K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation

A Project Report

submitted by

AKUNURI NARESH

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

and

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2015

THESIS CERTIFICATE

This is to certify that the thesis titled **K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation**, submitted by **Akunuri Naresh (EE10B077)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Dual Degree** in Electrical Engineering with Specialization in Communication Systems, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Arun Pachai Kannu
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 12th May 2015

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my guide, Dr. Arun Pachai Kannu for his invaluable support and guidance throughout my project duration. I have gained immense insight into signal processing techniques and image processing in this one year, which I believe will hold me in good stead in my future endeavors. I am indebted to him for patiently answering all my questions at times when I could not see any path forward.

I also thank all my friends Raj Kumar, Sailendra, Bharath Kumar, Rajesh, Rohith, Sai Vijayendra, Shashank, Raman, Shubam and Sathish for being extremely helpful and for their patience and guiding me through difficult tasks. They have been the best experience in my undergraduate life. There is always something new to learn from each person, I have learnt more than I ever learnt anything before coming here.

I would like to thank my labmate Haritha for being helpful to me throughout my project.

I would also like to thank my parents for being flexible and giving me moral support throughout my education. I believe that what I am today and what I will be tomorrow is a result of their wonderful upbringing. I am also thankful to my brother, who constantly gives me support and guides me through the life difficult situations. I dedicate this thesis to my family.

Last but not the least, I thank Professor Dr.A.N. Rajagopalan for his notes on Image Signal Processing.

Naresh

ABSTRACT

KEYWORDS: Atom decomposition, dictionary, sparse representation, K-SVD, psnr, median filter.

We study the sparse representation of signals using an overcomplete dictionary, in which signals are being described by linear combinations of dictionary atoms. We use pursuit algorithms that decompose signals with respect to a given dictionary. We implement an algorithm for adopting dictionaries to achieve sparse representations. Given a set of training signals, we build the dictionary that leads to the best representation for each member in the set, under strict sparsity constraints. We implement a new algorithm named the K-SVD algorithm which is an iterative method that alternates between sparse coding of the training signals based on the current dictionary and a process of updating the dictionary atoms to better fit the data. We analyze this algorithm and demonstrate some results on real image data.

We implement the algorithm and apply it to real world application: filling in missing pixels. We delete random pixels in the image and fill their values using the various dictionary atom decomposition. We then apply filter to the dictionary reconstructed image and corrupted image, calculate their psnr values with respect to original image. We show that after applying median filter, dictionary built image shows better psnr than corrupted image.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

OMP	Orthogonal Matching Pursuit
MOD	Method of Optimal Directions
SVD	Singular Value Decomposition
PSNR	Peak Signal to Noise Ratio
MSE	Mean Square Error
VQ	Vector Quantization
PCA	Principal Component Analysis
SRC	Sparse Representation-based Classification

NOTATION

y	data sample
D	Dictionary
x	Sparse Coefficient vector
r	residual
$\|\|\|\|_p$	p-norm
ϵ	tolerance
$\min$	Minimum
$\mathbb{R}$	Real numbers

CHAPTER 1

INTRODUCTION

Recently there has been a lot of interest towards the sparsity and its applications. The process of digitally sampling a signal leads to its representation as the sum of delta functions. This representation is convenient for the purpose of display and is mostly inefficient for analysis tasks. We require more meaningful representations which captures the useful characteristics of the signal.

A dictionary which comprises of the set of elementary signals or atoms is used to represent a signal. Every signal is uniquely represented as the linear combination of dictionary atoms, when the dictionary columns forms a basis. In a simple case, the dictionary is orthogonal, then the representation coefficients can be computed as inner products of the signal and the atoms. If dictionary forms bi-orthogonal basis, then the coefficients are equal to the inner products of the signal and the dictionary inverse. For years, orthogonal and bi-orthogonal dictionaries were used in variety of signal applications because of their mathematical simplicity. However their limited expressiveness eventually outweighed their simplicity. This led to the development of overcomplete dictionaries, having more atoms than the dimension of the signal, with which we can represent a wide range of signals.

We consider the dictionary $D = [d_1 d_2 \dots d_K] \in \mathbb{R}^{N \times K}$, where the columns represent the atoms, and $K \geq N$. Representing a signal $x \in \mathbb{R}^N$ using the dictionary can take one of two paths: (i) *analysis* path, (ii) *synthesis* path.

In *analysis* path, we use a set of linear filters, and the signal is represented using its inner products with the atoms,

$$\gamma = D^T x \tag{1.1}$$

In *synthesis* path, we use a set of atomic signals, and the signal is represented as a linear combination of the dictionary atoms,

$$x = D\gamma \tag{1.2}$$

In the *synthesis* approach, when D is overcomplete, the solution set satisfying 1.2 is infinitely large, with the degrees of freedom identified with the null-space of D. This gives the most informative representation of the signal with respect to some cost function $C(\gamma)$:

$$\gamma = Arg \min_{\gamma} C(\gamma) \quad \text{subject to} \quad x = D\gamma \tag{1.3}$$

Practical choice of $C(\gamma)$ gives the sparsity of the representation, and we also want the sorted coefficients to decay quickly. Solving 1.3 is commonly referred as *sparse coding*. We can achieve sparsity by choosing $C(\gamma)$ as some robust function, which is tolerant to large coefficients but aggressively penalizes small non-zero coefficients. Examples of cost functions include the Huber function and various ℓ^p cost functions with $0 \leq p \leq 1$.

Overcomplete dictionaries allow higher sparsity of signal representation than orthogonal basis that increase lot of flexibility, compactness and applicability. Applications that uses sparsity and overcompleteness are compression, feature extraction, regularization in inverse problems.

Exact determination of sparsest representation is an NP-hard problem. In this thesis, we review some of the methods and we address the issue of designing the proper dictionary under strictly sparsity constraints.

1.1 Dictionary Designing

An overcomplete dictionary that leads to sparsest representation can either be a set of prespecified functions or designed by adapting its content to fit a given set of examples.

Using an overcomplete dictionary $D \in \mathbb{R}^{n \times K}$ where the columns represent the signal-atoms ,$\{d_j\}_{j=1}^{K}$, a signal $y \in \mathbb{R}^n$ can be represented as a sparse linear combination of these atoms. The representation of y may either be exact, $y = Dx$, or approximate, $y \approx Dx$, satisfying $\|y - Dx\|_p \leq \epsilon$. The vector $x \in \mathbb{R}^K$ contains the sparse coefficients of the signal y.

A variety of dictionaries were developed from one of the two sources either a *mathematical model* of the data, or *a set of realizations* of the data. Dictionaries of first kind are analytic in nature and has fast implementation, while dictionaries of the second type deliver increased flexibility and the ability to adopt to specific signal data.

Choosing a prespecified dictionary gives simple and fast algorithms for the evaluation of the sparse representation. This include overcomplete wavelets, curvelets, contourlets, short-time Fourier transforms. Those dictionaries succeed in applications in which how closely they are to sparsely represent the given examples .

1.1.1 A Brief Intro to Analytic Dictionaries

Analytic dictionaries arise from a mathematical model of the signals. Dictionary atoms generally have analytic formulations and we have to know the mathematical properties of the signals.
Examples of analytic dictionaries are

$$\begin{aligned}
\text{Fourier:} \quad & \phi_k(x) = e^{i2\pi kx}, \\
\text{Gabor:} \quad & \phi_{k,n}(x) = \omega(x - \beta n)e^{i2\pi\alpha kx}, \\
\text{Wavelets:} \quad & \phi_{m,n}(x) = \alpha^{m/2} f(\alpha^m x - \beta n), \\
\text{Curvelets:} \quad & \phi_{m,n,l}(x) = \phi_m(R_{\Theta_l}(x - x_n^{m,l}))
\end{aligned}$$

There are fast algorithms for computing these transforms but they have limited expressiveness.

1.1.2 A Brief Intro to Trained Dictionaries

These dictionaries arise from a set of examples of the signal data itself. In this case, dictionary atoms are learned from actual data. These dictionaries quickly adapt to the trained signals and gives better performance in applications. Examples include MOD, K-SVD, Generalized PCA etc.

In this thesis, We take a different method for designing dictionaries based on learning. Our goal is to find the dictionary that yields sparse representation for the training signals. With ever-growing computational capabilities, computational cost may become insignificant in importance to the improved performance achievable by methods that adopt dictionaries for special classes of signals.

