1-1-2015

Effective Auto Encoder For Unsupervised Sparse Representation

Faria Mahnaz
Wayne State University,

EFFECTIVE AUTO ENCODER FOR UNSUPERVISED SPARSE REPRESENTATION

by

FARIA MAHNAZ

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2015

MAJOR: COMPUTER SCIENCE

Approved By:

Advisor Date

DEDICATION

To my family for their endless support and encouragement.

ACKNOWLEDGEMENTS

This MS thesis work is a story of several people working together towards a common goal. I would like to show my gratitude to my MS advisor Dr. Xue-wen Chen, who is the mastermind behind all of the work. His advice surely helped me along the way. I am also very thankful to my fellow colleagues in our Big Data and Business Analytics Group who helped me so much through my whole MS life. Without their support and help, I could not have accomplished this project. I have to specially mention Dr. Melih S. Aslan, who guided me as a mentor and spent a lot of time answering my silly questions. I have to mention the name of my friend Sumukhi Chandrashekar, Itauma Itauma and Robert Taniru here, as they always were with me and provided mental support. Also I would like to acknowledge the support of all my family and friends. Without the love and encouragement of my parents, this accomplishment could have never been possible. Last but not least, I would like to dedicate this effort to my husband, without whom this would not have been a reality. I will always cherish the conversations we held about this thesis work and its possibilities beyond academic boundaries.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The rise of large-volume unlabeled data with high dimensionality presents unprecedented challenges for machine learning and data mining societies. Relying on expensive labeled data, most of the traditional supervised learning methods are incapable of taking advantage of the enormous amount of unlabeled data in large scale data analytics. In the past, much effort has been made to learn compact representations (features) with unlabeled data. Among many dimensionality reduction methods such as PCA [38], ICA [37], and Factor Analysis [11], sparse representation learning is promising and has shown advantages [26, 6, 24, 39].

Inspired from early stage computational process of biological visual systems [30], sparse representation approximately represents data as a sparse linear combination of fixed basis vectors in a dictionary matrix. The dictionaries are often non-orthogonal, and due to the sparseness more interpretable [27]. Thus, representing the data into a lower dimensional space makes data analysis more efficient. Lee et al. [20] proposed efficient sparse coding algorithm, which produces low level gabor-like feature. Inducing sparsity penalty in traditional unsupervised models such as sparse restricted boltzman machine (RBM) [21], sparse auto-encoder [36] has shown improved results in many classification tasks. Hierarchical model for sparse representation learning was proposed to build high level features [19]. Greedy layer wise pre-training [12, 1] approach in deep learning [5] became very popular for deep hierarchical frameworks. Multi-layer of stacked sparse auto-encoder (SAE) [1, 35, 19], sparse Deep belief net (DBN), convolutional deep belief net (CDBN) [22] are few frameworks for learning sparse representation.

Despite of all these recent developments, current methods are limited when dealing with large scale data. Mostly, complex deep architecture and expensive training time are responsible for lack of good feature representations for large scale data. In most of the cases, researchers typically reduce the sizes of data sets and models in order to train networks in a practical amount of time. However, these reductions undermine the learning of high-level features.

The goal of large scale sparse representation learning is to extract features by learning the dictionary that captures high level structured information in large-scale data. Optimization of sparse representation learning has always been a difficult task due to its non-convex property. Different optimization techniques such as sub-gradient optimization [4], feature-sign [20], proximal method [14] have been applied. However, with the sheer size of data available today, new optimization techniques are urgently needed for unsupervised learning of large-scale sparse representation.

To address these problems, we draw upon the idea from sequential minimal optimization (SMO) [33] which can be considered as a Hessian free optimization. This decomposition method was first introduced for large scale support vector machines (SVM) [33], [15] and later for L_1 least square conventional sparse coding [23]. In this paper, we propose a fast algorithm for training an encoder-decoder based module, which efficiently extracts sparse and compact features from any data set with various size. The focus of this work is thus to provide a fast sparse representation learning algorithm which will work efficiently. Decomposition of the optimization problem in our model enables us to solve each subproblems analytically. We demonstrate experimentally that our algorithm is efficient than the commonly used sparse representation learning algorithm.

Contributions. The contributions of this paper are as follow:

I. A new formulation and solution for an unsupervised learning problem that is capable of extracting heterogeneous features with low computational cost and time;

II. A fast auto sparse encoder (FASE) module for large scale sparse feature representation learning that extracts details features from the larger dimensional patches;

III. Introducing the SMO for the first time for learning of sparse representation from unlabeled image data;

IV. An evaluation of execution time and classification accuracy of the proposed method and the alternative state-of-the-art unsupervised learning methods in two different experimental settings.

CHAPTER 2: BACKGROUND LITERATURE

This chapter provides an outline of the basic principles of sparse representation [30] upon which the remaining chapters are based. The chapter also introduces some of the notation that will be used throughout this thesis.

The work in this thesis tends to proposes a new unsupervised method that is able to learn sparse representation from unlabeled data efficiently. A novel learning algorithm has been derived based on the traditional sparse coding schemes [20]. Therefore this chapter is broken into two sections to describe the theoretical analysis or foundation of learning sparse features and some of the most common classical approaches used in machine learning for unsupervised sparse representation learning.

2.1 Learning Sparse Features

Sparse representation is a principle that a sample can be approximated by a sparse linear combination of basis vectors. To represent input image $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ as a weighted linear combination of a small no. of (unknown) basis vectors $\mathbf{d}_1, \ldots, \mathbf{d}_k$ (each in $\mathbb{R}^m$) also called bases so that each input $\mathbf{X}$ can be approximately decomposed as: $\mathbf{X} \approx \sum_{j=1}^{k} \mathbf{d}_j \mathbf{z}_j$ s.t. $\mathbf{z}_j$'s are mostly sparse vector.

A sparse representation uses more features where at any given time a significant number of the features will have a zero value. Non-orthogonal and redundant basis vectors can be extracted by learning of sparse representation. This principle has been applied for high dimensional data analysis because of the robustness, non-orthogonality and interpretability.

Notation: Matrices are always presented in uppercase bold (e.g., $\mathbf{X}$), vectors are in lower-case bold (e.g., $\mathbf{x}$) and scalars in lower-case (e.g., x) through out the paper. The matrix represented by $\mathbf{I}$ denotes an identity matrix. Any operator ^{T} on a vector or matrix in this paper represents the transpose. We may often omit vector indices (e.g.,

$\mathbf{d}_j$ refers to the j^{th} basis vector and $\mathbf{z}_j$ to the j^{th} sparse code vector) when referring to the variables being optimized. In these instances we assume that $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_k]$ and $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_k]^T$.

2.1.1 Theoretical Analysis

Sparse representation learning includes two different techniques. First one is termed as sparse coding, where a sparse vector representing a data sample is being learned for a fixed dictionary matrix. Second one is termed as dictionary learning, where the dictionary is being learned for given data sample. The formal mathematical formulation or theory of sparse representation can be analyzed from a Bayesian perspective for better understanding. For simplicity in this section, we drop the index j and denote each single day point $\mathbf{x} = \mathbf{x}_j$; $\mathbf{z} = \mathbf{z}_j = [z_1, \ldots, z_k]^T \in \mathbb{R}^k$. The model has the following formulation:

$$(\mathbf{x}|\mathbf{D}, \mathbf{z}, \lambda) = z_1\mathbf{d}_1 + z_2\mathbf{d}_2 \cdots + z_k\mathbf{d}_k + \xi = \mathbf{D}\mathbf{z} + \xi \tag{2.1}$$

where $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_k] \in \mathbb{R}^{m \times k}$ is called dictionary, $\mathbf{z}$ is a sparse code vector, and the error term is defined by ξ. $\mathbf{D}, \mathbf{z}$ and ξ are the model parameters. The model has the following constraints:

1. $\xi \sim \mathcal{N}(0, \Phi)$, where $\Phi = \phi I$ where ϕ is a positive scalar.

2. $\mathbf{d}_j \sim \mathcal{N}(0, \Delta)$ where $\Delta = I$.

3. $\mathbf{z}$ is independent of ξ .

