Analysis Dictionary Learning For Classification

Submitted by

Protim Bhattacharjee

under the guidance of

Dr. Angshul Majumdar

Asst. Professor, IIIT-Delhi

Indraprastha Institute of Information Technology
New Delhi
July 2016

Analysis Dictionary Learning For Classification

By
Protim Bhattacharjee

Submitted
in partial fulfilment of the requirements for the degree of Master of Technology

to

Indraprastha Institute of Information Technology Delhi
July, 2016

Certificate

I hereby certify that the thesis titled, **Analysis Dictionary Learning For Classification** being submitted by **Protim Bhattacharjee** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree.

July 2016

Dr. Angshul Majumdar
Deaprtment of Electronics and Communication
Indraprastha Institute of Information Technology Delhi
New Delhi 110020

Acknowledgement

I would like to thank Dr. Angshul Majumdar for his constant support and encouragement. He was patient to my weird and absurd ideas and dealt with every oddity with a smile. He has been a great mentor and guide, helping me at each step with punctilious discussion and judicious concern. I am grateful to him for giving a free reign to work at my pace, discovering and learning things as I go ahead.

My sincere thanks to Ms. Anupriya Gogna, for helping me decipher my own experiments with her percipient outlook. I learnt conscientious and diligence in one's work from her. I thank her for all the patience she showed me throughout the thesis work.

I would like to thank Snigdha Tariyal, Dhananjay Kimothi and Parag Aggarwal for their immense support, help, constant motivation, scientific and non scientific discussion which kept me going.

I am grateful to the institute for all its infrastructure support and to all my colleagues who helped me in various ways throughout the thesis.

Finally, this thesis is dedicated to my loving parents, my sister and grandma. I am thankful for their unconditional love and support over the years and having trust in my abilities, come what may.

Protim Bhattacharjee

Contents

List of Figures

List of Tables

Abstract

Data classification is the at the core of leading technologies today. With the explosion of data through mobility and growth of the Internet, analysis and classification of data is the immediate process after acquisition. Most of the information is hidden within the acquired data and needs to be extracted for further processing. Thus, feature extraction is an an important pre-processing task to classification. Till recently features were hand crafted which though accurate, were time consuming to generate and required human intervention. With the explosion of deep learning (since 2006), automatic feature generation has become the norm for most applications. Algorithms learn from the data and generate the required features adapted to various tasks, such as classification, reconstruction, denoising, sentiment analysis, data mining and the like. The most popular algorithms for automatic feature generation are Deep Belief Networks, Autoencoders and Convolutional Neural Networks(CNN) which also provide classification capabilities. All these algorithms and architectures derive motivation from the fact that human visual and audio cortex are compositional in nature and activate at various level of abstraction. Similarly, the aforementioned machines learn from raw data at multiple levels of abstraction with growing complexity. Such algorithms have proven to be very efficient and have also provided various benchmark tools and applications in the industry like Google Photos and Facebook's deepFace.

However, the major limitation to such tools are their enormous training times and humongous amount of data that is required to train them. In this thesis, a computationally simpler model for generating features through an Analysis Dictionary Learning approach is presented. In contrast with the synthesis dictionary learning approaches where the features are generated by solving an inverse problem via an iterative procedure, the analysis approach has the advantage of generating features from the data with minimal preprocessing by directly operating the data with the dictionary. The dictionary operates on the data and generates features, hence the framework is named as the analysis framework. Significant

improvement in test time feature generation is obtained as compared to other dictionary learning methods. Also, the learning procedure is computationally inexpensive and is flexible as any prior knowledge can easily be incorporated into the framework depending on the task the learnt features will be put to. To prove the versatility of the framework, the proposed approach is applied to various real world scenarios like digit recognition, speech recognition and Non Intrusive Load Monitoring (NILM). We have been able to establish state of the art results under varied conditions comparable to more complex deep learning techniques.

Chapter 1

State of The Art

In the past few years, dictionary learning has been extensively used for classification tasks. This can easily be seen from the expansive literature that is available. Most of the dictionary learning approaches to classification are data driven, i.e., the learning procedure is adapted to the available data. The dictionary is learnt in a way that is adapted for a particular task such as data classification, signal denoising and the like. Literature provides wide variety of supervised and unsupervised learning procedures. However, most of these techniques follow the synthesis framework. In this chapter, a discussion on the concept of dictionary learning is presented along with the popular algorithms used for classification. The analysis dictionary learning framework is introduced and discussed. State of the art for both synthesis and analysis framework is reviewed.

1.1 Dictionary Learning

Dictionary learning is a paradigm in which operators are designed according to the data and task at hand. In essence, the operator learning methodology is driven by the characteristics of the data instead of the characteristics of the operator itself. Sure enough operators can be constrained to have particular characteristics, but dictionary learning is focussed more on learning the peculiarities of the operator from the data and more often than not it is found that the learnt operator does have nice properties. From a mathematical point of view, the dictionary learning problem can be cast as follows:

$$x = Dz \tag{1.1.1}$$

Where,

x is the data vector

D is the dictionary

z are the coefficients

The expression in (1.1.1) is referred to as the synthesis dictionary learning framework. This framework expresses the signal as a linear combination of dictionary atoms, which are represented by the columns of the dictionary D. The coefficients of the linear combination are given by the vector z. Therefore, the aim here is to find a dictionary, D and coefficient vector z, such that the signal x is nearly equal to the product Dz. There are many flavours of such a framework. The unconstrained version of (1.1.1) can be written as

$$min_{D,z} \ \|x - Dz\|_2^2 \tag{1.1.2}$$

Here $||.||_2$ denotes the l_2 norm of a vector, which for a vector y of n dimensions is defined as $\sqrt{\sum_{i=1}^{n} y_i^2}$. The matrix version of equation (1.1.2) can be written as,

$$min_{D,Z} \ \|X - DZ\|_F^2 \tag{1.1.3}$$

$\|.\|_F$ denotes the Frobenius norm of a matrix. The Frobenius norm of a matrix A can be defined as $Tr\sqrt{AA^T}$, where Tr denotes the trace of a matrix. Here X is a matrix of the data stacked as columns and Z is a column sparse matrix. There are no constraints on the dictionary and on the coefficients in equations (1.1.2) and (1.1.3). The most popular way of solving such a problem is through the method of optimal directions [1], which uses a strategy of updating the two variables D and Z alternately through least square updates. In [1], the dictionary is called a frame and the number of frames required to represent the data is a free parameter. Problem (1.1.3) is also widely known as matrix factorization.

The unconstrained problem does not use any prior knowledge about the signal or datum x. With the advent of compressive sensing [2], sparsity became an important property of the signal which was exploited for faster reconstruction and denoising with a smaller number of signal observations. This framework is be represented as

$$y = Ax \tag{1.1.4}$$

Where,

y is the observations or measurements of the signal x

A is a $m \times n$ transformation matrix with $m < n$

x is the signal to be estimated or recovered

As $m < n$, the system of equations given by (1.1.4) in general would have an infinity of solutions. However, if A satisfies the Restricted Isometry Property (RIP) and x is s sparse, then with measurements, $m \geq Cs \ln\left(\frac{n}{s}\right)$, x can be recovered with a high probability [3]. The optimization problem for such a scenario can be written as,

$$min_x \ \|y - Ax\|_2^2 \quad s.t. \ \|x\|_0 < s \tag{1.1.5}$$

$\|.\|_0$ is the l_0 norm of a vector which counts the no. of non zero elements in a vector. Solving a l_0 problem is combinatorial in nature and as such is NP hard [4]. Greedy techniques are available for solving (1.1.5). Popular among them are Orthogonal Matching Pursuit (OMP) and its variants [5–9]. However, relaxing the l_0 norm to the l_1 norm, which for a n dimensional vector x maybe defined as $\sum_{i=1}^{n} |x_i|$, the problem becomes convex and can be cast in the following form,

$$min_x \ \|y - Ax\|_2^2 \ \ s.t. \ \ \|x\|_1 < t \tag{1.1.6}$$

where t is a parameter which in general is inversely proportional to the noise variance. The most common method of solving of (1.1.6) is by relaxing the constraint via the Lagrangian method and thus can be re-written as

$$min_x \|y - Ax\|_2^2 \ + \ \lambda\|x\|_1 \tag{1.1.7}$$

In equation (1.1.7), λ is the Lagrangian multiplier. Equations (1.1.6) and (1.1.7) are equivalent with proper choices of λ and t. The literature has a plethora of l_1 minimization techniques [10–14]. In the field of statistics (1.1.7) is known as LASSO (Least Angle Shrinkage and Selection Operator) [15]. The transformation matrix generally considered in previous works are sub Gaussian or Bernoulli matrices. There are two ways to decide on the sparsifying basis, i.e., the A matrix. One way is to exploit the mathematical model of the signal and accordingly choose off the shelf dictionaries like wavelets, contour-lets and the like or the other way is to learn a dictionary that does best on the training data for the given task at hand [16]. This thesis is concerned with the latter.

In the synthesis framework, the most popular algorithm for learning data adaptive dictionaries is the K-SVD algorithm [17]. The K-SVD solves (1.1.8) which is similar to (1.1.5) but now the dictionary is also learnt along with the features X.

$$min_{D,X} \ \|Y - DX\|_F^2 \ s.t. \ \|x_i\|_0 < t \ \forall \ i \tag{1.1.8}$$

Here the matrix X is column sparse and represents the coefficients and i denotes the sample number. Y is the matrix of training data and D is the learnt dictionary. If n is the number of training samples, then the dictionary has a dimension of $m \times p$, with $m << p$, where m is the number of measurements of the each of the signal in Y and X has a dimension of $p \times n$ and thus, Y has a dimension of $m \times n$. Equation (1.1.8) is solved in two steps. The sparse code update step, where the dictionary D is kept fixed and the sparse codes X is updated through an OMP like algorithm which can produce coefficients with predetermined number of zeros. The second step is the codebook update stage, where the dictionary D is learnt keeping the coefficient matrix X constant. The K-SVD algorithm updates each column of the dictionary one at a time along with new values of the contributing coefficients with an aim to minimize the Mean Squared Error (MSE) of the Frobenius term in (1.1.8). The update is a rank 1 SVD update with sparsity constraints provided by the value of t. Such dictionary learning methods have achieved state of the art performances in signal denoising, image inpainting, image restoration and the like [18–21].

1.2 Dictionary Learning for classification

The aforementioned applications are not the only domains where dictionary learning has made an impact. Over the past decade dictionary learning procedures have been extensively used for classification tasks [22]. These learning methods have been applied to various scenarios ranging from face recognition to anomaly detection. All flavours of learning have been used. Supervised, unsupervised and semi supervised techniques have been explored. Few of the most prevalent tools in classification using dictionary learning are the SRC classifier (Sparse Reconstruction Classifier), DKSVD (Discriminative KSVD) and the label consistent version of it, LC-KSVD(Label Consistent KSVD). All the mentioned algorithms are supervised dictionary learning procedures. In SRC [23] procedure, the dictionary is taken as the training samples and iteratively the sparse features over each of the test samples are calculated in the following two stage process:

$$1. \qquad min_z \ \|x - Dz\|_2^2 \ + \ \lambda\|z\|_1 \tag{1.2.1}$$

$$2. \qquad c = argmin_i\|x - D_i\delta_i(z)\| \tag{1.2.2}$$

The first step is the sparse coding step over the dictionary D, which is given by (1.2.1). The dictionary is initialized with the entire training data as columns of the dictionary. This step is performed for each test sample. In the second step, the test vector x is assigned to the c^{th} class according to the rule in (1.2.2). Here $\delta(.)$ is an indicator function which extracts the elements from the i^{th} class in the representation z learnt in the first stage. The first stage is the LASSO problem and can be solved efficiently. SRC has established state of the art results in the domain of Face Recognition.

Another very popular dictionary learning procedure for classification is DKSVD [24]. It adds a representation term to the classical reconstruction term in equation(1.1.3). The resulting optimization problem can be written as

$$min_{D,A,Z} \ \|X - DZ\|_F^2 \ + \ \lambda\|H - WZ\|_F^2 \ \ + \ \lambda_2\|W\|_F^2 \ s.t. \ \forall i, \ \|z_i\|_0 \leq T \tag{1.2.3}$$

Z,D and W are solved together as a KSVD problem. H is a binary matrix and each column of H has a one at the i^{th} position if the sample X_i belongs to class i. If a discriminative term is added to the framework of equation (1.2.3), Label Consistent KSVD or LC-KSVD [25] is obtained. In LC-KSVD, along with a classification error term a sparse code discrimination term is added. Hence at each iteration, the algorithm tries to find the most discriminative sparse code with a small classification error, also reducing the reconstruction error at the same time. The LC-KSVD formulation can be written as:

$$min_{D,W,A,Z}\ \|X - DZ\|_F^2\ +\ \lambda\|Q - AZ\|_F^2\ +\ \mu\|H - WZ\|_F^2\ s.t.\ \forall i, \|z_i\|_0 \leq T \tag{1.2.4}$$

Here Q is a binary matrix whose columns have a 1 where the sample from X and the dictionary D share the same label. The matrix Q is said to be the matrix of discriminative sparse codes and the linear transformation A converts the sparse Z to its most discriminative form Q.

In all these algorithms, one can train on the training samples directly or can code certain characteristics of the data into features, like spatial pyramidal features, cepstrum features, eigen faces and the like, and then train the dictionary on such feature vectors. Examples of such datasets are the CALTECH101 dataset in which the pyramidal features are used and random faces and eigen faces are popular in face recognition tasks.

Another important classification algorithm is the Fisher Discriminant Dictionary Learning (FDDL) [26]. This exploits the Fisher's discrimination criterion to learn discriminative sparse codes. Here, a projection operator is learnt such that the class centroids are separated after projection and at the same time the inter class variances of the features is minimized. Most of the discussed algorithms have their own classification schemes, which typically depend on the nature of the features learnt and the amount of data available (example FDDL uses a Global Classifier (GC) if amount of training samples is small in each class and Local Classifier (LC) if there is sufficient training data for each class). The aforementioned algorithms have state of the art performance in face recognition, hand writing recognition, scene classification and the like for dictionary learning techniques.

1.3 Analysis Dictionary Learning (ADL)

The algorithms discussed in sections (1.1) and (1.2) all belong to the synthesis framework in which given a signal a sparse coefficient vector is learnt such that, the signal can be

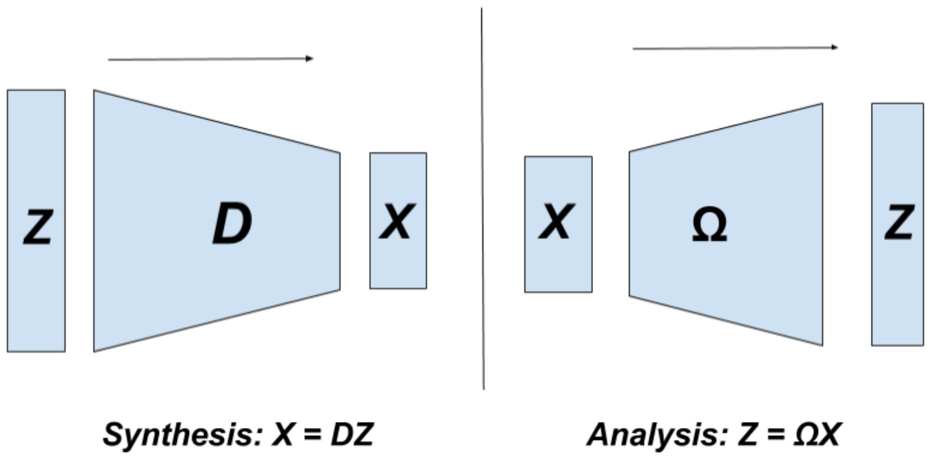


Figure 1.3.1: Synthesis versus Analysis Frameworks

represented by the linear combination of small number of atoms from a suitable dictionary. However, there exists an alternate framework in which the dictionary directly operates on the signal and produces sparse coefficients. This framework is known as the analysis framework. Also, sometimes known as the cospare learning framework, Analysis Dictionary Learning (ADL) has seen a lot of interest in the past few years [27–32]. The difference between the synthesis and analysis learning framework is depicted pictorially in figure (1.3.1).

