SPARSE MODELING OF HIGH-DIMENSIONAL DATA FOR LEARNING AND VISION

BY

JIANCHAO YANG

DISSERTATION
Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Doctoral Committee:
Professor Thomas S. Huang, Chair
Professor Narendra Ahuja
Professor Zhi-Pei Liang
Associate Professor Yi Ma
Dr. Kai Yu, NEC Laboratories America
Sparse representations account for most or all of the information of a signal by a linear combination of a few elementary signals called atoms, and have increasingly become recognized as providing high performance for applications as diverse as noise reduction, compression, inpainting, compressive sensing, pattern classification, and blind source separation. In this dissertation, we learn the sparse representations of high-dimensional signals for various learning and vision tasks, including image classification, single image super-resolution, compressive sensing, and graph learning.

Based on the bag-of-features (BoF) image representation in a spatial pyramid, we first transform each local image descriptor into a sparse representation, and then these sparse representations are summarized into a fixed-length feature vector over different spatial locations across different spatial scales by max pooling. The proposed generic image feature representation properly handles the large in-class variance problem in image classification, and experiments on object recognition, scene classification, face recognition, gender recognition, and handwritten digit recognition all lead to state-of-the-art performances on the benchmark datasets.

We cast the image super-resolution problem as one of recovering a high-resolution image patch for each low-resolution image patch based on recent sparse signal recovery theories, which state that, under mild conditions, a high-resolution signal can be recovered from its low-resolution version if the signal has a sparse representation in terms of some dictionary. We jointly learn the dictionaries for high- and low-resolution image patches and enforce them to have common sparse representations for better recovery. Furthermore, we employ image features and enforce patch overlapping constraints to improve prediction accuracy. Experiments show that the algorithm leads to surprisingly good results.

Graph construction is critical for those graph-orientated algorithms designed for the purposes of data clustering, subspace learning, and semi-supervised learning. We model the graph construction problem, including neighbor selection and
weight assignment, by finding the sparse representation of a data sample with respect to all other data samples. Since natural signals are high-dimensional signals of a low intrinsic dimension, projecting a signal onto the nearest and lowest dimensional linear subspace is more likely to find its kindred neighbors, and therefore improves the graph quality by avoiding many spurious connections. The proposed graph is informative, sparse, robust to noise, and adaptive to the neighborhood selection; it exhibits exceptionally high performance in various graph-based applications.

To this end, we propose a generic dictionary training algorithm that learns more meaningful sparse representations for the above tasks. The dictionary learning algorithm is formulated as a bilevel optimization problem, which we prove can be solved using stochastic gradient descent. Applications of the generic dictionary training algorithm in supervised dictionary training for image classification, super-resolution, and compressive sensing demonstrate its effectiveness in sparse modeling of natural signals.
To Iek Teng – my confidante, my partner
ACKNOWLEDGMENTS

First, I would like to express profound gratitude to my adviser Professor Thomas Huang for all the guidance, advice, and support he has given me during my Ph.D. study at UIUC. For the last five and half years, I have been inspired by his vision and passion to research, his attention and curiosity to details, his dedication to the profession, his intense commitment to his work, and his humble and respectful personality. During this most important period in my career, I thoroughly enjoyed working with him, and what I have learned from him will benefit me for my whole life.

I also would like to give my thanks to Professor Yi Ma for his suggestions and support for the last several years. I was deeply impressed and inspired by his passion and insights into research. I also feel very lucky to have had the chance to work with Dr. Kai Yu as a summer intern at NEC Laboratories America. From him, I learned not only techniques and skills but also his precious research insights. I am also very thankful to Professor Narendra Ahuja and Professor Zhi-Pei Liang for their advice and comments. I am fortunate to have them as my thesis committee members.