To understand the concept of sparse representation we should analysis the concept of Bayesian sparse representation. Given a new input $\mathbf{x}$ and a dictionary $\mathbf{D}$, by the above definition of sparse coding the statistical formulation is as follows

$$(\mathbf{x}|\mathbf{D}) = \mathbf{D}\mathbf{z} + \xi \tag{2.2}$$

Now let us assume that the code vector has Laplacian prior with zero mean and isotropic variance, that is

$$p(\mathbf{z}, \mathbf{D}) = \mathcal{L}(0, \Gamma) = \frac{1}{(2\gamma)^k} \exp^{\frac{\|\mathbf{x}\|_1}{\gamma}} \tag{2.3}$$

after takin the normal distribution

$$p(\mathbf{x}|\mathbf{D}, \mathbf{z}, \Phi) = \mathcal{N}(\mathbf{D}\mathbf{z}, \Phi) = \frac{1}{(2\pi)^{m/2}\Phi^{m/2}} \exp^{\frac{\|\mathbf{x}-\mathbf{D}\mathbf{z}\|_2^2}{2\Phi}} \tag{2.4}$$

Thus we can have the following posterior

$$p(\mathbf{z}|\mathbf{D}, \mathbf{x}, \Phi, \Gamma) \propto p(\mathbf{x}|\mathbf{D}, \mathbf{z}, \Phi)p(\mathbf{z}|\Gamma) \tag{2.5}$$

by applying the log function we have

$$L(\mathbf{z}) = \log p(\mathbf{x}|\mathbf{D}, \mathbf{z}) + \log p(x) = -\frac{\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2}{2\Phi} - \frac{\|\mathbf{z}\|_1}{\gamma} \tag{2.6}$$

where, c is a constant variable. Now the equivalent minimization problem is as follows:

$$\min_z \frac{1}{2}\|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1, \tag{2.7}$$

where $\lambda = \frac{\Phi}{\gamma}$. It is a general regression model with L_1 regularization also know as LASSO, equivalent to a maximum a posteriori estimation. As instance-based learning, the sparse coding methods without dictionary learning are able to classify complex data, but become very slow as the number of samples increases dramatically.

2.2 Classical Methods

2.2.1 Sparse Coding Representation

Sparse coding has been widely used to extract features for classification [34], [8]. Sparse coding proposed by Lee et al. [20] describes a class of algorithms, where the observed data sample $\mathbf{X} \in \mathbb{R}^{m \times n}$ is represented compactly by using a small number of unknown basis vectors $\mathbf{d}_j \in \mathbb{R}^m : 1 < j < k$ and sparse representation vectors $\mathbf{Z} \in \mathbb{R}^{k \times n}$, i.e., $\mathbf{X} \approx \sum_{j=1}^{k} \mathbf{d}_j \mathbf{z}_j$ where $k >> m$ for overcomplete representation. The objective is to minimize the reconstruction error $\mathbf{X} - \sum_{j=1}^{k} \mathbf{d}_j \mathbf{z}_j$, in order to discover good dictionary $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_k] \in \mathbb{R}^{m \times k}$ to represent input vectors as closely as possible, which can be formulated as a non-convex optimization problem:

$$\arg\min_{\mathbf{D},\mathbf{Z}} \frac{1}{2} \|\mathbf{X} - \mathbf{DZ}\|_F^2 + \beta \|\mathbf{Z}\|_1, \tag{2.8}$$

$$\text{subject to} : \|\mathbf{d}_j\|_2^2 \leq c \text{ for } j = 1, \dots, k,$$

where β is a sparsity penalty and c is a constant. This problem can be solved by two convex optimization problems until convergence. The L_1 regularized least squares problem, also termed as LASSO, is solved with respect to $\mathbf{Z}$, using the feature sign search algorithm. The L_2 constrained least squares problem is solved with respect to dictionary $\mathbf{D}$, using the Lagrange dual algorithm.

For learning sparse representation, the feature-sign search algorithm minimizes the following objective with respect $\mathbf{Z}$, while the dictionary remains fixed,

$$\arg\min_{\mathbf{Z}} \frac{1}{2} \|\mathbf{X} - \sum_{j=1}^{k} \mathbf{d}_j \mathbf{z}_j\|_F^2 + \beta \| \sum_{j=1}^{k} \mathbf{z}_j \|_1. \tag{2.9}$$

The algorithm is based on an active set optimization method and iteratively tries to guess the sign (positive, negative or zero) for nonzero coefficients, z_j's. Each of the terms $|z_j|_1$ can be replaced by either z_j (if positive sign), 0 (if zero) or $-z_j$ (if negative sign). Hence, the problem (Eq.2.8) will be reduced to a solvable standard unconstrained quadratic optimization problem (QP). The algorithm proceeds by repeatedly refining the guess for signs until it converges to an optimal solution. This algorithm is currently one of the state-of-the-art LASSO solvers, and efficiently learns feature representation from small dimensional image data sets. However, for larger image patches convergence becomes significantly slower and takes a long time to find optimal sparse vectors. Another drawback of sparse coding is that, the coefficients are not merely encouraged to be sparse; they are encouraged to remain close to 0, even when they are active.

For dictionary learning, the Lagrange dual algorithm [3] minimizes the following objective by solving the corresponding dual problem analytically with L_2 constraint,

$$\arg\min_{\mathbf{D}} \frac{1}{2}\|\mathbf{X}-\mathbf{DZ}\|_F^2, \tag{2.10}$$

$$\text{subject to} : \|\mathbf{d}_j\|_2^2 \leq c \text{ for } j = 1, \ldots, k.$$

The overall model setup is similar to a decoder module without the encoder part. While learning, two optimization problems are solved alternatingly in each iteration cycle. However, the drawback of this strategy is the computational expense associated with both problems. Also, the inference in sparse coding takes longer time, as there is no encoder module to make initial prediction. The inference of sparse representation becomes slower as the number of features increases. On the other hand, in this work, we implemented an algorithm using the idea from the SMO for solving the LASSO problem, which interestingly improves the inference efficiency, thus overall learning is optimized.

Sparse coding is also difficult to be integrated into a deep generative model of data (e.g. natural images). Recently Yu et al. [40] and Zeiler et al. [42] have shown some success at learning hierarchical sparse coding. However, generative models such as sparse deep belief net (DBN) and convolutional deep belief net (CDBN) [22] with their combination of feed-forward and feed-back connections during inference can learn a much higher quality features, than simple stacked feed-forward models like deep sparse coding models. We, therefore, expect that our fast inference algorithm will yield better performance than a feed-forward architecture by extracting complicated features by training an unsupervised model. Our algorithmic framework and better classification results, effectively make headway on the future challenges of deep learning [5].

2.2.2 Dictionary Learning Models

The Energy Based Model (EBM) [36] was proposed for learning sparse and overcomplete features. The model is basically a dictionary learning based model. An energy-based model has been proposed with a similar architecture of auto encoder. The model includes an encoding part as well as an decoding part. In order to learn sparse representations a non-linear logistic function with an adaptive bias has been placed between encoder and decoder to achieve sparsity. It controls the degree of sparsity of the representation as well as the entropy of each code unit. The model apply spercifying logistic activation function to induce the sparsity instead of L_1 regularization. In Later study Lecun et al. proposed Predictive Sparse Decomposition (PSD) [35] algorithm which is a special case of autoencoder. The model has demonstrated effectiveness in computation in a series of work. They differ in the specifics of the architecture where PSD has nonlinear encoder function and L_1 constraints on the representation. They claimed a fast inference procedure for training the model than one of the state of the art Feature Sign Search algorithm [20]. The disadvantage of both the methods is the optimization technique which is solely based on sub-gradient method [4].

Hierarchical model for sparse dictionary learning was proposed to build high level features [19]. Greedy layer wise pre-training [12, 1] approach in deep learning [5] became very popular for deep hierarchical frameworks. Multi-layer of stacked sparse auto-encoder (SAE) [1, 35, 19], sparse Deep belief net (DBN), convolutional deep belief net (CDBN) [22] are few frameworks for learning sparse representation. In recent years some popular dictionary learning models has been proposed by researchers such as K-means learning framework proposed by Coates et al. [9], hierarchical matching pursuit [2] and large scale dictionary learning model proposed by Zhang et al. [41]. Despite of all these recent developments, current methods are limited when dealing with large scale data. Mostly, complex deep architecture and expensive training time are responsible for lack of good feature representations for large scale data.

Optimization of sparse representation learning has always been a difficult task due to its non-convex property. Different optimization techniques such as sub-gradient optimization [4], feature-sign [20], proximal method [14] have been applied. However, with the sheer size of data available today, new optimization techniques are urgently needed for unsupervised learning of large-scale sparse representation.

