INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

B.TECH. PROJECT STAGE I

Dictionary Learning Algorithms and Application

Authors:
Deepali Adlakha

Supervisor:
Prof. Suyash Awate

A thesis submitted in fulfillment of the requirements for B.Tech. project Stage 1 and 2

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

May 2015

Acknowledgements

I would like to take this opportunity to express our deepest gratitude towards our guide Prof. Suyash Awate for his motivating support, guidance and valuable suggestions.

Abstract

Sparse and redundant representation modeling of data assumes an ability to describe signals as linear combinations of a few atoms from a pre-specified dictionary. As such, the choice of the dictionary that sparsifies the signals is crucial for the success of this model. In general, the choice of a proper dictionary can be done using one of two ways: i) building a sparsifying dictionary based on a mathematical model of the data, or ii) learning a dictionary to perform best on a training set. In this project, we explore the second route where we attach the dictionary to a set of examples it is supposed to serve. Moreover, we experiment the possibility of combining these sparse representations with the parts based representations in coherence with the learning by human receptive fields.

Contents

Chapter 1

Introduction

The process of digitally sampling a natural signal leads to its representation as the sum of Delta functions in space or time. This representation, while convenient for the purposes of display or playback, is mostly inefficient for analysis tasks. Signal processing techniques commonly require more meaningful representations which capture the useful characteristics of the signal-for recognition, the representation should highlight salient features; for denoising, the representation should efficiently separate signal and noise; and for compression, the representation should capture a large part of the signal with only a few coefficients. Interestingly, in many cases these seemingly different goals align, sharing a core desire for simplification.

1.1 Motivation for Sparse Representations

The responses of neurons in the primary visual cortex (V1), a region of the brain involved in encoding visual input, are modified by the visual experience of the animal during development. For example, most neurons in animals reared viewing stripes of a particular orientation only respond to the orientation that the animal experienced. The responses of V1 cells in normal animals are similar to responses that simple optimisation algorithms can learn when trained on images. However, whether the similarity between these algorithms and V1 responses is merely coincidental has been unclear. A number of experiments have been carried out where animals were reared with modified visual experience to test the explanatory power of optimisation algorithms. The images had been filtered for the algorithms in ways that mimicked the visual experience of the animals. It was proved that the changes in V1 responses in experiment were consistent with the algorithms. This is evidence that the precepts of the algorithms, notably sparsity, can be used to understand the development of V1 responses.[1]

1.2 Machine Learning

Linear data representations are widely used in signal processing and data analysis. A traditional method of choice for signal representation is of course Fourier analysis, but also wavelet representations are increasingly being used in a variety of applications. Both of these methods have strong mathematical foundations and fast implementations, but they share the important drawback that they are not adapted to the particular data being analyzed. Data-adaptive representations, on the other hand, are representations that are tailored to the statistics of the data. Such representations are learned directly from the observed data by optimizing some measure that quantifies the desired properties of the representation. This class of methods include principal component analysis (PCA), independent component analysis (ICA), sparse coding, and non-negative matrix factorization (NMF). Some of these methods have their roots in neural computation, but have since been shown to be widely applicable for signal analysis.

1.2.1 Analytic Versus Trained Dictionaries

The dictionaries described so far all roughly fall under the umbrella of Harmonic Analysis, which suggests modeling interesting signal data by a more simple class of mathematical functions , and designing an efficient representation around this model. For example, the Fourier dictionary is designed around smooth functions, while the wavelet dictionary is designed around piecewise-smooth functions with point singularities. The dictionaries of this sort are characterized by an analytic formulation, and are usually supported by a set of optimality proofs and error rate bounds. An important advantage of this approach is that the resulting dictionary usually features a fast implicit implementation which does not involve multiplication by the dictionary matrix. On the other hand, the dictionary can only be as successful as its underlying model , and indeed, these models tend to be over-simplistic compared to the complexity of natural phenomena.
Through the 1980's and 1990's, Machine Learning techniques were rapidly gaining interest, and promised to confront this exact difficulty. The basic assumption behind the learning approach is that the structure of complex natural phenomena can be more accurately extracted directly from the data than by using a mathematical description. One direct benefit of this is that a finer adaptation to specific instances of the data becomes possible, replacing the use of generic models.

1.2.2 Dictionary Learning

Using an over complete dictionary matrix that contains prototype signal-atoms for columns, , a signal can be represented as a sparse linear combination of these atoms. The representation of may either be exact or approximate, , satisfying . The vector contains the representation coefficients of the signal . In approximation methods, typical norms used for measuring the deviation are the l_p norms for $p = 1$, $p = 2$ or $p = \infty$. In this paper, we shall concentrate on the case of $p = 2$.

If $n < K$ and D is a full-rank matrix, an infinite number of solutions are available for the representation problem, hence constraints on the solution must be set. The solution with the fewest number of nonzero coefficients is certainly an appealing representation. This sparsest representation is the solution of either

$$\min_x \|x\|_0 \quad \textit{subject to} \quad y = Dx \tag{1.1}$$

$$\min_x \|x\|_0 \quad \textit{subject to} \quad \|y - Dx\|_2 < \epsilon \tag{1.2}$$

where $\|.\|$ is the l_0 norm, counting the nonzero entries of a vector.

1.3 Applications

Applications that can benefit from the sparsity and over-completeness concepts (together or separately) include compression, regularization in inverse problems, feature extraction, and more. Indeed, the success of the JPEG2000 coding standard can be attributed to the sparsity of the wavelet coefficients of natural images [2]. In denoising, wavelet methods and shift-invariant variations that exploit over-complete representation are among the most effective known algorithms for this task [3][4][5][6]. Sparsity and over-completeness have been successfully used for dynamic range compression in images [7], separation of texture and cartoon content in images [8], in-painting [9], and more.

1.4 Summary

Dictionary design has significantly evolved over the past decades, beginning with simple orthogonal transforms and leading to the complex over-complete analytic and trained dictionaries now defining the state-of-the-art. Substantial conceptual advancement has been made in understanding the elements of an efficient dictionary design - most notably adaptivity, multi-scale, geometric in-variance, and over completeness. However, with a wealth of tools already developed, much work remains to be done.

Chapter 2

Non Negative Sparse Coding

There is psychological and physiological evidence for parts-based representations in the brain, and certain computational theories of object recognition rely on such representations. But little is known about how brains or computers might learn the parts of objects. Non Negative Matrix Factorisation is in contrast to other methods, such as principal components analysis and vector quantization, that learn holistic, not parts-based, representations. Non-negative matrix factorization is distinguished from the other methods by its use of non-negativity constraints. These constraints lead to a parts-based representation because they allow only additive, not subtractive, combinations. When non-negative matrix factorization is implemented as a neural network, parts-based representations emerge by virtue of two properties: the firing rates of neurons are never negative and synaptic strengths do not change sign.

2.1 Non Negative Matrix Factorization

Definition 2.1. Given a non-negative matrix V, find non-negative matrix factors W and H such that [10]:

$$V = W \times H$$

How does NMF learn such a representation, so different from the holistic representations of Principal Components Analysis and Vector Quantization? To answer this question, it is helpful to describe the three methods in a matrix factorization framework. The image database is regarded as an n $\times$ m matrix V, each column of which contains n non-negative pixel values of one of the m patches. Then all three methods construct approximate factorizations of the form V $\approx$ WH, or

$$V_{ij} \approx (WH)_{ij} = \sum_{a=1}^{r} W_{ia}H_{aj}$$

The r columns of W are called basis images. Each column of H is called an encoding and is in one-to-one correspondence with a patch in V. An encoding consists of the coefficients by which a patch is represented with a linear combination of basis images. The dimensions of the matrix factors W and H are n $\times$ r and r $\times$ m, respectively. The rank r of the factorization is generally chosen so that $(n+m)r < nm$, and the product WH can be regarded as a compressed form of the data in V.

2.2 Parts-based and Sparse representations

A key contribution to the area of dictionary learning was provided by Olshausen and Field in 1996 [11]. In their widely celebrated paper, the authors trained a dictionary for sparse representation of small image patches collected from a number of natural images. With relatively simple algorithmic machinery, the authors were able to show a remarkable result, the trained atoms they obtained were incredibly similar to the mammalian simple-cell receptive fields. Thus, the receptive fields acquired through unsupervised learning of sparse representations of natural scenes have similar properties to primary visual cortex (V1) simple cell receptive fields. The finding was highly motivating to the sparse representation community, as it demonstrated that the single assumption of sparsity could account for a fundamental biological visual behavior.
In this chapter, we combine Sparse Coding with Non Negative Matrix Factorization and test it on natural images.

2.2.1 Sparse Coding

Assume that we observe data in the form of a large number of i.i.d. random vectors x^n, where n is the sample index. Arranging these into the columns of a matrix $\mathbf{X}$, then linear decompositions describe this data as $\mathbf{X} \approx \mathbf{AS}$. The matrix $\mathbf{A}$ is called the mixing matrix, and contains as its columns the basis vectors (features) of the decomposition. The rows of $\mathbf{S}$ contain the corresponding hidden components that give the contribution of each basis vector in the input vectors. We do an approximate reconstruction.

Our goal is to find a decomposition in which the hidden components are sparse, meaning that they have probability densities which are highly peaked at zero and have heavy tails. This basically means that any given input vector can be well represented using only a few significantly non-zero hidden coefficients. Combining the goal of small reconstruction error with that of sparseness, one can arrive at the following objective function [12] to be minimized:

$$C(\mathbf{A}, \mathbf{S}) = \frac{1}{2} \| \mathbf{X} - \mathbf{AS} \|^2 + \lambda \sum_{ij} f(S_{ij}) \tag{2.1}$$

where the squared matrix norm is simply the summed squared value of the elements, i.e. $\| \mathbf{X} - \mathbf{AS} \|^2 = \sum_{ij} [\mathbf{X}_{ij} - (\mathbf{AS})_{ij}]^2$. The trade-off between sparseness and accurate reconstruction is controlled by the parameter, whereas the form of f defines how sparseness is measured. To achieve a sparse code, the form of f must be chosen correctly: A typical choice is f(s) = $|s|$, although often similar functions that exhibit smoother behavior at zero are chosen for numerical stability.

There is one important problem with this objective: As f typically is a strictly increasing function of the absolute value of its argument, the objective can always be decreased by simply scaling up A and correspondingly scaling down S. The consequences of this is that optimization of 2.1 with respect to both A and S leads to the elements of A growing (in absolute value) without bounds whereas S tends to zero. More importantly, the solution found does not depend on the second term of the objective as it can always be eliminated by this scaling trick. In other words, some constraint on the scales of A or S is needed. We chose to fix the norms of the columns of A.

