ℓ_2 Sparse Coding + 20 Time-Domain Features	0.877	0.877	0.877
p-value (w.r.t. ℓ_2 Sparse Coding)	0.098	0.075	0.033
$\ell_{1,\infty}$ Sparse Coding + 20 Time-Domain Features	**0.887**	**0.882**	**0.884**
p-value (w.r.t. ℓ_2 Sparse Coding)	0.157	0.011	0.076
PCA + 20 Time-Domain Features	0.836	0.839	0.838
p-value (w.r.t. ℓ_2 Sparse Coding)	0.087	0.033	0.039
20 Time-Domain Features	0.531	0.832	0.682
p-value (w.r.t. ℓ_2 Sparse Coding)	0.001	0.008	0.001

(Se=0.869, Sp=0.849, MAcc=0.859). He also reports cross-validated scores that would be a more appropriate comparison to the values presented in Table 6.4: Se=0.942, Sp=0.888, MAcc=0.915. Comparing my method with Zabihi's shows that my method matches state-of-the-art cross-validated performance in specificity (0.882 vs 0.888), but falls short with respect to sensitivity (0.887 vs 0.942). One possible reason for this is that I include a parameter to constrain the sensitivity and specificity to be nearly equal while training the SVMs. Removing this constraint would likely improve either sensitivity or specificity at the expense of the other metric.

6.2.3 Discussion

The work presented in this section has three main contributions. First, it introduces sparse coding as a tool for unsupervised features extraction in heart sound classification. Second, it shows that sparse coding features can be augmented with additional features to improve classification performance. This contribution demonstrates the importance of domain

knowledge in a machine learning environment. While sparse coding can be used blindly to find features for a classifier, performance is enhanced when including both sparse coding and domain-specific features. Finally, this work describes the first use of the $\ell_{1,\infty}$ sparse coding method in a practical classification setting. The $\ell_{1,\infty}$ method outperforms standard sparse coding and PCA, which suggests that it is able to learn a better model of the spectral features.

6.3 Electrocardiograms

Like the PCG dataset, the ECG dataset is available through Physionet. This dataset consists of 8,528 ECG recordings, ranging from 9 to 61 seconds in duration. Each ECG signal has one of four labels: normal (59.5%), AF (8.9%), other rhythm (28.3%), or noisy (3.3%). The dataset is described in detail in [20].

Like the PCG dataset discussed earlier, the ECG dataset was released to the public as part of a Computing in Cardiology challenge, this time in 2017. The goal of the challenge was to determine if a short ECG recording shows a normal rhythm, atrial fibrillation (AF), an alternative rhythm, or is too noisy to classify. The work presented in this section was previously published in [95] and represents an entry to the 2017 CinC challenge.

6.3.1 Method

ECG Preprocessing

Figure 6.5 shows a plot of an ECG signal with the P, Q, R, S, and T fiducial points, as well as the RR-interval, marked. When examining the ECG signal, I used two algorithms to extract the times and voltage values of the fiducial points of the ECG beat. The first algorithm was provided by the Challenge organizers and is based on the Pan and Tompkins (P&T) method of R-peak detection [113, 114]. This method only extracted the times and voltages of the R-peaks. The second algorithm was an ECG delimiter based on the work of [115] that identified all parts of the PQRST complex. This second algorithm was developed

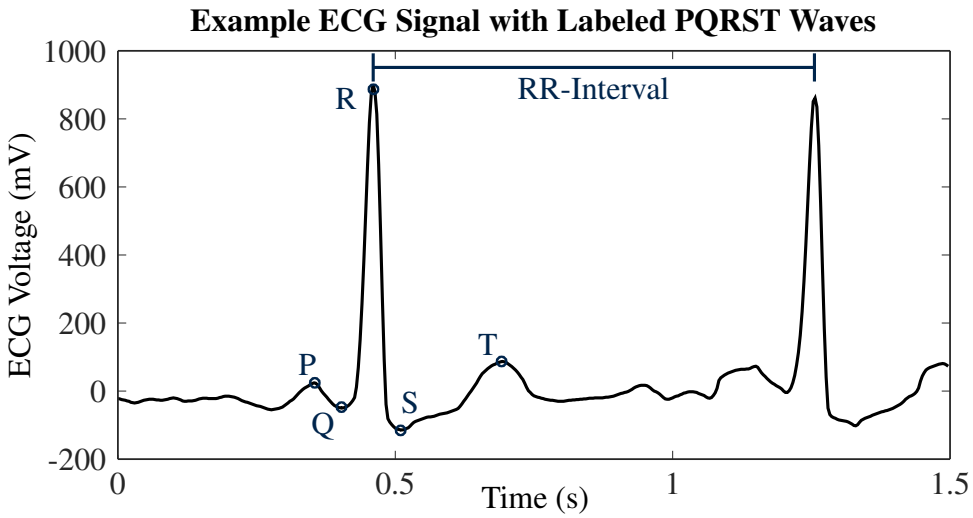

Figure 6.5: Plot of an ECG signal with the P, Q, R, S, and T fiducial points marked.

primarily by V. B. Aydemir and is explained in more detail in [95].