CHAPTER 3: SPARSE REPRESENTATION LEARNING

In this chapter a new encoder-decoder based module named fast auto sparse encoder (FASE) has been proposed. The derivation of the efficient learning algorithm for extracting sparse and compact features of high dimensional data, has been discussed in details.

This new optimization technique inspired by Sequential Minimal Optimization (SMO) [33] simultaneously learns a dictionary as well as creates sparse representations. This chapter also includes the empirical results on famous Caltech-101 [16] image database. The effectiveness of FASE algorithm in terms of computation time has been demonstrated.

3.1 Learning Framework

Suppose that, we are given an input sample $\mathbf{x} \in \mathbb{R}^m$ (e.g., image patches) in a high dimensional space. The representation learning algorithm consists of two modules. First one is a feed forward encoder module that maps the input to a code vector or latent representation $\mathbf{z} \in \mathbb{R}^k$. The other one is a decoder module that tends to reconstruct the input sample approximately by a linear combination (i.e., $\mathbf{x} \approx \mathbf{D}\mathbf{z}$) of k basis vectors in a dictionary matrix $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_k] \in \mathbb{R}^{m \times k}$. A non-linear encoding function $f(\mathbf{x}; \mathbf{W})$ has been used to map $\mathbf{x} \rightarrow \mathbf{z}$, where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k]^T \in \mathbb{R}^{k \times m}$ is a latent weight matrix. For sparse representation, L_1 constraint is imposed to induce sparsity to the internal code vector. Our proposed module named fast auto sparse encoder (FASE) is, in spirit, similar to a sparse auto-encoder framework [36]. While learning, for given n data samples in $\mathbb{R}^m$ represented by matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{m \times n}$, we want to learn a dictionary $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_k] \in \mathbb{R}^{m \times k}$, and sparse representation code vectors $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n] \in \mathbb{R}^{k \times n}$, so that each input sample $\mathbf{x}_j$ can be approximated by $\mathbf{D}\mathbf{z}_\mathbf{j}$. Now the optimization problem can be formulaized over $\mathbf{D}$, $\mathbf{Z}$ and $\mathbf{W}$ as below:

$$\arg \min_{\mathbf{D},\mathbf{Z},\mathbf{W}} \frac{1}{2}\|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 + \lambda\|\mathbf{Z}\|_1 + \frac{\alpha}{2}\|\mathbf{Z} - f(\mathbf{X};\mathbf{W})\|_F^2, \tag{3.1}$$

$$\text{subject to} : \|\mathbf{d}_i\|_2^2 \leq 1 \text{ for } i = 1, \ldots, k,$$

where $\lambda > 0$ is a parameter that controls the sparsity of the code vectors (features) and α is a penalty parameter. We consider $\|.\|_F$ and $\|.\|_1$ to represent Frobenius norm and elementwise L_1-norm respectively. In our experiment, we use sigmoid activation function, $f(\mathbf{X};\mathbf{W}) = (1 + exp^{-(\mathbf{W}\mathbf{X})})^{-1}$, and set α equals to 1. One can use different activation functions, such as, hyperbolic tangent function and rectifier linear unit.

We are interested in minimizing the objective given by Eq. 3.1. The first quadratic term measures the discrepancy between observed input and reconstructed input. The third quadratic term $\|\mathbf{Z} - f(\mathbf{X};\mathbf{W})\|_2^2$ penalizes the violation of constraint, $\mathbf{Z} = f(\mathbf{X};\mathbf{W})$; so that the system can predict the internal sparse representation as accurately as possible. Training objective is accomplished in two phases:

- Inferring the sparse code vector by solving minimization problem in Eq. 3.1 to achieve optimal $\mathbf{Z}^*$ for given $\mathbf{X}$ and fixed $\mathbf{W}$, $\mathbf{D}$ (as in sparse coding).

- Adjust both latent weights $\mathbf{W}$ and dictionary $\mathbf{D}$ in order to reconstruct the image sample as close as possible for given $\mathbf{X}$ and fixed $\mathbf{Z}^*$ (as in dictionary learning).

3.2 Sequential Optimization

Like most of sparse representation learning models, the optimization is a non-convex process. Moreover, L_1 penalty term makes the problem of finding optimal code vector $\mathbf{z}$ very difficult. In our learning model we introduce an SMO-based inference at each iteration, which is very fast and efficient for large size data, in comparison to current state-of-the-art methods.

3.2.1 Inferring latent feature

The SMO is generally a decomposition optimization method for quadratic programming problems. This method applies for optimization problem with equality and bound constraints. In each iteration, a few number of variables violating the optimality conditions are included in an working set, while the rest are fixed. In this method only a minimal number of variables (even one or two) are updated by a solver. This procedure iterates until no variable violates the optimality condition. Because the objective function is decreased in each iteration, the convergence to the optimal solution is guaranteed in regular cases. One of the properties of the SMO is that the subproblem with only minimal variables can be solved analytically. We now derive a new SMO solution for accelerated inference of sparse codes or features. For implementation simplicity, we assume that there is only a single data point in Eq. 3.1; it is trivial in our proposed formulation to handle multiple data point in a training set. When $\mathbf{D}$, $\mathbf{W}$ are fixed, finding the optimal $\mathbf{Z}$ requires solving n subproblems. The j^{th} subproblem finds $\mathbf{z}_j$ for $\mathbf{x}_j$. In this section, we drop the index j and denote $\mathbf{x} = \mathbf{x}_j$; $\mathbf{z} = \mathbf{z}_j = [z_1, \ldots, z_k]^T \in \mathbb{R}^k$. The objective function is then as follows:

$$\begin{aligned}\arg\min_{\mathbf{z}_1,\ldots,\mathbf{z}_k} R(\mathbf{z}) &= \frac{1}{2}\|\mathbf{x}-\mathbf{D}\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1 + \frac{1}{2}\|\mathbf{z}-f(\mathbf{x};\mathbf{W})\|_2^2 \\ &= \frac{1}{2}(\mathbf{z}^T\mathbf{H}\mathbf{z}+\mathbf{z}^T\mathbf{z}) + \mathbf{g}_d^T\mathbf{z} + \mathbf{z}^T\mathbf{g}_c + \lambda\|\mathbf{z}\|_1. \end{aligned} \tag{3.2}$$

The objective function in Eq. 3.2 is equivalent to the unconstrained non-smooth QP problem, where Hessian matrix, $\mathbf{H} = \mathbf{D}^T\mathbf{D}$, $\mathbf{g}_c = -\mathrm{sigm}(\mathbf{W}\mathbf{x})$, and $\mathbf{g}_d = -\mathbf{D}^T\mathbf{x}$. Let $\mathcal{A}$ be the set of a few working variables and $\mathcal{P}$ be the set of fixed variables. Let's assume that the vectors and matrices, $\mathbf{z}$, $\mathbf{g}_c$, $\mathbf{g}_d$ and $\mathbf{H}$ are properly arranged with respect to $\mathcal{A}$ and $\mathcal{P}$, so that $\mathbf{z} = \begin{vmatrix}\mathbf{z}_\mathcal{A}\\ \mathbf{z}_\mathcal{P}\end{vmatrix}$, $\mathbf{g}_c = \begin{vmatrix}\mathbf{g}_{c_\mathcal{A}}\\ \mathbf{g}_{c_\mathcal{P}}\end{vmatrix}$, $\mathbf{g}_d = \begin{vmatrix}\mathbf{g}_{d_\mathcal{A}}\\ \mathbf{g}_{d_\mathcal{P}}\end{vmatrix}$ and $\mathbf{H} = \begin{vmatrix}\mathbf{H}_{\mathcal{A}\mathcal{A}} & \mathbf{H}_{\mathcal{A}\mathcal{P}}\\ \mathbf{H}_{\mathcal{P}\mathcal{A}} & \mathbf{H}_{\mathcal{P}\mathcal{P}}\end{vmatrix}$. We use the indices $\mathcal{A}$ and $\mathcal{P}$ in the notations to denote the correspondence with working and fixed set respectively.