2.2.2 Non Negative Sparse Coding

In standard sparse coding, the data is described as a combination of elementary features involving both additive and subtractive interactions. As discussed in the previous chapter, the fact that features can 'cancel each other out' using subtraction is contrary to the intuitive notion of combining parts to form a whole. Thus, both the non-negativity constraints and the sparseness goal are important for learning parts-based representations. We propose to combine these two methods into non-negative sparse coding:

Definition 2.2. Non-negative sparse coding (NNSC) of a non-negative data matrix $\mathbf{X}$ (i.e. $\forall ij : X_{ij} > 0$) is given by the minimization of[12]

$$C(\mathbf{A}, \mathbf{S}) = \frac{1}{2} \| \mathbf{X} - \mathbf{AS} \|^2 + \lambda \sum_{ij} f(S_{ij}) \tag{2.2}$$

subject to the constraints $\forall ij : \mathbf{A}_{ij} \geq 0$, $\mathbf{S}_{ij} \geq 0$ and $\forall i : \|a_i\| = 1$, where a_i denotes the i^{th} column of $\mathbf{A}$. It is also assumed that the constant $\lambda \geq 0$. Notice that we have here chosen to measure sparseness by a linear activation penalty (i.e. f(s) = s). This particular choice is primarily motivated by the fact that this makes the objective function quadratic in S. This is useful in the development and convergence proof of an efficient algorithm for optimizing the hidden components **S**.

2.2.3 Algorithm for Non Negative Sparse Coding

Following is the algorithm for learning the dictionary as per Non Negative Sparse Coding. [12][13]

1. Initialize A^0 and S^0 to random strictly positive matrices of the appropriate dimensions, and rescale each column of A^0 to unit norm. Set t = 0.
2. Iterate until convergence:
   (a) $B = A^t - \mu(A^t S^t - X)(S^t)^T$
   (b) Any negative values in B are set to zero.
   (c) Rescale each column of B to unit norm, and then set $A^{t+1} = B$.
   (d) $S^{t+1} = S^t .* ((A^t + 1)^T X)./((A^t + 1)^T (A^t + 1) S^t + \lambda)$.
   (e) Increment t.

2.3 Implementation

The algorithm is run on the noisy dataset, both simulated and real, with very high values of the variance of Gaussian noise. We wish to depict that storing sparse representations of signals and recovering them through a dictionary also serves as a denoising mechanism. Various degrees of noise is added to explore the relevance of the sparsity parameter. The root means square error(RMSE) of the denoised image with respect to the original image is compared.

2.3.1 Tuning the Sparsity Parameter for Denoising

The algorithm for NNSC takes the sparsity parameter λ as input. Hence, dictionary learning can only proceed once this parameter is aptly set. If we don't put in the optimum value of λ, we may expect underfitting and overfitting. For smaller values of λ, for instance zero in case of Non Matrix Factorisation, leads to almost no constraint on the sparsity i.e. the number of atoms that are used to represent the signal. This is

an example of overfitting. High values of λ do not allow sufficient number of atoms to represent the signal, hence, a case of underfitting.
In order to find the optimum value of λ for denoising, we carry out five cross validation. We divide the dataset into five parts, dictionary learning is done on four of these and validation is done on one. Then we take the average of the RMSE values w.r.t to the original image. The plot of average RMSE w.r.t. lambda is U shaped, with the minimum of this curve pointing to the optimum value of λ.
Note that the optimum lambda we get is corresponding to one level of noise present in the dataset. If we consider some another dataset having different value for lambda, the procedure of tuning the parameter λ will have to be done again.

2.3.2 Datasets

The simulated dataset is built from ten 3×3 features. The dataset includes 1-,2- and 3-combinations of these features. For different level of noises in the dataset, we tune the parameter λ, hence obtain the ideal dictionary and reconstruct the data.
The real dataset consists of a 512×512 real image. For optimal value of λ for a given noise level, we obtain dictionaries of size 10 and 50 and show reconstruct image.

2.4 Experimental Results

2.4.1 Simulated Dataset

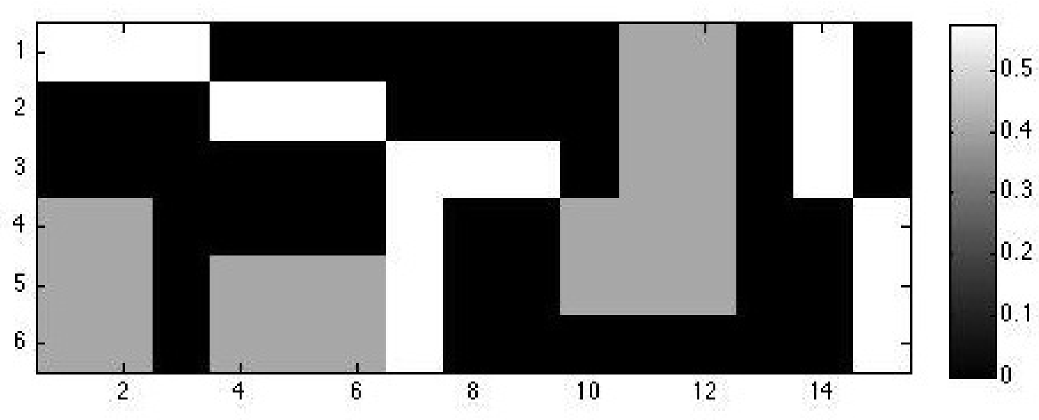

FIGURE 2.1: Actual Features of the original dataset

The two rows contain five atoms(3×3) each.

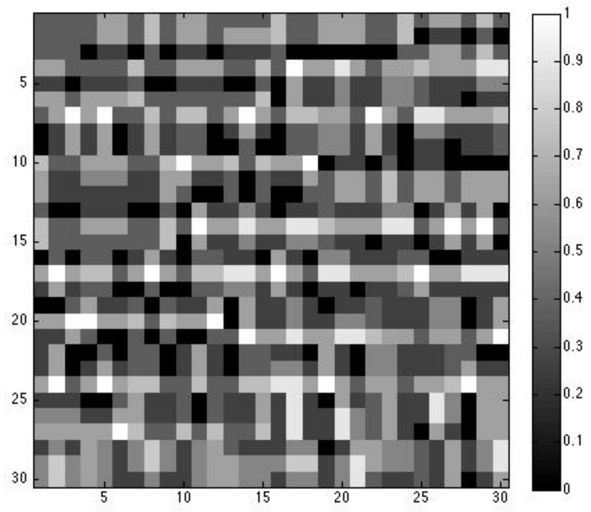

Figure 2.2: Original Hundred Patches in the Dataset

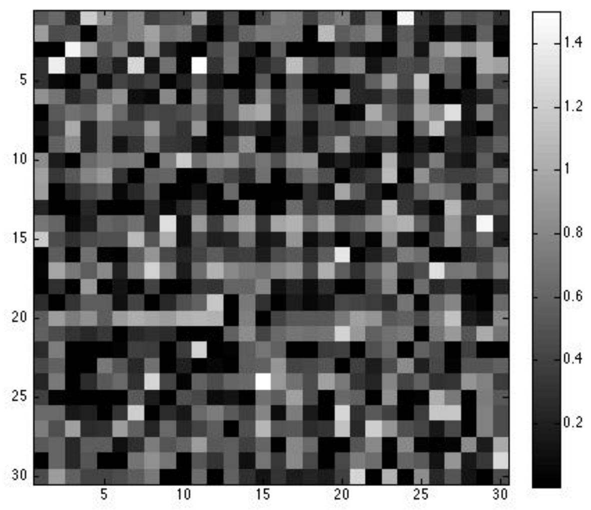

Figure 2.3: Noisy Hundred Patches in the Dataset in Run A

During Run A, the noise level added in the grayscale picture (values between 0-1) has standard deviation 0.3 and mean 0. The minimum average cross validation error is at λ = 0.15 as shown in the plot 2.4. The reconstructed patches and dictionary are shown in 2.5 and 2.6 respectively. The RMSE for noisy image is 0.2714 and RMSE for denoised image is 0.2130.

During Run B, the noise level added in the grayscale picture has standard deviation 0.1 and mean 0. From the RMSE plot it is clear that since degree of noise is low, no overfitting takes place and optimal value of the sparsity parameter turns out to be zero.

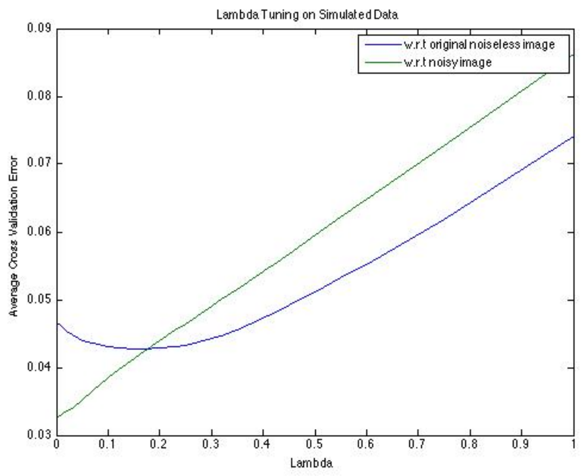

FIGURE 2.4: RMSE plot in the Run A

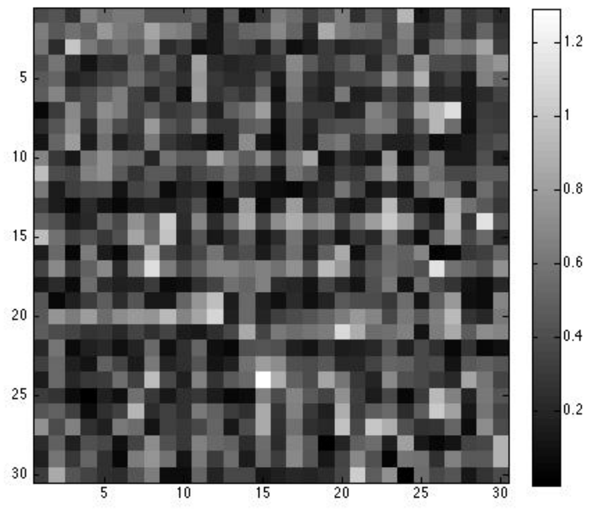

FIGURE 2.5: Reconstructed hundred patches at optimal λ in Run A

The reconstructed patches and dictionary are shown in 2.7 and 2.9 respectively. The RMSE for noisy image is 0.0966 and RMSE for denoised image is 0.0840.

We chose these two specific choices of noise levels to depict that in some cases where you have a risk of overfitting due to noise, a value of λ other than zero might correspond to the lowest RMSE. In cases where we don't have the overfitting phase, an increase in the sparsity parameter simply translates to increase in the RMSE. In that case an optimal λ value might be subjectively chosen based on the requirement of sparseness and exactness in the application.