Feature Ensemble

After processing the ECG signal with both R-peak detection algorithms as well as the in-house PQRST algorithm, several statistics and other measures are used as features in the classifier. These features are described in Table 6.5.

Sparse Coding

Because one symptom of AF is irregular heart beats, dictionary learning is applied to the RR-intervals of the ECG signal. The RR-intervals are easily extracted and contain information about the time between heart beats. In order for the linear dictionary to extract meaningful features, the RR-intervals are sorted and the sorted list is interpolated to have a length of 50. The purpose of this is to allow the 'short' and 'long' RR-intervals from each signal to line up correctly. For example, the sorted list of RR-intervals taken from a healthy heart is fairly flat, while the sorted list corresponding to AF has a non-negligible slope. After sorting and interpolating the RR-intervals, sparse coding is applied as an unsupervised feature extractor.

Table 6.5: Features used in SVM Classifier

Feature	Description
6 R-Peak Statistics	Mean, standard deviation, and median of R-peak values calculated from both R-peak algorithms
10 RR-Interval Statistics	Mean, standard deviation, mean, min, and max of RR-intervals calculated from both R-peak algorithms
8 Irregularity Features	AFEvidence, PACEvidence, IrregularityEvidence, and OriginCount, as defined in [116], calculated from both R-peak algorithms
6 Entropy Features	Approximate Entropy [117], Sample Entropy [118], and Coefficient of Sample Entropy [119] calculated from both R-peak algorithms
2 Algorithm Agreement Statistics	Mean and standard deviation of the difference between the statistics mentioned above calculated from the two R-peak algorithms (15 each)
5 ECG Statistics	Mean, standard deviation, skew, kurtosis, and median of ECG signal
1 Heart Rate	Number of R-peaks (using P&T method) divided by duration of sample
1 RR-Interval Interdecile Range	Interdecile (10%-90%) range of the in-house RR-intervals (after removing intervals larger than 2.6s)
9 PQRST RMS Statistics	Mean, standard deviation, and median of RMS values of P, R, and T waves
9 PQRST Interval Statistics	Mean, standard deviation, and median of PR, RT, and QS intervals
15 PQRST Amplitude Statistics	Mean, standard deviation, and median of P, R, T, S, and RS normalized amplitudes
1 Polarity Feature	Percent of the positive polarity R-peaks
1 Noise Feature	Number of noisy six-second segments
4 Sparse Code SVM Scores	Explained in Section 6.3.1

After learning a dictionary on the RR-intervals, sparse feature vectors are calculated for each ECG file. A cross-validated, linear, four-class, soft-margin SVM is trained on the sparse vectors using LIBSVM [105]. The four scores from the soft-margin SVM are a measure of how likely the ECG belongs to each of the four classes. These four scores are

used as features in the final classifier.

The procedure was repeated using PCA on the sorted RR-intervals to identify differences between sparse coding and a more conventional dimensionality-reduction approach.

Classification

After extracting the 78 features (4 sorted RR-interval soft-margin scores plus the ensemble of 74 features) from each ECG file, a 10-fold cross-validated RBF-kernel SVM is trained using LIBSVM [105]. The SVM classifies between normal, AF, other, and noisy files. The modified cuckoo search algorithm is used to select the SVM learning parameters [110]. I searched for parameters that maximized the average F1 scores of the normal, AF, and other classes[3]. The parameters chosen are also encouraged to reduce range of the F1 scores for these three classes.

This classification procedure was repeated without the 4 soft-margin scores to determine how much those scores contribute to the final classifier.

6.3.2 Results

The ten-fold cross-validated scores for the various classifiers are reported in Table 6.6. The sparse coding and PCA features report similar performance for the $F1_{normal}$ and $F1_{noisy}$ scores. However, sparse coding outperforms PCA when considering the $F1_{AF}$ and $F1_{other}$ scores. The classifier trained only using the feature ensemble performs much better. Adding the PCA and sparse coding soft-margin scores only modestly improves the performance. This suggests that the feature ensemble is rich enough that it includes most of the discriminating information that can be extracted from the RR-intervals using either method. Because of this, follow-up work focused on feature selection and changing the classifier to improve performance [120].