In the analysis framework we are concerned with the zeros of the signal over the dictionary. The basic framework of ADL can be written as

$$min_{\Omega,z}\ \|z - \Omega x\|_2^2\ s.t.\ \|z\|_0 < T \tag{1.3.1}$$

As is evident from the above equation, the analysis operator Ω acts on the signal x and generates a sparse feature vector z. The dimensions of the analysis dictionary Ω is p x d, the coefficient vector belongs to $\mathbb{R}^p$ and the signal x in $\mathbb{R}^d$. The two most important approaches to ADL has been, Analysis K-SVD [33] and Sparsity Transform Learning [34]. The Analysis K-SVD approach attempts to learn an analysis dictionary which can produce a clean signal from a noisy signal. It is a denoising framework which can be seen from the equation written below,

$$min_{\Omega}\ \|X - Y\|_F^2\ s.t.\ \|\Omega_{\Lambda_i} x_i\| = 0, rank(\Omega_{\Lambda_i}) = d - r, \|w_j\|_2^2 = 1\ \forall\ 1 < j < p \qquad (1.3.2)$$

Here X is the clean estimate of the noisy signal Y. and the dimension of the signal subspace is r and the co- rank is thus $d - r$, which is the rank of the co-support, which is the index set of zeros entries of Ωx or the index set of rows of the dictionary Ω which are orthogonal to the signal. Also, the notion of sparsity as in synthesis framework changes to co-sparsity, which is the number of zeros in Ωx. In mathematical terms, if co sparsity is l then

$$\|\Omega x\|_0 = p - l \qquad (1.3.3)$$

More will be discussed on co sparsity and the analysis framework in the next chapter in the context of classification.

Another paradigm for analysis dictionary learning is Sparsifying Transforms [35–38]. Introduced by Bresler, these suite of algorithms generate a sparsifying transform S, which when applied to the data generates sparse coefficients. The basic formulation for such a problem is same as equation (1.3.1). In this paradigm a lot of importance is given to learning good transform which has full rank and good condition number. Incorporating these constraints the general optimization problem can be written as

$$min_{S,Z} \|Z - SX\|_F^2\ -\ \lambda logdet(S) + \mu \|S\|_F^2\ s.t.\ \|Z_i\|_0 < \tau \qquad (1.3.4)$$

The *logdet* constraint is to ensure that the transform has full rank and no repeated rows and the Frobenius constraint ensures S does not become arbitrarily large.

These analysis operators have been put to use for various applications like image reconstruction, signal denoising and image restoration. However, the analysis framework has not been exploited for classification even though it has been shown that the co sparse model under certain suitable conditions is richer in the number of parameters available than the synthesis model [39]. In the recent past Sumit et. al. [40], used the sparsity transform framework to classify faces and hand written digits. They used the transform learning framework as a feature generator in a completely unsupervised framework and used off the shelf classifier like the SVM to classify the test data. They showed the efficacy of the procedure by comparing against well known synthesis dictionary learning techniques.

In this thesis we start by understanding the analysis framework further and formulate procedures which can exploit the nature of the analysis framework for classification. The next chapter explores various procedures and algorithms for classification. The subsequent chapters discuss empirical results and experiments on standard datasets along with a practical application to Non Intrusive Load Monitoring (NILM).

Chapter 2

Analysis Dictionary Learning for Classification

In this chapter, the analysis framework is discussed in the context of data classification. The intuition behind why such a framework should work is developed and compared with the existing literature. Algorithms are developed which learn discriminative co-sparse codes of a signal and unearth the inherent structure within a signal. Both unsupervised and supervised techniques are presented. Similarities with machine learning structures are discussed and a deep architecture is suggested for analysis dictionary learning.
Feature generation requires the maximum effort for any recognition or classification task. Here analysis dictionary learning is used to generate features from the data, following which an appropriate classifier is used to classify the test data.

2.1 Motivation

It is well known that natural signals have sparse representations in various analytical domains. For example images are known to be sparse in wavelet, curvelets, countorlets and DCT domains. Audio signals have a sparse structure under the Short Time Fourier Transform (STFT). The following figures emphasize the claim. As we see audio in STFT (Figure

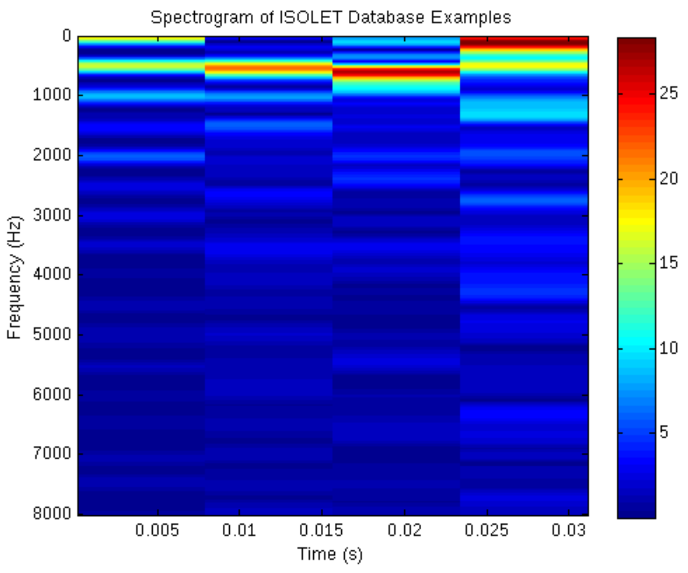


Figure 2.1.1: Audio is Sparse in STFT

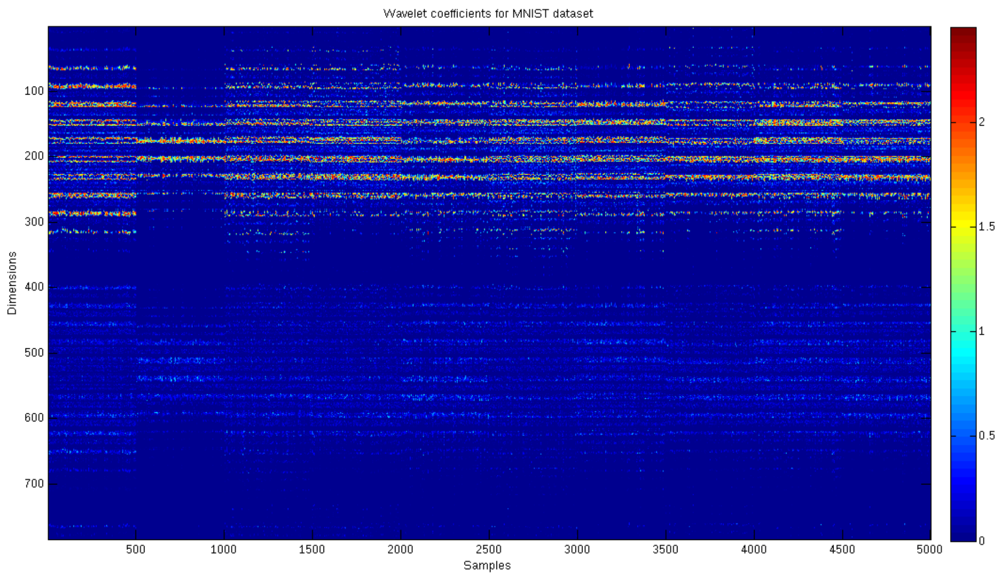


Figure 2.1.2: Images are Sparse in Wavelet Domain

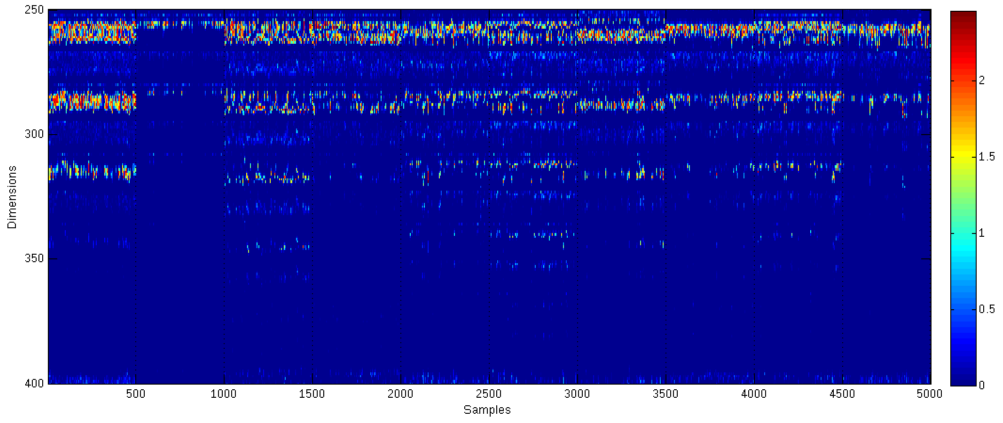


Figure 2.1.3: Discriminative Sparse Codes in Wavelet Domain

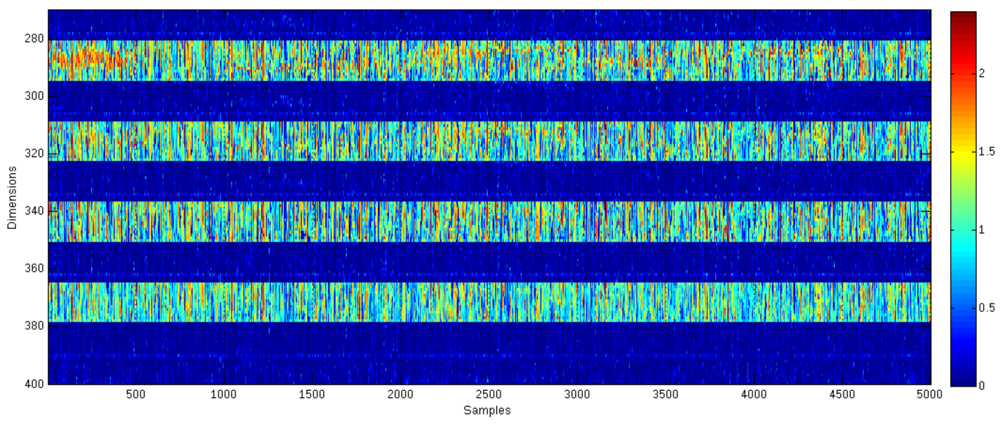


Figure 2.1.4: Discriminative Sparse Codes of Noisy Images in Wavelet Domain

2.1.1) and images in wavelets (here Daubechies wavelets were used to generate the figures) have extremely sparse representations. Also, if the wavelet coefficients are analysed closely we find a continuous sparsity pattern for certain samples and then the pattern changes. This is especially prominent at the lower end of the figure. Figure (2.1.3) exhibits this pattern. Vertical lines demarcate various classes. Hence, such features (i.e. coefficients) also have discriminatory power. A signal belonging to a particular class has a feature signature which is different from the signature of another class. However, when the data is noisy the sparse nature of the representations get distorted. The dictionary, in this case the wavelet dictionary, is not able to discover the inherent structure of the data, when contaminated with noise. This is easily seen in figures (2.1.3) and figure (2.1.4). Figure (2.1.5) and (2.1.6) show

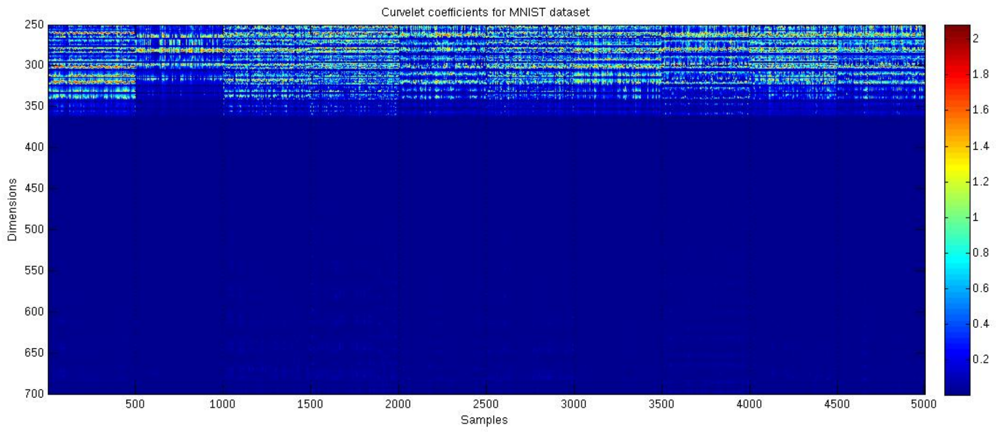


Figure 2.1.5: Discriminative Sparse Codes in Curvelet Domain

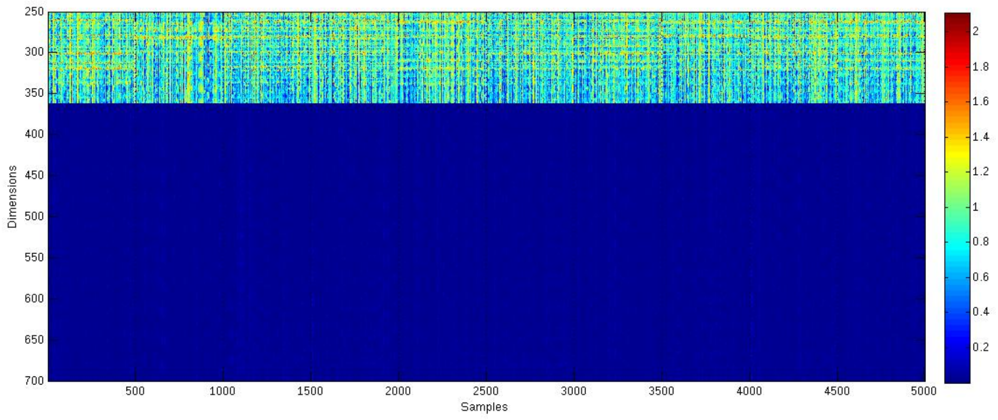


Figure 2.1.6: Discriminative Sparse Codes of Noisy Images in Curvelet Domain

Curvelet features for MNIST and noisy MNIST data. As we see the variation in patterns across classes is similar to the case in which wavelets are used. Hence, the question arises, is it possible to learn a dictionary which generates sparse features under two circumstances

1. When we do not explicitly know the domain in which the given signal is sparse
2. When we have a noisy version of the signal

In this chapter we will find answer to the above question and formulate a methodology to generate discriminative features.

2.2 Analysis Learning Framework

The basic learning problem in Analysis Dictionary Learning (ADL), is to learn a dictionary D such that when it operates on the data X it produces a sparse feature vector Z. Mathematically the learning problem can be stated as,

$$min_{D,Z} \ ||Z - DX||_F^2 \ \ s.t. \ \ ||Z_i||_0 < T_0 \tag{2.2.1}$$

Here i is the number of training samples available which are stored as the columns of the matrix X. Each column of X contains a sample from class c. Generally, there are multiple classes, i.e., $C > 2$, where C are the total number of classes. Z as can be easily seen to be column sparse with T_0 zeros in each column and represents the feature for the corresponding training signal which generated it when coded over D.