1.2 Organization of Thesis

The outline of the thesis is as follows. The chapter on *Basic theory* presents a brief description about the optimization problem and gives various algorithms and mathematics involved in the project. The chapter on *Dictionary designing methods* gives an in-depth analysis of various methods exists prior to the proposed method. The chapter on *K-SVD algorithm* presents the detailed description of the proposed algorithm and its implementation details. The chapter on *Applications to Image processing* presents the results obtained after applying the proposed algorithm to popular method in Image processing.

At the end of the report, We give a brief description about the future of the project and how the proposed algorithm can be applied to some of the popular Image Processing techniques.

CHAPTER 2

BASIC THEORY

We discuss the basics of some of the Sparse representation algorithms in this chapter to gain an understanding of the theory behind the project.

2.1 Sparse Representation

Sparse representation refers to the ability to describe the signals as a linear combinations of a few atoms from a prespecified dictionary. Sparse coding is the process of computing the representation coefficients x based on the given signal y and the dictionary D.
This problem of sparse coding can be either,

$$\min_x \|x\|_0, \quad \text{subject to} \quad y = Dx \tag{2.1}$$

or

$$\min_x \|x\|_0, \quad \text{subject to} \quad \|y - Dx\|_2 \leq \epsilon \tag{2.2}$$

Where $\|.\|_0$ is the ℓ^0 norm, which is counting the non-zero entries of a vector.

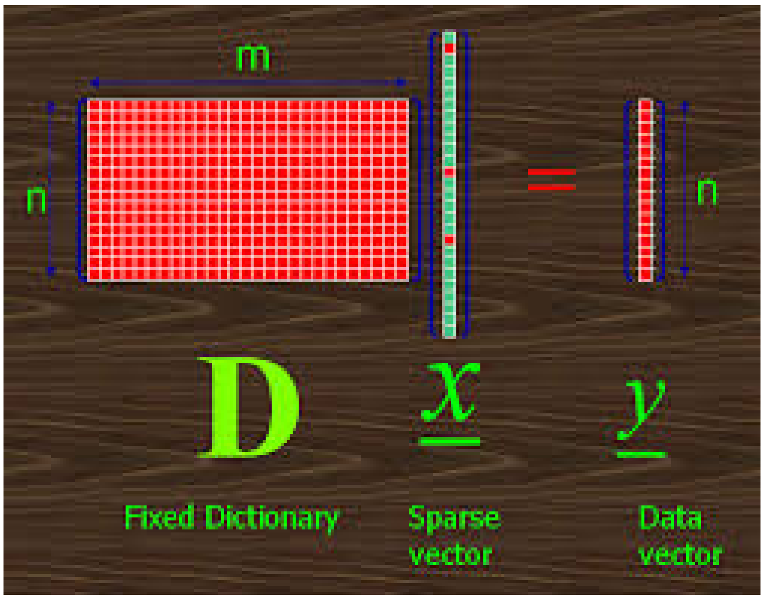


Figure 2.1: The sparse representation of a data vector.

As exact determination of sparsest representation proves to be a NP-hard problem, Sparse coding is generally done by a pursuit algorithm that finds an approximate solution. The simplest pursuit algorithms are the Matching Pursuit (MP), Orthogonal Matching Pursuit (OMP), and Order Recursive Matching Pursuit (ORMP). These are greedy algorithms and select the dictionary atoms sequentially. They are simple and involve the computations like inner products between the signal and the dictionary atoms.

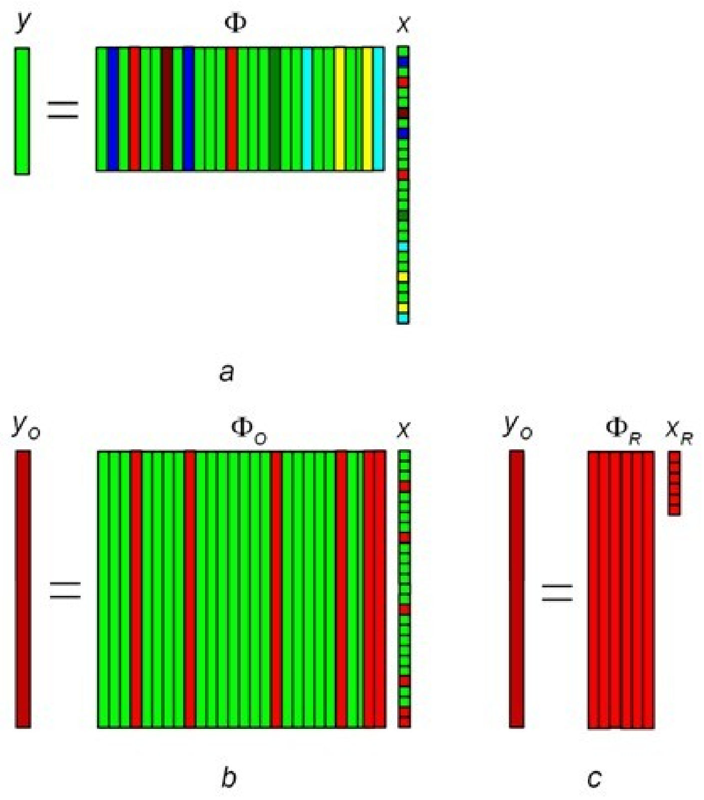
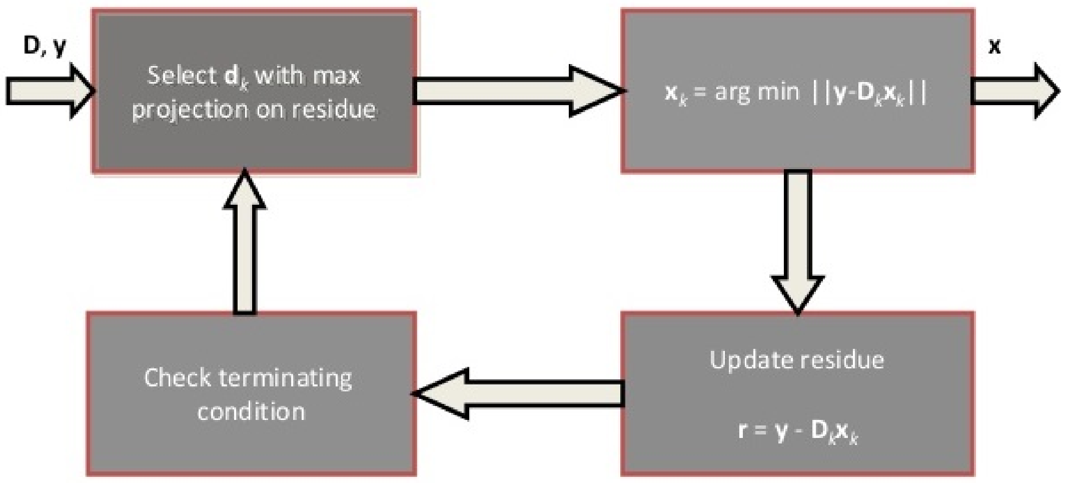


Figure 2.2: Detailed description of sparse coding

Another method to solve sparse coding is Basis Pursuit (BP). It exploits the property of convexification of the problems in 2.1 and 2.2, by replacing the ℓ^0-norm with an ℓ^1-norm. The Focal Under-determined System Solver (FOCUSS) is very similar, except it uses the ℓ^p-norm with $p \leq 1$.

2.2 Orthogonal Matching Pursuit

Orthogonal matching pursuit is a greedy step wise regression algorithm. Assuming dictionary atoms normalized, at each step this method selects the dictionary atom having the maximal projection onto the residual vector. This means that the algorithm selects the atom, which gives the maximum information and reduces the error in reconstruction. The following figure depicts the OMP algorithm pictorially.



D, y
Select $\mathbf{d}_k$ with max projection on residue
$\mathbf{x}_k$ = arg min ||$\mathbf{y}$-$\mathbf{D}_k\mathbf{x}_k$||
x
Update residue
r = y - $\mathbf{D}_k\mathbf{x}_k$
Check terminating condition

Figure 2.3: Orthogonal Matching Pursuit Algorithm.

Given a signal $y \in \mathbb{R}^n$, and a dictionary D with K ℓ^2-normalized columns $\{d_k\}_{k=1}^{K}$, start the method by setting $r_0 = y, k = 1$, and follow the steps.