During my Ph.D. study over the last five years, it has been a great honor for me to collaborate with Hasegawa-Johnson at UIUC, Professor Shuicheng Yan at National University of Singapore, Professor Jiebo Luo at Rochester University, Professor John Wright at Columbia University, Professor Yun Fu at University at Buffalo, and Professor Xu Han at University of Missouri. It has also been an honor to work with many excellent researchers during my summer internships, including Dr. Kai Yu, Dr. Jinjun Wang, and Dr. Fengjun Lv at NEC Laboratories America, Dr. Jie Yu at GE Global Research, and Dr. Zhe Lin and Dr. Scott Cohen at Adobe. I owe sincere gratitude to many student collaborators, including Bin Cheng, Hao Tang, Haichao Zhang, Zhaowen Wang, and Jiangping Wang. I also owe sincere thanks to all the students at IFP for helping me in both living and research during my Ph.D. study.
Last, I would like to give special thanks to my wife, Iek Teng Wu, for being very supportive and patient during the course of this Ph.D. program. It was the luckiest thing for me in this world to meet her at UIUC. Special gratitude and thanks also to my parents for their unconditional support and care throughout my life. Life would not be so beautiful without their love.
# TABLE OF CONTENTS

CHAPTER 1  INTRODUCTION ................................................. 1

CHAPTER 2  POOLING THE SPARSE FEATURES FOR IMAGE CLASSIFICATION ........................................... 8
  2.1  Introduction ......................................................... 8
  2.2  Feature Pooling Based on Sparse Representations ........................................... 11
  2.3  Efficient Highly Over-Complete Sparse Coding Using a Mixture Model ........................................... 24
  2.4  Conclusion ......................................................... 35

CHAPTER 3  SINGLE IMAGE SUPER-RESOLUTION VIA SPARSE REPRESENTATION ........................................... 36
  3.1  Introduction ......................................................... 36
  3.2  Image Super-Resolution from Sparsity ........................................... 40
  3.3  Learning the Dictionary Pair ........................................... 49
  3.4  Experimental Results ........................................... 54
  3.5  Conclusion ......................................................... 64

CHAPTER 4  LEARNING THE $\ell_1$-GRAPH FOR IMAGE ANALYSIS ........................................... 66
  4.1  Rationales on the $\ell_1$-graph ........................................... 68
  4.2  Learning with the $\ell_1$-graph ........................................... 72
  4.3  Experiments ......................................................... 75
  4.4  Conclusion ......................................................... 83

CHAPTER 5  BILEVEL SPARSE MODELING ........................................... 85
  5.1  Introduction ......................................................... 85
  5.2  Sparse Coding in a Single Feature Space ........................................... 86
  5.3  Bilevel Sparse Coding for Specific Tasks ........................................... 87
  5.4  Applications ......................................................... 93
  5.5  Conclusion ......................................................... 107

CHAPTER 6  SUMMARY ......................................................... 108

CHAPTER 7  PUBLICATIONS DURING PH.D. STUDY ........................................... 111

REFERENCES ......................................................... 114
CHAPTER 1
INTRODUCTION

A sparse signal is a signal that can be represented as a linear combination of a small number of elementary signals called atoms in a basis or an over-complete dictionary. The representation for a sparse signal seeks a solution to the linear system while requiring that the unknown has a few nonzero entries

$$\min_{\alpha} \|\alpha\|_0 \text{ s.t. } \Phi \alpha = x.$$  \hspace{1cm} (1.1)

$$\Phi \in \mathbb{R}^{d \times K}$$ (typically, \(d \leq K\)) is a real matrix whose columns have unit Euclidean norm, which is often referred to as a dictionary. The \(\ell_0\)-norm \(\| \cdot \|_0\) is a counting function \(\mathbb{R}^K \rightarrow \mathbb{R}\) that returns the number of nonzero components. \(\alpha\) is \(s\)-sparse if \(\| \alpha \|_0 \leq s\), and we refer to \(\alpha\) as a \textit{sparse representation} of the signal \(x\) with respect to the dictionary \(\Phi\). One natural variation is to relax the equality constraint to allow some error tolerance \(\epsilon \geq 0\), where the signal is contaminated with noise

$$\min_{\alpha} \|\alpha\|_0 \text{ s.t. } \|\Phi \alpha - x\|_2 \leq \epsilon.$$  \hspace{1cm} (1.2)