As is evident, the analysis framework is a signal model in which the analysed signal is assumed to be sparse. If $D \in \mathbb{R}^{pxd}$, $x \in \mathbb{R}^d$, then $\|z\|_0$ is small, where $z = Dx \in \mathbb{R}^p$. Thus, in the analysis model we are interested in the zeros of the signal. An important concept related to the zeros of the signal is the co-sparsity of the signal.

Let us start with a redundant analysis dictionary $D \in \mathbb{R}^{pxd}$ where $p > d$. Therefore, only d or less than d number of rows of D can be linearly independent. If rank of D is d then x has to be the zero vector. So, $\|Dx\|_0 = 0$. If $x \neq \mathbf{0}$ then there will be atleast $p - d$ no. of non zeros in z, because if l is the rank of D such that $l < d$, then $D_\Lambda x = 0$, where Λ indexes the l rows and the no. of zeros in z become $p - l$. Therefore, co-sparsity can be defined as $l = p - \|Dx\|_0$, i.e., the number of zeros in z and the index set of the zero entries in z is called the co-support and is denoted by Λ.

The analysis model can also be regarded as a generator framework which generates signals of a given co-sparsity. Consider an analysis dictionary $D \in \mathbb{R}^{pxd}$ and take randomly chosen l rows with $l < d$. Take a vector v with randomly generated Gaussian entries and project it to the subspace orthogonal to D_Λ. The vector so obtained is a l co-sparse vector.

$$x = (I_d - D_\Lambda^T (D_\Lambda D_\Lambda^T)^{-1})v$$

The co-rank of a signal x is defined as the rank of D_Λ. This data modelling framework belongs to a larger class of data models known as the Union of Subspaces(UoS). Given an analysis operator D, a signal x which is l co-sparse with respect to the rows of D can be thought of lying in the space orthogonal of the space spanned by the rows indexed by the co-support Λ. If $\mathcal{V}_\Lambda = span(d_i, i \in \Lambda)^\perp$ then

$$x \in \cup_{i \in \Lambda : |\Lambda| = l} \mathcal{V}_\Lambda$$

Therefore, the co-sparse signal belongs to the union of all the possible $\binom{p}{l}$ subspaces of dimension $d - l$.

A similar approach for synthesis models can be obtained. If $A \in \mathbb{R}^{mxn}$ is a synthesis dictionary and x has a k sparse representation in A, then x lies in the union of $\binom{n}{k}$ subspaces of dimension k. In order to compare the two models let us consider a situation where the dimensions of the subspaces are same, i.e., $k = d - l$. Also allow the same over completeness i.e. $p = n = 2d$. Then number of synthesis subspaces available are

$$log_2 \binom{n}{k} \approx n.H\left(\frac{k}{n}\right) \approx k.log_2\left(\frac{n}{k}\right)$$

where $H(t) = -tlog_2 t - (1-t)log_2(1-t)$ and using the fact that $k << n$ and Stirling's Approximation. The number of available analysis subspace are

$$log_2 \binom{p}{l} \approx n.H\left(\frac{k}{n}\right) \approx n.H(0.5) = n$$

As an example let us take $p = n = 2d = 700$, $l = 300, k = 50$, then number of synthesis subspaces available are 191 and the number of analysis subspace available are 700. Therefore, unless $\frac{d}{n} \approx 1$, the number of available analysis subspaces far exceeds the synthesis subspace. A more extensive analysis can be found in [29]. Therefore, using the analysis framework for classification maybe advantageous. However, getting high co-sparsity is not easy especially when we want $d \leq l < p$. In order for this to happen the rows of the analysis operator should exhibit high dependencies. However these intricacies will not be dwelt into further as this work is not directly concerned with the co-sparsity of the signal rather it deals with the subspaces that characterise the signal and the nature of the features generated by such a dictionary. The assumption is that signals belonging to different classes will exhibit different co-sparsity patterns and thus belong to different subspaces. Supervised and unsupervised settings are explored. The following sections discuss various algorithms by which we can generate discriminative features for signals belonging to different classes. As specified above the term feature refers to the sparse coefficient vector z

2.3 Unsupervised ADL

The optimization problem in (2.2.1), as noted before, is a NP hard problem. The l_0 norm constraint can be relaxed to the l_1 norm which is the nearest convex relaxation of the l_0 norm. Thus writing the Lagrangian for equation (2.2.1) with the norm relaxation we get,

$$min_{D,Z} \|Z - DX\|_F^2 + \lambda \|Z\|_1 + \mu \|D\|_F^2 \tag{2.3.1}$$

$$s.t. \|d\|_2^2 = 1$$

Here, λ is the Lagrangian multiplier and acts a regularizer trading off the representation error and the l_1 norm penalty on the features. Therefore, λ controls how sparse the feature is going to be. With l_1 norm penalty there won't be exact zeros in the solution but when λ

is set properly different classes do tend to have zeros at different positions. The evaluation of the features do not incorporate label information and hence equation (2.3.1) gives rise to an unsupervised learning procedure. The constraint on the Frobenius norm of the dictionary is to prevent scale ambiguities. We also constraint each atom to have unit norm so as to prevent D to become a zero matrix.

The procedure for solving (2.3.1) is an iterative procedure of alternate minimization of D and Z. In the first step Z is held constant and D is updated while in the second step D is held constant and Z is updated. The update for D is a simple least squares update. Holding Z constant the second term drops from (2.3.1). Representing (2.3.1) with f and Differentiating with respect to D, we get,

$$\nabla_D f = D(XX^T + \mu I)ZX^T) \tag{2.3.2}$$

Equating (2.3.2) to zero, we get the update equation for D as

$$D = ZX^T(XX^T + \mu I)^{-1} \tag{2.3.3}$$

Here I stands for the identity matrix of the appropriate dimension. While solving for Z, we note that the problem can be cast as a proximal gradient problem of the form

$$f(x) + g(x)$$

where $f(x)$ is a convex function and $g(x)$ is a smooth differentiable function not necessarily convex. For such a problem the solution can be obtained by the proximal operator,

$$prox_f(x) = argmin_u(f(u) + \frac{1}{2t}\|u - x\|_2^2) \tag{2.3.4}$$

and the update equation at iteration k becomes

$$x^k = prox_f(x^{k-1} - t\nabla g(x^{k-1})), k > 1 \tag{2.3.5}$$

For the problem defined by equation (2.3.1), the proximal operator reduces to the soft thresholding operator and the Z update at iteration k can be written as

$$Z^k = SoftThresholding(D^k X, threshold) \tag{2.3.6}$$

Where the soft thresholding operator is defined as follows,

$$SoftThreshold(x, T) := sign(x)max(0, |x| - T) \tag{2.3.7}$$

The threshold value in (2.3.7) for unsupervised ADL is the regularizer λ for equation (2.3.1). The complete algorithm for solution to equation (2.3.1) is written in the following algorithm:

```
Input: X, tol, $\lambda$,$\mu$
initialization D, Z = DX;
while $\|Z - DX\|_F^2 < tol$ do
    At iteration k;
    $D^k = Z^{k-1}X^T(XX^T + \mu I)^{-1}$;
    $Z^k = SoftThreshold(D^k X, \lambda)$;
    Normalize Atoms of $D$;
    $k = k + 1$;
end
```

Algorithm 1: Unsupervised ADL

The above algorithm does not use any label information from the training signals. Thus it can be treated as an unsupervised learning scheme and we name it Unsupervised Analysis Dictionary Learning (ADL).

In the next section, various discriminative constraints are discussed which can be easily incorporated into this analysis dictionary learning framework. A number of supervised dictionary learning techniques are discussed along with their intuition and solutions to the respective optimization problems. These supervised algorithms promote group co-sparse patterns among the signals of the same class and work as effective feature extractors for clean and noisy signals.

2.4 Supervised Analysis Dictionary Learning

Supervised dictionary learning involves exploitation of training sample labels to learn features adapted to a particular class. As previously mentioned the matrix X contains training signals stacked as columns. Each of the signals belong to a particular class c. Signals belonging to the same class will be represented as X_c. Intuitively, signals which belong to the same class should have similar features or similar signatures. This is also in line with the concept of group sparsity. Thus, three supervised analysis dictionary learning schemes are introduced to promote the group co-sparsity within signals of the same class.

1. Row sparsity constrained ADL
2. Low Rank constrained ADL
3. Label Consistent ADL

Each of the above algorithm will be discussed in detail in the following subsections.

2.4.1 Row Sparsity Constrained ADL

When signals belong to the same class c and are stacked in columns of a matrix as $[X_1|X_2|...|X_c]$, then they can be assumed to have a similar feature signature or in other words along the rows of the matrix $Z_c(= DX_c)$ the signals should have few non zero values at similar positions, or succinctly, the matrix Z_c should be row sparse. Here, we promote signals from the

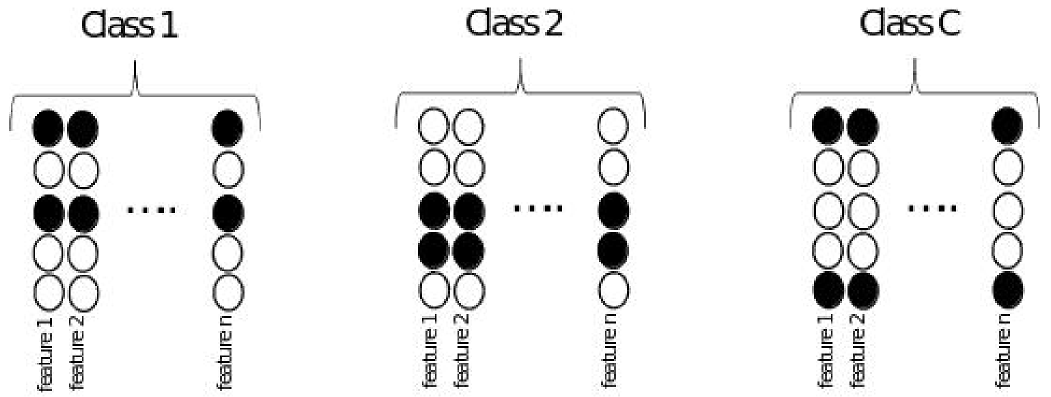


Figure 2.4.1: Row Sparse Features for Various Classes

same class to belong to the same subspace.

$$X_c \in \cup_{i \in \Lambda_c : |\Lambda_c| = l_c} \mathcal{V}_{\Lambda_c}$$

Overlap between the subspaces of the various classes are allowed. The following figure displays the scheme more efficiently,

As is evident from the figure such a scheme should be able to generate discriminative sparse features between various classes.

The row sparse penalty is easily implemented by the mixed $l_{2,1}$ norm [41]. The $l_{2,1}$ constraint acts on each class separately and constraints the l_2 norm of the rows in the matrix Z_c . It performs l_1 thresholding of the l_2 norm of the rows. The $l_{2,1}$ norm is defined as

$$\|X\|_{2,1} = \Sigma_{j=1}^{n} \|X^{j\rightarrow}\|_2 \tag{2.4.1}$$

Where $\|X^{j\rightarrow}\|$ is the vector whose entries form the j^{th} row of X.

The optimization problem can be written as

$$min_{D,Z} \|Z - DX\|_F^2 + \lambda \Sigma_C \|Z_c\|_{2,1} + \mu \|D\|_F^2 \tag{2.4.2}$$

$$s.t. \|d\|_2^2 = 1$$

The total number of classes is denoted by C. The above problem is also solved via alternate minimization between D and Z. The update for D remains the same as equation (2.3.3). The update for Z is done on a class by class basis by solving the following optimization problem at each iteration

$$min_{Z_c} \|D_k X_c - Z_c\|_F^2 + \lambda \|Z_c\|_{2,1} \tag{2.4.3}$$

Equation (2.4.3) can be solved efficiently via the method stated in [42] in a closed form. The update equation for the matrix Z_c can thus be written at iteration k as

$$Z_c^k = signum(D_k X_c) max(0, |D_k X_c| - \frac{\lambda}{\alpha}\Gamma) \tag{2.4.4}$$

where α is the maximum eigen value of $D^T D$. Γ at iteration k is given by the following expression

$$\Gamma = diag(\|Z_{k-1}^{j\rightarrow}\|)|Z_{k-1}^{j\rightarrow}| \ \forall j \tag{2.4.5}$$

Thus the row sparse ADL algorithm becomes,

```
**Input**: X, tol, $\lambda$, $\mu$
initialization D, Z = DX ;
while $\|Z - DX\|_F^2 < tol$ do
    At iteration k;
    $D^k = Z^{k-1}X^T(XX^T + \mu I)^{-1}$;
    for $class = 1 : C$ do
        $Z^k_{class} = signum(D^k X_{class}) max(0, |D_k X_{class}| - \frac{\lambda}{\alpha}\Gamma)$;
    end
    $k = k + 1$;
    Normalize Atoms of $D$;
end
```

Algorithm 2: Row Sparse ADL

In the above algorithm at each iteration we update the feature matrix Z class wise and then learn a dictionary which would generate row sparse features.

2.4.2 Low Rank ADL

Another way of looking at similarity of features is to consider linear dependence of the features. Signals belonging to the same class will have higher dependencies among themselves and a lower dependency on the other class. Thus if features belonging to the same class are stacked as columns in a matrix they should form a low rank matrix. The input data may or may not be low rank due to various perturbations like noise, rotation and scaling. However, a dictionary maybe learnt which cleans, scales and aligns the signals thus resulting in a low rank feature matrix Z_c.

Instead of imposing constraints on the dictionary to be rotation corrective and carry out denoising, a low rank constraint may be imposed upon the feature matrix which in an alternate minimization procedure learns a dictionary which brings out the dependencies among the various classes. This also imposes discriminative constraints on the features. The following figure explains the scheme more clearly.