(1) Select the index of the next dictionary element $i_k = arg\max_w |\langle r_{k-1}, d_w\rangle|$;

(2) Update the current approximation $y_k = arg\min_{y_k} \|y - y_k\|_2^2$, such that $y_k \in \mathrm{span}\{\mathbf{d}_{i1}, \mathbf{d}_{i2}, \ldots, \mathbf{d}_{ik}\}$;

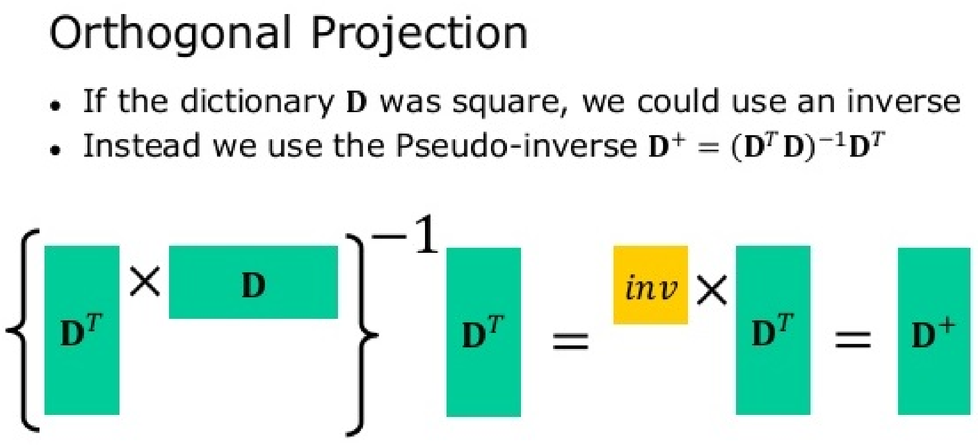
(3) Update the residual $r_k = y - y_k$

In the second step of the algorithm, orthogonal projection is used.

Algorithm 1 Orthogonal Matching Pursuit

```
function [$x$] = OMP($y, D, T, \epsilon$)
$y \Leftarrow$ data sample of size $n \times 1$
$D \Leftarrow$ dictionary matrix of size $n \times K$
$T \Leftarrow$ sparsity threshold
$\epsilon \Leftarrow$ error tolerance
$r_0 = y$
for $k < T$ do
    $p \Leftarrow D^T r_{k-1}$
    $l_k \Leftarrow$ add to list index where column $|p|_i$ is maximum
    $D_k \Leftarrow$ atoms from $D$ which have entries in $l_k$
    $x_k \Leftarrow D_k^\dagger y$
    $r_k \Leftarrow y - D_k x_k$

    if $\|r_k\|_2^2 < \epsilon$ then
        break;
    end if
end for
Return $x \Leftarrow x_k$
```



Orthogonal Projection

- If the dictionary $\mathbf{D}$ was square, we could use an inverse
- Instead we use the Pseudo-inverse $\mathbf{D}^+ = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T$

$$\left\{ \mathbf{D}^T \times \mathbf{D} \right\}^{-1} \mathbf{D}^T = inv \times \mathbf{D}^T = \mathbf{D}^+$$

Figure 2.4: Orthogonal Projection.

The algorithm can be stopped after a predetermined number of steps, hence after having selected a fixed number of atoms. OMP can easily be programmed to get a representation with an a priori fixed number of non-zero entries which is used in the training of dictionaries.

2.3 Basis Pursuit

The BP algorithm replaces ℓ^0-norm in 2.1 with ℓ^1-norm. Hence the problem formulation in BP becomes

$$\min_x \|x\|_1 \quad \text{subject to} \;\; y = Dx \tag{2.3}$$

in the exact representation case and

$$\min_x \|x\|_1 \quad \text{subject to} \;\; \|y - Dx\| \leq \epsilon \tag{2.4}$$

in the approximate case. Solution of the above formulation gives rise to linear programming and efficient solvers for such problems exist. Recent work on iterated shrinkage algorithms provide highly efficient and methods for minimizing approximate case resembles the OMP.

CHAPTER 3

DICTIONARY DESIGN METHODS

In this chapter we see some of the methods for dictionary design. All of the methods proposed follows the same outline of two stages namely, the sparse coding stage and the dictionary update stage. The difference between the methods is in the method used for the calculation of sparse coefficients and in the procedure used for updating the dictionary matrix.

3.1 The MOD Method

This method closely follows 3.3 outline, with a sparse coding stage that uses either OMP or $FOCUSS$ followed by an update of the dictionary. The MOD involves simple way of updating the dictionary. Assuming that the sparse coding for each example is known, we define the errors $e_i = y_i - Dx_i$. The overall representation mean square error is given by

$$||E||_F^2 = ||[e_1, e_2, \ldots, e_N]||_F^2 = ||Y - DX||_F^2 \tag{3.1}$$

Where matrix Y is concatenation of all examples y_i and similarly the representation matrix X. The notation $||A||_F$ stands for the Frobenius norm, defined as $||A||_F = \sqrt{\sum_{ij} A_{ij}^2}$.

Assuming X is fixed, we find an update to D such that the error is minimized. Taking the derivative w.r.t. D, we obtain,

$$D^{n+1} = YX^{(n)^T}(X^{(n)}X^{(n)^T})^{-1} \tag{3.2}$$

The method is simple to implement but it involves computing a complex matrix inversion.

3.2 Unions of Orthonormal Bases

This method takes a dictionary composed as a union of orthonormal bases

$$D = [D_1, D_2, \dots, D_L]$$

where $D_i \in I\mathbb{R}^{n \times n},\ \ i = 1, 2, \dots, L$ are orthonormal matrices.

The coefficients of the sparse representations concatenated in the matrix X can be decomposed into L parts, each referring to a different orthonormal basis. Thus

$$X = [X_1, X_2, \dots, X_L]^T$$

where X_j is the matrix containing the coefficients of the orthonormal basis D_i.

The advantage of this algorithm is the relative simplicity of the pursuit agorithm needed for the sparse coding stage. The coefficients are found using the block coordinate relaxation algorithm where the optimization problem is solved as a sequence of simple shrinkage steps, such that at each stage X_i is computed while keeping all other columns of X fixed.

Assuming coefficients are known, the algorithm updates each orthonormal basis D_j sequentially. The update is done by computing the residual matrix

$$E_j = [e_1, e_2, \dots, e_N] = Y - \sum_{i \neq j} D_i X_i \tag{3.3}$$

Then, using the singular value decomposition of the matrix $E_j X_j^T = U \Lambda V^T$, the update of the jth orthonormal basis is done by $D_j = UV^T$. This is obtained by solving a constrained least squares problem with $\|E_j - D_j X_j\|_F^2$ as the cost function, assuming fixed coefficients X_j and error E_j. This way the proposed algorithm updates each matrix D_j separately, by considering the data matrix Y in the residual matrix E_j.

3.3 K-means Process

A codebook that contains K representatives is used to represent a family of signals $Y = \{y_i\}_{i=1}^{N}$ and $N \gg K$ by nearest neighbor assignment. This gives efficient compression of those signals as clusters in $\mathbb{R}^n$ surrounding the chosen codewords.