However, solving the above sparse representation or approximation problem directly is combinatorially NP-hard [1]. In the past several years, there have been exciting breakthroughs in the study of high dimensional sparse signals. Much of the excitement centers around the discovery that under surprisingly broad conditions, a sufficiently sparse linear representation in Equation (1.1) and Equation (1.2) can be correctly and efficiently computed by greedy methods or convex optimization (the \(\ell_0 - \ell_1\) equivalence). By formulating the sparse representation problem as a convex optimization, we have

$$\min_{\alpha} \|\alpha\|_1 \text{ s.t. } \Phi \alpha = x,$$  \hspace{1cm} (1.3)

or

$$\min_{\alpha} \|\alpha\|_1 \text{ s.t. } \|\Phi \alpha - x\|_2 \leq \epsilon.$$  \hspace{1cm} (1.4)
Replacing the combinatorial $\ell_0$-norm formulation with the $\ell_1$-norm yields convex optimization problems that admit tractable and efficient algorithms. In the concrete sense, the $\ell_1$-norm is the tightest convex relaxation for the $\ell_0$-norm. Results from [2] and [3] demonstrate that the convex relation succeeds well in the presence of the restricted isometry property (RIP) on $\Phi$ with order $s$

$$ (1 - \delta_s)\|\alpha\|_2^2 \leq \|\Phi\alpha\|_2^2 \leq (1 + \delta_s)\|\alpha\|_2^2, \quad (1.5) $$

where $\|\alpha\|_0 \leq s$ and constant $\delta_s < 1$. Although no known deterministic matrix yields a substantially good RIP, certain random matrices, e.g., Gaussian and Bernoulli, establish much stronger RIP bounds, which explains the randomness in compressive sampling [4, 5].

Sparse representation has proven to be an extremely powerful tool for acquiring, representing, and compressing high dimensional signals. In the more general sense, sparsity constraints have emerged as a fundamental type of regularizer for many ill-conditioned or under-determined linear inverse problems, which arise throughout the engineering and mathematical sciences. In the past several years, variations and extensions of sparsity promoting $\ell_1$-norm minimization have been applied to many vision and machine learning tasks [6], such as segmentation [7, 8], denoising and inpainting [9, 10, 11, 12], background modeling [13, 14], photometric stereo [15], face recognition [16, 17], audio classification [18, 19], bioinformatics [20], human action recognition [21], graph construction and label propagation [22, 23], and image classification [24, 25, 26, 27].

In this dissertation, we will explore several different areas in computer vision and learning based on sparse modeling. Specifically, we will discuss image classification, single image super-resolution, and graph learning, all based on sparse representations of the data in terms of some dictionary. We will also discuss a general bilevel sparse coding algorithm for learning dictionaries that generate more informative and meaningful sparse representations for various tasks in vision and signal recovery.

**Image Classification** Image classification, which aims to categorize the input image into one of the predefined classes, is among the most important and fundamental problems in computer vision. Successful image classification is critical in many applications, such as automatic video surveillance, robotics, security, image and video retrieval, human and computer interface, and automatic check scanning.
Despite a long history of active research in this area, image classification performance is still far from being used in real applications. The major difficulty for generic image classification has its roots in the vast variation of natural images that arises from viewpoint changes, lighting changes, deformation of non-rigid or articulated objects, intra-class appearance variations, the presence of occlusions, misalignments, the quality of the captured images, and background noise. The loss of image correspondence as a result of these variations makes the distribution of each image class highly complicated and intractable. Finding a compact generic image feature representation for discriminative analysis thus becomes extremely difficult.

Depending on the classification tasks, image classification can be as specific as face recognition, gender recognition, object recognition, scene categorization, action recognition, or digit recognition. For facial analysis and digit recognition, where the image correspondences can be roughly kept, many holistic approaches have been explored [28, 29, 30, 25, 31]. Appearance-based holistic approaches for face recognition, such as Eigenfaces [32] and Fisherfaces [33], have proved to be effective in face recognition experiments using large databases. In [16] and [17], sparse representation is applied to face recognition on well-aligned frontal face images. The algorithm, although very simple, achieves surprisingly good results when dealing with illumination changes and occlusions. However, these holistic approaches are sensitive to image misalignment, such as pose variations and expression variations. Unlike face images, the correspondences for object recognition and scene classification hardly exist, and holistic approaches in this scenario easily fail. For such categorization tasks with large within-class variations, the bag-of-features methods, which represent an image as an orderless collection of local features, have recently demonstrated an impressive level of performance [34, 35]. Over the last several years, extensive research efforts have been devoted to improve the bag-of-features model [36, 37, 38, 39, 40, 41, 42, 43]. However, these approaches are limited by either their performance or computational costs.