The decomposition of $\mathcal{R}(\mathbf{z})$ in Eq. 3.2 can be the following L_1 least square subproblem:

$$\begin{aligned} R(\mathbf{z}_\mathcal{A}) = \frac{1}{2}\begin{vmatrix}\mathbf{z}_\mathcal{A}^T & \mathbf{z}_\mathcal{P}^T\end{vmatrix}\begin{vmatrix}\mathbf{H}_{\mathcal{A}\mathcal{A}} & \mathbf{H}_{\mathcal{A}\mathcal{P}}\\ \mathbf{H}_{\mathcal{P}\mathcal{A}} & \mathbf{H}_{\mathcal{P}\mathcal{P}}\end{vmatrix}\begin{vmatrix}\mathbf{z}_\mathcal{A}\mathbf{z}_\mathcal{P}\end{vmatrix} &+ \frac{1}{2}\begin{vmatrix}\mathbf{z}_\mathcal{A}^T & \mathbf{z}_\mathcal{P}^T\end{vmatrix}\begin{vmatrix}\mathbf{z}_\mathcal{A}\\ \mathbf{z}_\mathcal{P}\end{vmatrix} \\ +\begin{vmatrix}\mathbf{g}_{d_\mathcal{A}}^T & \mathbf{g}_{d_\mathcal{P}}^T\end{vmatrix}\begin{vmatrix}\mathbf{z}_\mathcal{A}\\ \mathbf{z}_\mathcal{P}\end{vmatrix} &+ \begin{vmatrix}\mathbf{z}_\mathcal{A}^T & \mathbf{z}_\mathcal{P}^T\end{vmatrix}\begin{vmatrix}\mathbf{g}_{c_\mathcal{A}}\\ \mathbf{g}_{c_\mathcal{P}}\end{vmatrix} \\ &+\lambda\|\mathbf{z}_\mathcal{A}\|_1 + \lambda\|\mathbf{z}_\mathcal{P}\|_1 \\ = \frac{1}{2}(\mathbf{z}_\mathcal{A}^T\mathbf{H}_{\mathcal{A}\mathcal{A}}\cdot\mathbf{z}_\mathcal{A} + \mathbf{z}_\mathcal{A}^T\cdot\mathbf{z}_\mathcal{A}) &+ (\mathbf{H}_{\mathcal{A}\mathcal{P}}\mathbf{z}_\mathcal{P} + \mathbf{g}_{d_\mathcal{A}})^T\mathbf{z}_\mathcal{A} \\ &+\mathbf{z}_\mathcal{A}^T\mathbf{g}_{c_\mathcal{A}} + \lambda\|\mathbf{z}_\mathcal{A}\|_1 + const. \end{aligned} \tag{3.3}$$

Analytical solution:

To avoid the expensive Hessian matrix ($\mathbf{H}$) calculation in each iteration for the objective update, we solve the subproblems analytically in a closed form. The minimal subproblem with only one active variable z_1 is given as follows:

$$\begin{aligned}\min_{z_1} R(z_1) = \frac{1}{2}h_{11}z_1^2 + \frac{1}{2}z_1^2 + (\mathbf{H}_{1\mathcal{P}}\mathbf{z}_{\mathcal{P}} + g_{d_1})z_1 \\ +z_1 g_{c1} + \lambda\|z_1\| \\ = \frac{1}{2}h_{11}z_1^2 + b_1 z_1 + \lambda\|z_1\|, \end{aligned} \tag{3.4}$$

where $\mathcal{A} = \{z_1\}$, $\mathcal{P} = \{k-1 \text{ remaining variable}\}$, $b_1 = \mathbf{H}_{1\mathcal{P}}\mathbf{z}_{\mathcal{P}} + g_{d_1} + g_{c_1}$.

This is the minimal subproblem of Eq. 3.2, which can be solved analytically as the following procedure. Let us separate the interval into $z_1 \geq 0$ and $z_1 \leq 0$. For positive interval the objective $R(z_1)$ becomes,

$$R(z_1) = \frac{1}{2}(h_{11}+1)z_1^2 + (b_1+\lambda)z_1.$$

Taking first-order derivative and setting it to zero, we have $z_1^{(+)} = \frac{-b_1-\lambda}{(h_{11}+1)}$. Therefore, for interval $z_1 \geq 0$, the optimal solution is:

$$z_1^{(+)^*} = \begin{cases} z_1^{(+)}, & \text{if } z_1 \geq 0 \\ \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

Similarly, for negative interval, we have the optimal solution as:

$$z_1^{(-)^*} = \begin{cases} z_1^{(-)}, & \text{if } z_1 \leq 0 \\ \\ 0, & \text{otherwise,} \end{cases} \tag{3.6}$$

where $z_1^{(-)} = \frac{-b_1+\lambda}{(h_{11}+1)}$. By considering both together, the optimal solution is the one among $z_1^{(+)^*}$ and $z_1^{(-)^*}$, whichever obtains the minimum objective value, that is: $\arg\min_{z_1^{(+)*}, z_1^{(-)*}} R(z_1)$.

Now we can see that,

$$z_1^{(+)} \geq 0 \Longrightarrow \frac{-b_1 - \lambda}{(h_{11}+1)} \geq 0 \Longrightarrow b_1 \leq -\lambda.$$

Similarly,

$$z_1^{(-)} \leq 0 \Longrightarrow \frac{-b_1 + \lambda}{(h_{11}+1)} \leq 0 \Longrightarrow b_1 \geq \lambda.$$

So, if $b_1 \leq -\lambda$ or $b_1 \geq \lambda$, the solution of Eq.3.4 is $z_1^* = \frac{-b_1-\lambda}{(h_{11}+1)}$ or $z_1^* = \frac{-b_1+\lambda}{(h_{11}+1)}$, respectively. Otherwise, $z_1^* = 0$.

Therefore, each subproblem has following analytical optimal solution:

$$z_1^* = \begin{cases} \frac{-\text{sgn}(b_1)(|b_1|-\lambda)}{(h_{11}+1)}, & \text{if } |b_1| \geq \lambda \\ \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

without loss of generality, this is the analytical solution to Eq. 3.2, where mathematical sign function is represented by operator sgn. This is the rule of updating an active or working variable coefficient. To obtain this rule, we follow a general proposition [23] given below which is very significant.

Proposition 1. The solution to the following problem

$$\min_z f(z) = z^2 + bz + \lambda|z|$$

is analytically

$$z^* = \begin{cases} -\text{sgn}(b)(|b| - \lambda), & \text{if } |b| \geq \lambda \\ \\ 0, & \text{otherwise.} \end{cases}$$

Optimality condition:

The Karush Kuhn Tucker (KKT) necessary and sufficient optimality condition of

Eq. 3.2 is:

$$\frac{\partial R(\mathbf{z})}{\partial \mathbf{z}} = \mathbf{z}^T(\mathbf{H}+\mathbf{I}) + \mathbf{g}_d^T + \mathbf{g}_c^T + \frac{\partial(\lambda\|\mathbf{z}\|_1)}{\partial \mathbf{z}} = 0. \tag{3.8}$$

However, because $\lambda\|\mathbf{z}\|_1$ is not differentiable, we employ the sub-differential concept [7]. Hence we have the optimality condition,

$$s_i = (\mathbf{H}_{i:} + \mathbf{I}_{i:})\mathbf{z} + g_{d_i} + g_{c_i} = \begin{cases} \lambda & \text{if } z_i < 0 \\ \in [-\lambda, \lambda] & \text{if } z_i = 0 \\ -\lambda & \text{if } z_i > 0, \end{cases} \tag{3.9}$$

where we use $\mathbf{s}_i$ to define the optimality condition for each variable and $\mathbf{H}_{i:}$ to denote the corresponding row of $\mathbf{H}$ matrix.