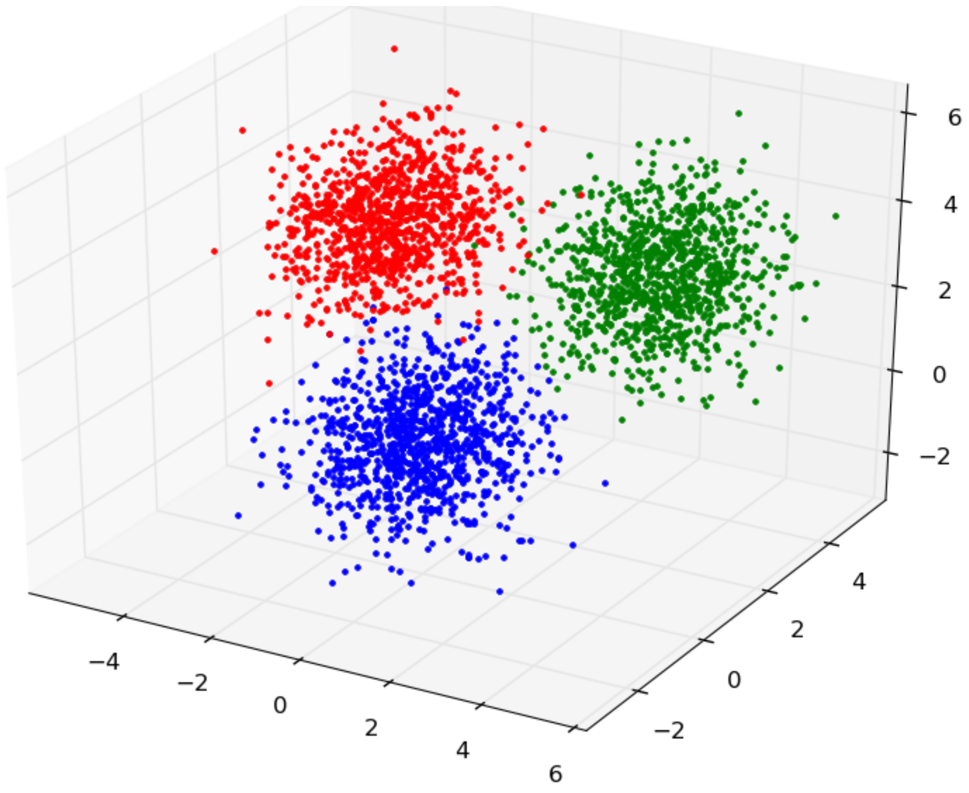


Figure 3.1: An example of K-means to three classes

We denote the codebook by $C = [c_1, c_2, c_3, \ldots, c_K]$, the codewords being the columns. When C is given, each signal is represented as the closest codeword (under ℓ^2-norm distance). We have $y_i = Cx_i$, where $x_i = e_j$ is a vector from the trivial basis, with all zeros except a one at the jth position.

Choose j such that

$$\forall_{k \neq j} \|y_i - Ce_j\|_2^2 \leq \|y_i - Ce_k\|_2^2$$

This is an extreme case of sparse coding where only one atom is allowed to construct signal y_i, and the coefficient is forced to be 1. The representation MSE per y_i is defined as $e_i^2 = \|y_i - Cx_i\|_2^2$, and the overall MSE is

$$E = \sum_{i=1}^{K} e_i^2 = \|Y - CX\|_F^2 \tag{3.4}$$

The VQ training problem is to find a codebook C that minimizes the error E, subject to the sparse structure of X, whose columns must be taken from the trivial basis, mathematically,

$$\min_{C,X}\{\|Y - CX\|_F^2\} \text{ subject to } \forall i,\ x_i = e_k$$

for some k (3.5)

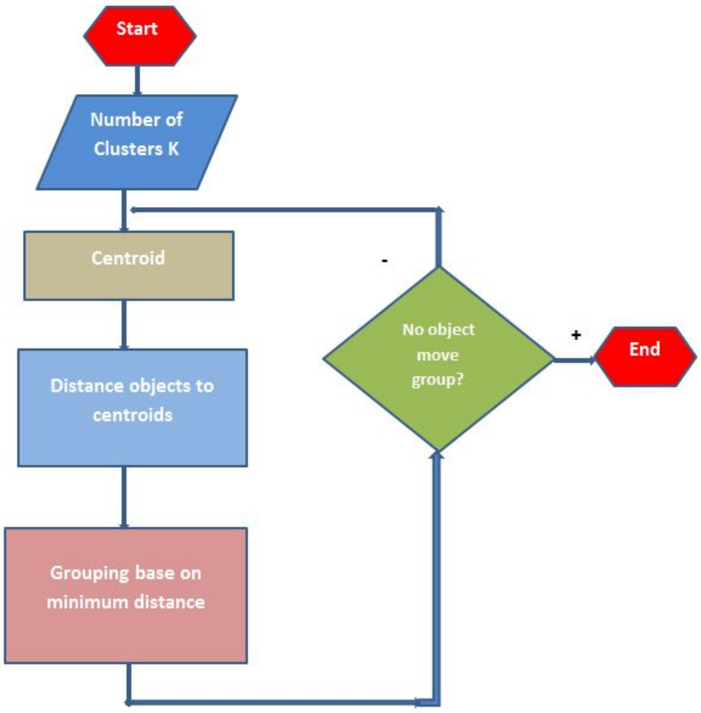

Start
Number of Clusters K
Centroid
Distance objects to centroids
Grouping base on minimum distance
No object move group?
-
+
End

Figure 3.2: Flow chart of K-means process

The K-means algorithm is an iterative method and it is used for designing the optimal codebook. There are two stages in each iteration: one for sparse coding that evaluates X and one for updating the codebook C.

3.3.1 Pseudo-code

Algorithm 2 K-Means Process

```
Task: Find the best possible codebook to represent the data samples $\{y_i\}_{i=1}^N$ by nearest neighbor, by solving
    $\min_{C,X}\{\|Y - CX\|_F^2\} \quad subject\ to\ \forall i,\ x_i = e_k$ for some $k$.

Initialization: Set the codebook matrix $C^{(0)} \in \mathbb{R}^{n\times K}$, Set $J$ =1.

Repeat until convergence(use stop rule):

. Sparse Coding Stage: Partition the training samples $Y$ into $K$ sets
    $(R_1^{(J-1)}, R_2^{(J-2)}, \ldots, R_K^{(J-1)})$,
  each holding the sample indices most similar to the column $c_k^{(J-1)}$,
    $R_k^{(J-1)} = \{i|\forall l \neq k,\ \|y_i - c_k^{(J-1)}\|_2 < \|y_i - c_l^{(J-1)}\|_2\}$
. Codebook Update Stage: For each column $k$ in $C^{(J-1)}$, update it by
    $c_k^{(J)} = \frac{1}{|R_k|}\sum_{i\in R_k^{(J-1)}} y_i.$

. Set $J = J$+1.
```

The sparse coding stage assumes a known codebook and evaluates X that minimizes the error function E. Similarly, the dictionary update stage fixes X and finds an update to codebook C so as to minimize the error function E. Clearly, at each iteration, either a reduction or no change in the MSE is ensured.

3.4 Singular Value Decomposition

The SVD is a factorization of a real(complex) matrix. The SVD of an $m \times n$ matrix A is a factorization of the form

$$A = U \sum V^*$$

where, U is an $m \times m$ real(complex) unitary matrix,

$\sum$ is an $m \times n$ rectangular diagonal matrix,

V^* is an $n \times n$ real(complex) unitary matrix

and A^* is complex conjugate transpose of A.

The diagonal entries $\sum_{i,i}$ are known as the singular values of A.

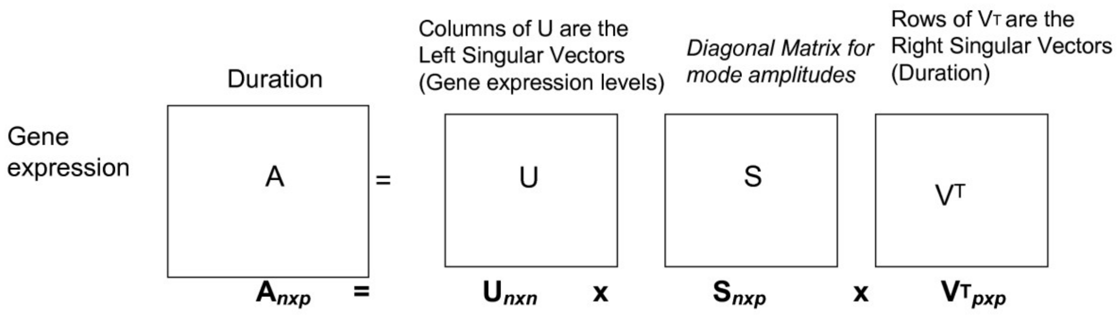

Duration
Gene expression
A
=
Columns of U are the Left Singular Vectors (Gene expression levels)
U
Diagonal Matrix for mode amplitudes
S
Rows of VT are the Right Singular Vectors (Duration)
VT
Anxp = Unxn x Snxp x VTpxp

Figure 3.3: Singular Value Decomposition

Applications that uses SVD are computing the pseudo inverse, least squares fitting of data, determing the rank, range and null-space of a matrix and Low-rank matrix approximation.

CHAPTER 4

K-SVD ALGORITHM

In this chapter, we introduce the K-SVD algorithm for training of dictionaries. The algorithm is flexible and works with any pursuit algorithm. It is simple and direct generalization of the K-means process. The algorithm is efficient due to an effective sparse coding and a Gauss-Seidel-like accelerated dictionary update stage. The algorithm's steps are coherent with each other, both working towards the minimization of the overall objective function.