In this dissertation, we propose an efficient and effective generic image representation that aims to apply to all the above image classification tasks. By max pooling over different spatial locations across different spatial scales over sparse representations of the local image descriptors in terms of a pre-trained over-complete dictionary, our image feature representation possesses the following desirable properties:
• The feature representation is robust to most of the image variations mentioned above.

• Unlike most histogram based representations, our feature fits linear kernel well and thus is applicable to large-scale scenarios.

• The feature representation is generic and can be applied to various image classification tasks.

**Image Super-resolution** In most digital imaging applications, high-resolution images or videos are usually desired for later image processing and analysis. The desire for high image resolution stems from two principal application areas: improvement of pictorial information for human interpretation; and helping representation for automatic machine perception. The image resolution is limited by the imaging system, and building imaging chips and optical components to capture high-resolution images could be prohibitively expensive and not practical in most real applications, *e.g.*, widely used surveillance cameras and web cameras. In other scenarios such as satellite imagery, it is difficult to use high-resolution sensors due to physical constraints. An alternative way to address this resolution problem is to accept the image degradations and use signal processing techniques to postprocess the captured images, balancing the computational cost with the hardware cost. These techniques are specifically referred to as super-resolution reconstruction (SR). Super-resolution techniques are tools that construct high-resolution (HR) images from several observed low-resolution (LR) images, thereby increasing the high-frequency components and removing the degradations caused by the imaging processing of the low-resolution camera. SR arises in many areas, such as

1. Surveillance video [44, 45]: frame freeze and zoom region of interest (ROI) in video for human perception; resolution enhancement for automatic target recognition.

2. Medical imaging (CT, MRI, ultrasound) [46, 47]: several images limited in resolution quality can be acquired, and SR techniques can be applied to enhance the resolution.

3. Remote sensing [48]: several images of the same area are provided, and an improved resolution image can be sought.
4. Video standard conversion, *e.g.*, from NTSC video signal to HDTV signal.

5. Enlarging consumer photographs, zoom web videos, etc.

Image super-resolution has become an active research area since the seminal work by Tsai and Huang in 1984. Many techniques have been proposed over the last two decades [49] representing approaches from the frequency domain to the spatial domain, and from signal processing to machine learning perspectives. Depending on the availability of low-resolution observations, SR techniques can be applied to multi-frame super-resolution based on reconstruction constraints or single-frame super-resolution based on machine learning techniques. Compared to the reconstruction based SR techniques, machine learning based techniques can break the super-resolution limit with additional training data. Over the last several years, the machine learning based techniques have achieved substantial progress [50, 51, 52, 53]. However, most of these approaches fall into heuristics, and their performances are limited due either to overly smoothness or unwanted artifacts.

In this dissertation, we focus on the single frame image super-resolution problem, *i.e.*, to recover a high-resolution image from a single low-resolution input. Unlike those heuristic approaches, we formulate this ill-posed inverse problem as a sparse signal recovery problem inspired by the recent compressive sensing theories.

**Graph Construction for Image Analysis** An informative graph, directed or undirected, is critical for those graph-oriented algorithms designed for the purposes of data clustering, subspace learning, and semi-supervised learning. Data clustering often starts with a pairwise similarity graph and is then transformed into a graph partition problem [54]. The pioneering works on manifold learning, *e.g.*, ISOMAP [55], Locally Linear Embedding [56], and Laplacian Eigenmaps [57], all rely on graphs constructed in different ways. Moreover, most popular subspace learning algorithms, *e.g.*, Principal Component Analysis [58], Linear Discriminant Analysis [33], and Locality Preserving Projections [59], can all be explained within the graph embedding framework as claimed in [28]. Also, most semi-supervised learning algorithms are driven by certain graphs constructed over both labeled and unlabeled data. Zhu *et al.* [60] utilized the harmonic property of the Gaussian random field over the graph for semi-supervised learning. Belkin
and Niyogi [61] instead learned a regression function that fits the labels at labeled data and also maintains smoothness over the data manifold expressed by a graph.