During Run A, the dictionary coefficients of the dataset for optimal and zero values of λ a and represented in figures 2.10 and 2.11 respectively. Since the RMSE at optimal

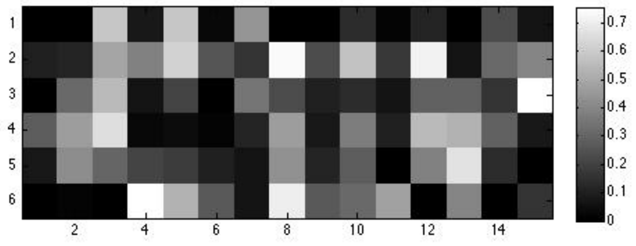

FIGURE 2.6: Dictionary at $\lambda = 0.15$ (optimal) in Run A

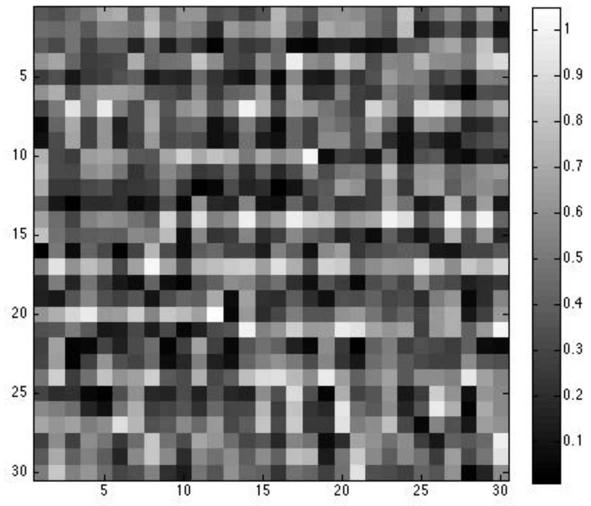

FIGURE 2.7: Reconstructed hundred patches at optimal λ in Run B

λ is low and the representation is more sparse, it is the best fit for our data.

2.4.2 Real Dataset

We chose the image of lena for exploring Non negative Sparse Coding in the domain of natural images. The reconstructed patches The noise level added in the grayscale picture (values between 0-1) has standard deviation 0.1 and mean 0. and dictionary at the optimal value of λ (0) are shown in 2.15 and 2.14 respectively. The RMSE for noisy image is 0.0981 and RMSE for denoised image is 0.0648.

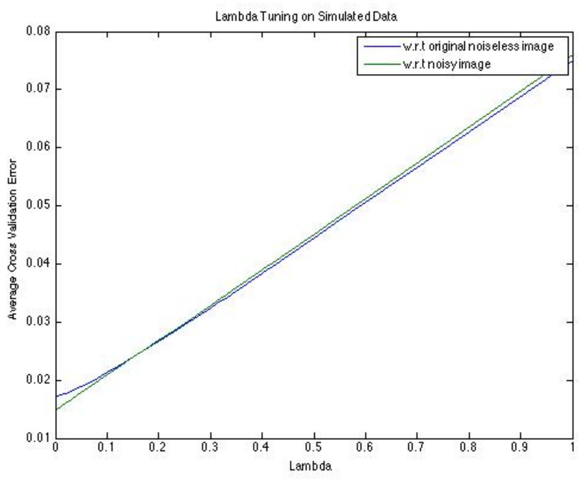

FIGURE 2.8: RMSE plot in the Run A

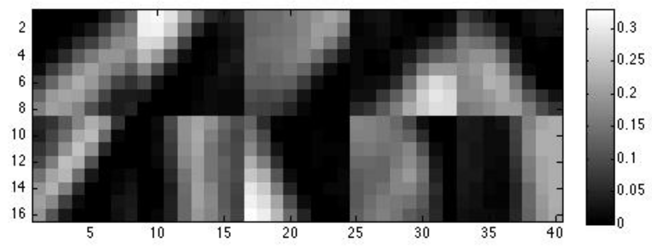

FIGURE 2.9: Dictionary at $\lambda = 0.25$ (optimal) in Run B

We repeated the experiment with the same level of noise but this time, a dictionary is learnt over 100 atoms. The reconstructed patches and dictionary at the optimal value of λ (0.01) are shown in 2.18 and 2.17 respectively. The RMSE for noisy image is 0.0981 and RMSE for denoised image is 0.0731.

What we see is that though now the features in the image, for instance the strands of hair are more explicit, hence better image quality but the RMSE has increased a bit.

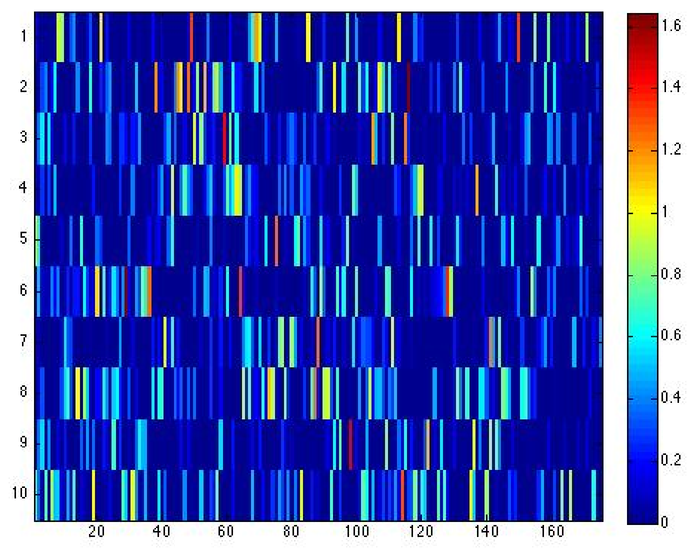

FIGURE 2.10: Dictionary Coefficients for Optimal Sparsity Parameter in the Run A

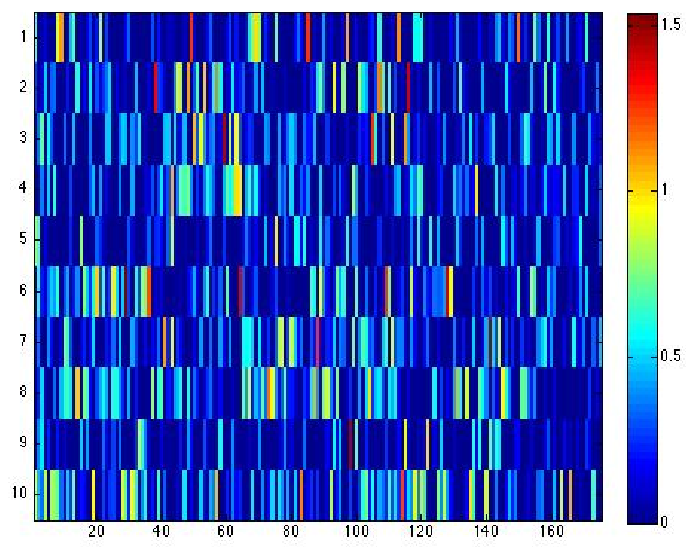

FIGURE 2.11: Dictionary Coefficients for Zero Sparsity Parameter in the Run A

2.5 Summary

In this chapter, we introduced the concept of Non Negative Sparse Coding. It combines sparse representations along with parts based representation. We saw its application in denoising. We calculated the optimal sparsity parameters for each noise level and showed results on Simulated and Real dataset.