4.1 K-SVD–Generalization of the K-means

In sparse representation problem each input signal is represented by a linear combination of codewords, which are also called dictionary atoms. The coefficient vector is allowed more than one non-zero entry, and it can have arbitrary values. Now the problem can be described as searching the best dictionary for the sparse representation of the example set Y

$$\min_{D,X}\{\|Y-DX\|_F^2\} \quad \text{subject to } \forall i,\ \|x_i\|_0 \le T_0 \tag{4.1}$$

A objective function can be modified as

$$\min_{D,X}\{\sum_i \|x_i\|_0\} \quad \text{subject to } \|Y-DX\|_F^2 \le \epsilon \tag{4.2}$$

for a fixed value of ϵ. In this thesis, we deal with the first formulation.

We minimize 4.1 iteratively. Fixing D, we will find the best coefficient matrix X. As finding the optimal X is impossible, we find an approximate solution by using any pursuit method. Any pursuit method can be used for finding X with a fixed and predetermined number of non-zero entries T_0.

After the sparse coding stage, in second step we search for a better dictionary. This stage updates one column at a time, fixing all columns in D except one, d_k, and finds a new column $\hat{d_k}$ and calculates new coefficients to best reduce the value of MSE. Whereas in other methods, we find a better D with fixed X. Here, as we change the columns of D sequentially, we allow change in the corresponding coefficients.

The process of updating one column of D at a time is solving a problem having solution based on the singular value decomposition (SVD). Since, we are changing the coefficients while updating the dictionary columns, the method accelerates convergence, because the subsequent column updates will be based on more recent coefficient values.

4.2 K-SVD Description

Our objective function is

$$\min_{D,X}\{\|Y - DX\|_F^2\} \text{ subject to } \forall i,\ \ \|x_i\|_0 \leq T_0 \tag{4.3}$$

First, Let's consider the sparse coding stage, where we assume that the dictionary D is fixed. we will take above optimization problem and solve for sparse representation coefficients. The cost function can be rewritten as

$$\|Y - DX\|_F^2 = \sum_{i=1}^{N} \|y_i - Dx_i\|_2^2 \tag{4.4}$$

Therefore the optimization problem can now be decomposed as N distinct problems of the form

$$\min_{x_i}\{\|y_i - Dx_i\|_2^2\} \quad \text{subject to } \|x_i\|_0 \leq T_0, \quad \text{for } i = 1, 2, \ldots, N. \tag{4.5}$$

This problem can be solved using the pursuit algorithms discussed above. If T_0 is small enough, the solution is a good approximation to the ideal solution which is numerically impossible to compute.

The second stage is the process of updating the dictionary along with the non-zero coefficients. Assuming both X and D are fixed, we solve for one column in the dictionary d_k and the sparse coefficients that correspond to it, the kth row in matrix X, which we denote as x_T^k. Now, the objective function 4.1 can be written as

$$\|Y-DX\|_F^2 = \|Y-\sum_{i=1}^{K} d_i x_T^i\|_F^2 = \|(Y-\sum_{i\neq k} d_i x_T^i)-d_k x_T^k\|_F^2 = \|E_k-d_k x_T^k\|_F^2 \tag{4.6}$$

where, the matrix E_k is equal to the error for all the N examples when the kth atom is removed. We have decomposed the multiplication term DX into the sum of K rank-1 matrices. Among them, K-1 terms are assumed fixed, except the one term at the kth position remain in question.

At this stage, we can apply the SVD algorithm to find new d_k and x_T^k. The SVD finds the closest rank-1 matrix that approximates E_k, which will eventually minimize the error function in 4.6. However, in such a solution x_T^k is very likely to be filled, as we are not enforcing the sparsity constraint while updating the dictionary column d_k.

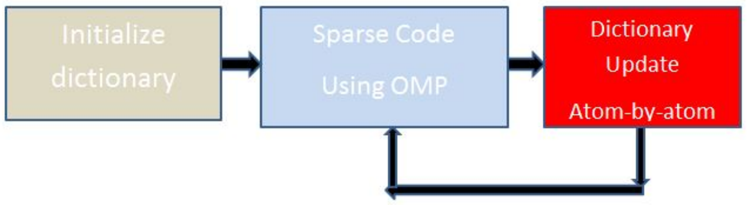


Figure 4.1: Flow chart of k-svd algorithm

The solution to the above problem is simple and intuitive. We define ω_k as the group of indices representing examples y_i that use the atom d_k, i.e., those indices where $x_T^k(i)$ is not equal to zero. Thus

$$\omega_k = \{i|1 \leq i \leq K,\ x_T^k(i) \neq 0\} \tag{4.7}$$

Algorithm 3 K-SVD Algorithm

```
Task: Find the best dictionary to represent the data samples $\{y_i\}_{i=1}^N$ as sparse compositions, by solving
        $\min_{D,X}\{\|Y - DX\|_F^2\}$  $subject\ \ to\ \forall i,\ \|x_i\|_0 \leq T_0$.

Initialization: Set the dictionary matrix $D^{(0)} \in \mathbb{R}^{n \times K}$ with $l^2$ normalized 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 example $y_i$, by approximating the solution of
        $i = 1, 2, \ldots, N,\ \ \min_{x_i}\{\|y_i - Dx_i\|_2^2\}$  $subject\ \ to\ \ \|x_i\|_0 \leq T_0$

. Codebook Update Stage: For each column $k = 1, 2, \ldots, K$ in $D^{(J-1)}$, update it by

- Define the group of examples that use this atom, $\omega_k = \{i|1 \leq i \leq N,\ x_T^k(i) \neq 0\}$.

- Compute the overall representation error matrix $E_k$, using

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

- Restrict $E_k$ by choosing only the columns corresponding to $\omega_k$, and calculate $E_k^R$.

- Apply SVD decomposition $E_k^R = U\Delta V^T$. Choose the updated dictionary column $\hat{d_k}$ to be the first column of $U$. Update the coefficient vector $x_R^k$ to be the first column of $V$ multiplied by $\Delta(1,1)$.

. Set $J = J$+1.
```

Define Ω_k as a matrix with ones at $(\omega_k(i), i)th$ positions and zeros elsewhere. This matrix will be of size $N \times |\omega_k|$. We multiply x_T^k with Ω_k, i.e., $x_R^k = x_T^k \Omega_k$. x_R^k is a row vector of length ω_k and it's a result of discarding all the zero entries in x_T^k. Similarly, after multiplying Y and Ω_k, $Y_k^R = Y\Omega_k$ results in a matrix of size $n \times |\omega_k|$ that includes a subset of examples that are using the d_k atom. The same is with $E_k^R = E_k\Omega_k$, means a selection of error columns that correspond to examples that use the column d_k.

Now we will solve the problem 4.6 and achieve minimization with respect to both d_k and x_T^k, and force the solution of $\hat{x_T^k}$ to have the same support as the original x_T^k. This problem is equivalent to solving the minimization of

$$\|E_k\Omega_k - d_k x_T^k \Omega_k\|_F^2 = \|E_k^R - d_k x_R^k\|_F^2 \tag{4.8}$$

and now we can use SVD and achieve sparsity by considering the restricted matrix E_k^R, SVD decomposes it to $E_k^R = U\Delta V^T$.

We find the solution for the atom $\hat{d_k}$ as the first column of U, and the coefficient vector x_R^k as the first column of V multiplied by $\Delta(1,1)$. In this solution we 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.
The K-means process calculates K computations of means to update the codebook, K-SVD uses K SVD computations to update the dictionary, each column at a time. A full description of algorithm is presented in 3.

In the K-SVD algorithm, we go through the columns and always use the recently updated coefficients as they are calculated from the preceding SVD steps. Parallel versions of this algorithm can also be considered where all the updates of the previous dictionary are done using the same coefficient matrix X. Experiments show that while this version of algorithm also converges, it gives an inferior solution and typically requires more than four times the number of computations.

4.2.1 Convergence of K-SVD

The important question is will the K-SVD algorithm converge? Assuming that sparse coding stage is perfectly done, resulting in the best approximation to the signal y_i that contains maximum of T_0 non-zero coefficients. In this case, and assuming a fixed dictionary, each sparse coding step tries to minimize the total error $||Y - DX||_F^2$. Also, in the each update step for d_k, MSE is either decreased or stays constant, while not violating the sparsity constraint. After performing a number of such steps the algorithm ensures a monotonic MSE reduction, and therefore, the algorithm converges to a local minimum.