There exist two popular ways for graph construction, one of which is the $k$-nearest-neighbor method, and the other is the $\epsilon$-ball based method, where, for each datum, the samples within its surrounding $\epsilon$ ball are connected, and then various approaches, e.g., binary, Gaussian-kernel [57] and $\ell_2$-reconstruction [56], can be used to further set the graph edge weights. Since the ultimate purposes of the constructed graphs are for tasks such as data clustering, subspace learning, and semi-supervised learning, the following graph characteristics are desired:

1. **Robustness to data noise.** Data noise is inevitable especially for visual data, and robustness is a desirable property for a satisfactory graph construction method. The graph constructed by $k$-nearest-neighbor or $\epsilon$-ball methods is founded on pair-wise Euclidean distance, which is very sensitive to data noise. It means that the graph structure is easy to change when unfavorable noise enters.

2. **Sparsity.** Recent research on manifold learning [57] shows that a sparse graph characterizing locality relations can convey valuable information for classification purposes. Also for applications with large-scale data, a sparse graph is the necessary choice due to storage limitation.

3. **Datum-adaptive neighborhood.** Another observation is that the data distribution may vary greatly at different areas of the feature space, which results in a distinctive neighborhood structure for each datum. Both $k$-nearest-neighbor and $\epsilon$-ball methods, however, use a fixed global parameter to determine the neighborhoods for all the data, and hence fail to offer such datum-adaptive neighborhoods.

Inspired by sparse representation based face recognition work [16], we propose a new graph construction procedure for high-dimensional data with a sparse structure assumption, called $\ell_1$-graph, which utilizes overall contextual information of the data samples instead of only pairwise Euclidean or kernel distances, as done conventionally. The neighboring samples of a datum and the corresponding ingoing/outgoing edge weights are simultaneously determined by solving a sparse representation problem through $\ell_1$-norm minimization. The proposed $\ell_1$-graph naturally possesses the above desirable properties. The graph is also informative in that it can uncover the underlying data similarity (semantic information). This
ability is partially derived from a simple but important property of the data: although the images (or their features) are in very high-dimensional spaces, in many applications, images belonging to the same class exhibit degenerate structure, i.e., they lie on or near low-dimensional subspaces. Therefore, projecting a signal to its nearest and lowest dimensional subspace is likely to find its kindred neighbors.

The rest of this dissertation is organized as follows. Chapter 2 discusses the literature of image classification and presents a novel generic image feature representation based on multi-scale max pooling over sparse codes of local image descriptors. Chapter 3 introduces the single image super-resolution problem and provides a principled solution based on sparse signal recovery. Chapter 4 explores the sparse representation for signal space modeling and presents a graph construction procedure with explicit sparsity constraint. As all these applications are based on the sparse representations of the data in terms of some learned dictionary, in Chapter 5 we propose a generic bilevel sparse coding model for learning more informative sparse representations for the tasks to be performed. Finally, Chapter 6 summarizes this dissertation with discussions for future exploration.
CHAPTER 2

POOLING THE SPARSE FEATURES FOR IMAGE CLASSIFICATION

2.1 Introduction

Recently, bag-of-features (BoF) and spatial pyramid matching (SPM) [38] have become extremely popular in various visual recognition tasks, e.g., object recognition and scene classification. The BoF model treats an image as a loose collection of unordered appearance descriptors (e.g., the SIFT descriptor [62]) extracted from local patches, quantizes them into discrete “visual words,” and then computes a compact histogram representation for semantic image classification. However, the BoF model discards all the spatial information of the local descriptors, which is informative or even crucial for discriminative analysis, and therefore its performance is limited. The SPM method overcomes this difficulty by partitioning the image into increasingly finer spatial sub-regions and computes histograms of local descriptors from each sub-region. Typically, $2^l \times 2^l$ sub-regions in different scales $l = 0, 1, 2$ are used. Other spatial partitions, such as $3 \times 1$, have also been attempted to incorporate domain knowledge for images, for example, with “sky” on top and “ground” on bottom. The resulting “spatial pyramid” representation is a computationally efficient extension of the orderless BoF model, contributes as the major component in many challenging tasks such as PASCAL, and has shown very promising performance in various benchmarks.