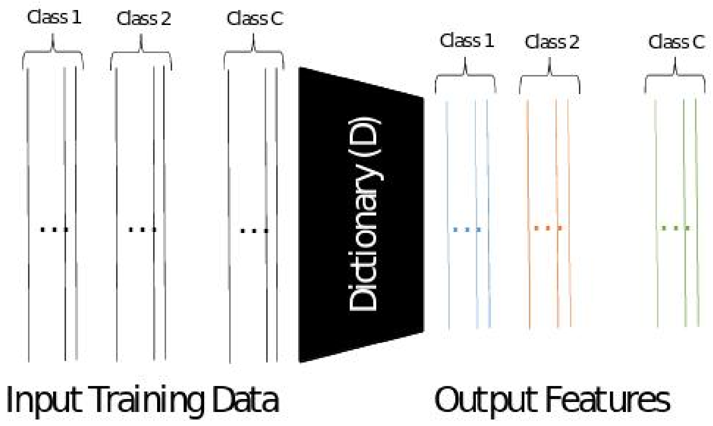


Figure 2.4.2: Dependencies Among Features of Various Classes

In the above figure the different colors denote the respective linear dependencies in each class. This can be easily formulated in term of the nuclear norm of the feature matrix of each class Z_c. Nuclear norm is the sum of the singular values of matrix. For a m x n matrix X, nuclear norm can be written as

$$\|X\|_* = \Sigma_{i=1}^{min(m,n)} \sigma_i \tag{2.4.6}$$

The nuclear norm is the nearest convex surrogate of the rank of a matrix. Using this definition we can formulate the optimization problem as follows,

$$min_{D,Z} \|Z - DX\|_F^2 + \lambda \Sigma_c \|Z_c\|_* + \mu \|D\|_F^2 \tag{2.4.7}$$

$$s.t. \|d\|_2^2 = 1$$

As before an alternate minimization approach is adopted for the solving (2.4.7). The update rule for D is same as equation (2.3.3). Like in the case of row sparse ADL, the Z

matrix update is a class wise affair. However, the update rule is different. Z can be easily solved via singular value shrinkage [43]. The class wise optimization problem is given below

$$min_{Z_c}\Sigma_c\|D_kX_c - Z_c\|_F^2 + \lambda\|Z_c\|_* \tag{2.4.8}$$

Details of solving (2.4.7) is given in the following algorithm.

```
**Input**: X, tol, $\lambda$, $\mu$
initialization D, Z = DX ;
**while** $\|Z - DX\|_F^2 < tol$ **do**
    At iteration k;
    $D^k = Z^{k-1}X^T(XX^T + \mu I)^{-1}$;
    **for** $class = 1 : C$ **do**
        $U\Sigma V^T = SVD(D_kX_c)$;
        $Z^k_{class} = USoft_{\lambda/2}(\Sigma)V^T$;
        where $Soft_{\lambda/2} = diag(\Sigma).max(0, diag(\Sigma) - \lambda)$;
    **end**
    Normalize Atoms of $D$;
    $k = k + 1$;
**end**
```

Algorithm 3: Low Rank ADL

2.4.3 Label Consistent ADL

The aforementioned supervised learning algorithms do not take into consideration labels of the training data explicitly. They learn the feature in a class wise manner forcing the dictionary to be able to produce a good representation. Taking cue from LC-KSVD [25], a Label Consistent ADL (LC-ADL) is proposed. Here a classifier mapping is learnt from the feature space to the target space. This provides additional supervision in the form of a label consistency term. The map learnt from the feature space to the target space also affects the learning of the dictionary and via the target label mapping supervision is introduced. The optimization problem can be stated as

$$min_{D,Z,W}\|Z-DX\|_F^2+\lambda_1\|Z\|_1+\lambda_2\|H-WZ\|_F^2++\mu_1\|D\|_F^2+\mu_2\|W\|_F^2 \tag{2.4.9}$$

$$s.t.\|d\|_2^2=1$$

Here, H is a binary matrix of label targets. For example, if there are a total of three classes, then corresponding to the signal belonging to class 1 H will have a vector of $(1,0,0)^T$, for signals belonging to class 2, corresponding columns of H will have the vector $(0,1,0)^T$ and for class 3 the corresponding columns H will be $(0,0,1)^T$. W is a matrix which maps the features to labels. The first Frobenius term is the feature generation term while the second Frobenius term is the label consistency term. In this case the optimization problem is over three variables D, W and Z. The three sub problems become

1. $$min_D\|Z-DX\|_F^2+\mu_1\|D\|_F^2 \tag{2.4.10}$$

2. $$min_W\|H-WZ\|_F^2+\mu_2\|W\|_F^2 \tag{2.4.11}$$

3. $$min_Z\left\|\begin{pmatrix} DX \\ \sqrt{\lambda_2}H \end{pmatrix}-\begin{pmatrix} I \\ \sqrt{\lambda_2}W \end{pmatrix}\right\|+\lambda_1\|Z\|_1 \tag{2.4.12}$$

Equations (2.4.10) and (2.4.11) can easily be solved via a least squares update as in equation (2.3.3), equation (2.4.12) can be solved via Iterative Soft Thresholding Algorithm (ISTA) [11]. ISTA solves a problem of the following type:

$$min_x \|y - Tx\|_F^2 + \lambda\|x\|_1 \tag{2.4.13}$$

Solution to the above optimization problem is an iterative procedure and has the following basic steps:
Landweber Iteration: $B = x_{k-1} + \frac{1}{\alpha}T^T(y - Tx_{k-1})$
SoftThresholding: $x_k = signum(B)max(0, |B| - \frac{\lambda}{2\alpha})$

α is the maximum eigenvalue value of $T^T T$. The LC-ADL algorithm can be summarized as follows:

```
Input: X, tol, $\lambda_1$, $\lambda_2$, $\mu_1$,$\mu_2$
initialization D, Z = DX, W ;
while $\|Z - DX\|_F^2 < tol$ do
    At iteration k;
    $D^k = Z^{k-1}X^T(XX^T + \mu_1 I)^{-1}$;
    $A = \begin{pmatrix} D_k X \\ \sqrt(\lambda_2) H \end{pmatrix}$;
    $B = \begin{pmatrix} I \\ \sqrt(\lambda_2) W_{k-1} \end{pmatrix}$;
    $Z_k = ISTA(A, B, \lambda_1)$;;
    $W_k = HZ_k^T(Z_k Z_k^T + \mu_2 I)^{-1}$;
    Normalize Atoms of $D$;
    $k = k + 1$;
end
```

Algorithm 4: Label Consistent ADL

2.5 Connection With RBMs

Restricted Boltzmann Machine (RBM) is a learning unit which learns a latent representation of an input applied to it. It has a fully connected latent and input layer. It is a restricted because units from the same layer are not interconnected. The energy function a RBM minimizes to learn a feature is the Boltzmann function. It learns the representation at its hidden layer and the connection weights. If the energy function is the euclidean cost then the entire RBM framework reduces to the analysis dictionary learning framework.

In a RBM the aim is to learn the network weight and the output features such that the similarity between the projected data (at the input) and the features ($Z^T DX$) is maximized. In the proposed method, the cost function is modified - instead of maximizing similarity; the Euclidean distance between the projection of the data (DX) and the generated features

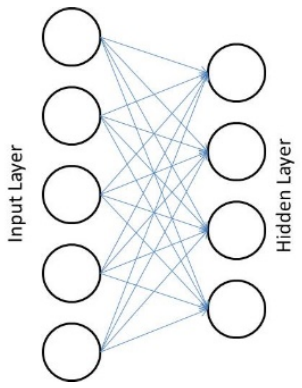


Figure 2.5.1: Generic Structure of a RBM

(Z) is minimized.

2.6 Deep Analysis Dictionary Learning

A RBM is generally a part of a Deep Belief Network (DBN). In a DBN RBMs are stacked on top of each other and the features learnt by a particular level is then fed as input to the consequent level. So, a DBN is a multilayer feature extractor, and hence the word deep. Each RBM in a DBN is trained in a standalone fashion and then is fine tuned using back-propagation. It is a completely unsupervised method of learning and is used as a feature generator for a supervised algorithm. A similar approach can be employed with ADL as the architecture of the basic units is almost the same. Single layer analysis units can be stacked to form a deeper network and can be tuned with the any of the algorithms presented in section 2.3 and 2.4. The optimization problem for such a task can be formulated as follows

$$min_{Z^{(n)}D^{(n)}}\|D^{n-1}D^{n-2}...D^1X - Z^n\|_F^2 + \mu\Sigma_i\|D^{(i)}\|_F^2 + \lambda R(Z^{(n)})$$

$$s.t.\|d_j^{(n)}\|_2 = 1$$

Here, $R(.)$ can be any of the penalties that have been applied to Z . The number of

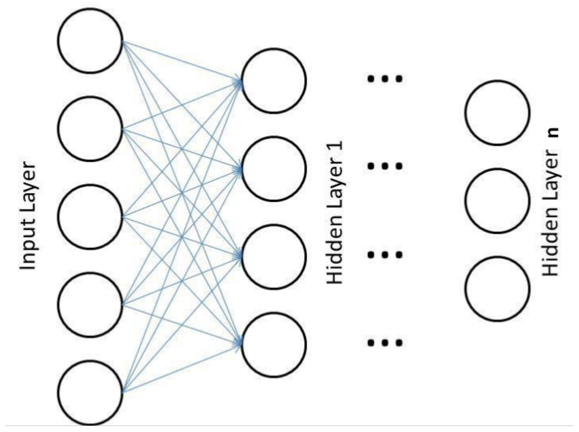


Figure 2.6.1: Deep Analysis Dictionary Learning

variables involved is large and hence direct optimization is not feasible. Thus a greedy layer wise training may work in training such a deep dictionary learning framework.

2.7 Advantages of Analysis Learning For Classification

The advantage of the analysis learning framework in a classification scenario is its ability to generate a feature vector fast at test time. In all the algorithms once the learning procedure is over, during test the feature is generated by multiplying the incoming test data with the learnt analysis dictionary. If x_{test} denotes the test sample and z_{test} denotes the corresponding feature, then the relation between z_{test} and x_{test} can be written as

$$z_{test} = Dx_{test} \tag{2.7.1}$$

However, in case of an synthesis learning framework the following l_1 optimization problem needs to solved

$$min_z \|x_{test} - Dz_{test}\| + \lambda \|z_{test}\|_1 \tag{2.7.2}$$

Hence, the complexity of generating a feature in a synthesis framework increases. The complexity of generating a feature in an analysis framework with a dictionary size of mxn is

$O(mn)$, which is the same as the complexity of a vector matrix multiplication. On the other hand solving (2.7.2) is iterative in nature and the per iteration cost is $O(mn^2)$ and $O(n^{0.5})$ number of iterations is required, thus the total complexity becomes $O(mn^{2.5})$, which is large compared to $O(mn)$.

In the next chapter the algorithms discussed in this chapter will be used to gauge classification performance on standard datasets. Empirical evidence will be given for their performance, convergence and the features generated by these algorithms will be studied.

Chapter 3

Experimental Results and Discussion

This chapter discusses the various experiments and simulations which were performed to show the efficacy of the proposed algorithms. The empirical convergence of the algorithms are discussed along with the parameter selection procedure. Then the performance of the algorithms on various datasets and their robustness to noise and missing data is presented. We compare the proposed work with state of the art synthesis learning algorithms, deep learning and the analysis classification procedure suggested by Chellapa et. al.. A comparison of the features generated by our algorithms and the algorithm proposed by Chellapa et. al. is presented. Lastly, we discuss the advantage of the proposed work with respect to training time and time taken for test feature generation. The four proposed algorithms were coded and executed in MATLAB. Simulations were performed on an Intel machine with i5 processor working at 1.72 GHz. MATLAB version R2014a was used.

3.1 Empirical Convergence

The following graphs show the convergence of the proposed algorithms. The objective function is plotted against the iterations of the algorithm.

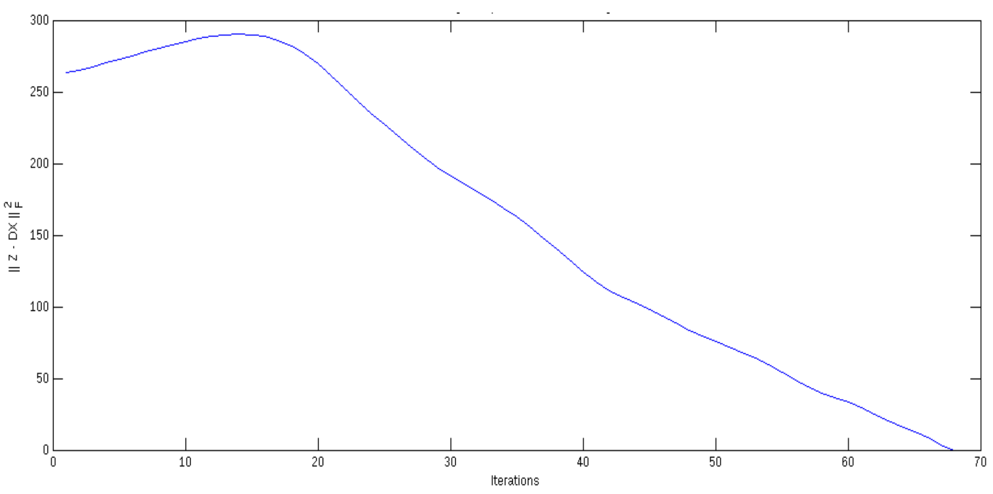


Figure 3.1.1: Empirical Convergence for Unsupervised ADL

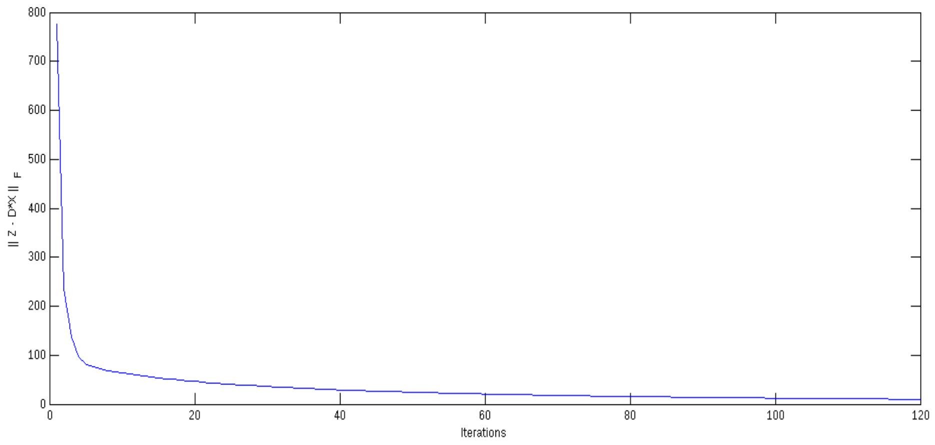
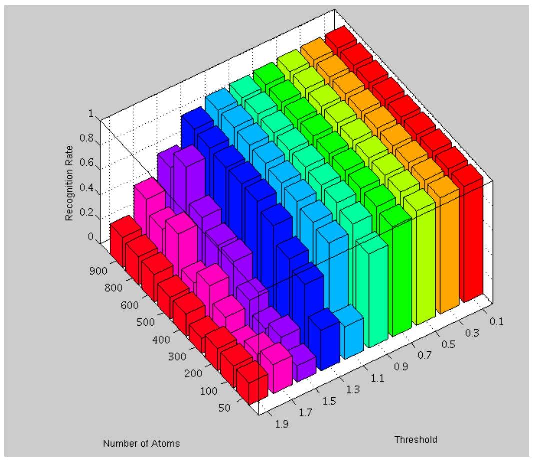


Figure 3.1.2: Empirical Convergence for Supervised ADL

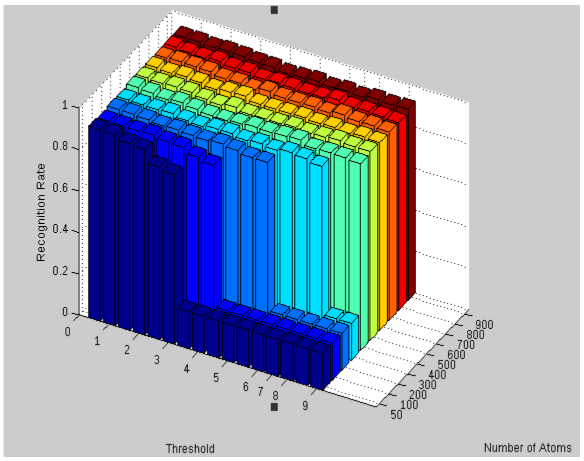
Figure (3.1.2) shows the convergence for row sparse ADL, while figure (3.1.3) shows the convergence for low rank ADL. The convergence for the generation term and label consistency term in Label Consistent ADL is shown in figure (3.1.4) and figure (3.1.5) respectively.