4.3 K-Means from K-SVD

When the value of T_0 is 1, the problem boils down to the gain shape VQ and the K-SVD algorithm becomes a method for its codebook training. When T_0 =1, the coefficient matrix X has only one non-zero entry per column. Thus, computing the error E_k^R in 4.8 gives

$$E_k^R = E_k\Omega_k = (Y - \sum_{j\neq k} d_j x_T^j)\Omega_k = Y\Omega_k = Y_k^R \tag{4.9}$$

This is because Ω_k takes only those columns in E_k that use the d_k atom implying that $\forall j,\ \ x_T^j\Omega_k = 0$.

So, in this particular case where $T_0 = 1$, the SVD is being calculated on the group of examples in ω_k. Also, the K updates of the columns of D become independent of each other, implying that a sequential process or a parallel process, both lead to same algorithm. We could further constrain our sparse coding stage and limit the non-zero entries of X to be 1, which gives us the classical clustering problem. In this case, we have x_R^k is filled with ones, i.e., $x_R^k = 1^T$. Then the K-SVD gives the error matrix as $E_k^R = Y_k^R$ by a rank-1 matrix $d_k.1^T$. The solution is the mean of the columns of Y_k^R, which is exactly what the K-means does in clustering problem.

CHAPTER 5

APPLICATIONS TO IMAGE PROCESSING

The proposed algorithm is applied to natural image data, in order to understand the practicality of the method and sparse coding.

5.1 Filling in Missing Pixels

The popular open database known as yale database is considered for training. The database contains gray scale images of 38 persons with each 65 face images taken under different illumination conditions. The images are of size 192 x 168. We have taken 64 images of one person for training and set aside one image for testing. We down sample the training set by four to get 256 images of size 48 x 42 for training. We take patches of size 8 x 7 each time. The dictionary was initialized with data signals. We update the dictionary for this training set of examples and OMP was used for sparse coding stage because of its simplicity. The number of non-zero coefficients(T_0) in sparse coefficient matrix is set to 10 and number of iterations for K-SVD algorithm were set to 80.

$$Y - 56 \times 256 \tag{5.1}$$

$$D - 56 \times 200 \tag{5.2}$$

$$X - 200 \times 256 \tag{5.3}$$

We take the test image and for each block the following steps are executed for r in the range of $\{0.2, 0.9\}$

(i) We delete a fraction r of the pixels in each block.

(ii) The coefficients of the corrupted block are constructed using the K-SVD built dictionary using OMP. The projections in OMP includes only the non corrupted pixels, and for this the dictionary columns are normalized so that the non corrupted indexes in each element have unit norm. The resulting coefficient vector for the block B is denoted as x_B

(iii) The reconstructed block is chosen as $\hat{B} = Dx_B$

5.1.1 Results

The psnr values in db for different percentage of missing pixels(randomly deleted) for a fixed dictionary atoms is shown in Table 5.1.

Table 5.1: Variation of psnr values for different missing pixels percentage.

Image	23.2	35.7	53.6	76.7
Corrupted	13.0091	11.3874	9.8739	8.7464
Dictionary	16.6156	17.5919	13.7363	14.9235
Corrupted-median	18.5215	15.0893	10.5837	8.0929
Dictionary-median	16.84	18.3762	13.8923	15.5152
Corrupted-Gaussian	17.3678	15.2354	10.9249	10.8398
Dictionary-Gaussian	16.9933	18.5	14.1658	15.5934

The psnr values in db for different different dictionary atoms for a fixed percentage of missing pixels is shown in Table 5.2.

Table 5.2: Variation of psnr values for different dictionary atoms.

Image	100	150	200	250
Corrupted	9.0655	10.355	11.3874	10.5059
Dictionary	14.7245	15.0456	17.5919	14.792
Corrupted-median	12.8427	14.0294	15.0893	13.7508
Dictionary-median	14.8619	15.8656	18.3762	14.9926
Corrupted-Gaussian	12.8402	14.1095	13.3	14.1918
Dictionary-Gaussian	14.9995	16.0453	18.5	15.1971

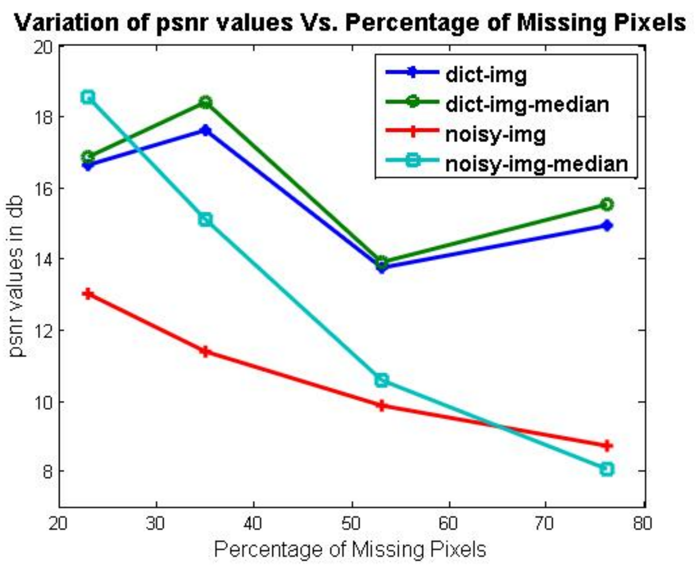


Figure 5.1: Graph showing variation of psnr values with percentage of missing pixels

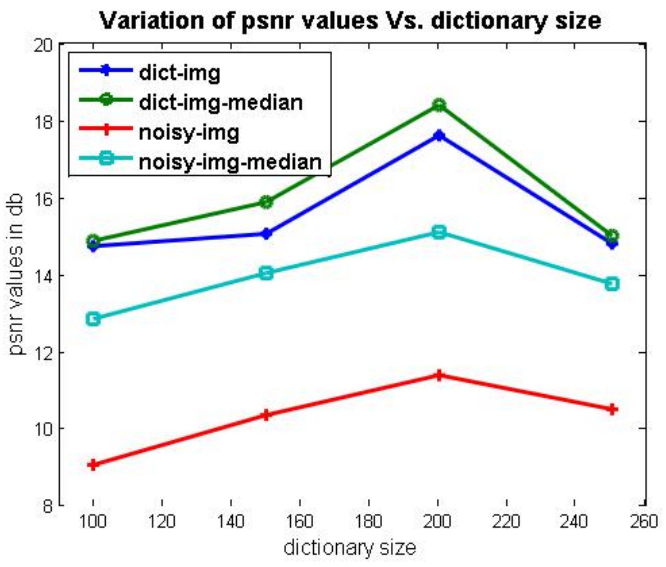
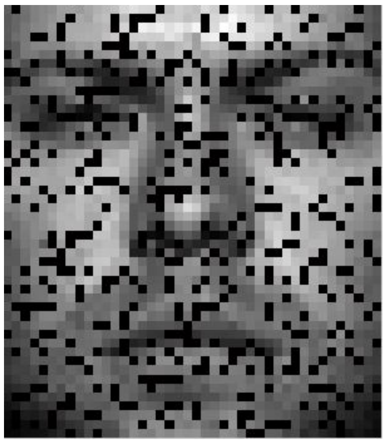
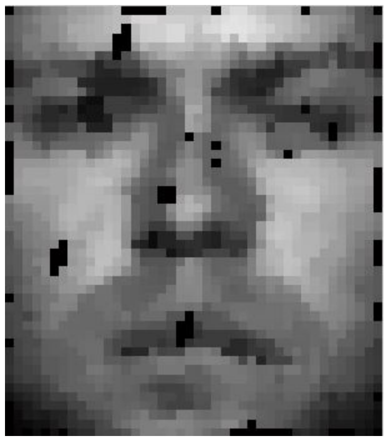
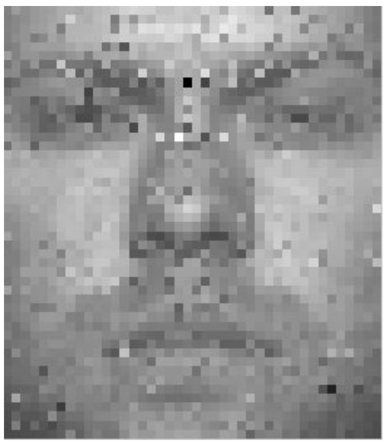


Figure 5.2: Graph showing variation of psnr values with varying dictionary atoms

The following results has been observed when 13 pixels in each block are deleted resulting in 23.2 percent missing pixels in the test image.