The bag-of-features image representation model effectively accounts for the large appearance variations in general images. Various extensions based on the BoF image representation model have been proposed in the past few years, leading to substantial improvements in the image categorization area. Gemert et al. [40] proposed soft assignment methods to address the vector quantization error problem of hard assignment in the traditional bag-of-words model. Empirically, people find that histograms (from either hard assignment or soft assignment), which only consider zero-order statistics, have to work with nonlinear Mercer kernels, e.g.,
the chi-square or intersection kernel, to achieve a good performance. Accordingly, the nonlinear classifier has to afford additional computational complexity, bearing $O(n^3)$ in training and $O(n)$ for testing in SVM, implying poor scalability for large-scale applications. Perronnin and Dance [41] proposed a fisher kernel approach based on the Gaussian Mixture Model (GMM) to aggregate the set of descriptors by considering their zero-, first- and second-order statistics. By mapping the image into a high-dimensional feature space, the obtained image feature works rather well with linear classifiers. Zhou et al. [63] proposed a similar approach by incorporating the spatial information of the local descriptors. In [26], we proposed a different approach by max pooling over the sparse representations of the local descriptors with respect to a learned over-complete dictionary. Our new feature works exceptionally well with a linear kernel. A later work [64] proposed to encode the descriptor locality for highly over-complete sparse coding, which achieves a remarkably good performance on very challenging image classification tasks. In summary, the stream of the most successful works based on BoF image representation for learning image features can be summarized into the following modules [65]:

1. **Image Representation**: Images are usually represented as a collection of (coordinated) local descriptors, e.g., image raw patches, keypoints, HOG [66], SIFT, or LBP [67].

2. **Nonlinear Coding**: The local descriptors are then transformed into some codes with desired properties, such as compactness, sparseness, or statistical independence.

3. **Feature Pooling**: These codes are then summarized (over different spatial locations) to get a compact feature representation, e.g., through average pooling (for histogram) and max pooling.

4. **Discriminant Training**: Classifiers are then trained based on the feature representations, such as linear SVM or kernel SVM (chi-square or intersection kernel).

In this chapter, we discuss the techniques of hierarchical spatial max pooling technique based on sparse coding recently published in [26] and [64]. The proposed generic image representation will be applied to various image classification tasks including object recognition, scene categorization, action recognition, face recognition, gender recognition, and handwritten digit recognition.
2.1.1 Related Work

Sparse representation has been successfully applied to many inverse problems, e.g., image restoration [68], and also well applied to classification tasks [16, 25, 26]. Wright et al. [16, 17] cast the recognition problem as one of finding a sparse representation of the test image in terms of the training set as a whole, up to some sparse error due to occlusion. The algorithm achieves impressive results on public datasets, but fails to handle practical face variations such as alignment and pose. Both [16] and [17] utilize the training set as the dictionary for sparse coding, and the sparse representation is modeled directly as the classifier. Others tried to train a compact dictionary for sparse coding [25, 69], and the sparse representations of the signals are used as image features trained later with some classifier, e.g., SVM. These holistical sparse coding algorithms on the entire image, on one hand, hold robustness to corruptions such as noise and occlusions, as shown in [16]. On the other hand, the underlying linear subspace assumption considerably limits the applications and performances of these approaches, e.g., face expression is known to be nonlinear.

Instead of sparse coding holistically on the entire image, learning sparse representations for local descriptors has also been explored for classification purposes. Raina et al. [42] described an approach using sparse coding to construct high-level features, showing that sparse representations perform much better than conventional representations, e.g., raw image patches. In [26], we proposed a hierarchical structure where the sparse coding model is applied over the handcrafted SIFT features, followed by spatial pyramid max pooling. Applied to general image classification tasks, the proposed approach achieves state-of-the-art performance on several benchmarks with a simple linear classifier. Different network structures were also proposed for fast inference for sparse coding algorithms [70, 71]. However, these models are difficult to train and the supervised training cannot guarantee the sparsity of the data representation.