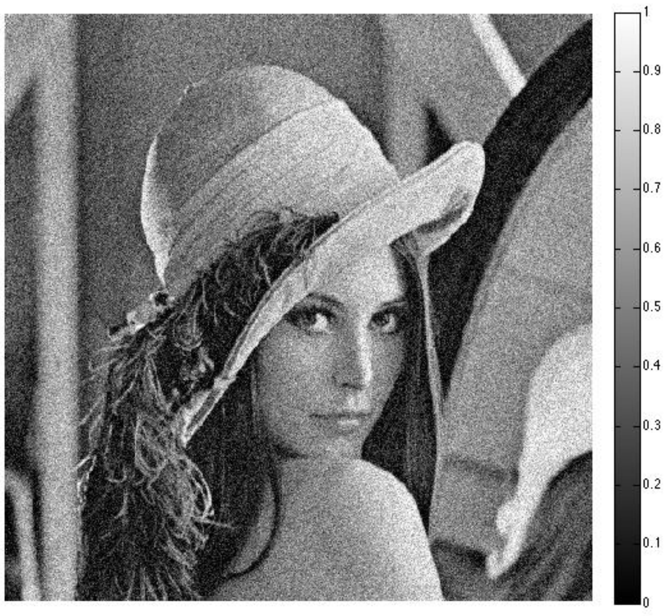

FIGURE 2.12: Noisy Image

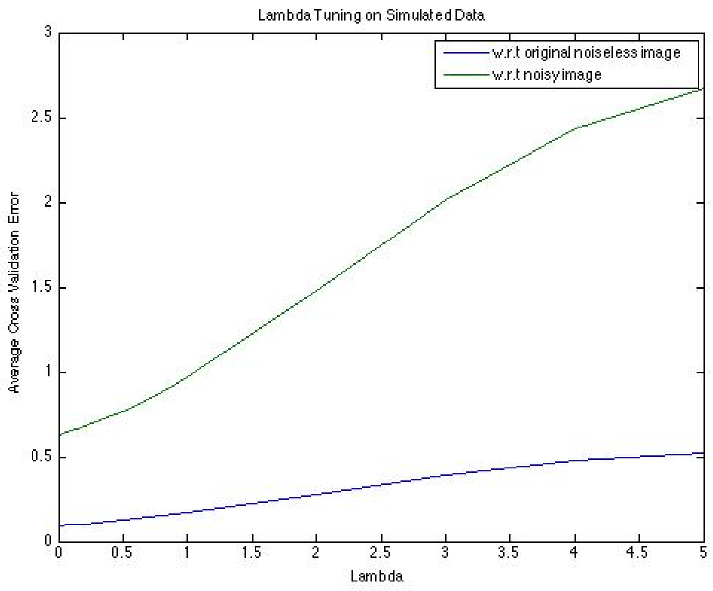

FIGURE 2.13: RMSE plot for the Real Image

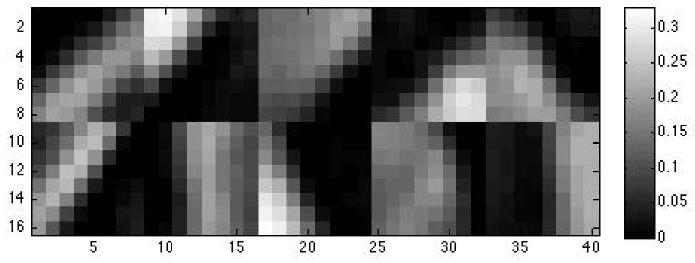

FIGURE 2.14: Dictionary atoms obtained at optimal value of Sparsity parameter

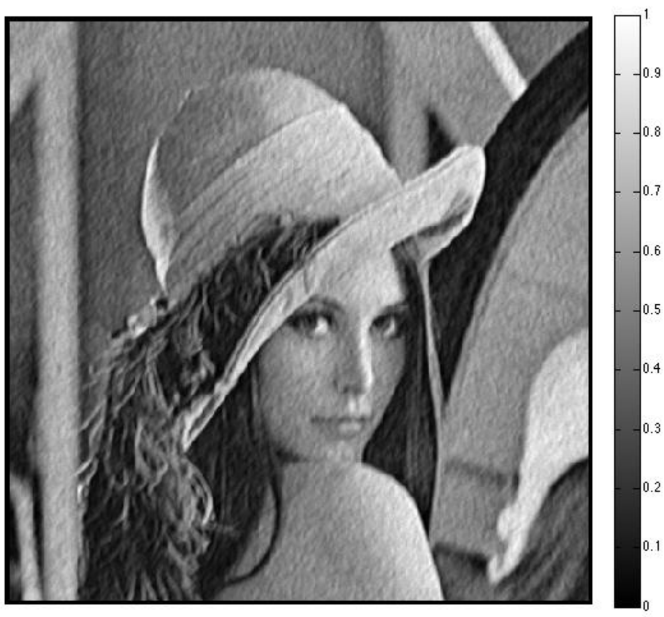

FIGURE 2.15: Reconstructed image at optimal value of Sparsity parameter

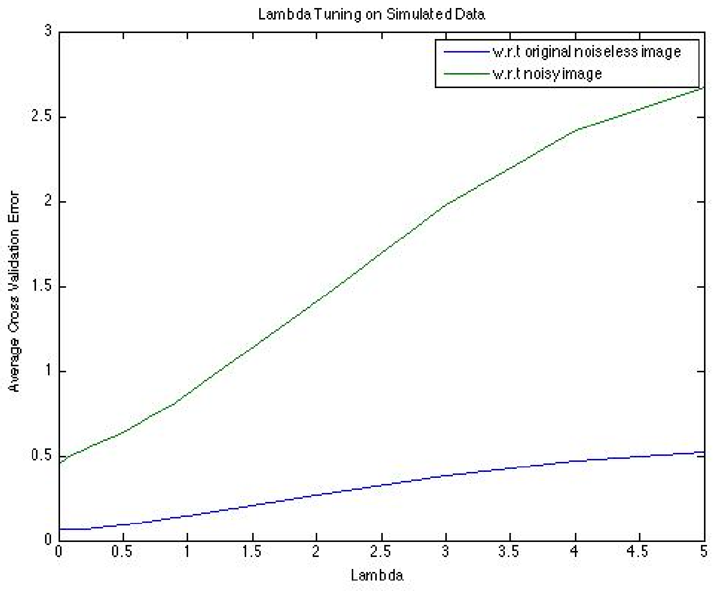

Figure 2.16: RMSE plot for the Real Image (for 100 atoms)

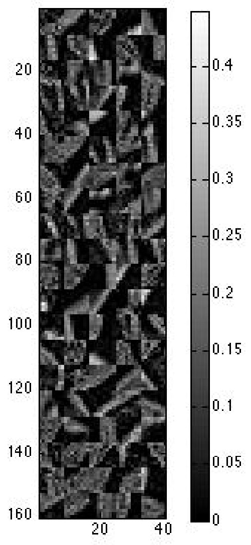

Figure 2.17: Dictionary atoms obtained at optimal value of Sparsity parameter (for 100 atoms)

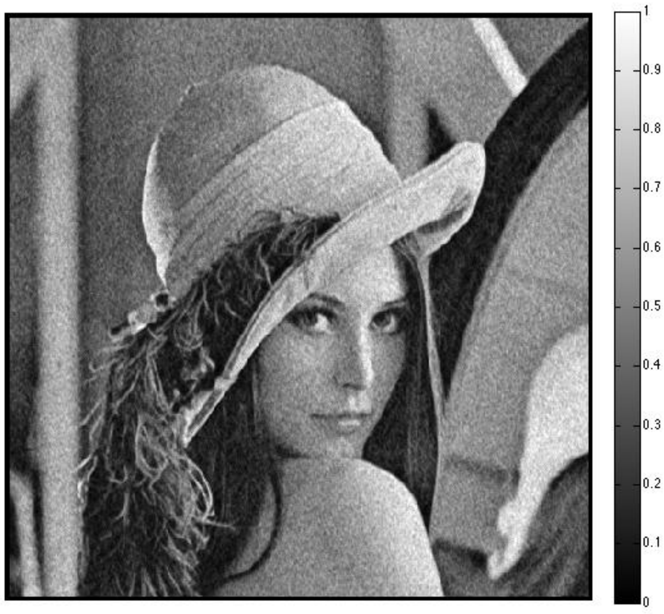

FIGURE 2.18: Reconstructed image at optimal value of Sparsity parameter (for 100 atoms)

Chapter 3

K-SVD

Given a set of training signals, we seek the dictionary that leads to the best representation for each member in this set, under strict sparsity constraints. We present a new method — the K-SVD algorithm [14] generalizing the K-means clustering process. K-SVD is an iterative method that alternates between sparse coding of the examples based on the current dictionary and a process of updating the dictionary atoms to better fit the data. The update of the dictionary columns is combined with an update of the sparse representations, thereby accelerating convergence. The K-SVD algorithm is flexible and can work with any pursuit method (e.g., basis pursuit, FOCUSS, or matching pursuit). We analyze this algorithm and demonstrate its results in applications on real image data.

3.1 Sparse Coding

Sparse coding is the process of computing the representation coefficients based on the given signal and the dictionary. This process, commonly referred to as "atom decomposition," requires solving 1.1 or 1.2.
Exact determination of sparsest representations proves to be an NP-hard problem. Thus, approximate solutions are considered instead, and in the past decade or so several efficient pursuit algorithms have been proposed. The simplest ones are the matching pursuit and the orthogonal matching pursuit [15] (OMP) algorithms. These are greedy algorithms that select the dictionary atoms sequentially. These methods are very simple, involving the computation of inner products between the signal and dictionary columns, and possibly deploying some least squares solvers. Both 1.1 and 1.2 are easily addressed by changing the stopping rule of the algorithm.

3.1.1 Orthogonal Matching Pursuit

The OMP algorithm aims to approximate the solution of one of the two following problems, the sparsity-constrained sparse coding problem, given by

$$\gamma^* = \arg\min_{\gamma} \|x - D\gamma\|^2 \quad \textit{subject to} \quad \|\gamma\|_0 \leq K$$

and the error-constrained sparse coding problem, given by

$$\gamma^* = \arg\min_{\gamma} \|\gamma\|_0 \quad \textit{subject to} \quad \|x - D\gamma\|^2 \leq \epsilon$$

For simplicity, we assume hereon that the columns of D are normalized to unit l^2-length. The greedy OMP algorithm selects at each step the atom with the highest correlation to the current residual. Once the atom is selected, the signal is orthogonally projected to the span of the selected atoms, the residual is recomputed, and the process repeats (see Algorithm 1). Note that line 5 is the greedy selection step, and line 7 is the orthogonalization step.

The computation in line 7 will usually not be carried out explicitly due to its high cost. A practical implementation employs a progressive Cholesky or QR update process to reduce the work involved in the matrix inversion. In a nutshell, the computation

$$\gamma_I = (D_I x)^+ = (D_I^T D_I)^{-1} x$$

requires the inversion of the matrix $D_I^T D_I$ which remains non-singular due to the orthogonalization process which ensures the selection of linearly independent atoms. The matrix $D_I^T D_I$ is an SPD (symmetric positive-definite) matrix which is updated every iteration by simply appending a single row and column to it, and therefore its Cholesky factorization requires only the computation of its last row. It is readily verified that given the Cholesky factorization of $A_p = L_p L_p^T \ \epsilon \ \mathbb{R}^{(n-1)\times(n-1)}$, the Cholesky factorization of

$$A = \begin{pmatrix} A_p & v \\ v^T & c \end{pmatrix} \quad \epsilon \quad \mathbb{R}^{n\times n}$$

is given by $A = LL^T$, with

$$L = \begin{pmatrix} L_p & 0 \\ w^T & \sqrt{c - w^T w} \end{pmatrix}, \quad w = L_p^{-1} v$$

Algorithm 2 gives the full OMP-Cholesky implementation.

Algorithm 1. Orthogonal Matching Pursuit

```
• Input: Dictionary D, signal x, target sparsity K or target error $\epsilon$
• Output: Sparse representation $\gamma$ such that x $\approx D\gamma$
• Init I := (), r := x, $\gamma$ := 0
• while (stopping criterion not met) do
      $k^* = \arg\max_k |d_k^T r|$
      I := (I; $k^*$)
      $\gamma_I = (D_I x)^+$
      r := x - $D\gamma_I$
  end while
```

Algorithm 2. OMP Cholesky[16]

```
• Input: Dictionary D, signal x, target sparsity K or target error $\epsilon$
• Output: Sparse representation $\gamma$ such that x $\approx D\gamma$
• Init I := (), L := [1] , r := x, $\gamma$ := 0 $\alpha := D^T$ x, n := 1
• while (stopping criterion not met) do
      $k^* = \arg\max_k |d_k^T r|$
      if n > 1 then
          w := Solve for w (Lw = $D_I^T k^*$)
          $L = \begin{pmatrix} L_p & 0 \\ w^T & \sqrt{c - w^T w} \end{pmatrix}$
      end if
      I := (I; $k^*$)
      $\gamma_I$ := Solve for c (L$L^T$c = $\alpha_I$)
      r := x - $D\gamma_I$
      n := n + 1 end while
```

3.2 KSVD

3.2.1 KSVD Overview

$$\min_{D,X} \|Y - DX\|_F^2 \quad \textit{subject to} \quad \forall i \quad \|x_i\|_0 \leq T_0 \tag{3.1}$$

Let us first consider the sparse coding stage, where we assume that is fixed, and consider the above optimization problem as a search for sparse representations with coefficients summarized in the matrix . The penalty term can be rewritten as

$$\|Y - DX\|_F^2 \;=\; \sum_{i=1}^{n} \|y_i - Dx_i\|_2^2 \tag{3.2}$$

Therefore the problem posed in can be decoupled to distinct problems of the form

$$\min_{D,X} \|y_i - Dx_i\|_2^2 \quad \textit{subject to} \quad \|x_i\|_0 \leq T_0 \;\forall i \;=\; 1 \;\; to \;\; n \tag{3.3}$$

This problem is adequately addressed by the pursuit algorithms, and if T_0 is small enough, their solution is a good approximation to the ideal one that is numerically in-feasible to compute.