3.2 Parameter Selection

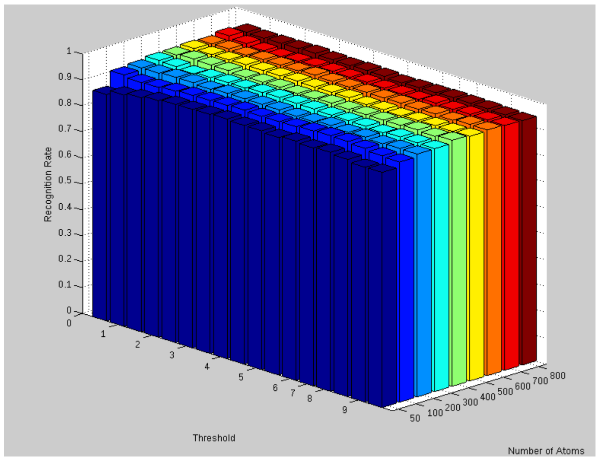
The following bar graphs (figure 3.2.1.) show the variation of recognition rate with the parameters of the proposed algorithms. The dictionary size and the threshold parameters which yield the best recognition rate are used to cross validate a classifier. The classifiers considered were Neural Networks and SVMs.



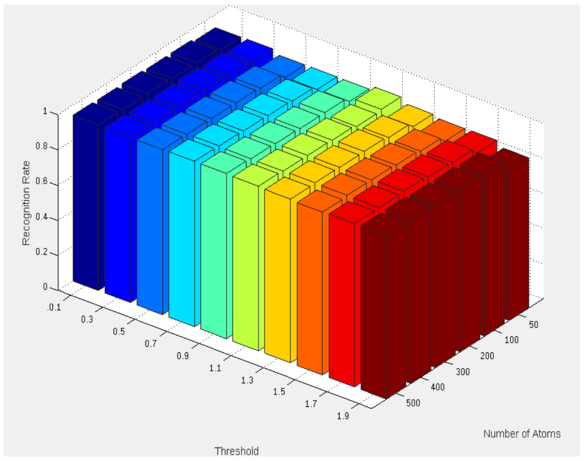
(a) Variation of Recognition Rate With Parameters for Unsupervised ADL



(b) Variation of Recognition Rate With Parameters for Row Sparse ADL



(c) Variation of Recognition Rate With Parameters for Low Rank ADL



(d) Variation of Recognition Rate With Parameters for LC-ADL with $\lambda_2 = 1.6$

Figure 3.2.1

3.3 Results on Digit Recognition

The proposed algorithms were used as feature generators for digit recognition tasks. The datasets used for digit recognition were the MNIST [44] dataset, it's variations [45] and the USPS dataset. The original MNIST dataset has 60 thousand training images and 10 thousand test images of dimensions 28x28 ($\mathbb{R}^{784}$). The dataset has five variations. The *mnist-rot*, where the digits have been rotated by an angle chosen uniformly between 0 and 2π radians. Background pixels whose value is uniformly chosen between 0 and 255 is inserted in *mnist-back-rand.* Random patches from black and white images are inserted as backgrounds in *mnist-back-image.* The perturbations in *mnist-back-image* and *mnist-rot* are combined in *mnist-rot-back-image.* Each of these variations have 12 thousand training samples and 50 thousand test samples of the same dimension as the original dataset. The features learnt from various proposed algorithms were fed to a neural network or a SVM classifier. The proposed algorithms were compared against Label Consistent KSVD (LC-KSVD), Disciminative KSVD (DKSVD) [24], the transform learning classification method proposed by Chellapa et. al.[40] (Chellapa-ADL) and two deep learning methods the Deep Belief Networks with RBMs as the basic unit and Stacked Denoising Auto Encoders(SDAE). Chellapa ADL uses SVM with RBF kernels as classifiers, LC-KSVD and DKSVD have their inbuilt linear classifiers. Logistic regression has been used along with DBN-3 and SDAE for classification. The toolbox used for DBN-3 and SDAE can be found in [46]. For wavelets, curvelets and DCT SPARCO Toolbox was used [47].

Before going into the recognition rate results, a discussion on the features generated by the proposed algorithms are in order. The next figure shows the MNIST dataset and its variations.

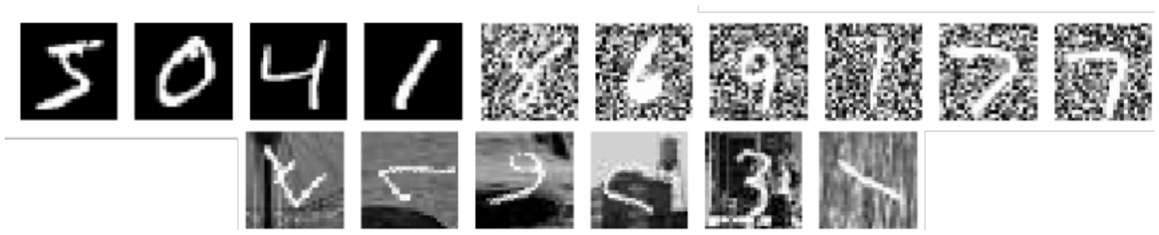


Figure 3.3.1: MNIST Dataset and its Variations

In the experiments the original MNIST and *mnist-rot* datasets were not preprocessed. The *mnist-back-image*, *mnist-back-rand* and *mnist-rot-back-image* datasets were preprocessed by thresholding the images with a threshold of 0.9. The final images fed to the algorithms after preprocessing are shown below,

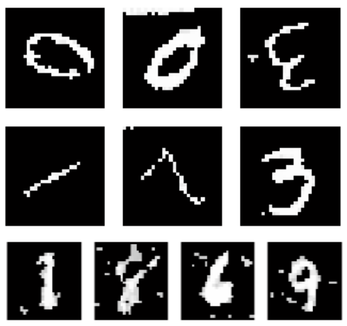
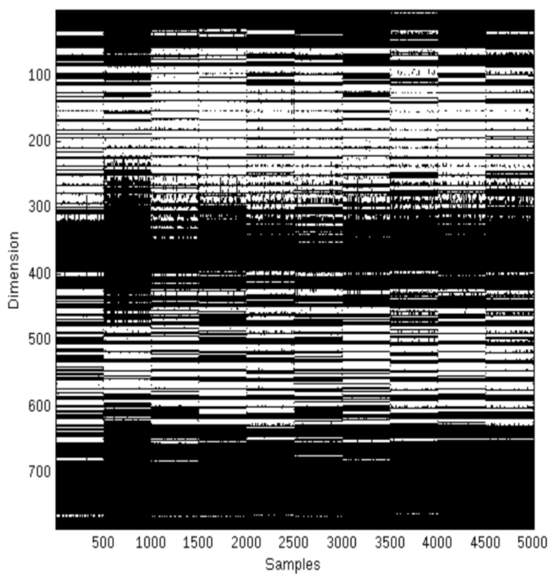


Figure 3.3.2: MNIST Dataset and its Variations

Let us look at the wavelet features of the clean MNIST and the *mnist-back-image*. In figure (3.3.3) the black rows define the rows in the feature which had less than 10% contribution to the $l_{2,1}$ energy of the feature. On the left the clean MNIST features are presented. Here certain discriminative rows can be found between various classes. Five hundred samples from each class was taken. On the right we find that under noise the features lose their discriminative nature. Figure 3.3.4 show the features learnt by the Row Sparse ADL algorithm. Here also the black rows define those coefficients in the feature whose contribution to the $l_{2,1}$ norm was less than 10%. In this case better discriminative nature of the features is observed across each class. Even though, a lot of subspace options are blinded out in the noisy case but one can still find atoms which are unique to each of the class. To drive the point home



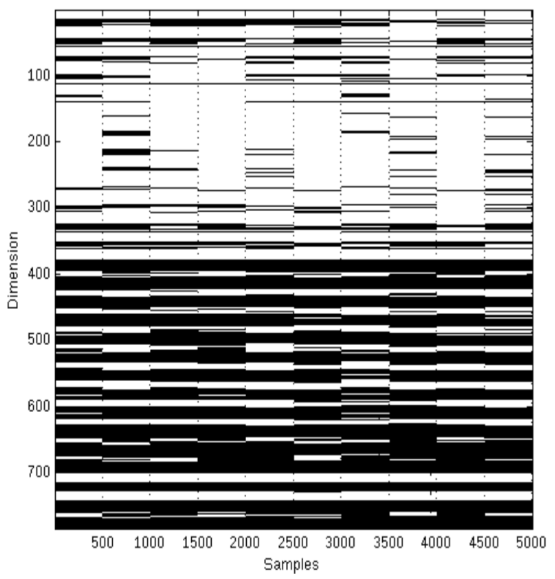
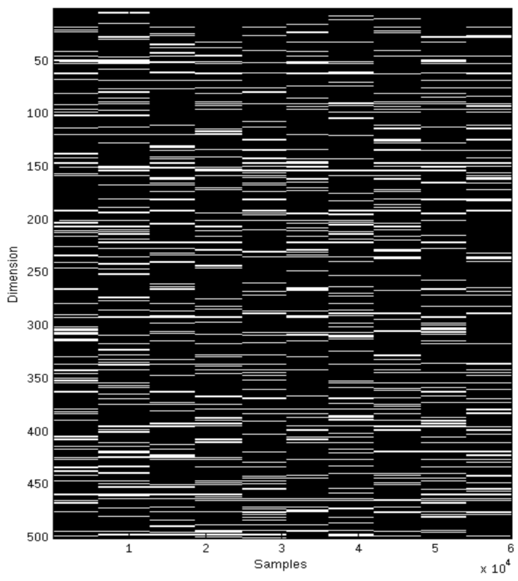


Figure 3.3.3: Wavelet Features For MNIST Clean **(on Left)** and MNIST with background images **(on Right)**

that adapted analysis dictionaries do work better, let us look at two other parameters, the amount of overlap between atoms of various classes and the recognition rate with wavelet features and features generated by Row Sparse ADL.



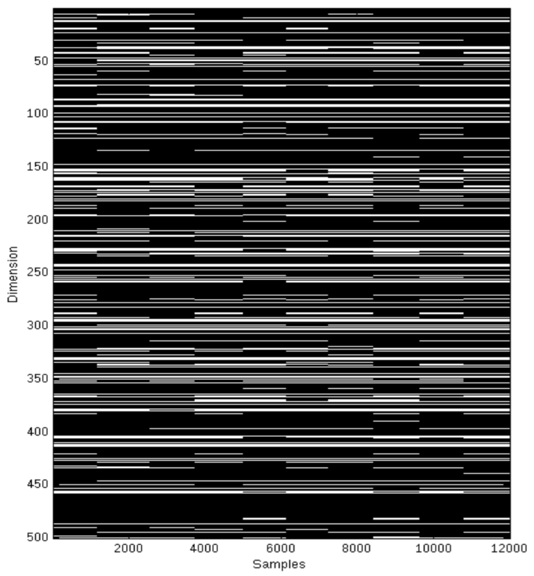


Figure 3.3.4: Features For MNIST Clean **(on Left)** and MNIST with background images **(on Right)** for Row Sparse ADL

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	1	0.12	0.17	0.25	0.12	0.25	0.16	0.18	0.2	0.17
Class 1	0.12	1	0.17	0.21	0.17	0.15	0.17	0.16	0.24	0.14
Class 2	0.17	0.17	1	0.18	0.17	0.18	0.13	0.14	0.22	0.19
Class 3	0.25	0.21	0.18	1	0.15	0.32	0.16	0.19	0.29	0.19
Class 4	0.12	0.17	0.17	0.15	1	0.2	0.17	0.24	0.22	0.35
Class 5	0.25	0.15	0.18	0.32	0.2	1	0.19	0.22	0.27	0.17
Class 6	0.16	0.17	0.13	0.16	0.17	0.19	1	0.13	0.18	0.16
Class 7	0.18	0.16	0.14	0.19	0.24	0.22	0.13	1	0.17	0.32
Class 8	0.2	0.24	0.22	0.29	0.22	0.27	0.18	0.17	1	0.26
Class 9	0.17	0.14	0.19	0.19	0.35	0.17	0.16	0.32	0.26	1

Table 3.3.1: Overlap Rate for Co-Supports from Different Classes For Row Sparse ADL Using MNIST Clean Images

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	1	0.71	0.76	0.79	0.75	0.8	0.75	0.68	0.78	0.72
Class 1	0.71	1	0.7	0.73	0.71	0.72	0.74	0.68	0.82	0.72
Class 2	0.76	0.7	1	0.8	0.71	0.75	0.77	0.65	0.75	0.68
Class 3	0.79	0.73	0.8	1	0.77	0.83	0.74	0.73	0.81	0.75
Class 4	0.75	0.71	0.71	0.77	1	0.79	0.71	0.77	0.82	0.82
Class 5	0.8	0.72	0.75	0.83	0.79	1	0.75	0.69	0.83	0.75
Class 6	0.75	0.74	0.77	0.74	0.71	0.75	1	0.66	0.77	0.7
Class 7	0.68	0.68	0.65	0.73	0.77	0.69	0.66	1	0.76	0.85
Class 8	0.78	0.82	0.75	0.81	0.82	0.83	0.77	0.76	1	0.8
Class 9	0.72	0.72	0.68	0.75	0.82	0.75	0.7	0.85	0.8	1

Table 3.3.2: Overlap Rate for Co-Supports from Different Classes For Wavelets Using MNIST Clean Images

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	1	0.62	0.64	0.64	0.61	0.71	0.68	0.67	0.69	0.66
Class 1	0.62	1	0.76	0.75	0.68	0.75	0.72	0.72	0.72	0.73
Class 2	0.64	0.76	1	0.7	0.71	0.71	0.75	0.66	0.73	0.71
Class 3	0.64	0.75	0.7	1	0.67	0.77	0.71	0.69	0.77	0.74
Class 4	0.61	0.68	0.71	0.67	1	0.71	0.75	0.73	0.75	0.82
Class 5	0.71	0.75	0.71	0.77	0.71	1	0.77	0.7	0.83	0.74
Class 6	0.68	0.72	0.75	0.71	0.75	0.77	1	0.66	0.74	0.72
Class 7	0.67	0.72	0.66	0.69	0.73	0.7	0.66	1	0.7	0.82
Class 8	0.69	0.72	0.73	0.77	0.75	0.83	0.74	0.7	1	0.75
Class 9	0.66	0.73	0.71	0.74	0.82	0.74	0.72	0.82	0.75	1

Table 3.3.3: Overlap Rate for Co-Supports from Different Classes For Row Sparse ADL using *mnist-back-image*

!

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	1	0.72	0.8	0.8	0.73	0.77	0.77	0.71	0.79	0.7
Class 1	0.72	1	0.72	0.72	0.75	0.69	0.69	0.73	0.71	0.69
Class 2	0.8	0.72	1	0.84	0.75	0.74	0.74	0.71	0.79	0.71
Class 3	0.8	0.72	0.84	1	0.75	0.77	0.73	0.75	0.79	0.74
Class 4	0.73	0.75	0.75	0.75	1	0.74	0.73	0.76	0.78	0.79
Class 5	0.77	0.69	0.74	0.77	0.74	1	0.72	0.71	0.81	0.7
Class 6	0.77	0.69	0.74	0.73	0.73	0.72	1	0.66	0.75	0.68
Class 7	0.71	0.73	0.71	0.75	0.76	0.71	0.66	1	0.74	0.8
Class 8	0.79	0.71	0.79	0.79	0.78	0.81	0.75	0.74	1	0.75
Class 9	0.7	0.69	0.71	0.74	0.79	0.7	0.68	0.8	0.75	1

Table 3.3.4: Overlap Rate for Co-Supports from Different Classes For Wavelets using *mnist-back-image*

As is clear, in the clean case, Row Sparse ADL produces features whose co-support is different for different classes and there is very little overlap. The highest is 35% overlap for the classes 4 and 9 as we would expect as the digits 4 and 9 are similar to each other. On the other hand there is significant overlap among the co-support produced by the wavelet dictionary. In the noisy case, we see significant overlap in the co-supports of both the dictionaries. However, a closer look tells shows that the overlap between class 4 and class 9 in the Row Sparse ADL is 82% and for the wavelet dictionary it is 79%. While for digits 2 and 3 co-supports produced by Row Sparse ADL have an overlap of 70% and the same for the wavelet system is 84%. Taking another example between the digits 1 and 8, Row Sparse ADL has a co-support overlap of 69% while the wavelet dictionary has an overlap of 79%. On the whole we can deduce that under noisy conditions Row Sparse ADL is able to maintain better discrimination of subspaces. This gives us the motivation to use adaptive dictionaries. Next we look at the recognition rate for the test cases. The recognition rate are stated based on preprocessed data. The wavelet features and the Row Sparse features were both fed to a SVM classifier with RBF kernel. The wavelet system yielded a recognition rate of 0.77, while the SVM trained on Row Sparse features could classify with a recognition rate of 0.86.