[3]The F1 score is defined as the harmonic mean of the recall and precision: $F1_{normal} = (2 \times Nn)/(\sum N + \sum n)$, where Nn is the number of correctly labeled normal files, $\sum N$ is the total number of files with a ground truth label of normal, and $\sum n$ is the total number of files predicted as normal by the classifier. The aim of the 2017 CinC/PhysioNET Challenge was to maximize ($F1_{normal}$ + $F1_{AF}$ + $F1_{other}$) / 3

Table 6.6: Cross-Validated ECG Results

	Score	$F1_{normal}$	$F1_{AF}$	$F1_{other}$	$F1_{noisy}$
PCA	0.451	0.787	0.276	0.291	0.021
Sparse Coding	0.580	0.796	0.593	0.351	0.021
74 Features	0.779	0.876	0.766	0.697	0.499
PCA + 74 Features	**0.780**	0.871	**0.771**	0.698	0.472
Sparse Coding + 74 Features	0.779	**0.878**	0.760	**0.701**	**0.509**

When tested on the hidden challenge data, the algorithm using sparse coding and the feature ensemble achieved a score of 0.78. The F1 scores for normal, AF, and other files were 0.88, 0.80, and 0.65, respectively. The F1 score for noisy files was not provided. The final official challenge score was 0.77. These results are not state-of-the-art; Teijeiro, et. al. reported the best results with a score of 0.83 on the unseen challenge data [121]. However, using sparse coding in the ECG classification example emphasizes what was already demonstrated in the previous section on PCG classification: sparse coding is a valuable tool that can be used with other domain-specific features to develop a robust classifier.

6.3.3 Discussion

When looking at the results of the classifier without the feature ensemble, sparse coding outperforms PCA. This suggests that sparse coding is able to learn a better model of the RR-interval data as it pertains to AF detection. However, both methods pale in comparison to using the 74 domain-specific features. While adding PCA or sparse coding to the 74 features results in modest performance gains, it appears that the feature ensemble already contains most of the discriminating information. Future work could explore using sparse coding on more than just the RR-interval data. In addition, the feature ensemble presented here could be incorporated into different challenge solutions for more accurate AF detection from ECG recordings.

6.4 Summary

In this chapter, I have applied sparse coding to three separate classification tasks. When applied to disease detection in chickens, the frozen dictionary approach outperformed previous attempts at solving that problem. When applied to detecting abnormal cardiac events from PCG or ECG recordings, sparse coding is shown to work in conjunction with other, disease-specific features to create a robust classifier. In all cases, the direct comparisons between sparse coding and PCA show that sparse coding is able to better model the imbalanced datasets.

CHAPTER 7
CONCLUSION

7.1 Summary of Contributions

The work I have presented has the potential to increase the effectiveness of using sparse coding to model imbalanced datasets. I explored experimentally both the limits of sparse coding with respect to important algorithmic parameters and the inherent assumption of linearity in traditional sparse coding. In addition, I applied sparse coding to three classification problems. Recall the three research questions introduced in Chapter 1:

1. How well can sparse coding model known features of a dataset, especially features that occur infrequently?

2. To what extent can sparse coding model features when they are combined nonlinearly?

3. Can the model learned from sparse coding be used to successfully classify imbalanced datasets?

The work presented in this thesis addressed these questions and resulted in the publication of two journal papers [88, 94] and four conference papers [92, 87, 93, 95]. Two additional journal papers [89, 120] and two conference papers [90, 91] have been submitted for peer review.

I did not directly address any of these questions in Chapter 3. Instead, I used that chapter to introduce the frozen-ℓ_2, $\ell_{1,\infty}$, $\ell_2 \rightarrow \ell_{1,\infty}$, and frozen-$\ell_{1,\infty}$ sparse coding algorithms. While many other modified sparse coding algorithms have been developed and discussed in the literature and serve various purposes, they do not address the problem of dataset

imbalance [73, 75, 78, 79]. The modifications introduced in this dissertation are designed to encourage the dictionaries to more effectively model imbalanced datasets.