We now turn to the second, and slightly more involved, process of updating the dictionary together with the nonzero coefficients. Assume that both D and X are fixed and we put in question only one column in the dictionary and the coefficients that correspond to it, the k^{th} row in X, denoted as x_T^k Returning to the objective function 3.4, the penalty term can be rewritten as

$$\|Y - \sum_{j=1}^{k} d_j x_j^T\|_F^2 \;=\; \|(Y - \sum_{j=1}^{k} d_j x_j^T) - d_k x_k^T\|_F^2 \;=\; \|E_k - d_k x_k^T\|_F^2 \tag{3.4}$$

We have decomposed the multiplication DX to the sum of K rank-1 matrices. Among those, K-1 terms are assumed fixed, and one—the k^{th}—remains in question. The matrix E_k stands for the error for all the examples when the k^{th} atom is removed.

Here, it would be tempting to suggest the use of the SVD to find alternative d_k and x_k^T. The SVD finds the closest rank-1 matrix (in Frobenius norm) that approximates E_k, and this will effectively minimize the error as defined in (21). However, such a step will be a mistake, because the new vector x_k^T is very likely to be filled, since in such an update of d_k we do not enforce the sparsity constraint.

A remedy to the above problem, however, is simple and also quite intuitive. Define ω_k as the group of indices pointing to examples (y_i) that use the atom d_k, i.e., those where x_T^k(i) is nonzero. Thus

$$\omega_k = \{i | 1 \leq i \leq k, x_T^k(i) \neq 0\}$$

Define Ω_k as a matrix of size N $\times$ $|\omega_k|$ with ones on the (ω_T^k(i),i)entries and zeros elsewhere. When multiplying $x_R^k = x_T^k \Omega_k$, this shrinks the row vector x_T^k, by discarding of the zero entries, resulting with the row vector x_R^k of length $|\omega|$. Similarly, the multiplication $Y_R^k = Y\Omega_k$ creates a matrix of n $\times$ $|\omega_k|$. that includes a subset of the examples that are currently using the d_k atom. The same effect happens with $E_R^k = E\Omega_k$, implying a selection of error columns that correspond to examples that use the atom d_k. Hence, now we can minimise the following function:

$$\|E_k\Omega_k - d_k x_T^k \Omega_k\|_F^2 = \|E_k^R - d_k x_R^k\|_F^2 \tag{3.5}$$

This can be done via SVD. Taking the restricted matrix E_R^k, SVD decomposes it to E_R^k $= U\Delta V$. We define the solution for d_k^* as the first column of U, and the coefficient vector x_R^k as the first column of V multiplied by $\Delta(1,1)$. Note that, in this solution, we necessarily have that i) the columns of D remain normalized and ii) the support of all representations either stays the same or gets smaller by possible nulling of terms.

3.2.2 KSVD and KMeans

While Kmeans applies computations of means to update the codebook, K-SVD obtains the updated dictionary by SVD computations, each determining one column.
When $T_0 = 1$ in 3.1, , the coefficient matrix X has only one nonzero entry per column. Thus, computing the error E_R^k in 3.5 yields

$$E_k^R = E_k\Omega_k = (Y - \sum_{j \neq k} d_j x_T^j)\Omega_k = Y\Omega_k = Y_k^R \tag{3.6}$$

This is because the restriction Ω_k takes only those columns in E_k that use the $d_k$atom, and thus necessarily, they use no other atoms, implying that for all j, $x_T^j \Omega_k = 0$.
This shows that the kmeans update of the cluster centroids could be interpreted as a sequential process.

3.2.3 Algorithm for KSVD

Task: Find the best dictionary to represent the data samples $\{y_i\}_{i=1}^N$ as sparse compositions, by solving

$$\min_{D,X} \|Y - DX\|_F^2 \quad \textit{subject to} \quad \forall i \quad \|x_i\|_0 \leq T_0$$

Initialisation: Set the dictionary matrix $D^0 \epsilon \mathbb{R}^{n \times K}$ with l^2 normalised columns. Set J = 1

Repeat until convergence(stopping rule):

- Sparse Coding Stage: Use any Pursuit Algorithm to compute the representation vectors x_i for each sample y_i by approximating the solution of

$$\min_{D,X} \|y_i - Dx_i\|_2^2 \quad \textit{subject to} \quad \|x_i\|_0 \leq T_0 \ \forall i \ = \ 1 \ \textit{to} \ n$$

- Codebook Update Stage: For each column k=1,2,3... K in D^{J-1} update it by
  - Define the group of examples that use this atom

$$\omega_k = \{i|1 \leq i \leq k, x_T^k(i) \neq 0\}$$

  - Compute the overall representation Error Matrix, E_k, by

$$E_k = (Y - \sum_{j \neq k} d_j x_T^j)$$

  - Restrict E_k by choosing only the columns corresponding to ω_k, and obtain E_k^R
  - Apply SVD Decomposition $E_k^R = U\Delta V$. Choose the updated dictionary atom d_k^* to be the first column of U. Update the coefficient vector x_k^R to be the first column of V multiplied by Δ(1,1).
- Set J = J+1

3.2.4 Implementation

We added gaussian noise of mean 0 and standard deviation 5 to the 0-255 image. Then we used 15, 25 and 70 atoms respectively to reconstruct the images. RMSE values of noisy image, denoised image with 15, 20 and 70 atoms, with respect to original image are 0.0303, 0.0619, 0.0602 and 0.0606. Corresponding images and dictionaries are shown. The denoised image obtained via kMeans with 70 atoms has RMSE 0.0609 with respect to original image.

It is quite evident that RMSE may not always indicate how good the dictionary is. An image which has better quality visually may have larger RMSE. RMSE does not take into account other quantitative factors as edge strength measure etc. RMSE alone can't decide how good the image is.

Figure 3.1: Original Image

3.3 Summary

We studied and implemented generalized k-Clustering algorithm kSVD for Dictionary Learning. We explored OMP and OMP Cholesky for Sparse Coding. We analyzed the results on natural images.