3.2.2 Dictionary and latent weight learning

In each iteration we have another phase after one cycle of inference. When $\mathbf{z}$ is learned, we apply the gradient descent update rule to adjust the dictionary $\mathbf{D}$ and latent weight $\mathbf{W}$ while the value of $\mathbf{z}$ remains fixed. We perform only one step update and proceed to next iteration cycle. We are interested in following optimization problem over the parameter $\mathbf{D}$ and $\mathbf{W}$:

$$\arg\min_{\mathbf{W},\mathbf{D}} J(\theta) = \frac{1}{2}\|\mathbf{x}-\mathbf{D}\mathbf{z}\|_2^2 + \frac{1}{2}\|\mathbf{z}-f(\mathbf{x};\mathbf{W})\|_2^2$$

$$+\frac{\alpha_1}{2}\|\mathbf{D}\|_2^2 + \frac{\alpha_2}{2}\|\mathbf{W}\|_2^2, \tag{3.10}$$

where θ denotes the collection of parameters and α_1, α_2 are the regularization parameters. We use the L_2-norm regularization for both dictionary $\mathbf{D}$ and weight $\mathbf{W}$. After each update, the column of dictionary matrix has been normalized to have unit L_2-norm to assure the uniqueness of sparse linear combination. The update rules are as follows:

$$\mathbf{D} \leftarrow \mathbf{D} - \eta_1 \nabla_{\mathbf{D}} J(\theta),$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta_2 \nabla_{\mathbf{W}} J(\theta),$$

where gradient calculations are given by $\nabla_{\mathbf{D}} J(\theta)$ and $\nabla_{\mathbf{W}} J(\theta)$ with respect to $\mathbf{D}$ and $\mathbf{W}$ correspondingly. The learning rates are η_1, η_2.

3.3 FASE Algorithm

In proposed FASE method, we successfully remove the need of large Hessian-vector matrix multiplication, which makes the method more suitable for large-scale application. Algorithm 1 shows the proposed learning method.

3.3.1 Implementation details

In each iteration, the proposed algorithm tends to select and update the coefficients, which violate the KKT optimality condition. The selection of a violating variable and a working variable is implemented systematically in constant time to optimize the procedure. The algorithm proceeds by making a choice of working variable. In each iteration, a violating variable z_1 must be set as a working variable, and should be updated analytically to optimize. After the update of z_1, the vector $\mathbf{s}$ (defined in Eq. 3.9) should be updated in order to obtain the optimality condition again and choose the new violating variable z_1. Intuitively, $\mathbf{s}$ can be updated by its definition. However, it would take linear time to update each element s_i. In fact, if we keep a record of its previous value (denoted by $\mathbf{s}_{i'}$), $\mathbf{s}_i$ can be updated in constant time. Suppose, $\mathbf{z}_{old}$ represents the coefficients of the old z_1, and $\mathbf{z}_{new}$ represents the coefficients of the updated z_1. We derive that, $s_{i_{old}} = (h_{i1}+1)z_{i_{old}} + \mathbf{H}_{i\mathcal{P}}\mathbf{z}_{\mathcal{P}_{old}} + g_{d_i} + g_{c_i}$, $s_{i_{new}} = (h_{i1}+1)z_{i_{new}} + \mathbf{H}_{i\mathcal{P}}\mathbf{z}_{\mathcal{P}_{new}} + g_{d_i} + g_{c_i}$, and $\mathbf{H}_{i\mathcal{P}}\mathbf{z}_{\mathcal{P}_{new}} = \mathbf{H}_{i\mathcal{P}}\mathbf{z}_{\mathcal{P}_{old}}$.

Algorithm 1 Unsupervised Sparse Feature Learning

```
Input: Sample data $\mathbf{x} \in \mathbb{R}^m$.
Load parameters
Random Initialization: $\mathbf{W}_{m \times k}$, $\mathbf{D}_{k \times m}$.
While ($t <$ maximum iteration) or convergence Do
    1. Normalize $\mathbf{D}$ to unit norm.
    2. Start FOR: each subproblem DO
      2.a Initialize, $\mathbf{z}_{init} = f(\mathbf{x}; \mathbf{W})$.
      2.b Run SMO to estimate optimal $\mathbf{z}$:
        • Estimate active and passive sets:
          $\mathcal{A} = \{z_1\}$,
          $\mathcal{P} = \{k - 1 \text{ remaining variable}\}$
        • Calculate
          $\mathbf{H} = \mathbf{D}^T\mathbf{D}$,
          $\mathbf{g}_c = -\text{sigm}(\mathbf{W}\mathbf{x})$,
          $\mathbf{g}_d = -\mathbf{D}^T\mathbf{z}$.
        • Iteratively select a variable $z_1$, and compute
          $b_1 = \mathbf{H}_{1\mathcal{P}}\mathbf{z}_{\mathcal{P}} + g_{d_1} + g_{c_1}$.
        • Check the optimality condition as in Eq. 3.9.
        • If $z_1$ violates condition:
            update $z_{1_{new}}$ using Eq. 3.7
            update $b_{1_{new}} = s_{1_{old}} - (h_{11} + 1)z_{1_{old}}$,
            where $s_{1_{old}} = (h_{11} + 1)z_{1_{old}} + b_1$.
      For all variables, update the optimality condition as:
            $s_i = (h_{i1} + 1)(z_{new} - z_{old}) + g_{d_i} + g_{c_i}$.
      • Terminate if no variable violates optimality condition.
     2.c Calculate new objective value using Eq. 3.1.
     2.d Estimate and update $\mathbf{W}$, $\mathbf{D}$ using gradient descent method.
            update $\mathbf{D} \leftarrow \mathbf{D} - \eta_1 \nabla_{\mathbf{D}} J(\theta)$,
            update $\mathbf{W} \leftarrow \mathbf{W} - \eta_2 \nabla_{\mathbf{W}} J(\theta)$.
    3. End FOR
End While
```

We thus can update each element s_i by the following equation which takes constant time:

$$s_i = (h_{i1} + 1)(z_{new} - z_{old}) + g_{d_i} + g_{c_i}.$$

Similar idea also applies to the computation of b_1 before updating z_1. According to the definition, b_1 can also be updated in linear time. However, it can actually be updated in constant time as well. We can update b_1 in constant time as following:

$$b_{1_{new}} = s_{1_{old}} - (h_{11} + 1) z_{1_{old}}.$$

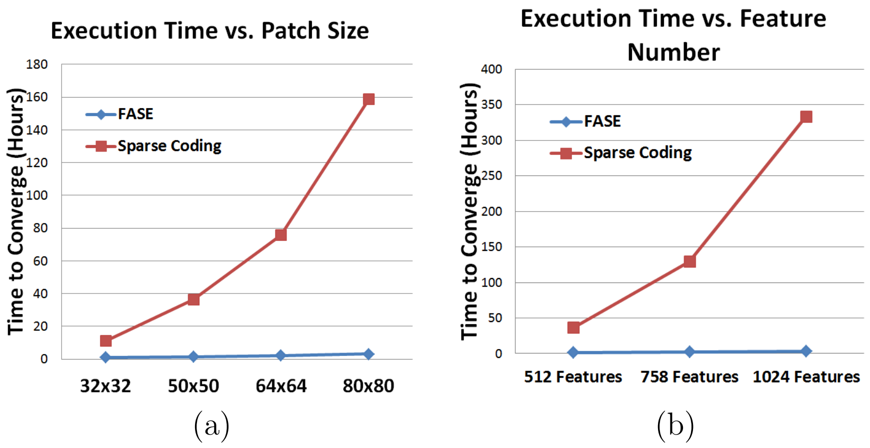
In order to optimize the selection of working variable, a meticulous procedure has been followed. We measure the difference between s_i and its corresponding desired values (Eq. 3.9) for each variables. The difference d_i is given by the following equation:

$$d_i = \begin{cases} |s_i - \lambda| & \text{if } z_i < 0 \\ |s_i| - \lambda & \text{if } z_i = 0 \\ |s_i - (-\lambda)| & \text{if } z_i > 0. \end{cases}$$

The variable with the maximum d_i value is then selected as the working variable. Also note that, before the iterative update of the method, $\mathbf{z}$ and $\mathbf{s}$ need to be initialized. So, $\mathbf{z}_{init} = f(\mathbf{x}; \mathbf{W})$ and $\mathbf{s}_{init} = (\mathbf{H}_{i:} + \mathbf{I})\mathbf{z_{init}} + g_{d_i} + g_{c_i}$, respectively. This initialization makes the iterative update very efficient, as $\mathbf{z}$ is eventually sparse.