In Chapter 4, I answered the first question. In a controlled environment, standard sparse coding fails to accurately model anomalous features. This illustrates why previous attempts at using sparse coding focused on classification based on large reconstruction errors [8, 10]. The newly-introduced frozen-ℓ_2 and the $\ell_{1,\infty}$ algorithms, however, directly model highly imbalanced datasets (with a balance of 17% or smaller). Naturally, there are some limitations associated with these two algorithms: the frozen-ℓ_2 algorithm requires general labels and the $\ell_{1,\infty}$ algorithm requires a low-noise environment. But if these conditions are met, the two modifications are new tools that can be used to extract intuitive features from imbalanced datasets.

Chapter 5 addressed the second question. I showed that sparse coding fails, in general, to model nonlinearly preprocessed data. However, if the data has harmonic structure sparse coding can learn an effective model even after some nonlinear processing. This helps explain why matrix decomposition techniques have been successfully applied in nonlinear audio applications when the audio has harmonic structure [14, 17]. My research suggests that for applications without harmonic structure, sparse coding can be improved by taking into account the nonlinear structure of the data.

Finally, I addressed the third question in Chapter 6, where I used sparse coding in three classification applications. I achieved state-of-the-art performance in chicken disease detection using the frozen-ℓ_2 method of sparse coding. I reported competitive results for cardiac disease detection using the $\ell_{1,\infty}$ method on PCG signals. In addition, I used standard sparse coding to extract useful features from ECG signals in order to detect atrial fibrillation. My main contribution in this area was not the classification results; others have used neural networks to achieve state-of-the-art results in detecting cardiac diseases [112, 121]. Rather, I demonstrated that the modified sparse coding algorithms produce better features than standard sparse coding. In addition, this work shows that sparse coding features can

be used together with additional, domain-specific features for improved classification.

7.2 Future Directions

Developing algorithms that can learn good models from data in an unsupervised manner continues to be an important research task [6, 122, 123]. I believe two areas of investigation would be particularly useful.

7.2.1 Disciplined Parameter Selection

Using the minimum angle metric, as I did in Chapter 4 has some limitations. In order to calculate this metric, the underlying structure of the data must be known a priori. This makes it difficult to correlate the metric to classification accuracy in a real application setting. Future work may explore other metrics that can be calculated using just the original and reconstructed vectors with the hope of finding a metric that corresponds to classification accuracy.

One option for this is to use a large dictionary, analyze how many dictionary elements are not used regularly, and reduce the dictionary size accordingly [124]. However, this 'number of unused elements' metric may not work for imbalanced datasets. Another option is to analyze dictionaries based on the reconstruction error of the training data. In my work, I noticed a dramatic drop in reconstruction error as a function of dictionary size once the dictionary was large enough to model the underlying structure of the dataset. By learning dictionaries of different sizes on the dataset, this inflection point can be found and the smallest dictionary size that reports an acceptable reconstruction error can be used.

The current practice seems to be selecting the dictionary size in an arbitrary manner, sometimes related to the dimension of the original data (e.g. to ensure learning an over-complete or undercomplete dictionary) [125, 126]. Other parameters, such as the fidelity-sparsity tradeoff parameter λ, are also chosen arbitrarily. A more rigorous study in this area, as well as studies involving other parameters in sparse coding and machine learning,

could result in more structured ways to choose these parameters.

7.2.2 Nonlinear Sparse Coding

In the investigation on nonlinear sparse coding, my work focused on synthetic datasets. Again, the rationale behind this approach was to see how well sparse coding could recover known structure in a dataset after nonlinear manipulation. My research suggests that for some datasets and for some nonlinearities, linear sparse coding does a good job at modeling the data. I also looked into learning dictionaries that take into account a specific nonlinearity.

One focus of current nonlinear sparse coding research is the 'spike-and-slab' model, where dictionary elements are combined using a max function rather than addition [81, 86]. Future work could look into developing a general framework for nonlinear sparse coding. Nguyen begins to do this with kernel dictionary learning [79], but his method of multiplying the dictionary matrix by a kernel matrix results in sparse coefficients that are simply a linear combination of a different feature matrix. A method for truly nonlinear sparse coding would be a more effective tool for modeling datasets after nonlinear processing.

REFERENCES

[1] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.

[2] S. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[3] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[4] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special issue on learning from imbalanced data sets," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 1–6, Jun. 2004.

[5] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Comput. Surv.*, vol. 49, no. 2, 31:1–31:50, Aug. 2016.

[6] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[7] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220 –239, 2017.

[8] J. Zhou, D. Semenovich, A. Sowmya, and J. Wang, "Sparse dictionary reconstruction for textile defect detection," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 1, Dec. 2012, pp. 21–26.

[9] G. Boracchi, D. Carrera, and B. Wohlberg, "Novelty detection in images by sparse representations," in *Intelligent Embedded Systems (IES), 2014 IEEE Symposium on*, Dec. 2014, pp. 47–54.