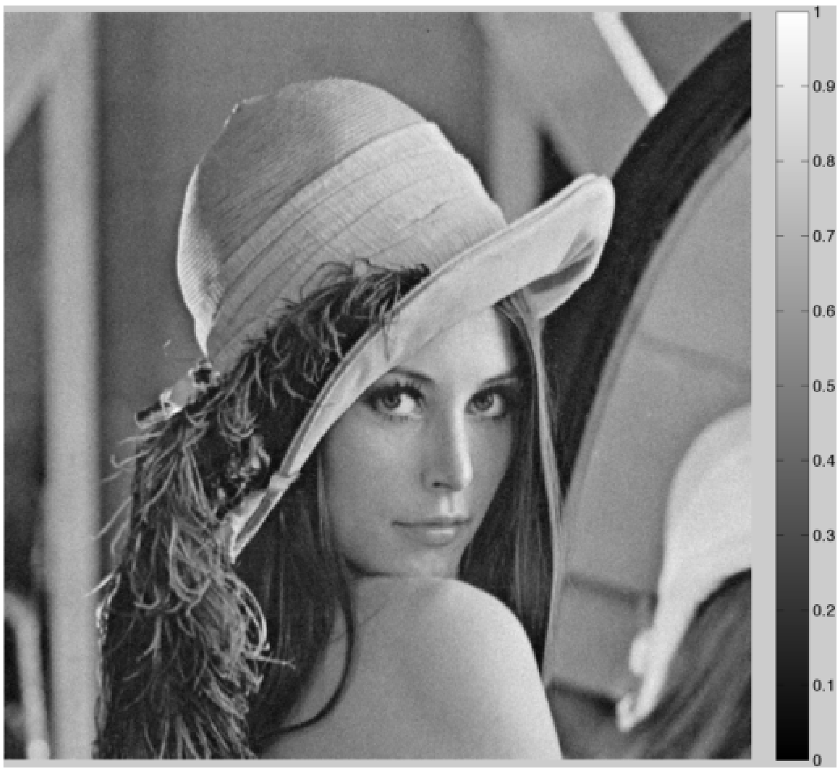

FIGURE 3.2: Noisy Image

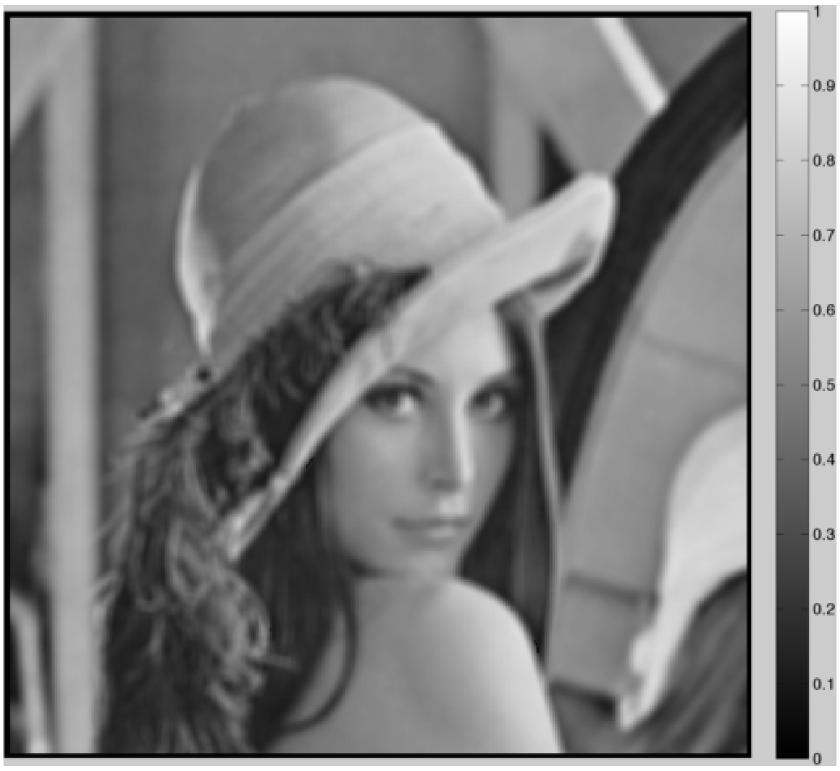

FIGURE 3.3: Denoised image with 15 atoms

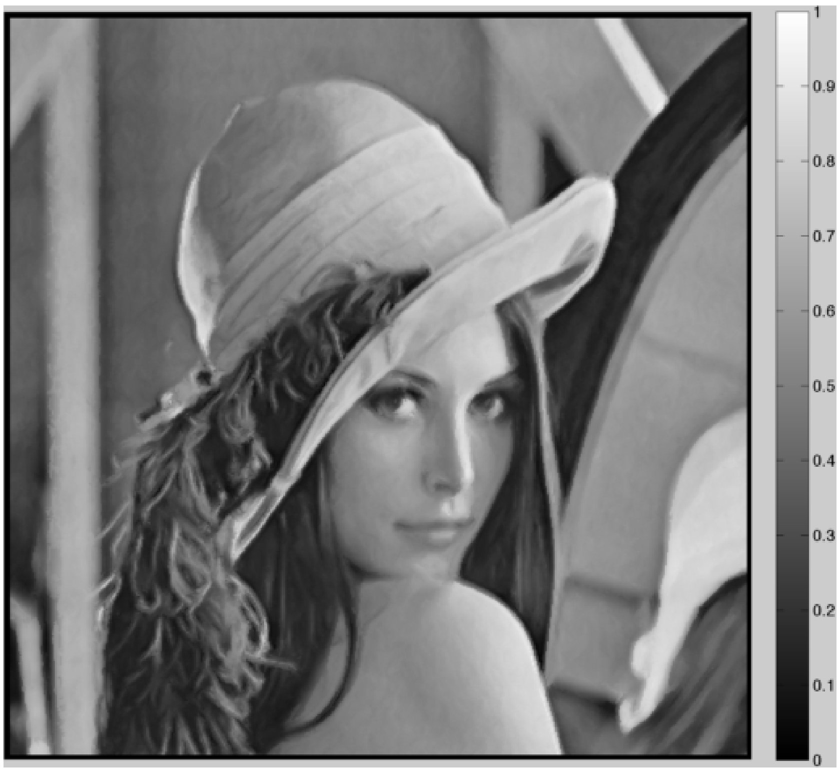

FIGURE 3.4: Denoised image with 25 atoms

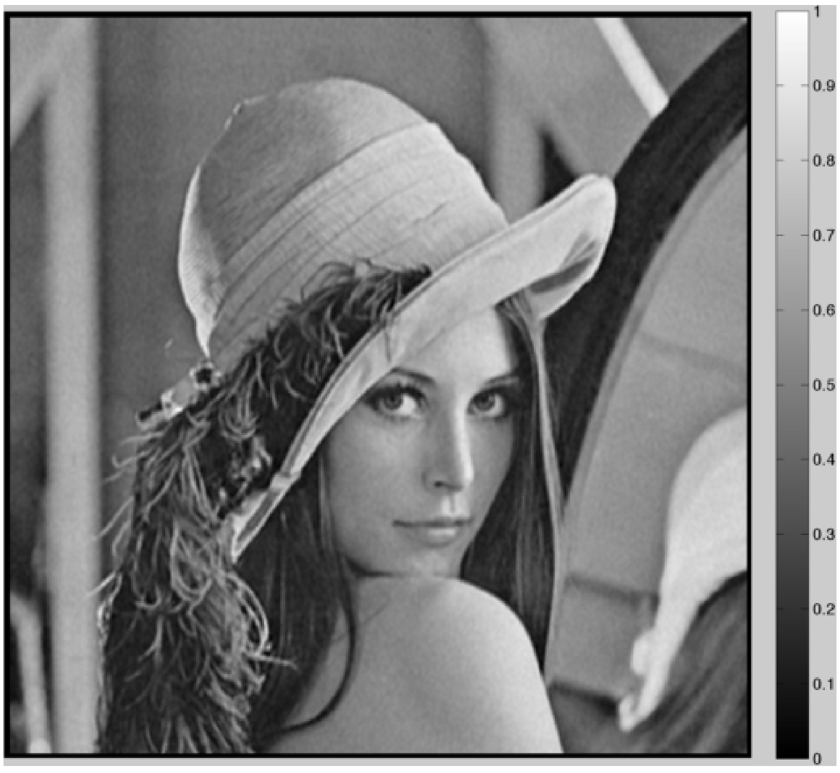

FIGURE 3.5: Denoised image with 70 atoms

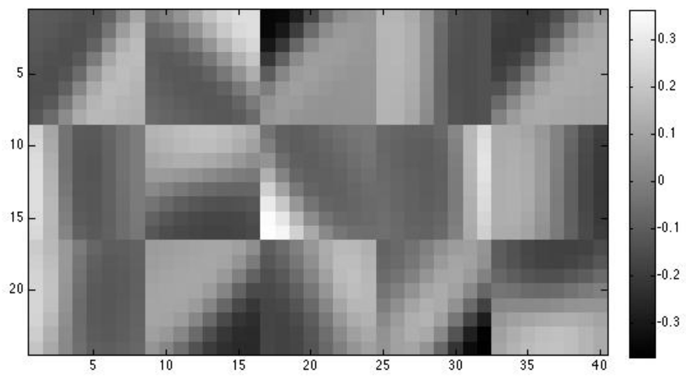

FIGURE 3.6: Dictionary with 15 atoms

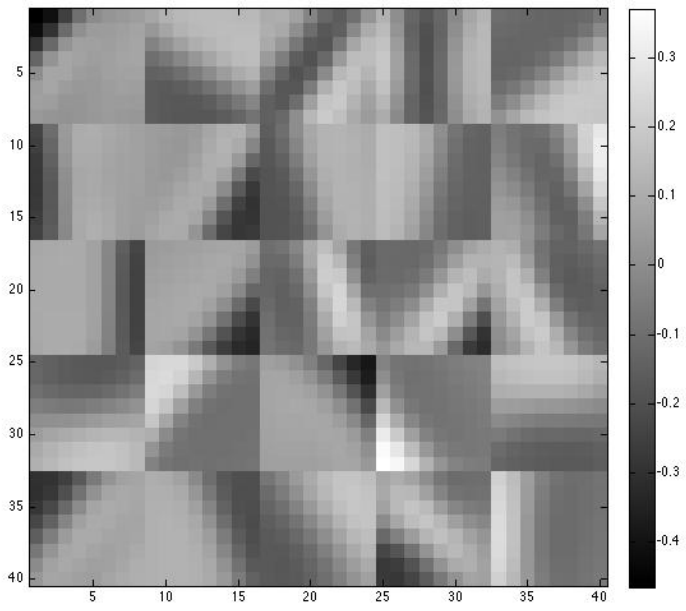

FIGURE 3.7: Dictionary with 25 atoms

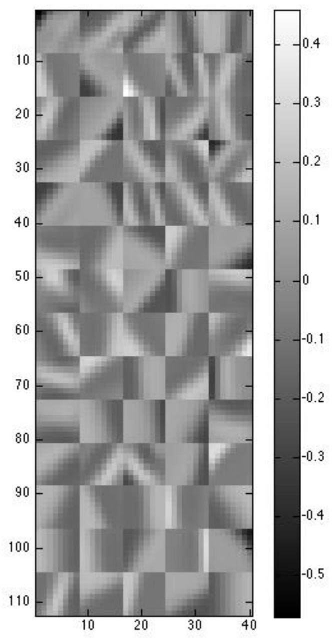

FIGURE 3.8: Dictionary with 70 atoms

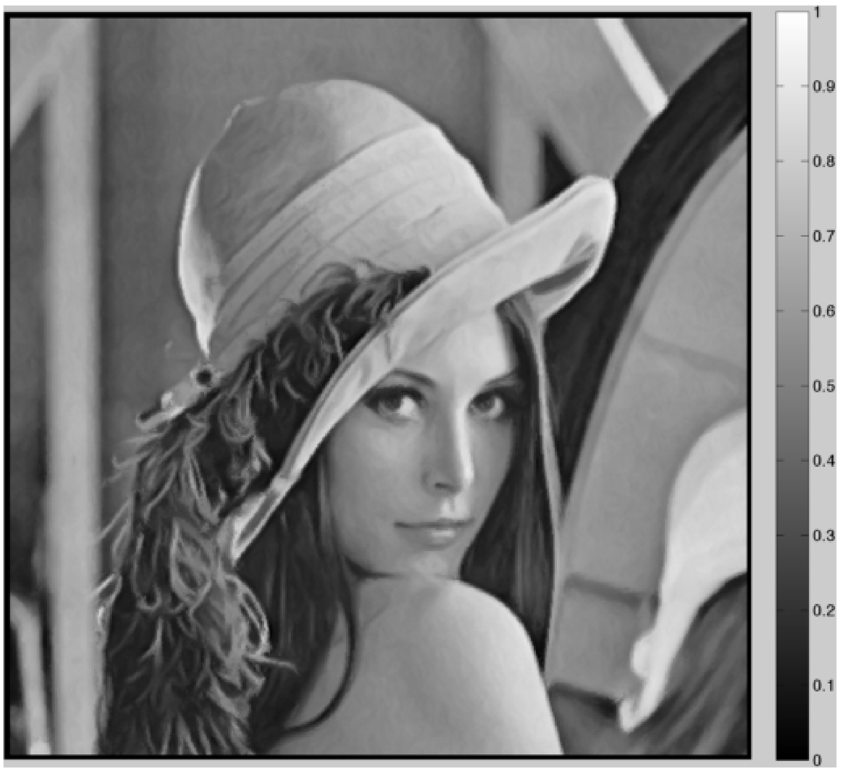

FIGURE 3.9: Denoised image with Kmeans(70 atoms)

Chapter 4

Re-weighted l1 Minimization

[17]

It is now well understood that it is possible to reconstruct sparse signals exactly from what appear to be highly incomplete sets of linear measurements and that this can be done by constrained l1 minimization. The algorithm consists of solving a sequence of weighted l1-minimization problems where the weights used for the next iteration are computed from the value of the current solution.

4.1 Introduction

Mathematically speaking and under sparsity assumptions, one would want to recover a signal x0 ϵ $\mathbb{R}^n$, e.g., the coefficient sequence of the signal in the appropriate basis, by solving the combinatorial optimization problem

$$\min_{x \epsilon \mathbb{R}^n} \|x\|_0 \quad \textit{subject to} \quad y = \phi x \tag{4.1}$$

where $\|x\|_{l_0} = \mid i : x_i \neq 0 \mid$

This is a common sense approach which simply seeks the simplest explanation fitting the data. In fact, this method can recover sparse solutions even in situations in which m $\ll$ n. This is of little practical use, however, since the optimization problem is non-convex and generally impossible to solve as its solution usually requires an intractable combinatorial search. A common alternative is to consider the convex problem

$$\min_{x \epsilon \mathbb{R}^n} \|x\|_1 \quad \textit{subject to} \quad y = \phi x \tag{4.2}$$

where $\|x\|_{l_1} = \sum_{i=1}^{n} \mid x_i \mid$

this problem is convex. It can actually be recast as a linear program and is solved efficiently. The programs 4.1 and 4.1 differ only in the choice of objective function, with the latter using an l1 norm as a proxy for the literal l0 sparsity count.