(a) corrupted image


(b) corrupted image after applying median filter


(a) dictionary reconstructed image

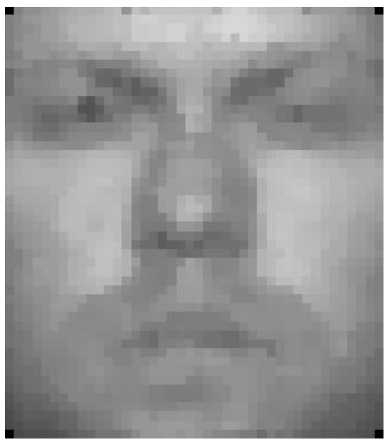
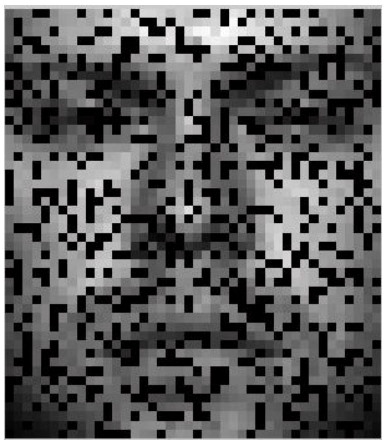
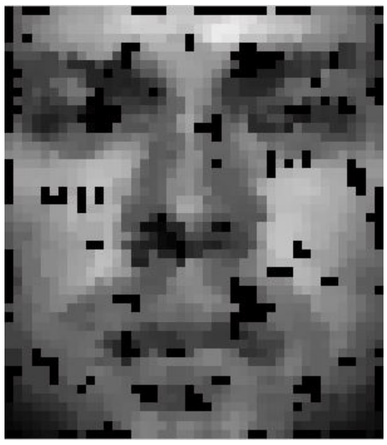
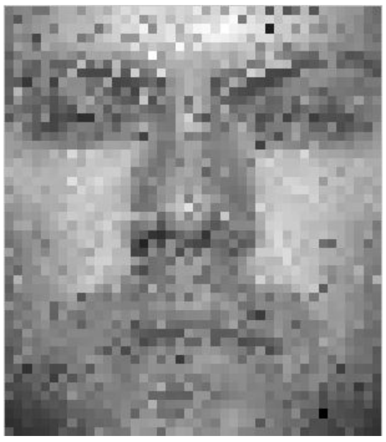

(b) dictionary reconstructed image after applying median filter

Figure 5.4: The figure shows reconstruction for 23 percent missing pixels

The following results has been observed when 20 pixels in each block are deleted resulting in 35.7 percent missing pixels in the test image.


(a) corrupted image


(b) corrupted image after applying median filter


(a) dictionary reconstructed image

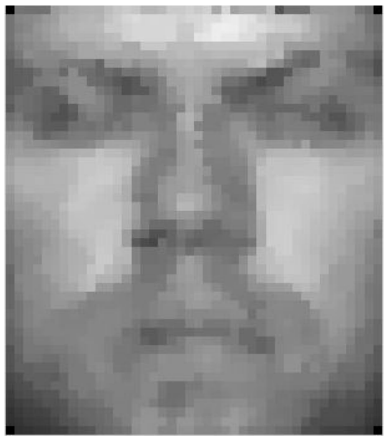
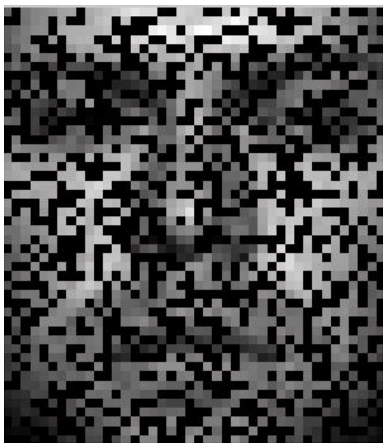
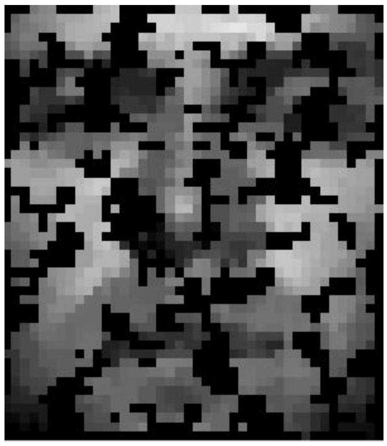
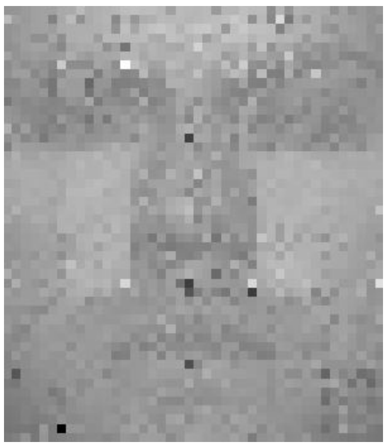

(b) dictionary reconstructed image after applying median filter

Figure 5.6: The figure shows reconstruction for 35 percent of missing pixels

The following results has been observed when 30 pixels in each block are deleted resulting in 53.5 percent missing pixels in the test image.


(a) corrupted image


(b) corrupted image after applying median filter


(a) dictionary reconstructed image

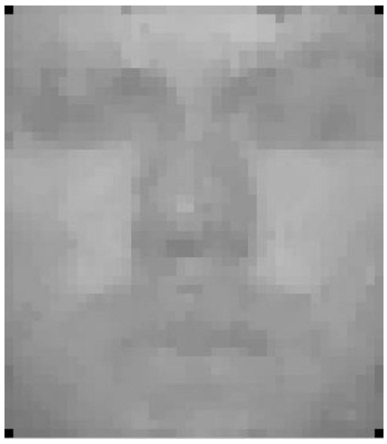
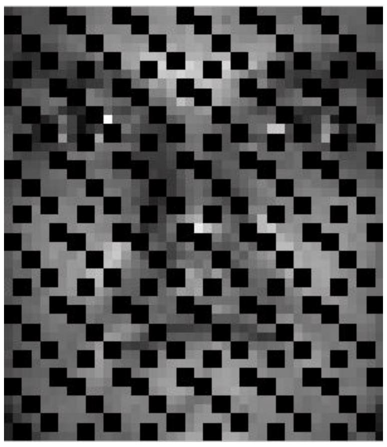
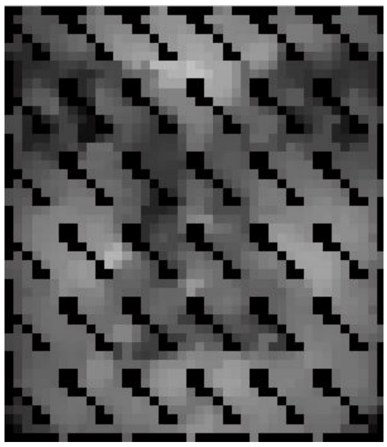
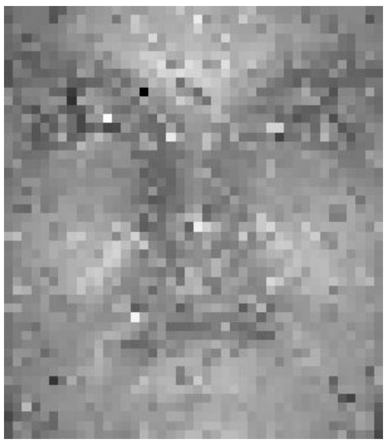

(b) dictionary reconstructed image after applying median filter

Figure 5.8: The figure shows reconstruction for 53 percent of missing pixels

The following are results for patch removal of images.