[10] B. Zhao, L. Fei-Fei, and E. P. Xing, "Online detection of unusual events in videos via dynamic sparse coding," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, Jun. 2011, pp. 3313–3320.

[11] Y. Cong, J. Yuan, and J. Liu, "Abnormal event detection in crowded scenes using sparse representation," *Pattern Recognition*, vol. 46, no. 7, pp. 1851–1864, 2013.

[12] X. Mo, V. Monga, R. Bala, and Z. Fan, "Adaptive sparse representations for video anomaly detection," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 24, no. 4, pp. 631–645, Apr. 2014.

[13] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[14] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised learning of sparse features for scalable audio classification.," in *ISMIR*, vol. 11, 2011, p. 2011.

[15] T. Virtanen, J. F. Gemmeke, B. Raj, and P. Smaragdis, "Compositional models for audio processing: Uncovering the structure of sound mixtures," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 125–144, Mar. 2015.

[16] A. Hurmalainen, R. Saeidi, and T. Virtanen, "Noise robust speaker recognition with convolutive sparse coding.," in *INTERSPEECH*, 2015, pp. 244–248.

[17] M. Kim and P. Smaragdis, "Mixtures of local dictionaries for unsupervised speech enhancement," *IEEE Signal Processing Letters*, vol. 22, no. 3, pp. 293–297, Mar. 2015.

[18] P. Smaragdis, C. Fevotte, G. J. Mysore, N. Mohammadiha, and M. Hoffman, "Static and dynamic source separation using nonnegative factorizations: A unified view," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 66–75, May 2014.

[19] M. Rizwan, B. T. Carroll, D. V. Anderson, W. Daley, S. Harbert, D. F. Britton, and M. W. Jackwood, "Identifying rale sounds in chickens using audio signals for early disease detection in poultry," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2016, pp. 55–59.

[20] G. D. Clifford, C. Liu, B. Moody, I. Silva, Q. Li, A. Johnson, and R. G. Mark, "AF classification from a short single lead ECG recording: The PhysioNet Computing in Cardiology Challenge 2017," in *2017 Computing in Cardiology Conference (CinC)*, vol. 44, Sep. 2017.

[21] G. D. Clifford, C. Liu, B. Moody, J. Millet, S. Schmidt, Q. Li, I. Silva, and R. G. Mark, "Recent advances in heart sound analysis," *Physiological Measurement*, vol. 38, E10, Aug. 2017.

[22] F. Provost, "Machine learning from imbalanced data sets 101," in *Proceedings of the AAAIŠ2000 workshop on imbalanced data sets*, 2000, pp. 1–3.

[23] H. He and E. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[24] X. Zou, Y. Feng, H. Li, and S. Jiang, "Srot: Sparse representation-based oversampling technique for classification of imbalanced dataset," *IOP Conference Series: Earth and Environmental Science*, vol. 81, no. 1, p. 012 201, 2017.

[25] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, 2003, pp. 49–56.

[26] T. Imam, K. M. Ting, and J. Kamruzzaman, "Z-SVM: An SVM for improved classification of imbalanced data," in *AI 2006: Advances in Artificial Intelligence: 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 264–273, ISBN: 978-3-540-49788-2.

[27] N. V. Chawla, "C4.5 and imbalanced data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure," in *Proceedings of the ICML*, vol. 3, 2003.

[28] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, Jan. 2006.

[29] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623 – 1637, 2015.

[30] H. Yin and K. Gai, "An empirical study on preprocessing high-dimensional class-imbalanced data for classification," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, 2015, pp. 1314–1319.

[31] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1, pp. 245–271, 1997.

[32] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[33] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication*, vol. 25, no. 1Ű3, pp. 133–147, 1998.

[34] M. F. McKinney, J. Breebaart, *et al.*, "Features for audio and music classification.," in *ISMIR*, vol. 3, 2003, pp. 151–158.

[35] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," *Ann Arbor*, vol. 1001, no. 48109, p. 2, 2010.

[36] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[37] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[38] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 –52, 1987, Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[39] F. Nie, J. Yuan, and H. Huang, "Optimal mean robust principal component analysis," in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, 2014, pp. 1062–1070.

[40] S. Yi, Z. Lai, Z. He, Y. ming Cheung, and Y. Liu, "Joint sparse principal component analysis," *Pattern Recognition*, vol. 61, pp. 524 –536, 2017.