3.3.2 Convergence Analysis

Now we can briefly analyze the the convergence of the FASE algorithm to the global optimum of Eq. 3.1. We base our analysis on a traditional SMO approach [32], [31]. In the inference phase, the LASSO problem in Eq. 3.2 is solved by the SMO. The derived KKT conditions in Eq. 3.9 are necessary and sufficient conditions for an optimal point of this type of positive definite QP problem. As long as SMO optimizes and updates the working variables at every iteration, each step will decrease the objective function according to Osuna et al. theorem [31]. The theorem simply proves that the large QP problem can be broken down into a series of smaller subproblems and maintain a feasible point that obeys all of the constraints. Therefore, a sequence of subproblems given by Eq. 3.3 that always have at least one violation variable to



(a) (b)

FIGURE 3.1: a) Efficiency of the methods with respect to various patch size. (b) Efficiency of the methods when using different number of features.

optimize, will be guaranteed to converge to the optimum. Also in each iteration cycle, after the inference loop, one step of gradient descent update takes place to optimize the problem in Eq. 3.10. Thus, the combination of two optimization problem ensures the overall convergence of the objective value given by Eq. 3.1.

3.4 Experiments and Result

To assess the efficiency of the proposed method with respect to the execution time of convergence and computational cost, we first test methods on a natural image database with various patch size and number of features. We use the Caltech-101 database that consists of 102 categories. We select randomly 30 images per class and pre-process them as in [16]: The images are converted to gray-scale, and then down-sampled and zero padded to 143×143 pixels. Finally, we normalize the images to have the standard Gaussian distribution. The convergence of each method is evaluated with the relative change in objective function value. The method is stopped when this change drops below a preset threshold. We compare the proposed method with the sparse coding algorithm proposed by Lee et al. [20] with respect to the execution time. This traditional sparse coding method has been used for the baseline comparison of the efficiency of methods. We use the code given by the author as it is fairly optimized and easy to modify. All algorithms were implemented using MATLAB on a machine with 64-bit AMD dual-processor systems, with 6GB of RAM and speed of 2.93GHz.

First, we perform experiments using 10,000 patches with varying sizes to evaluate the efficiency and scalability of the methods. Figure **??**(a) shows the comparison with respect to various patch size while we estimate 512 dimensional code vector per training sample. As shown in the figure, the execution time of sparse coding method is exponentially increasing with the larger patch size, while ours changes linearly. The figure shows that the time of the proposed method is increased from 1.04 hours to 3.1 hours while using 32×32 and 80×80 patch size, respectively. However, the sparse coding method [20] took 11.3 hours and 6.5 days for 32×32 and 80×80 patch sizes, respectively.

Second, we evaluate the efficiency of the methods with varying number of features using the same data. We randomly select $10,000$ patches of 50×50 pixels. Figure **??**(b) shows execution time for each method when the number of the learned features is increased from 512 to 1024. Our execution time is approximately 1.3 and 3.4 hours for corresponding feature numbers; whereas the sparse coding take about 1.5 and 14 days to complete the learning process. Our FASE performs significantly fast while learning large number of features from bigger size data.

Fig. **??** shows some examples of learned dictionaries using FASE and sparse coding methods for visualization. The proposed method is able to extract good features from the large patches which have been avoided because of high execution time and computational cost in the unsupervised learning process. According to Coates et al. [9] larger input size can capture complex features that cover a larger area of images. Also, larger image patches, small stride number, and learning large number features are the key factors to achieve good results. However, we see that learning large

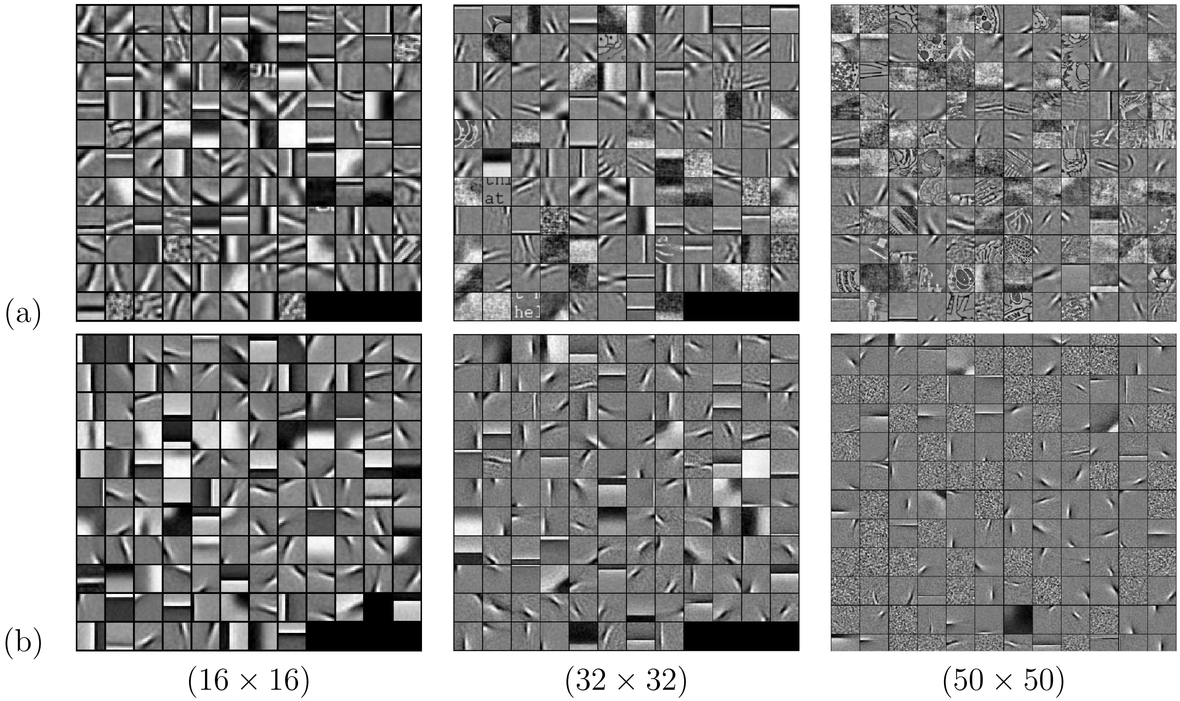


Figure 3.2: Example of learned dictionaries using various dimensional image patches with (a) the proposed FASE and (b) sparse coding [20] methods.

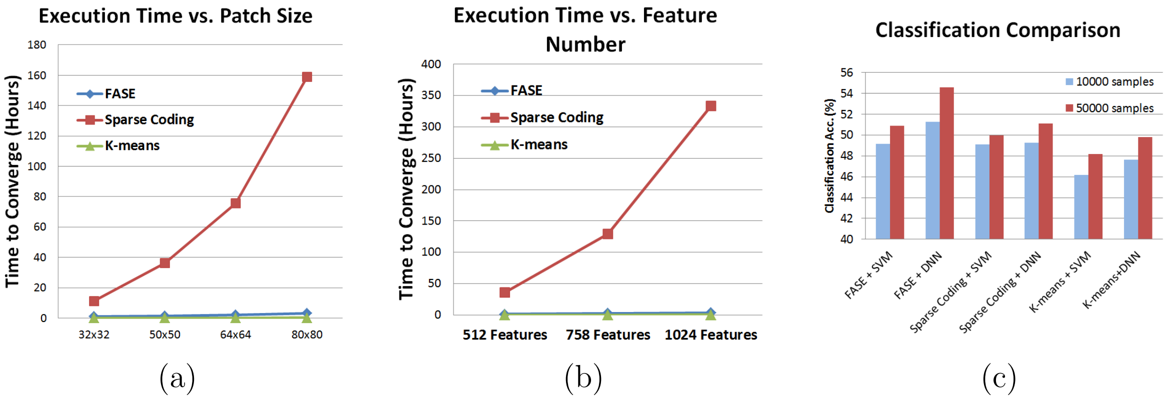
patches increases the dimensionality of the space. The key result of our experiments is that our proposed FASE speed up the convergence process by omitting expensive Hessian-vector matrix multiplication in each iteration. In comparison, state-of-the-art LASSO solver feature-sign method takes days to find coefficients for the sparse coding method [20]. Our algorithm successfully learned Gabor-like edges and also recognizes image-like complex structures. This result demonstrates the scalability of our algorithm. However, effectiveness of using these features can be better understood by using bigger data set and running it with more computational resources.

CHAPTER 4: SPARSE FEATURES AND CLASSIFICATION

This chapter evaluates our proposed method on object classification tasks. The performance of our proposed method has been assessed on several data sets including CIFAR-10 [17], Caltech-101 [10] and AR face database [25]. The method has been compared with various alternative state of the art methods in terms of the execution time and classification accuracy. All algorithms were implemented using MATLAB on a machine with 64-bit AMD dual-processor systems, with 6GB of RAM and speed of 2.93GHz.