Next we look into the features generated by the Low Rank ADL. Consider figure (3.3.5).

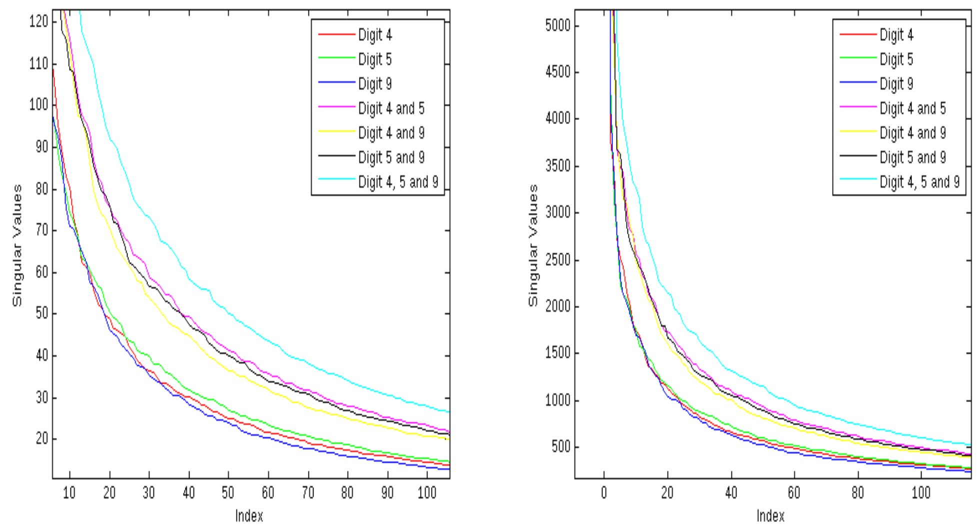


Figure 3.3.5: Comparison of the class rank in MNIST **(on Left)** and class rank of features generated by Low Rank ADL **(on Right)**

The class rank of the features generated by Low Rank ADL have a faster decay rate of the singular values, thus denoting the fact that the learnt features when grouped as a class have only the essential information contained in them. The slope for digit 4 in the MNIST dataset is -1.11 while the slope for the digit 4 feature is -25.6. Also, the learnt features maintain the discriminative ability by ensuring that each class has a lower rank than combined classes. Figure (3.3.5) also shows that there is high amount of dependencies within each class. Thus the learnt dictionary is able to produce discriminative low rank feature for each class.

Another aspect is initializing the dictionary for the various algorithms. A number of variations were tried. The one which gave consistently better performance on classification task was used for the experiments. First the dictionary was initialized with curvelets and DCT. It was found that features generated by such initializations could not give good classification performance. Thus, in most of the experiments, the dictionary was either initialized by samples from the training dataset, or as a random matrix with Gaussian entries.

The following tables show the recognition rate for the various proposed algorithms and the comparison with other algorithms for MNIST and its variations.

	MNIST(60k)	*mnist-rot*	*mnist-rot-back-image*	*mnist-back-rand*
Unsupervised ADL	0.953	0.81	0.39	0.885
Row Sparse ADL	0.958	0.81	0.392	0.884
Low Rank ADL	0.964	0.79	0.391	0.839
LC-ADL	0.972	0.875	0.371	0.884
LC-KSVD	0.933	0.754	0.487	0.877
DKSVD	0.936	0.754	0.492	0.863
Chellapa-ADL	0.973	0.829	**0.579**	0.875
DBN-3	**0.988**	0.897	0.526	**0.933**
SDAE	0.987	**0.905**	0.562	0.897

Table 3.3.5: Recognition Rate For MNIST and its Variations With Neural Net as Classifier for Proposed Methods

	MNIST(60k)	*mnist-rot*	*mnist-rot-back-image*	*mnist-back-rand*	*mnist-back-image*
Unsupervised ADL	0.978	0.887	0.556	0.893	0.846
Row Sparse ADL	0.982	0.888	0.552	0.894	**0.86**
Low Rank ADL	0.971	0.852	0.555	0.901	0.855
LC-ADL	0.972	0.875	0.51	0.901	0.82
LC-KSVD	0.933	0.754	0.487	0.877	0.806
DKSVD	0.936	0.754	0.492	0.863	0.819
Chellapa-ADL	0.973	0.829	**0.579**	0.875	0.851
DBN-3	**0.988**	0.897	0.526	**0.933**	0.837
SDAE	0.987	**0.905**	0.562	0.897	0.833

Table 3.3.6: Recognition Rate For MNIST and its Variations With SVM as Classifier with RBF Kernel for Proposed Methods

As is clear from the tables the SVM classifier performs better in classifying the data with the proposed algorithms. Thus, for further experiments only SVM is used as a classifier with a RBF kernel. Also, the proposed algorithms all surpass the synthesis algorithms, LC-KSVD and DKSVD. They also perform better than the analysis classification suggested by Chellapa et. al. It is interesting to note that in almost all cases the shallow ADL provides comparable recognition rates with respect to complex feature generation tools such as DBN and SDAE, which are deep learning tools and have much higher complexity compared to the simple analysis dictionary learning. Also, in some cases, classification based on features generated by the proposed algorithms beat those generated by DBN. For *mnist-back-image* the proposed algorithms beat deep learning techniques and synthesis techniques. An improvement of 3.24% over SDAE and an improvement of 2.7% over DBN is observed. A maximum improvement of 7% over synthesis learning algorithms was found. For *mnist-rot-back-image*, the proposed features beat the DBN in final recognition rates by a good margin and provide an maximum improvement of almost 6% and have recognition rate almost equal to that of SDAE. In case of *mnist-back-rand*, the classification based on features of Low Rank ADL and LC-ADL yield better results than SDAE. On the whole the proposed feature generation algorithms provided good recognition rates when compared with the state of the art feature generation algorithms at a much lower complexity.

To completely convince ourselves that the proposed algorithms do provide better recognition rates than synthesis learning algorithms and comparable with deep learning algorithms, experiments were conducted on the USPS dataset. The USPS dataset contains 7291 training samples of dimension 16 x 16 and has 2007 samples for testing. No preprocessing was done to the dataset.

Algorithm	Recognition Rate
Unsupervised ADL	0.953
Row Sparse ADL	**0.956**
Low Rank ADL	**0.956**
LC-ADL	0.946
LC-KSVD	0.939
DKSVD	0.95
Chellapa-ADL	0.945
DBN-3	0.941
SDAE	0.952

Table 3.3.7: Recognition Rate For USPS dataset, with SVM as Classifier with RBF Kernel for Proposed Methods

Chellapa et. al. proposed to feed the coefficients learnt by the transform learning into a classifier. Transform learning performs l_0 thresholding upon generating the feature by application of the transform. Figure (3.3.6) shows the features learnt by the Chellapa ADL and the proposed unsupervised ADL. Even though unsupervised ADL uses l_1 thresholding it is able to recognise significant zeros of the signal.

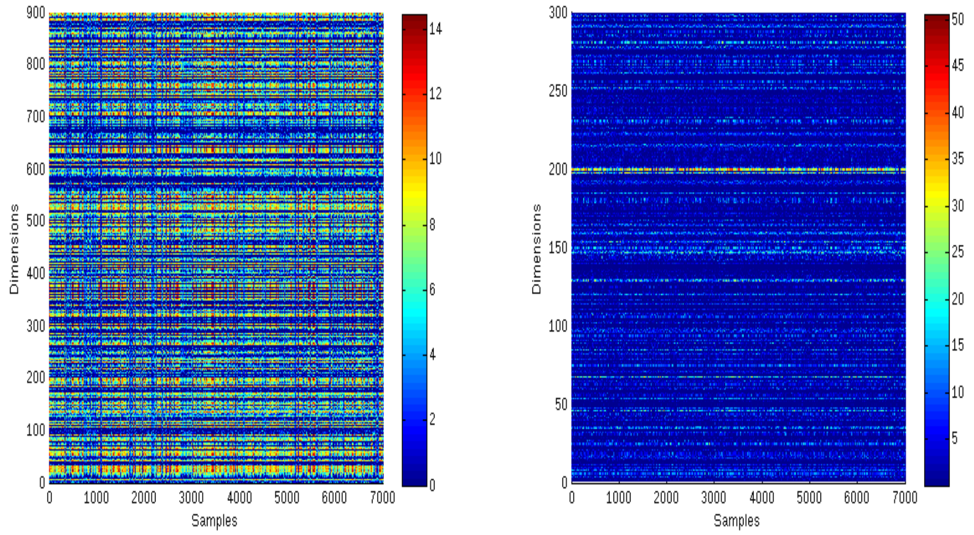


Figure 3.3.6: Feature Comparison For Chellapa ADL **(on Left)** and Unsupervised ADL **(on Right)** For USPS Dataset

Hence, even by relaxing the l_0 constraint to l_1, Unsupervised ADL are able to identify the zeros of the signal. So, it is not required to perform a non convex thresholding step.

Another set of experiments were performed to gauge the feature generation ability of the proposed algorithms in case of missing data. The following graph shows the performance of various algorithms with increasing number of missing pixels.

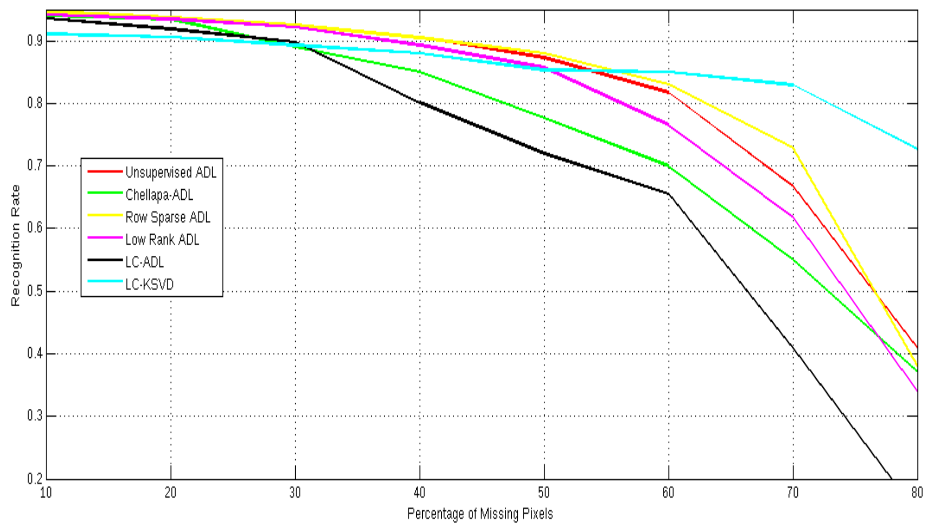


Figure 3.3.7: Variation in Recognition Rate with Missing Pixels for Various Algorithms

It is observed from the above graph that Row Sparse ADL performs best among the proposed algorithms followed by unsupervised ADL, Low Rank ADL, Chellapa ADL and LC-ADL. LC-KSVD performs the best, which is expected due to the KSVD type of algorithm used for solving the problem and performance of KSVD on reconstructing from missing data is well known.

3.4 Other Datasets

In order to see how the algorithm extends to other classification tasks, experiments were performed on the ISOLET dataset. This dataset consists alphabets spoken twice by 150 persons. Therefore there are 52 samples from each speaker. The aim is to recognise the alphabet which was spoken. There are 6238 training samples and 1559 testing samples. No pre-processing was used.

Algorithm	Recognition Rate
Unsupervised ADL	0.964
Row Sparse ADL	0.963
Low Rank ADL	**0.966**
LC-ADL	0.959
LC-KSVD	0.85
DKSVD	0.856
Chellapa-ADL	0.928
DBN-3	0.962
SDAE	0.95

Table 3.4.1: Recognition Rate For ISOLET dataset, with SVM as Classifier with RBF Kernel for Proposed Methods

In this case the proposed algorithms features when used with a SVM classifier out perform the synthesis as well as deep learning methods.

3.5 Feature Generation Time

The following table shows the time required by various algorithms to generate features at test time. Here we record the time taken to generate features for the entire test data. To have a fair comparison the number of dictionary atoms was fixed to 700 for all the algorithms. Clean MNIST dataset was used for comparison.

Algorithm	Time (s)
Proposed Algorithms	**0.13**
Chellapa-ADL	0.46
LC-KSVD	0.325
DKSVD	0.338

Table 3.5.1: Time Required for Feature Generation at Test Time For MNIST

As is noted the proposed algorithms are fast in generating test features in comparison to other algorithms. Chellapa-ADL requires more time because a l_0 thresholding is required after application of the transform. In the synthesis case an inverse problem is solved for generating the coefficients, hence greater the time required to generate features at test time.

Chapter 4

Application: Consumer Appliance Device Classification

Non Intrusive Load Monitoring (NILM) systems need to identify devices from their electrical characteristics. This helps in understanding human activity (e.g. a stove switched on means cooking) and also helps in management of electrical resources. The goal for energy efficient buildings can be identified as

1. Use of smart and energy efficient technologies to reduce energy consumption and
2. To reduce wastage by identifying (and thereby preventing) the operations of appliances in non-working hours.

The analysis dictionary learning framework is applied to the latter goal and used for consumer appliance device classification. The following sections discuss the motivation, the application framework and results. The proposed framework is compared with the state-of-the-art.

4.1 Motivation

Conserving energy and reducing consumption of energy is of prime importance in energy efficient buildings. However, wastage can only be reduced when we know that a device is 'ON' during a lean period. This is of utmost importance in commercial buildings where quite often than not electrical appliances are left switched on even during non working hours, which leads to energy wastage. Therefore, accurately detecting and classifying devices on a power line is an important task. This falls under the broad category of Non Intrusive Load Monitoring (NILM). NILM systems provide complete information on the current state and power consumed by a device. Power consumption readings in a scenario of event detection is not really helpful but the state of the device is. For example an A.C. may consume a lot of power but a fan or LED light may not, but in order to reduce wastage one just needs to know which device is on. This is a common scenario in residential places where one forgets to switch off a CFL or a computer CPU. So, the proposed method of classification in the thesis is applied to classify and identify the 'ON' devices. The methodology of detection is discussed next.

4.2 Methodology

Usually in NILM, the signal is the reading acquired by a smart meter at regular intervals of time. A few household appliances, such as electric toasters and irons, are simple on/off loads that show a sharp increase in the power consumption when switched on. These patterns can be identified with data gathered from smart meters [48]. However, most residential and commercial appliances are not simple on/off. loads. They are multi-modal (refrigerator, AC, washer etc.) [49] or continuously time varying (CPU, printers etc.) [50]. Detecting the operation of these appliances has challenged researchers in the past few years. Multi-modal loads can still be modelled by stochastic finite state machines (Factorial Hidden Markov Model or Product of Experts), but the continuously varying loads are the hardest to model by such classical techniques.