[41] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[42] C. Poultney, S. Chopra, and Y. L. Cun, "Efficient learning of sparse representations with an energy-based model," in *Advances in neural information processing systems*, 2006, pp. 1137–1144.

[43] H. Asari, B. A. Pearlmutter, and A. M. Zador, "Sparse representations for the cocktail party problem," *Journal of Neuroscience*, vol. 26, no. 28, pp. 7477–7490, 2006. eprint: `http://www.jneurosci.org/content/26/28/7477.full.pdf`.

[44] Y.-l. Boureau and Y. L. Cun, "Sparse feature learning for deep belief networks," in *Advances in neural information processing systems*, 2007, pp. 1185–1192.

[45] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, Jun. 2010.

[46] K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "Fast inference in sparse coding algorithms with applications to object recognition," *arXiv preprint arXiv:1010.3467*, 2010.

[47] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[48] K. Lee, Z. Hyung, and J. Nam, "Acoustic scene classification using sparse feature learning and event-based pooling," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*, IEEE, 2013, pp. 1–4.

[49] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

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

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

[52] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009. eprint: `http://dx.doi.org/10.1137/060657704`.

[53] M. Elad, M. A. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[54] J. Mairal, F. Bach, and J. Ponce, "Sparse modeling for image and vision processing," *Foundations and TrendsÂő in Computer Graphics and Vision*, vol. 8, no. 2-3, pp. 85–283, 2014.

[55] K. Huang and S. Aviyente, "Sparse representation for signal classification," in *Advances in neural information processing systems*, 2006, pp. 609–616.

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

[57] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce, "Discriminative sparse image models for class-specific edge detection and image interpretation," in *Computer Vision âĂŞ ECCV 2008*, ser. Lecture Notes in Computer Science, D. Forsyth, P. Torr, and A. Zisserman, Eds., vol. 5304, Springer Berlin Heidelberg, 2008, pp. 43–56, ISBN: 978-3-540-88689-1.

[58] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM journal on computing*, vol. 24, no. 2, pp. 227–234, 1995.

[59] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[60] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, 2006.

[61] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 6–18, Jan. 2006.

[62] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[63] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "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*, vol. 1, Nov. 1993, pp. 40–44.

[64] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, Feb. 2012.

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

[66] E. van den Berg and M. P. Friedlander, "Probing the pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2009. eprint: `http://dx.doi.org/10.1137/080714488`.

[67] K. Koh, S.-J. Kim, and S. Boyd, "An interior-point method for large-scale l1-regularized logistic regression.," *Journal of Machine learning research*, vol. 8, no. 7, pp. 1519–1555, 2007, Software available at `https://stanford.edu/~boyd/l1_ls/`.

[68] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, Feb. 2003.

[69] I. Tosic and P. Frossard, "Dictionary learning," *Signal Processing Magazine, IEEE*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[70] K. Engan, S. Aase, and J. Hakon Husoy, "Method of optimal directions for frame design," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 5, 1999, pp. 2443–2446.

[71] M. Aharon, M. Elad, and A. M. Bruckstein, "K-SVD and its non-negative variant for dictionary design," *Proc. SPIE*, vol. 5914, pp. 11–13, 2005.

[72] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[73] P. O. Hoyer, "Non-negative sparse coding," in *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, 2002, pp. 557–565.

[74] A. S. Charles, B. A. Olshausen, and C. J. Rozell, "Learning sparse codes for hyperspectral imagery," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 5, pp. 963–978, 2011.

[75] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, Montreal, Quebec, Canada: ACM, 2009, pp. 689–696, ISBN: 978-1-60558-516-1.

[76] ——, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 19–60, 2010.

[77] C. Lu, J. Shi, and J. Jia, "Online robust dictionary learning," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, Jun. 2013, pp. 415–422.

[78] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., Curran Associates, Inc., 2009, pp. 1033–1040.

[79] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2012, pp. 2021–2024.

[80] Y. Xie, J. Ho, and B. Vemuri, "On a nonlinear generalization of sparse coding and dictionary learning," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 1480–1488.

[81] J. Bornschein, M. Henniges, G. Puertas, and J. Lucke, "Sparse codes of V1 simple-cells and the emergence of globular receptive fields - a comparative study," in *Com-*

putational and Systems Neuroscience (Cosyne) Meeting, Salt Lake City, UT, Feb. 2011.

[82] J. Bornschein, M. Henniges, and J. Lücke, "Are V1 simple cells optimized for visual occlusions? A comparative study," *PLoS Comput Biol*, vol. 9, no. 6, 2013.