4.1 Object Classification

In machine learning area object classification has always been a challenging problem. The linear support vector machine (SVM) and a deep neural network (DNN)have been used for the classification in our method. The rectified linear units (ReLU) [28] activation function and the dropout regularization method [13] are used to improve



(a) (b) (c)

Figure 4.1: (a) Efficiency of the methods with respect to various patch size. (b) Efficiency of the methods when using different number of features. (c) Classification comparison when using various number of samples in the learning stage and with two different classifiers (i.e., linear SVM and DNN).

the classification accuracy of the DNN. The ReLU is used for all hidden layers to produce sparse activity vectors and learn much faster than ordinary logistic units. To avoid the over-fitting problem in DNNs, we use the dropout method as a regularization which randomly sets each unit's activation in all hidden layers to be 0 with a probability p. The value of p can be chosen based on the specific problem at hand (a typical value of p is 0.5). We apply dropout only in the training phase. In the classification, the DNNs have an architecture in dimension of $\{(dim) - (A * dim) - (B * dim) - (C)\}$ with $lr = 0.01$, $momentum = 0.5$, $dropout\ fraction\ (df) = 0.5$, $ep = 1000$, and where dim is the dimension of each feature vector, $A = 1.2 * dim$, $B = 1.5 * dim$, and C is the class number. The constants in the hidden layers are chosen empirically. In the unsupervised learning part, we train the entire labeled training set of images before the classification step. We also should note that, all images in our experiments are locally normalized to have the Gaussian distribution.

4.1.1 Evaluation on CIFAR-10

To show the quality of our feature learning method in classification, we first apply the proposed method on CIFAR-10 database. The CIFAR-10 database [17] consists of 10 classes with $50,000$ training and $10,000$ testing images in size of $32 \times 32 \times 3$. We process the data using contrast normalization and whitening as in [9]. We randomly trained $40,000$ image patches in size of $6 \times 6 \times 3$. We learn 512 dictionaries with our fast unsupervised learning algorithm. Then we extract features from $50,000$ training images following the convolutional extraction process of Coates et al. [9]. In this experiment we use stride 1 with 6×6 patches to obtain a dense feature extraction. The non-linear mapping transforms the input patches into a new representation with 512 features using the learned dictionaries. Then we use pooling for dimensionality reduction; and 2048 pooled features are used to train the DNN classifier.

Our result using this set up is reported and compared with other methods in

FIGURE 4.2: Images from Caltech-101 databases with selected categories. Each row shows samples from different classes.

Table 4.1. The goal here is to show that the proposed method extracts good features for the classification. We use the available codes [9, 20] for the comparison in our experiments. We obtain better or at least comparable results with the closest state-of-the-art methods [18], [9], [20]. The method proposed in [8] reported an accuracy of 81.5%, which is slightly higher than our classification accuracy of 78.8%. However, the method [8] applies a preset soft-thresholding activation function for encoding, whereas our FASE algorithm tends to learn the encoding function by training. Therefore, the direct comparison between these two methods is beyond the scope of this work, as the encoding mechanism is fairly different.

4.1.2 Evaluation on Caltech-101

As we briefly describe above, Caltech-101 database consists of 102 categories including a background class. Figure 4.2 shows example images from the database. We use 30 training and up to 30 testing images per each category. We also follow the

Table 4.1: Precision results for the CIFAR-10 database

Method	Acc. (%)
3-Way Factored RBM (3 layers)[18]	65.3
Sparse auto-encoder(SAE) + Linear SVM [9]	73.4
Sparse RBM + Linear SVM [21]	72.4
K-means (trangle) + Linear SVM [9]	77.9
Sparse coding + DNN 2-layers [20]	73.3
FASE + DNN 2-layers	**78.8**

Table 4.2: Precision results for the Caltech-101 database

Methods	Acc. (%)
IPSD + PCA + Gaussian kernel SVM [16]	54.0
K-means (trangle) + Linear SVM [9]	48.1
Sparse coding + DNN 2-layers [20]	51.1
FASE + DNN 2-layers	**54.6**

same normalization and pre-processing steps as we did in the Section **??**. We train our proposed and alternative methods on 50,000 patches which are selected randomly from 3060 training images. The size of each patch is chosen as 32×32.

The number of learned feature is 256 in our experiment. After the dictionaries are learned, we follow the same feature extraction procedure as we explained above. We compare our FASE with the available two algorithms and closest state-of-the-art single layer methods as shown in Table 4.2. The proposed method achieves 54.6% classification accuracy that is comparable to the closest works. Using the same DNN, the sparse coding method [20] obtains 51.1% classification accuracy; whereas our learning speed is much more faster. Invariant predictive sparse decomposition (IPSD) method [16] obtains 54.0% classification rate. As can be seen, our method does not only estimate features faster than the sparse coding method [20], but also it extracts important features that are able to achieve better classification accuracy when the feature map size is relatively larger (i.e., 32×32). Also, learning method using K-means technique [9] gets 48.1% classification accuracy which is much lower than ours. The proposed method obtains better classification accuracy as can be seen on the experiments on both CIFAR-10 and Caltech-101 databases. Figure **??**(c) shows the classification results by using the linear SVM and DNN classifiers after using $10,000$ and $50,000$ unlabeled patches for each method. With these results, we show how various sample numbers affect the feature learning stage by using different classifiers to make the comparison.

4.2 Face Recognition

4.2.1 Evaluation on AR Face Database

The classification quality is also measured on AR [25] face database. The aligned AR database [25] contains 100 subjects (50 men and 50 women), with 26 different images per subject which totals to $2,600$ images taken in two sessions. In this database, there are facial expression (neural, smile, anger, scream), illumination, and occlusion (sunglass, scarf) challenges. In our experiment, we use all images without the occlusion challenges for both the unsupervised learning and classification steps. Figure 4.3 shows some example images from a subject. We segment four essential facial regions with sizes of 39 x 51 (left eye and right eye), 30 x 60 (mouth), and 45 x 42 (nose).

Figure 4.4 shows our pipeline that we follow for the AR database. First, we learn the dictionary for each facial region separately. We believe that better representations are obtained by running the unsupervised learning for each region. The features of the labeled data are extracted using the learned dictionary. Before the classification, we combine the features extracted from four regions, and train the classifier.

In our experiment, we follow a scenario described in [41] which reported one of the state-of-the-art recognition rates. We select a subset of 1400 images which are composed of 14 images per subject with the facial expression and illumination

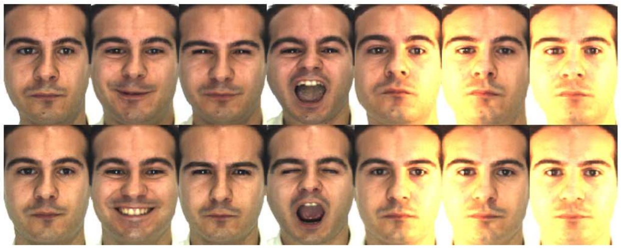


FIGURE 4.3: Images from one subject in *AR* database with various facial expression and illumination.

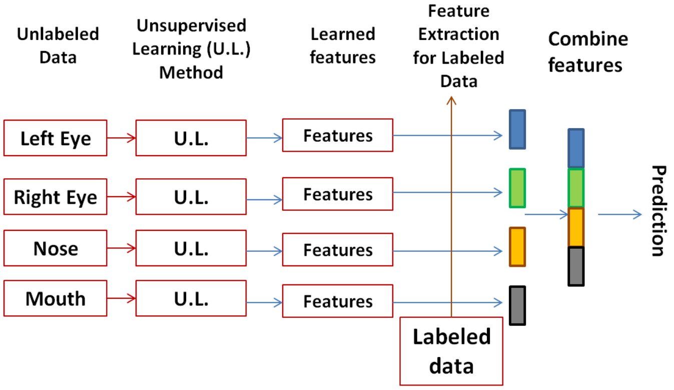


FIGURE 4.4: The framework for the classification of AR data.

changes. Various train-test image partitions are used. We conduct 10 runs for train-test procedure to get the average recognition rate for each partition.

In Table 4.3, we show the face recognition rates obtained by the proposed frame-

TABLE 4.3: Comparison of face recognition rates on *AR* database.