In the last few years, an alternate technique to smart meter sensing has emerged as a viable method for detecting time-varying power patterns in appliances. The method is based on detecting the unique electromagnetic emissions, generated by the switched mode power supply within the appliance, on the power line [51, 52]. The electromagnetic emissions are of two types: the differential signal between the phase and neutral power lines; and the common mode signal between these lines and the earth. The common-mode signal was demonstrated as a far more robust feature vector for classification in comparison to the differential signal in [52]. This is because the primary power signal (110V/230V) and its harmonics, which interfere heavily with the differential signal [50], are not present on the common-mode signal measurements. Also, most appliances, today, are required to be fitted with high quality differential mode filters that regulate their emissions. Therefore, Differential Mode Electromagnetic (DM EMI) emissions form an unreliable feature for detecting appliances and classification techniques yield poor results. Thus, the Common Mode Electromagnetic (CM EMI) data is collected from various devices and the learning algorithm is run on various aggregated instances of the devices. In this work four devices and a fifth class, background noise, were chosen to form a part of the data. The four devices were CFL, CPU, Laptop Charger (LC) and LCD. The reason behind choosing these four devices was that they are commonly found in residential and commercial places and are the ones which are left 'ON'.

4.3 Feature Vector

The data obtained from a CM EMI sensing device are voltage values with a time stamp. The sampling frequency of the data was 15.625 Mhz. The data consisted of five instances of measurement for each device and their corresponding background noise. Using the raw data directly classification does not yield much. The data needed to be converted to a usable form. Frequency domain techniques were used. The Fourier domain data was generated but they were not discriminative enough to give good classification. As in [53], the cepstrum features were tried and they were found to have good discriminative power. The cepstrum

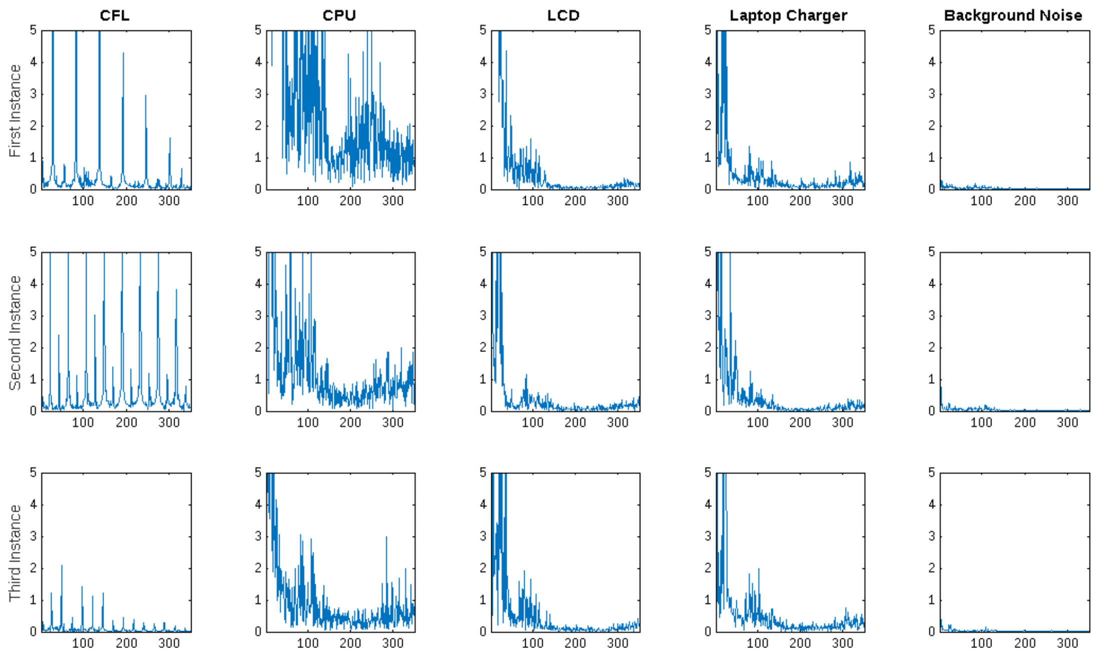


Figure 4.3.1: Fourier Features For Various Devices. x axis on each subplot define frequency in kHz and the y axis in each subplot defines magnitude of the coefficients.

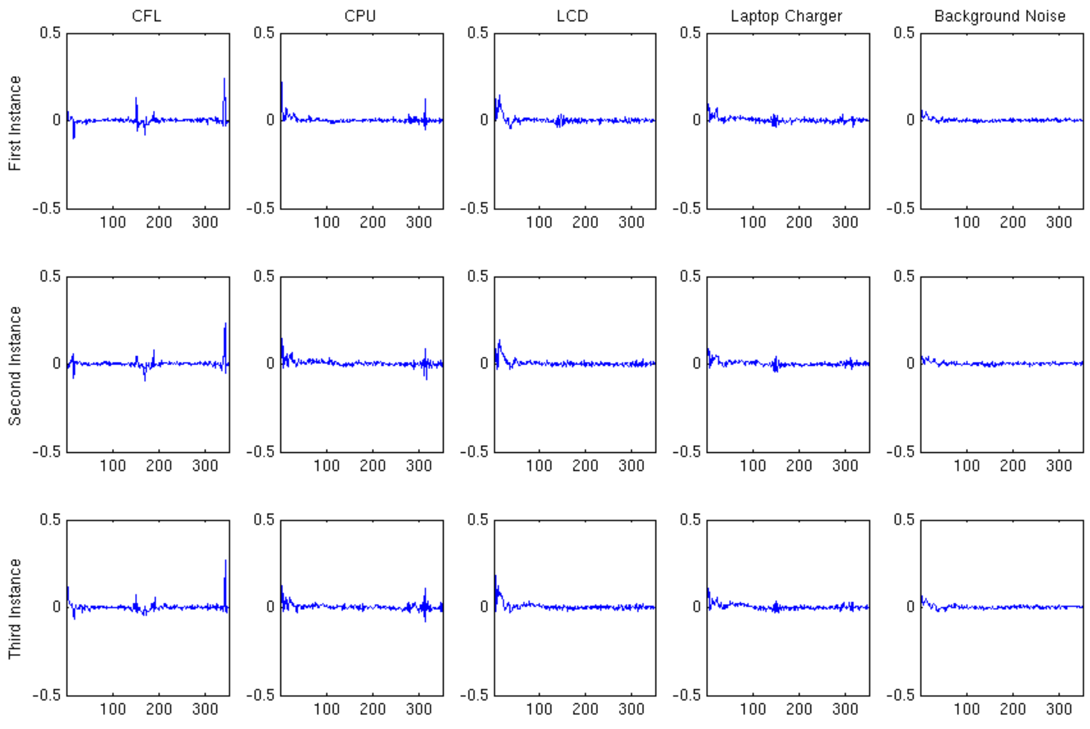


Figure 4.3.2: Cepstrum Features for Various Devices. x axis on each subplot define frequency in kHz and the y axis in each subplot defines magnitude of the coefficients.

of a signal is the inverse Fourier transform of the logarithm of its Fourier coefficients.

$$\text{cepstrum of a signal f(t)} = \mathcal{F}^{-1}(log(|\mathcal{F}(f(t))|))$$

Figure (4.3.2) shows the cepstrum features on which the learning was done. We can identify various peaks of the devices at various frequencies which discriminate the data. For example, both CFL and CPU have peaks near 20 and 330 kHz. However, CFL also has a peak around 170 kHz which is missing in the CPU signal and thus distinguishes the two. Laptop Charger has a similar signature as that of CFL, however misses the important peak at 330 kHz and has smaller magnitude peaks near 170 kHz which is enough information to distinguish the two devices. Similarly, the LCD and background noise have a relatively flat cepstrum but at low frequencies the LCD cepstrum has sharper and a larger number of peaks. Hence the cepstrum of the devices form a good discriminative set of features on which learning can be done.Such features were not possible with neither the Fourier domain nor the time domain data.Three techniques were used for learning. ElectriSense [51], Conditional Likelihood Maximization [54] and the proposed analysis dictionary learning framework with neural network and SVM as classifiers.

4.4 Learning Algorithms

The algorithms against which the proposed algorithm is compared are the ElectriSense and Conditional Likehood Maximization. The Electrisense was developed to specifically classify devices from EMI data. By itself ElectriSense is just a Gaussian Mixture Model(GMM) for event detection. It has an event segmentation and feature extraction procedure which works as follows. When the sensor is switched on, it reads the first twenty five frequency vectors and then stores it as the baseline noise signature. Then the system reads a window every twenty five frequency vectors and mean aggregates them. A difference vector is calculated from the baseline thus segmenting the event. The magnitude of the difference vector components is compared with a threshold and a event is detected. Thereafter, a Gaussian function is fitted to the peaks in the difference vectors to estimate the amplitude, mean and variance parameters. A feature vector is constructed with the parameters learnt and the central frequency component and a one neighbour nearest neighbour with euclidean distance metric with inverse weighting is used to classify the event. The difference between the proposed approach and ElectriSense is that the proposed method works directly on the disaggregated data and does not involve segmentation as a separate procedure. Moreover, ElectriSense works on DM EMI data. However, in the current work, experiments were performed with CM EMI data which has already been discussed to be more robust for classification.

Conditional Likelihood Maximization is an information theoretic approach to feature selection which picks up good selection vectors from the given training signals based on the likelihood of improving prediction accuracy. It uses mutual information between the selected feature and the target it approximates. The entire optimization procedure is complex and cannot be explained here. These two algorithms were compared with the proposed framework.For completeness, a brief discussion of the hardware used is presented next.

4.5 Hardware

The hardware was designed by the Circuits and Systems group at IIITD. A sensor similar to the one proposed in [52] is used for collecting both common-mode (CM) and differential mode (DM) EMI data injected by an Appliance Under Test (AUT) on the power line. The experimental set up is shown in figure (4.5.1) and (4.5.2). The sensor directly interfaces with the phase, neutral and earth power lines through an extension cord through which the AUT is powered. The DM EMI is measured from the differential across the phase and neutral lines. A high pass filter is introduced to remove the 230V, 50Hz power signal from the measurement. The CM EMI is measured directly from the earth currents. The measurements are stored in an internal buffer within the sensor and then uploaded to an external computer for further processing through wired interface. These figures have been adopted from [52] and a detail study about the sensor can be found therein.

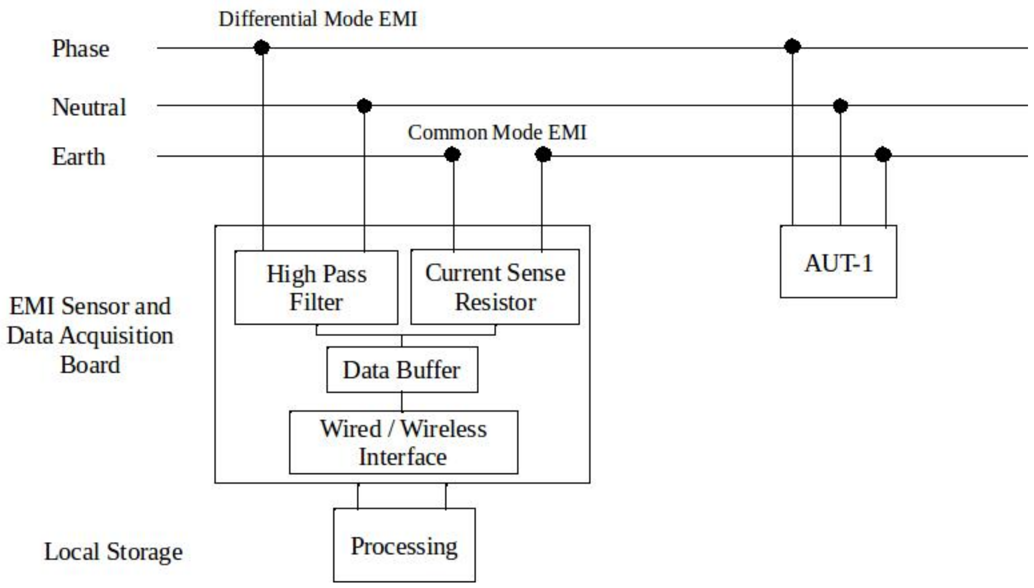


Figure 4.5.1: Sensor Flowchart for Data Acquisition

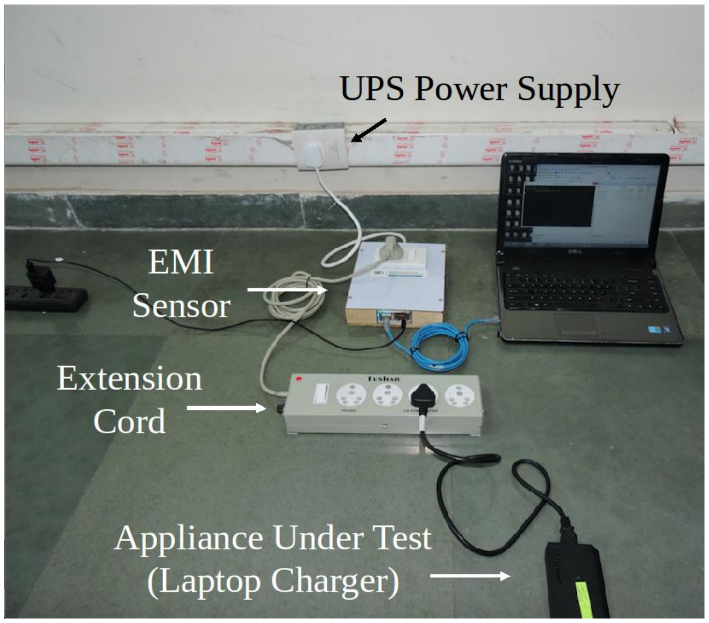


Figure 4.5.2: Test Setup

4.6 Experiments and Results

For the experiments the cepstrum features were generated for the collected data. There were five instance for each device and each instance had 1500 readings from the meter. The readings from all the devices for a given instance was aggregated together to form a matrix. Each column of matrix was the cepstrum feature of a particular device. The training set consisted of one instance of each device and the rest four instances were used as testing data. This is the most difficult and most practical situation to be observed in a residential or commercial building where access to training data is always scarce. Thus there were 1500 samples for each device to train on and 6000 samples for testing for each device. The background noise data was used from the LCD due to various stability reasons of the electrical signal. The following confusion matrices show the recognition rates for each device upon classification by various methods. The results reported are the average results across all the five instances taken as training. Neural Networks and SVM were used as classifier for the proposed methods. LC in the following tables denote Laptop Charger. First the features learnt from the data are shown followed by the results.