[83] J. A. Shelton, A.-S. Sheikh, J. Bornschein, P. Sterne, and J. Lücke, "Nonlinear spike-and-slab sparse coding for interpretable image encoding," *PLoS ONE*, vol. 10, no. 5, pp. 1–25, May 2015.

[84] A.-S. Sheikh, J. A. Shelton, and J. Lücke, "A truncated em approach for spike-and-slab sparse coding," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2653–2687, 2014.

[85] A. B. Patel, T. Nguyen, and R. G. Baraniuk, "A probabilistic theory of deep learning," vol. abs/1504.00641, 2015.

[86] A.-S. Sheikh and J. Lücke, "Select-and-sample for spike-and-slab sparse coding," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016, pp. 3934–3942.

[87] B. Whitaker and D. Anderson, "Learning anomalous features via sparse coding using matrix norms," in *2015 IEEE Signal Processing Workshop (SPW2015)*, Salt Lake City, USA, Aug. 2015, pp. 196–201.

[88] B. T. Carroll, B. M. Whitaker, W. Dayley, and D. V. Anderson, "Outlier learning via augmented frozen dictionaries," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1207–1215, Jun. 2017.

[89] B. M. Whitaker and D. V. Anderson, "Mixed matrix norms in sparse coding for imbalanced datasets," Submitted.

[90] ——, "Employing linear matrix factorizations after nonlinear processing," in *IEEE Statistical Signal Processing Workshop (SSP) 2018*, Submitted.

[91] ——, "Using block coordinate descent to learn sparse coding dictionaries with a matrix norm update," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2018*, Accepted.

[92] B. Whitaker, B. Carroll, W. Daley, and D. Anderson, "Sparse decomposition of audio spectrograms for automated disease detection in chickens," in *Proceedings of the IEEE GlobalSIP*, IEEE, Dec. 2014, pp. 1122–1126.

[93] B. M. Whitaker and D. V. Anderson, "Heart sound classification via sparse coding," in *2016 Computing in Cardiology Conference (CinC)*, Sep. 2016, pp. 805–808.

[94] B. M. Whitaker, P. B. Suresha, C. Liu, G. D. Clifford, and D. V. Anderson, "Combining sparse coding and time-domain features for heart sound classification," *Physiological Measurement*, vol. 38, no. 8, p. 1701, 2017.

[95] B. M. Whitaker, M. Rizwan, V. B. Aydemir, J. M. Regh, and D. V. Anderson, "AF classification from ECG recording using feature ensemble and sparse coding," in *2017 Computing in Cardiology Conference (CinC)*, vol. 44, Sep. 2017.

[96] P. Smaragdis, "From learning music to learning to separate," in *Forum Acusticum*, Citeseer, 2005.

[97] ——, "Probabilistic decompositions of spectra for sound separation," in *Blind Speech Separation*, Springer, 2007, pp. 365–386.

[98] A. Kumar and B. Raj, "Audio event detection using weakly labeled data," in *Proceedings of the 2016 ACM on Multimedia Conference*, ser. MM '16, Amsterdam, The Netherlands: ACM, 2016, pp. 1038–1047, ISBN: 978-1-4503-3603-1.

[99] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, Jun. 2001.

[100] H. Liu, M. Palatucci, and J. Zhang, "Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, Montreal, Quebec, Canada: ACM, 2009, pp. 649–656, ISBN: 978-1-60558-516-1.

[101] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 2.1*, http://cvxr.com/cvx, Mar. 2014.

[102] M. Lebrun and A. Leclaire, "An implementation and detailed analysis of the K-SVD image denoising algorithm," *Image Processing On Line*, vol. 2, pp. 96–133, 2012.

[103] E. Vincent, N. Bertin, R. Gribonval, and F. Bimbot, "From blind to guided audio source separation: How models and side information can improve the separation of sound," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 107–115, May 2014.

[104] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, Aug. 1980.

[105] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 3 2011, Software available at `www.csie.ntu.edu.tw/~cjlin/libsvm`.

[106] B. T. Carroll, D. V. Anderson, W. Daley, S. Harbert, D. F. Britton, and M. W. Jackwood, "Detecting symptoms of diseases in poultry through audio signal processing," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, Dec. 2014, pp. 1132–1135.

[107] C. Liu, D. Springer, Q. Li, B. Moody, R. A. Juan, F. J. Chorro, F. Castells, J. M. Roig, I. Silva, A. E. Johnson, Z. Syed, S. E. Schmidt, C. D. Papadaniil, L. Hadjileontiadis, H. Naseri, A. Moukadem, A. Dieterlen, C. Brandt, H. Tang, Maryam Samieinasab, M. R. Samieinasab, R. Sameni, R. G. Mark, and G. D. Clifford, "An open access database for the evaluation of heart sound algorithms," *Physiological Measurement*, vol. 37, no. 12, pp. 2181–2213, 2016.