	Acc. (%) with various Train		
Methods	*2 Train*	*4 Train*	*5 Train*
PCA [41]	34.94	50.71	56.13
LPP [41]	55.07	68.12	71.58
NPE [41]	40.45	52.95	61.12
ONPP [41]	62.20	77.25	81.76
EPP [41]	72.45	83.86	86.23
Sparse Filt.+SVM [29]	58.83	78.50	82.22
K-means+SVM [9]	65.24	82.25	85.56
FASE+SVM	**72.83**	**85.25**	**89.12**

work, alternative methods that were reported in [41], and publicly available algorithms [29, 9]. We follow the same framework as shown in Fig. 4.4 for each method [29, 9] to get fair comparison. The results show that our framework is better than the similar single layer learning and state-of-the-art methods applied on the AR face database.

CHAPTER 5: CONCLUSION

In this thesis work we presented a new auto encoder-decoder based method for unsupervised learning of sparse representation. To estimate sparse code vectors, we propose to use SMO method for fast convergence. To avoid the expensive Hessian matrix calculation needed for coefficient update, we solve the subproblem analytically in a closed-form solution. The proposed method is applied to derive sparse representations from unlabeled image data, which is increasingly available. Our results suggest that the proposed learning method is able to learn features from larger images with low computational cost and execution time. We have tested the proposed method on several datasets and the experimental results demonstrate its effectiveness. We also discussed the scalability issue in details and extend the module in a parallel and distributed system. We outline the derivation of proposed algorithm in a Map-reduce environment. Our future goal would be the implementation of the algorithm for applications in the area of Big data [5] research. Another direction for future work might be to integrate our feature extraction scheme into a hierarchical model such as sparse deep belief net (DBN), convolutional deep belief net (CDBN) [22] for challenging machine learning and computer vision tasks.

REFERENCES

[1] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems 19* (2007), 153.

[2] Bo, L., Ren, X., and Fox, D. Multipath sparse coding using hierarchical matching pursuit. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on* (2013), IEEE, pp. 660–667.

[3] Boyd, S., and Vandenberghe, L. *Convex optimization.* Cambridge university press, 2004.

[4] Boyd, S., Xiao, L., and Mutapcic, A. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter 2004* (2003), 2004–2005.

[5] Chen, X., and Lin, X. Big data deep learning: Challenges and perspectives. *IEEE Access 2*, 8 (2014), 514–525.

[6] Cheng, H., Liu, Z., Yang, L., and Chen, X. Sparse representation and learning in visual recognition: theory and applications. *Signal Processing 93*, 6 (2013), 1408–1425.

[7] Clarke, F. H. *Optimization and nonsmooth analysis*, vol. 5. Siam, 1990.

[8] Coates, A., and Ng, A. Y. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (2011), pp. 921–928.

[9] Coates, A., Ng, A. Y., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics* (2011), pp. 215–223.

[10] Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding 106*, 1 (2007), 59–70.

[11] Hastie, T., Tibshirani, R., Friedman, J., and Franklin, J. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer 27*, 2 (2005), 83–85.

[12] Hinton, G., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation 18*, 7 (2006), 1527–1554.

[13] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[14] Jenatton, R., Mairal, J., Bach, F. R., and Obozinski, G. R. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (2010), pp. 487–494.

[15] Joachims, T. Making large scale svm learning practical. Tech. rep., Universität Dortmund, 1999.

[16] Kavukcuoglu, K., Ranzato, M., Fergus, R., and Le-Cun, Y. Learning invariant features through topographic filter maps. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 1605–1612.

[17] Krizhevsky, A., and Hinton, G. Learning multiple layers of features from tiny images. Tech. rep., Computer Science Department, University of Toronto, Tech. Rep 1 (4) 7, 2009.

[18] Krizhevsky, A., Hinton, G. E., et al. Factored 3-way restricted boltzmann machines for modeling natural images. In *International Conference on Artificial Intelligence and Statistics* (2010), pp. 621–628.

[19] Le, Q. V. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (2013), IEEE, pp. 8595–8598.

[20] Lee, H., Battle, A., Raina, R., and Ng, A. Y. Efficient sparse coding algorithms. In *Advances in neural information processing systems* (2006), pp. 801–808.

[21] Lee, H., Ekanadham, C., and Ng, A. Y. Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems* (2008), pp. 873–880.

[22] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 609–616.

[23] Li, Y., Caron, R. J., and Ngom, A. A decomposition method for large-scale sparse coding in representation learning. In *Neural Networks (IJCNN), 2014 International Joint Conference on* (2014), IEEE, pp. 3732–3738.

[24] Mairal, J., Elad, M., and Sapiro, G. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on 17*, 1 (2008), 53–69.

[25] Martinez, A., and Benavente, R. The AR face database. *Computer Vision Center, Technical Report 24* (1998).

[26] Mei, X., Ling, H., and Jacobs, D. W. Sparse representation of cast shadows via l1-regularized least squares. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 583–590.

[27] Mitchell, T. M. The need for biases in learning generalizations. Tech. rep., Computer Science Department, Rutgers University, New Brunswick, MA, 1980.

[28] Nair, V., and Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *in: Proc. Int'l Conf. Machine Learning* (2010), pp. 807–814.

[29] Ngiam, J., Chen, Z., Bhaskar, S. A., Koh, P. W., and Ng, A. Y. Sparse filtering. In *Advances in Neural Information Processing Systems* (2011), pp. 1125–1133.

[30] Olshausen, B. A., et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature 381*, 6583 (1996), 607–609.

[31] Osuna, E., Freund, R., and Girosi, F. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop* (1997), IEEE, pp. 276–285.

[32] Platt, J., et al. Sequential minimal optimization: A fast algorithm for training support vector machines.

[33] Platt, J., et al. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods-support vector learning 3* (1999).

[34] Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 759–766.

[35] Ranzato, M., and Hinton, G. E. Modeling pixel means and covariances using factorized third-order boltzmann machines. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 2551–2558.

[36] Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (2007), IEEE, pp. 1–8.

[37] van Hateren, J. H., and van der Schaaf, A. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B: Biological Sciences 265*, 1394 (1998), 359–366.

[38] Wall, M. E., Rechtsteiner, A., and Rocha, L. M. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*. Springer, 2003, pp. 91–109.

[39] Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., and Ma, Y. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31*, 2 (2009), 210–227.

[40] Yu, K., Lin, Y., and Lafferty, J. Learning image representations from the pixel level via hierarchical sparse coding. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 1713–1720.

[41] Zang, F., Zhang, J., and Pan, J. Face recognition using elasticfaces. *Pattern Recognition 45*, 11 (2012), 3866–3876.

[42] Zeiler, M. D., Taylor, G. W., and Fergus, R. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 2018–2025.

ABSTRACT

EFFECTIVE AUTO ENCODER FOR UNSUPERVISED SPARSE REPRESENTATION

by

FARIA MAHNAZ

August 2015

Advisor: Dr. Xue-wen Chen

Major: Computer Science

Degree: Master of Science

High dimensionality and the sheer size of unlabeled data available today demand new development in unsupervised learning of sparse representation. Despite of recent advances in representation learning, most of the current methods are limited when dealing with large scale unlabeled data. In this study, we propose a new unsupervised method that is able to learn sparse representation from unlabeled data efficiently. We derive a closed-form solution based on the sequential minimal optimization (SMO) for training an auto encoder-decoder module, which efficiently extracts sparse and compact features from any data set with various size. The inference process in the proposed learning algorithm does not require any expensive Hessian computation for solving the underlying optimization problems. Decomposition of the non-convex optimization problem in our model enables us to solve each subproblems analytically. Using several image datasets including CIFAR-10, CALTECH-101 and AR face database, we demonstrate the effectiveness in terms of computation time and classification accuracy. Proposed method discovers dictionaries that are able to capture low level features in larger dimensional patches in quite lower executional time than the other alternatives. Then by detailed experimental results, we present that

our module outperforms various similar single layer state-of-the-art methods including Sparse Filtering and K-Means clustering method.

AUTOBIOGRAPHICAL STATEMENT

FARIA MAHNAZ

EDUCATION

- Master of Science (Computer Science), July 2015
  Wayne State University, Detroit, MI, USA

- Master of Science (Theoretical Mathematics), January 2012
  University of Dhaka, Dhaka, Bangladesh

- Bachelor of Engineering (Mathematics), June 2010
  University of Dhaka, Dhaka, Bangladesh