(a) corrupted image


(b) corrupted image after applying median filter


(a) dictionary reconstructed image

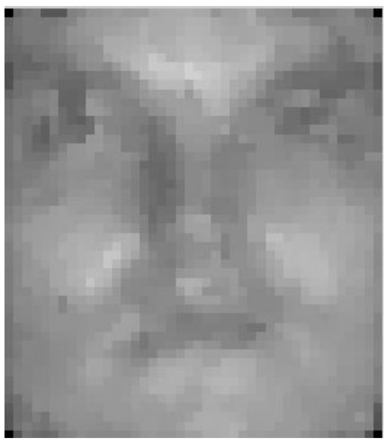
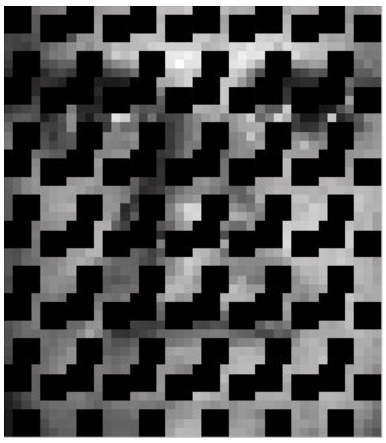
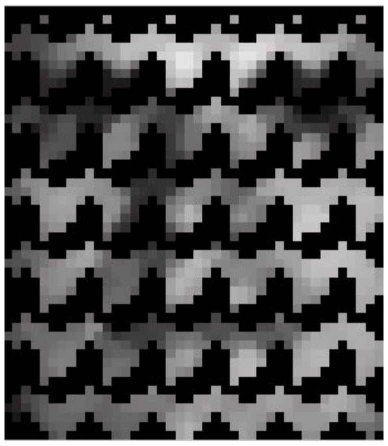
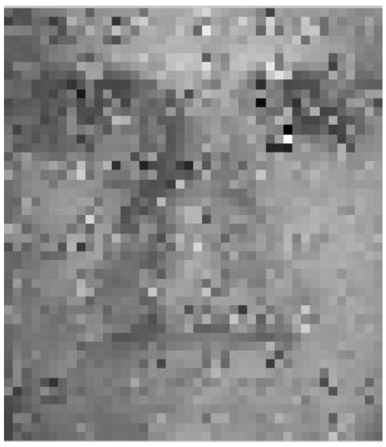

(b) dictionary reconstructed image after applying median filter

Figure 5.10: The figure shows reconstruction for 2x2 patch removal of an image


(a) corrupted image


(b) corrupted image after applying median filter


(a) dictionary reconstructed image

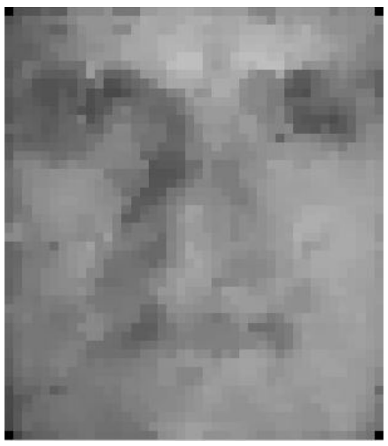

(b) dictionary reconstructed image after applying median filter

Figure 5.12: The figure shows reconstruction for 3x3 patch removal of an image

5.2 Combining with SRC

One of the applications of the algorithm is face recognition using SRC algorithm. Both corrupted and reconstructed images are sent through SRC algorithm and the following results have been observed.

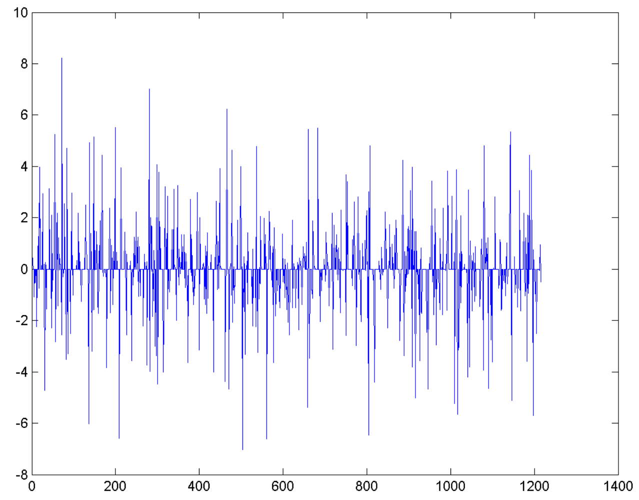


Figure 5.13: Figure shows the sparsity coefficients for a Corrupted image

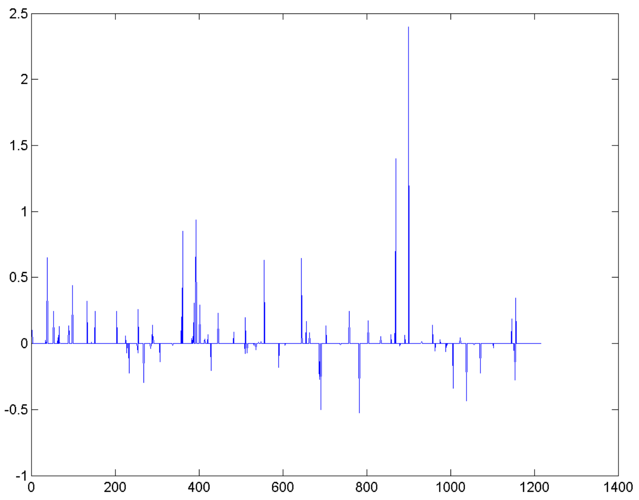


Figure 5.14: Figure shows the sparsity coefficients for a Reconstructed image

The following bar graph shows the results for different types of corrupted images using standard SRC algorithm.

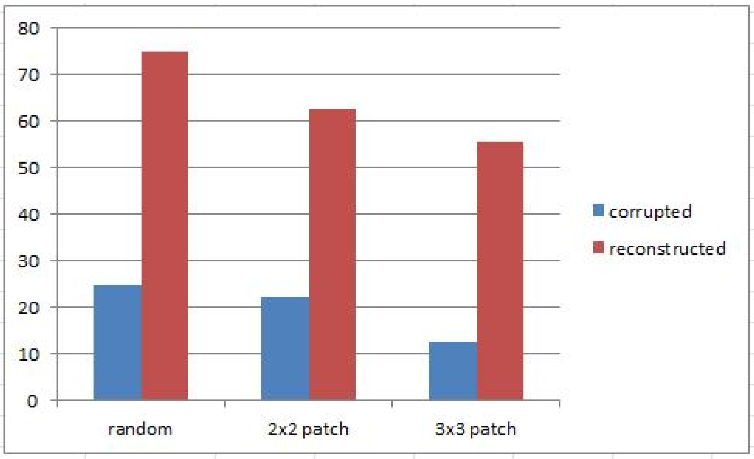


Figure 5.15: Bar graph showing recognition rate for various corrupted images using SRC

It has been inferred that the recognition rate has improved as compared to corrupted images rate. We can see that the recognition rate is higher in case of reconstructed image than in corrupted image case.

CHAPTER 6

CONCLUSION

In this thesis we have extensively studied the sparse representation of signals and dictionary design methods. We have seen methods to solve sparse coding and some algorithms for dictionary design. We have seen a new method K-SVD for the problem extending the K-means algorithm, a popular method used in clustering schemes. We have extended the algorithm for applications in Image processing: Filling in Missing Pixels. We have shown various results to support that the algorithm gives better results. We have applied the algorithm to patch removed images and got some good results. We have sent the reconstructed images through the face recognition algorithm(SRC) and observed that there has been a significant improvement in rate of recognition.

6.1 Future Work

- Exploration of the significance of different pursuit methods when using with K-SVD algorithm
- Multi-scale approaches and tree-based training where the number of columns K is allowed to increase during the algorithm
- Handling the scalability problem of the K-SVD when working with larger image patches.
- A study of the effect of introducing weights to the atoms

CHAPTER 7

REFERENCES

[1] **Michal Aharon, Michael Elad, and Alfred Bruckstein**. "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation". *IEEE Transactions on Signal Processing*, 54(11): 4311-4322, November 2006.

[2] **Ron Rubinstein, Alfred M. Bruckstein, and Michael Elad** . "Dictionaries for Sparse Representation Modeling". *IEEE Proceedings.*

[3] **Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad**. "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition" in *Conf.Rec. 27th Asilomar Conf.Signals, Syst. Comput.*,1993, vol.1.

[4] **Priyam Chatterjee**. "Denoising using the K-SVD Method". June 2007.

[5] **Ivana Tosic, and Pascal Frossard**. "Dictionary Learning". *IEEE Signal Processing Magazine*, February 2011.

[6] **Mostafa Sadeghi, Massoud Babaie-Zadeh, and Christian Jutten**. "Dictionary Learning for Sparse Representation: A Novel Approach". *IEEE Signal Processing Letters*, 20(12):1195-1198, December 2013.