[108] S. E. Schmidt, C Holst-Hansen, C Graff, E Toft, and J. J. Struijk, "Segmentation of heart sound recordings by a duration-dependent hidden markov model," *Physiological Measurement*, vol. 31, no. 4, p. 513, 2010.

[109] D. B. Springer, L. Tarassenko, and G. D. Clifford, "Logistic regression-HSMM-based heart sound segmentation," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 4, pp. 822–832, Apr. 2016.

[110] S. Walton, O. Hassan, K. Morgan, and M. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons & Fractals*, vol. 44, no. 9, pp. 710–718, 2011.

[111] C. Potes, S. Parvaneh, A. Rahman, and B. Conroy, "Ensemble of feature-based and deep learning-based classifiers for detection of abnormal heart sounds," in *2016 Computing in Cardiology Conference (CinC)*, vol. 43, Sep. 2016, pp. 621–624.

[112] M. Zabihi, A. B. Rad, S. Kiranyaz, M. Gabbouj, and A. K. Katsaggelos, "Heart sound anomaly and quality detection using ensemble of neural networks without segmentation," in *2016 Computing in Cardiology Conference (CinC)*, Sep. 2016, pp. 613–616.

[113] J. Behar, J. Oster, and G. D. Clifford, "Combining and benchmarking methods of foetal ECG extraction without maternal or scalp electrode data," *Physiological Measurement*, vol. 35, no. 8, p. 1569, 2014.

[114] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no. 3, pp. 230–236, Mar. 1985.

[115] A. Martinez, R. Alcaraz, and J. J. Rieta, "A new method for automatic delineation of ECG fiducial points based on the Phasor Transform," *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, pp. 4586–4589, 2010.

[116] S. Sarkar, D. Ritscher, and R. Mehra, "A detector for a chronic implantable atrial tachyarrhythmia monitor," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 3, pp. 1219–1224, Mar. 2008.

[117] S. M. Pincus, "Approximate entropy as a measure of system complexity.," *Proceedings of the National Academy of Sciences*, vol. 88, no. 6, pp. 2297–2301, 1991. eprint: `http://www.pnas.org/content/88/6/2297.full.pdf`.

[118] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *American Journal of Physiology - Heart and Circulatory Physiology*, vol. 278, no. 6, H2039–H2049, 2000. eprint: `http://ajpheart.physiology.org/content/278/6/H2039.full.pdf`.

[119] D. E. Lake and J. R. Moorman, "Accurate estimation of entropy in very short physiological time series: The problem of atrial fibrillation detection in implanted ventricular devices," *American Journal of Physiology - Heart and Circulatory Physiology*, vol. 300, no. 1, H319–H325, 2011.

[120] M. Rizwan, B. M. Whitaker, and D. V. Anderson, "AF detection from ECG recordings using feature selection, sparse coding, and ensemble learning," *Physiological Measurement*, Submitted.

[121] T. Teijeiro, C. A. Garcia, D. Castro, and P. Felix, "Arrhythmia classification from the abductive interpretation of short single-lead ecg records," in *2017 Computing in Cardiology Conference (CinC)*, vol. 44, Sep. 2017.

[122] L. Yin, Y. Ge, K. Xiao, X. Wang, and X. Quan, "Feature selection for high dimensional imbalanced data," *Neurocomputing*, vol. 105, pp. 3 –11, 2013, Learning for Scalable Multimedia Representation.

[123] S. Maldonado, R. Weber, and F. Famili, "Feature selection for high-dimensional class-imbalanced data sets using support vector machines," *Information Sciences*, vol. 286, pp. 228 –246, 2014.

[124] M. Zhou, H. Chen, L. Ren, G. Sapiro, L. Carin, and J. W. Paisley, "Non-parametric bayesian dictionary learning for sparse image representations," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds., Curran Associates, Inc., 2009, pp. 2295–2303.

[125] C. N. Yu, P. Mirowski, and T. K. Ho, "A sparse coding approach to household electricity demand forecasting in smart grids," *IEEE Transactions on Smart Grid*, vol. 8, no. 2, pp. 738–748, 2017.

[126] C. Bao, H. Ji, Y. Quan, and Z. Shen, "Dictionary learning for sparse coding: Algorithms and convergence analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, pp. 1356–1369, 2016.