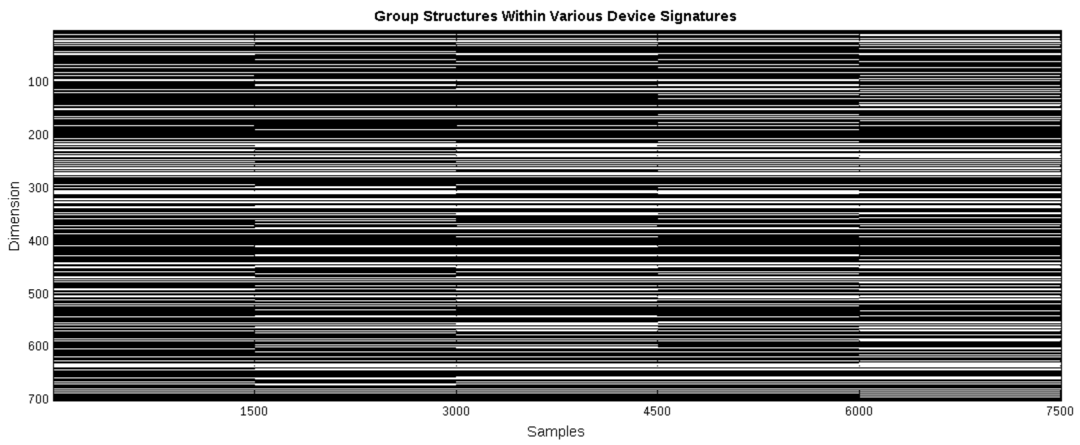


Figure 4.6.1: Row Sparse Features For the Devices

As we can see, the row sparse algorithm can recover the group structure of the various devices. The black rows indicate the atoms to which the signal is perpendicular. As expected there are common directions to which all the signals are orthogonal, but each device has a class specific signature as is evident from Figure (4.6.1). The low rank features for the various

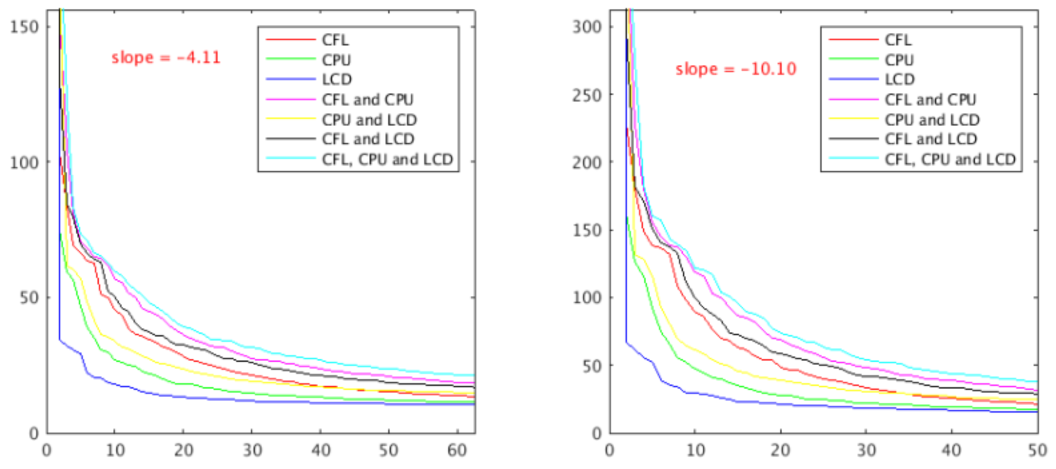


Figure 4.6.2: Low Rank Features For the Devices

devices are shown in figure (4.6.2). The low rank algorithm learns the dependencies among he classes and hence each class has a lower rank as compared to a mixture of two or more classes. This is more evident on case of CFL. CFL has a slower decay rate of the singular values than CPU and LCD, however any mixture of CFL and another class we see the decay rate of singular values is far slower. Also we can see that for the CFL data we find almost 60% faster decay rate in the low rank data as compared to the original cepstrum data. Thus we find that the low rank ADL algorithm generates the most low rank feature for each class.

	CFL	CPU	LCD	LC	Background Noise
CFL	0	0	0	0	0
CPU	0.75	0.5	0.25	0.25	0.25
LCD	0	0	0	0	0
LC	0.25	0.5	0.75	0.75	0.5
Background Noise	0	0	0	0	0.25

Table 4.6.1: Confusion Matrix: GMM + kNN (ElectriSense)

	CFL	CPU	LCD	LC	Background Noise
CFL	0.50	0.38	0.20	0.31	0.2
CPU	0.09	0.51	0	0.005	0
LCD	0.18	0.001	0.79	0.003	0
LC	0.13	0.1	0.002	0.68	0
Background Noise	0.09	0	0	0	0.8

Table 4.6.2: Confusion Matrix: CLM + Neural Network

	CFL	CPU	LCD	LC	Background Noise
CFL	0.29	0.25	0.01	0.16	0.001
CPU	0.09	0.49	0.065	0.07	0.065
LCD	0.28	0.052	0.84	0.08	0.05
LC	0.23	0.2	0.082	0.69	0.08
Background Noise	0.1	0	0	0	0.8

Table 4.6.3: Confusion Matrix: CLM + SVM

	CFL	CPU	LCD	LC	Background Noise
CFL	0.66	0.008	0.004	0.031	0.06
CPU	0.017	0.88	0.052	0.001	0
LCD	0.033	.094	0.8	0.26	0
LC	0.04	0.02	0.14	0.68	0
Background Noise	0.26	0	0	0.02	0.94

Table 4.6.4: Confusion Matrix: Unsupervised ADL + Neural Network

	CFL	CPU	LCD	LC	Background Noise
CFL	0.7	0.01	0.001	0.01	0.05
CPU	0.017	0.88	0.035	0	0
LCD	0.028	0.09	0.8	0.25	0
LC	0.034	0.02	0.16	0.73	0
Background Noise	0.23	0	0	0.02	0.95

Table 4.6.5: Confusion Matrix: Row Sparse ADL + Neural Network

	CFL	CPU	LCD	LC	Background Noise
CFL	0.72	0.009	0.006	0.021	0.051
CPU	0.004	0.87	0.023	0.001	0
LCD	0.041	0.12	0.8	0.25	0
LC	0.021	0.02	0.17	0.70	0
Background Noise	0.22	0	0	0.03	0.95

Table 4.6.6: Confusion Matrix: Low Rank ADL + Neural Network

	CFL	CPU	LCD	LC	Background Noise
CFL	0.53	0.051	0	0.009	0.002
CPU	0.05	0.71	0.07	0.009	0
LCD	0.042	0.19	0.8	0.3	0
LC	0.013	0.05	0.13	0.603	0
Background Noise	0.37	0	0.005	0.084	0.998

Table 4.6.7: Confusion Matrix: Label Consistent ADL + Neural Network

	CFL	CPU	LCD	LC	Background Noise
CFL	0.78	0.03	0	0.0025	0.07
CPU	0.071	0.89	0.01	0	0
LCD	0.056	0.073	0.8	0.29	0
LC	0.01	0.007	0.2	0.704	0
Background Noise	0.085	0	0.005	0.084	0.932

Table 4.6.8: Confusion Matrix: Unsupervised ADL + SVM

	CFL	CPU	LCD	LC	Background Noise
CFL	0.77	0.084	0.002	0.005	0.007
CPU	0.14	0.855	0.021	0	0
LCD	0.012	0.0054	0.8	0.32	0
LC	0	0.007	0.18	0.69	0
Background Noise	0.08	0	0	0	0.993

Table 4.6.9: Confusion Matrix: Row Sparse ADL + SVM

	CFL	CPU	LCD	LC	Background Noise
CFL	0.77	0.037	0.001	0.006	0.07
CPU	0.09	0.89	0.011	0	0
LCD	0.05	0.067	0.803	0.28	0
LC	0.005	0.01	0.18	0.72	0
Background Noise	0.08	0	0	0	0.933

Table 4.6.10: Confusion Matrix: Low Rank ADL + SVM

	CFL	CPU	LCD	LC	Background Noise
CFL	0.731	0.04	0.001	0.0053	0.072
CPU	0.105	0.89	0.0138	0	0
LCD	0.07	0.06	0.8	0.32	0
LC	0.005	0.007	0.18	0.672	0
Background Noise	0.09	0	0	0	0.93

Table 4.6.11: Confusion Matrix: Label Consistent ADL + SVM

Method	Accuracy
ElectriSense	30
CLM + Neural Network	65.87
CLM + SVM	62.46
Unsupervised ADL + Neural Network	79.29
Row Sparse ADL + Neural Network	81.12
Low Rank ADL + Neural Network	80.85
Label Consistent ADL + Neural Network	80.20
Unsupervised ADL + SVM	82.05
Row Sparse ADL + SVM	81.9
Low Rank ADL + SVM	**82.3**
Label Consistent ADL + SVM	80.43

Table 4.6.12: Classification Accuracies With Various Methods (In Percentage)

4.7 Conclusion

As can be seen from the confusion matrices for CM EMI data, the proposed algorithms always perform better on an per device basis as well as on the entire dataset as a whole. Moreover, the SVM classifier fairs better as compared to the neural network. The CLM method is able to distinguish between LCD, Laptop Charger and noise. However, it fails miserably in distinguishing CPUs and CFLs. As is clearfrom the results, ElectriSense procedure is tailor made for DM EMI data and fails in the CM EMI data framework. The proposed methods are more robust across devices and detect all devices with reasonable accuracy.

Bibliography

[1] K. Engan, S. O. Aase, and J. H. 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 vol.5.

[2] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[3] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb 2006.

[4] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, Apr. 1995. [Online]. Available: http://dx.doi.org/10.1137/S0097539792240406

[5] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Commun. ACM*, vol. 53, no. 12, pp. 93–100, Dec. 2010. [Online]. Available: http://doi.acm.org/10.1145/1859204.1859229

[6] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, Dec 2007.

[7] T. T. Cai and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with

noise," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4680–4688, July 2011.

[8] 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.

[9] A. Majumdar and R. K. Ward, "On the choice of compressed sensing priors and sparsifying transforms for mr image reconstruction: An experimental study," *Image Commun.*, vol. 27, no. 9, pp. 1035–1048, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.image.2012.08.002

[10] E. Candes, M. Wkin, and S. Boyd, "Enhancing sparsity by reweighted l1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, Dec. 2008.

[11] Selesnick, "Sparse signal restoration," 2009. [Online]. Available: http://cnx.org/content/m32168/

[12] A. Y. Yang, Z. Zhou, A. G. Balasubramanian, S. S. Sastry, and Y. Ma, "Fast l1-minimization algorithms for robust face recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 3234–3246, Aug 2013.

[13] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Img. Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1137/080716542

[14] S. Becker, J. Bobin, and E. J. Candès, "Nesta: A fast and accurate first-order method for sparse recovery," *SIAM J. Img. Sci.*, vol. 4, no. 1, pp. 1–39, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1137/090756855

[15] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[16] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, June 2010.

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

[18] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[19] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Trans. Img. Proc.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIP.2006.881969

[20] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, Jan 2008.

[21] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Modeling & Simulation*, vol. 7, no. 1, pp. 214–241, 2008. [Online]. Available: http://dx.doi.org/10.1137/070697653

[22] K. Shu and W. Donghui, "A brief summary of dictionary learning based approach for classification." [Online]. Available: arXiv:1205.6391

[23] 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.

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

[25] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent k-svd: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, Nov 2013.

[26] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 543–550. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126286

[27] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Transactions on Signal Processing*, vol. 61, no. 9, pp. 2341–2355, May 2013.

[28] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2138–2150, June 2013.

[29] S. Nam, M. E. Davies, M. Elad, and R. Gribonval, "Cosparse analysis modeling - uniqueness and algorithms," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 5804–5807.

[30] B. Ophir, M. Elad, N. Bertin, and M. D. Plumbley, "Sequential minimal eigenvalues - an approach to analysis dictionary learning," in *Signal Processing Conference, 2011 19th European*, Aug 2011, pp. 1465–1469.

[31] M. Seibert, J. Wrmann, R. Gribonval, and M. Kleinsteuber, "Learning co-sparse analysis operators with separable structures," *IEEE Transactions on Signal Processing*, vol. 64, no. 1, pp. 120–130, Jan 2016.

[32] E. M. Eksioglu and O. Bayir, "K-svd meets transform learning: Transform k-svd," *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 347–351, March 2014.

[33] R. Rubinstein, T. Peleg, and M. Elad, “Analysis k-svd: A dictionary-learning algorithm for the analysis sparse model,” *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, Feb 2013.

[34] S. Ravishankar and Y. Bresler, “Learning sparsifying transforms for image processing,” in *2012 19th IEEE International Conference on Image Processing*, Sept 2012, pp. 681–684.

[35] ——, “Learning overcomplete sparsifying transforms for signal processing,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 3088–3092.

[36] B. Wen, S. Ravishankar, and Y. Bresler, “Learning overcomplete sparsifying transforms with block cosparsity,” in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 803–807.

[37] S. Ravishankar, B. Wen, and Y. Bresler, “Online sparsifying transform learning part i: Algorithms,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 625–636, June 2015.

[38] S. Ravishankar and Y. Bresler, “l0 sparsifying transform learning with efficient optimal updates and convergence guarantees,” *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2389–2404, May 2015.

[39] M. Elad, P. Milanfar, and R. Rubinstein, “Analysis versus synthesis in signal priors,” in *Signal Processing Conference, 2006 14th European*, Sept 2006, pp. 1–5.

[40] S. Shekhar, V. M. Patel, and R. Chellappa, “Analysis sparse coding models for image-based classification,” in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 5207–5211.

[41] R. Chartrand and B. Wohlberg, "A nonconvex admm algorithm for group sparsity with sparse groups," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6009–6013.

[42] A. Majumdar and R. K. Ward, "Synthesis and analysis prior algorithms for joint-sparse recovery," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012, pp. 3421–3424.

[43] A. Majumdar and R. Ward, "Some empirical advances in matrix completion," *Signal Process.*, vol. 91, no. 5, pp. 1334–1338, May 2011. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2010.12.005

[44] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[45] [Online]. Available: http://www.iro.umontreal.ca/ lisa/twiki/bin/view.cgi/Public/MnistVariations

[46] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," Master's thesis, 2012.

[47] E. v. Berg, M. P. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and Ö. Yılmaz, "Sparco: A testing framework for sparse reconstruction," Dept. Computer Science, University of British Columbia, Vancouver, Tech. Rep. TR-2007-20, October 2007.

[48] M. Zeifman, "Disaggregation of home energy display data using probabilistic approach," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 23–31, February 2012.

[49] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, Dec 1992.

[50] S. Barker, S. Kalra, D. Irwin, and P. Shenoy, "Empirical characterization and modeling

of electrical loads in smart homes," in *Green Computing Conference (IGCC), 2013 International*, June 2013, pp. 1–10.

[51] S. Gupta, M. S. Reynolds, and S. N. Patel, "Electrisense: Single-point sensing using emi for electrical event detection and classification in the home," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, ser. UbiComp '10. New York, NY, USA: ACM, 2010, pp. 139–148. [Online]. Available: http://doi.acm.org/10.1145/1864349.1864375

[52] M. Gulati, S. S. Ram, and A. Singh, "An in depth study into using emi signatures for appliance identification," in *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, ser. BuildSys '14. New York, NY, USA: ACM, 2014, pp. 70–79. [Online]. Available: http://doi.acm.org/10.1145/2674061.2674070

[53] S. Kong, Y. Kim, R. Ko, and S. K. Joo, "Home appliance load disaggregation using cepstrum-smoothing-based method," *IEEE Transactions on Consumer Electronics*, vol. 61, no. 1, pp. 24–30, February 2015.

[54] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *J. Mach. Learn. Res.*, vol. 13, pp. 27–66, Jan. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2188385.2188387

Publications

Accepted:

1. P. Bhattacharjee, P. Khurana and A. Majumdar, Low-rank matrix recovery from non-linear observations, 2015 IEEE International Conference on Digital Signal Processing (DSP), P 623-627 2015 IEEE.

2. P. Khurana, P. Bhattacharjee and A. Majumdar, Matrix factorization from non-linear projections: application in estimating T2 maps from few echoes, Magnetic Resonance Imaging, 33(7), 927-931, Elsevier.

Submitted:

1. P. Bhattacharjee, S. Banerjee, A. Majumdar, M. Gulati and S.S. Ram, Supervised Analysis Dictionary Learning for Consumer Electronics Appliance Classification. (IEEE Transactions of Consumer Electronics)