4.2 An Iterative Algorithm for Reweighted l1 Minimization

Consider the weighted l1 minimization problem

$$\min_{x \epsilon \mathbb{R}^n} \sum_{i=1}^{n} w_i x_i \quad \textit{subject to} \quad y = \phi x \tag{4.3}$$

where w1,w2,...,wn are positive weights. This problem is seen as a relaxation of weighted l0 minimisation problem

$$\min_{x \epsilon \mathbb{R}^n} \|Wx\|_0 \quad \textit{subject to} \quad y = \phi x \tag{4.4}$$

Whenever the solution to 4.1 is unique, it is also the unique solution to 4.4 provided that the weights do not vanish. However, the corresponding l1 relaxations 4.2 and 4.3 will have different solutions in general. Hence, one may think of the weights (w_i) as free parameters in the convex relaxation, whose values if set wisely could improve the signal reconstruction.
The most intuitive of all is: the weights can be inversely proportional to the true signal magnitude, i.e., that

$$w_i = \begin{cases} \frac{1}{|x_{0,i}|} & x_{0,i} \neq 0, \\ \inf & x_{0,i} = 0 \end{cases} \tag{4.5}$$

4.2.1 Algorithm

We propose a simple iterative algorithm that alternates between estimating x0 and redefining the weights. The algorithm is as follows:

1. Set the iteration count l to zero and and $w_i^{(0)} = 1$, i = 1,......n

2. Solve the weighted l1 minimisation problem

$$x^{(l)} = \arg\min \|W^{(l)}x\|_{l_1} \quad \textit{subject to} \quad y = \phi x \tag{4.6}$$

3. Update the weights: for each i=1,....,n,

$$w_i^{(l+1)} = \frac{1}{\mid x_i^l \mid + \epsilon} \tag{4.7}$$

4. Terminate on convergence or when l attains a specified maximum number of iterations l_m. Otherwise, increment 1 and go to step 2.

We introduce the parameter $\epsilon \geq 0$ in step 3 in order to provide stability and to ensure that a zero-valued component in $x^{(l)}$ does not strictly prohibit a nonzero estimate at the next step.

Using an iterative algorithm to construct the weights tends to allow for successively better estimation of the nonzero coefficient locations. Even though the early iterations may find inaccurate signal estimates, the largest signal coefficients are most likely to be identified as nonzero. Once these locations are identified, their influence is downweighted in order to allow more sensitivity for identifying the remaining small but nonzero signal coefficients.

4.3 Analytical Justification

The iterative reweighted algorithm falls in the general class of Majorization- Minimization (MM) algorithms. To establish this connection, consider the problem

$$\min_{x \epsilon \mathbb{R}^n} \sum_{i=1}^{n} \log\left(\mid x_i \mid + \epsilon\right) \quad \textit{subject to} \quad y = \phi x \tag{4.8}$$

which is equivalent to

$$\min_{x,u \epsilon \mathbb{R}^n} \sum_{i=1}^{n} \log\left(u_i + \epsilon\right) \quad \textit{subject to} \quad y = \phi x, \mid x_i \mid \leq u_i, i = 1, \ldots\ldots n \tag{4.9}$$

The equivalence means that if x^* is a solution to 4.8, then $(x^*, \mid x^* \mid)$ is a solution to 4.9. And conversely, if (x^*, u^*) is a solution to 4.9, then x^* is a solution to 4.8.

4.9 is of general form

$$\min_{v} g(v) \quad \textit{subject to} \quad v \epsilon C \tag{4.10}$$

where C is a convex set. In 4.9, the function g is concave and, therefore, below its tangent. Thus, one can improve on a guess v at the solution by minimizing a linearization of g around v. This simple observation yields the following MM algorithm: starting with $v(0) \epsilon C$, inductively define

$$v^{(l+1)} = \arg\min g(v^{(l)}) + \nabla g(v^{(l)}).(v - v^{(l)}) \quad \textit{subject to} \quad v \epsilon C \tag{4.11}$$

Each iterate is now the solution to a convex optimization problem. In the case 4.9 of interest, this gives

$$(x^{(l+1)}, u^{(l+1)}) = \arg\min \sum_{i=1}^{n} \frac{u_i}{u_i^{(l)} + \epsilon} \quad \textit{subject to} \quad y = \phi x, \mid x_i \mid \leq u_i, i = 1, ...n \tag{4.12}$$

which is of course equivalent to

$$x^{(l+1)} = \arg\min \sum_{i=1}^{n} \frac{\mid x_i \mid}{\mid x_i^{(l)} \mid + \epsilon} \quad \textit{subject to} \quad y = \phi x \tag{4.13}$$

Now, we are able to make sense of the iterative algorithm.

4.4 Re-weighted l1 Minimization in case of Non Sparse Coding

4.4.1 Approach for Sparse Coding

Given non-negative data y and non-negative dictionary ϕ, we have to find non=negative coefficients x.

$$\min_{x \epsilon \mathbb{R}^n} \sum_{i=1}^{n} \log{(x_i + \epsilon_1)} \quad \textit{subject to} \quad \|y - \phi x\|_2 < \epsilon, x_i > 0, i = 1, ...n \tag{4.14}$$

where ϵ is related to the noise in the data.

Here the function to be minimized is concave and the constraints are convex. Just as before, as per the MM algorithm, we will minimize

$$x^{(l+1)} = \arg\min \sum_{i=1}^{n} \frac{x_i}{x_i^{(l)} + \epsilon_1} \quad \textit{subject to} \quad \|y - \phi x\|_2 < \epsilon, x_i > 0, i = 1, ...n \tag{4.15}$$

where ϵ_1 is a small positive number.

In such a problem, we use Interior Point Methods to find a solution. But to apply interior point method, one has to provide a starting point in the convex set.

Our convex set is composed of two types of constraints, one pertaining to the norm and the other pertaining to the non negativity. We use projection onto convex sets to find a point satisfying both the constraints. Since we know how to project on the two sets individually, we can keep on projecting on the two sets alternately, and the solution converges to the point needed if the intersection is non zero.

Projecting onto the convex set involving the norm requires us to solve the following convex problem:

$$\min_{x \epsilon \mathbb{R}^n} \|x - x_p\|_2 \quad \textit{subject to} \quad \|y - \phi x\|_2 < \epsilon \tag{4.16}$$

where x_p is the point to be projected onto the convex set.

Projecting onto the convex set involving non negativity, simply thresholds the negative entries to zero.

Hence, the sparse coding can be done by interior point method.

4.4.2 Approach for Dictionary Learning

In the standard form, Non-negative sparse coding (NNSC) of a non-negative data matrix $\mathbf{X}$ (i.e. $\forall ij : X_{ij} > 0$) is given by the minimization of[12]

$$C(\mathbf{A}, \mathbf{S}) = \frac{1}{2} \| \mathbf{X} - \mathbf{AS} \|^2 + \lambda \sum_{ij} f(S_{ij}) \tag{4.17}$$

subject to the constraints $\forall ij : \mathbf{A}_{ij} \geq 0$, $\mathbf{S}_{ij} \geq 0$ and $\forall i : \|a_i\| = 1$, where a_i denotes the i^{th} column of $\mathbf{A}$. It is also assumed that the constant $\lambda \geq 0$.

In our case, f corresponds to the log penalty function.

In order to update the dictionary, we do projected gradient descent on the above optimization function. Not that this optimization function is different from the one we use for Log Sparse Coding.

4.5 Implementation

4.5.1 Simulated Dataset

We have 110 10*10 atoms as shown. The dataset is formed by 1- and 2- combination of these atoms.

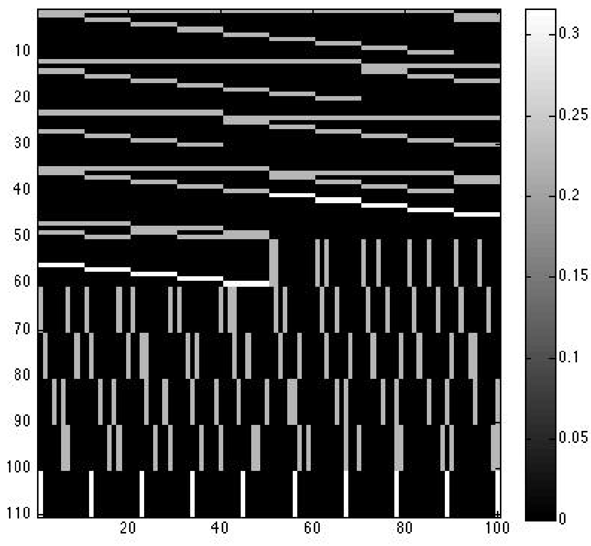

Figure 4.1: Features of the dataset

When we do standard and log sparse coding with the perfect dictionary (shown in the previous slide), the RMSE values for some random 20 patches is 22.0555 and 4.3337e-04 respectively.

Total number of non-zero coefficients for 20 atoms with standard sparse codes is 1249, and with log sparse codes is 342.

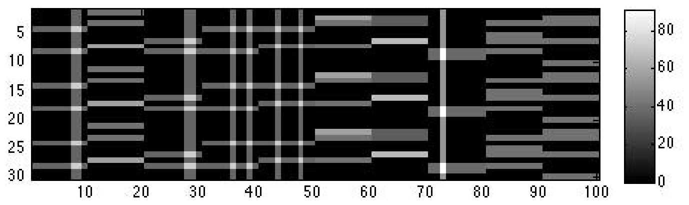

FIGURE 4.2: Random 10 atoms reconstructed with Standard, Log Sparse Coding and Original

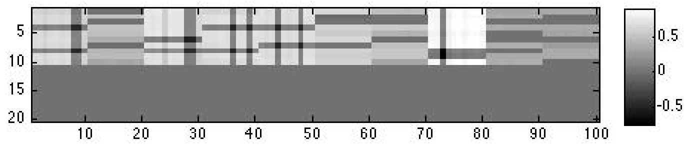

FIGURE 4.3: Differences w.r.t to Original Data

We are taking atoms of size 10*10. We get 10 atoms consisting of horizontal bars at different positions. We get 10 atoms consisting of vertical bars at different positions. We take 2-combinations of horizontal and vertical bar atoms individually. Hence, we have 10+10+45+45=110 atoms.

We form a dataset containing 1 and 2 combinations of these 110 atoms. Then we

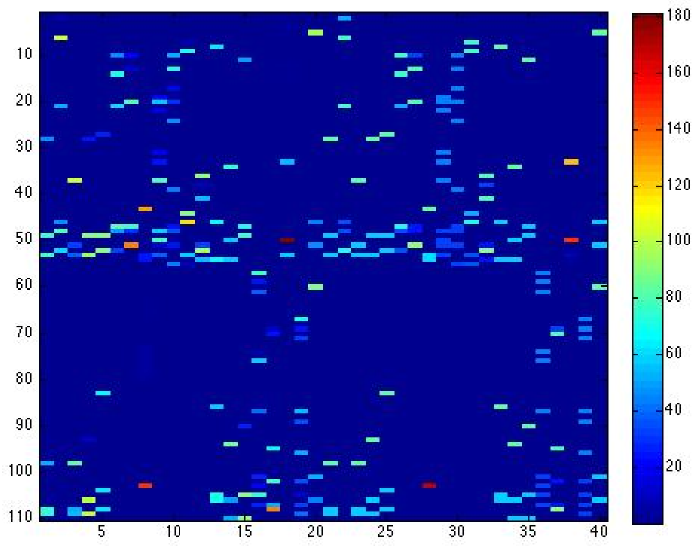

FIGURE 4.4: Coefficients yielded by Standard and Log Sparse Codes

randomly pick 1000 patches from this dataset and train on it and test it on randomly picked 100 atoms.

For the same perfect dictionary(actual 110 atoms), the sparse codes yielded by Log Sparse Coding gives better results, lower rmses as shown.

If the dictionary is imperfect, then ϵ not only reflects the noise factor but also the inability of the dictionary to represent the atoms to the fullest. Hence, ϵ has to be tuned. So, this time, I tuned epsilon for each atom, and the rmses were close enough as shown.

Now, if I do dictionary learning itself with different sparse codes, I get the following results.

We use the following imperfect dictionary.

When we do standard and log sparse coding with the imperfect dictionary (shown in the previous slide), the RMSE values for some random 20 patches is 282.0954 and 281.8776 respectively.

Total number of non-zero coefficients for 20 atoms with standard sparse codes is 589, and with log sparse codes is 183.

Following are the dictionary with the Log Sparse Coding and Standard Sparse Coding.

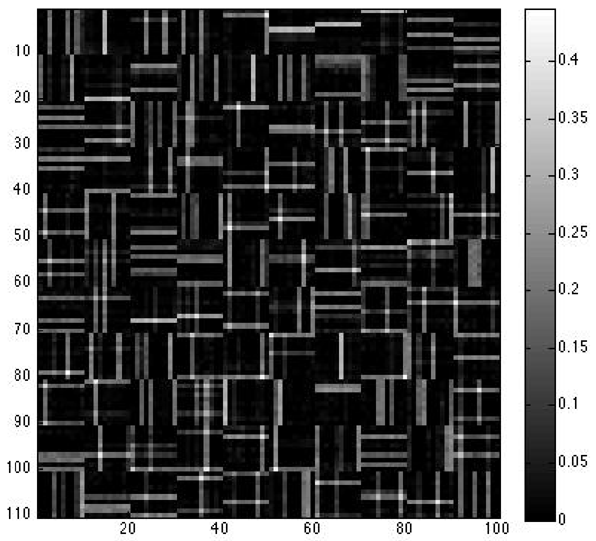

Figure 4.5: Dictionary Atoms

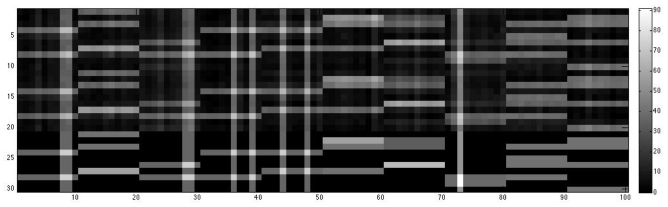

Figure 4.6: Random 10 atoms reconstructed with Standard, Log Sparse Coding and Original

When we do standard and log sparse coding with the respective dictionaries (shown in the previous slide), the RMSE values for some random 20 patches is 119.2053 and 222.7440 respectively. [Why?]

Total number of non-zero coefficients for 20 atoms with standard sparse code dictionary is 1448, and with log sparse code dictionary is 997.

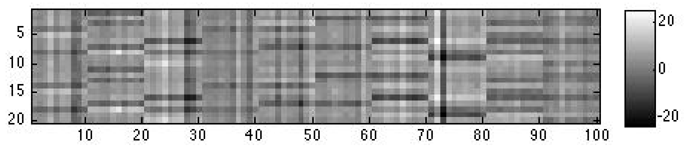

FIGURE 4.7: Differences w.r.t to Original Data

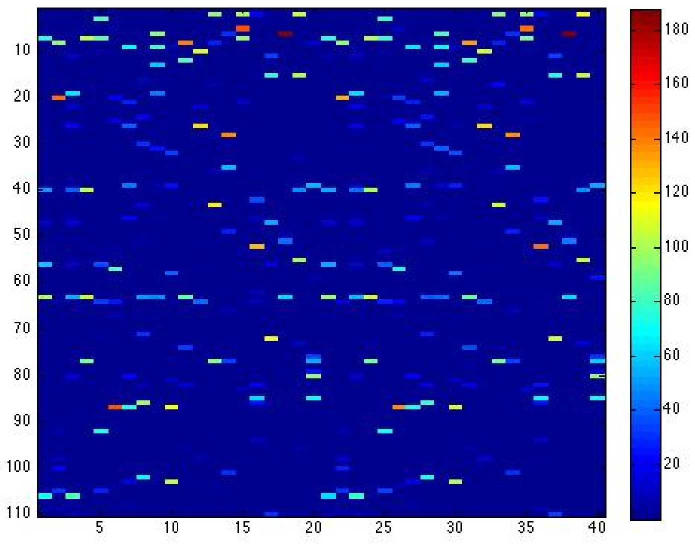

FIGURE 4.8: Coefficients yielded by Standard and Log Sparse Codes

4.5.2 Real Dataset

We have 10 (92*112) Face images each of 35 people. We train on 5 per person and test on 5. We do similar experiments as before on this learnt dictionary of 256 atoms.

Total number of non-zero coefficients for 5 atoms with standard sparse code dictionary is 1280, and with log sparse code dictionary is 91. RMSE of data patches wrt Standard

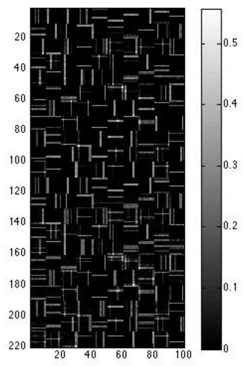

Figure 4.9: Dictionary Atoms

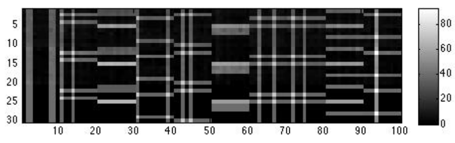

Figure 4.10: Random 10 atoms reconstructed with Standard, Log Sparse Coding and Original

and Log Sparse Codes are 231.4024 and 185.7587 respectively.

Image Reconstruction with Log Sparse Coding with Relaxed Constraints on ϵ. The final RMSEs wrt standard and log sparse coding based dictionaries are 734.2352 and 1886.8.

4.6 Summary

In this chapter, we introduced the concept of Non Negative Sparse Coding with Log Penalty. We saw its results on simulated datasets and face database. It has worked decent.

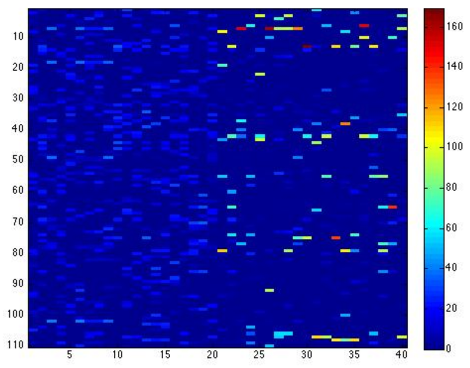

FIGURE 4.11: Coefficients with Log Sparse Coding and Standard Sparse Coding

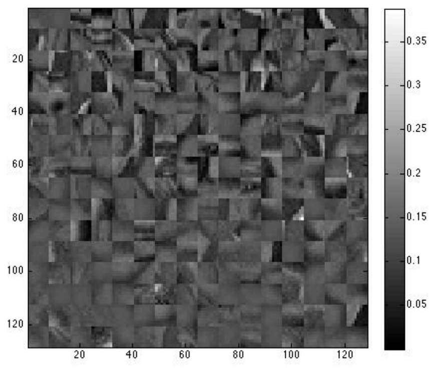

FIGURE 4.12: Dictionary Atoms

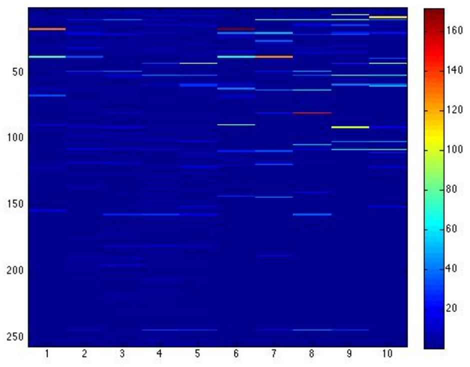

FIGURE 4.13: Coefficients using Log and Standard Sparse Coding

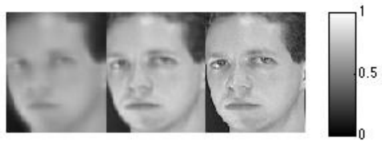

FIGURE 4.14: Reconstruction using Log, Standard Sparse Coding and Original

Bibliography

[1] Jonathan J Hunt, Peter Dayan, and Geoffrey J Goodhill. Sparse coding can predict primary visual cortex receptive field changes induced by abnormal visual input. *PLoS computational biology*, 9(5):e1003005, 2013.

[2] Michael W Marcellin, Michael J Gormish, Ali Bilgin, and Martin P Boliek. An overview of jpeg-2000. In *Data Compression Conference, 2000. Proceedings. DCC 2000*, pages 523–541. IEEE, 2000.

[3] David L Donoho, Iain M Johnstone, et al. Ideal denoising in an orthonormal basis chosen from a library of bases. *Comptes rendus de l'Académie des sciences. Série I, Mathématique*, 319(12):1317–1322, 1994.

[4] Ronald R Coifman and David L Donoho. *Translation-invariant de-noising.* Springer, 1995.

[5] Eero P Simoncelli, William T Freeman, Edward H Adelson, and David J Heeger. Shiftable multiscale transforms. *Information Theory, IEEE Transactions on*, 38(2): 587–607, 1992.

[6] J-L Starck, Emmanuel J Candès, and David L Donoho. The curvelet transform for image denoising. *Image Processing, IEEE Transactions on*, 11(6):670–684, 2002.

[7] R Gastaud and JL Starck. Dynamic range compression: a new method based on wavelet transform. In *Astronomical Data Analysis Software and Systems Conference, Strasbourg*, 2003.

[8] J-L Starck, Michael Elad, and David L Donoho. Image decomposition via the combination of sparse representations and a variational approach. *Image Processing, IEEE Transactions on*, 14(10):1570–1582, 2005.

[9] Michael Elad, J-L Starck, Philippe Querre, and David L Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Applied and Computational Harmonic Analysis*, 19(3):340–358, 2005.

[10] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[11] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1. *Vision research*, 37(23):3311–3325, 1997.

[12] Patrik O Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 557–565. IEEE, 2002.

[13] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[14] Michal Aharon, Michael Elad, and Alfred Bruckstein. k-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–4322, 2006.

[15] Sheng Chen, Stephen A Billings, and Wan Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control*, 50(5):1873–1896, 1989.

[16] Ron Rubinstein, Michael Zibulevsky, and Michael Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *CS Technion*, page 40, 2008.

[17] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 14 (5-6):877–905, 2008.