2.1.2 Organization

The remainder of this chapter is organized as follows. Section 2.2 presents our framework of hierarchical feature pooling based on sparse codes of local image descriptors for image analysis. Section 2.3 extends this sparse coding based framework with a mixture model for efficient encoding with highly-over-complete
dictionaries. Extensive experiments on various benchmark datasets validate the effectiveness of our framework.

2.2 Feature Pooling Based on Sparse Representations

In this section, we present the framework of hierarchical feature pooling based on sparse representations of local descriptors for image classification. Based on the bag-of-features image representation in a spatial pyramid, we transform the local image descriptors, e.g., SIFT descriptor, into sparse representations with respect to a learned dictionary, from which max pooling is performed over different spatial blocks and cross different spatial scales to obtain the final image feature representation. By hierarchical spatial max pooling, our image feature representation achieves different levels of local translation invariance in different spatial scales.

2.2.1 Structured Image Representation

A fundamental problem in visual classification is to design good image feature representations, which can achieve good trade-off between discriminative power and invariance. While the raw image pixel representation has the highest discriminative power, it also loses the invariance property when measured using standard Euclidean distance, especially for general purpose image classification tasks, where image correspondences are not available at all. In object and scene classification, the most popular image representation is to regard the image as a loose collection of local descriptors, e.g., SIFT descriptors, HOG descriptors, or raw image patches, which achieves translation invariance for the local descriptors.

In our hierarchical model, we also represent the image as a collection of local image descriptors, which can be SIFT descriptors or image patches depending on the specific tasks. Specifically, the image is represented $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{d \times n}$, where $x_i \in \mathbb{R}^d$ denotes the $i$-th local descriptor of the image in a column vector. To capture the informative spatial information of the local descriptors, we reorganize the local descriptors in a spatial pyramid as in [38]

$$X = [Y_{11}^0, Y_{11}^1, Y_{12}^1, Y_{21}^1, Y_{22}^1, ..., Y_{44}^2] ,$$  (2.1)
where $Y_{ij}^s$ denotes the collection of local descriptors falling into the $(i, j)$-th spatial block in $s$-th spatial scale. In each spatial scale $s$, the image is divided into $2^s \times 2^s$ spatial blocks, within which the local descriptors are grouped together, e.g., $Y_{11}^0$ denotes the first spatial scale and contains all the local descriptors in the image, and $Y_{11}^1$ contains the local descriptors in the upper left quarter of the image. Within each spatial block, the local descriptors $Y_{ij}^s$ are orderless, and therefore, a feature vector defined on them will be invariant to local translations of the descriptors within that spatial block. More generally, the partition configuration of the image into local spatial blocks can be arbitrary, depending on the domain knowledge of the data. For example, a spatial partition of $3 \times 1$ will make more sense for images with “sky” on top and/or “ground” at bottom.

### 2.2.2 Nonlinear Encoding

Natural image signals are evidently high-dimensional sparse signals, *i.e.*, the signal can be represented as a linear combination of relatively few base elements in a basis or an over-complete dictionary. The sparsity level of the signal determines its degree of freedom or intrinsic dimension given the dictionary, and thus seeking sparse representations has the motivation of finding the correct and simpler model for the signals. To achieve sparser representations, people have gone beyond linear transformations, such as Fourier transform, DCT, wavelet, and contourlet, to nonlinear transforms, such as sparse coding with respect to an over-complete dictionary.

In the traditional bag-of-words model, the local descriptors are modeled by vector quantization using a codebook or dictionary trained from $K$-means. Given the dictionary $B$ (can be over-complete) and a signal $x$, the nonlinear encoding scheme for vector quantization can be cast into the following optimization

\[
\begin{align*}
\min_{z} & \quad \|x - Bz\|_2^2 \\
\text{s.t.} & \quad \|z\|_0 = 1, \|z\|_1 = 1, z \succeq 0,
\end{align*}
\]  

(2.2)

where $\|z\|_0 = 1$ is a cardinality constraint, meaning that only one element of $z$ is nonzero, $z \succeq 0$ means that all the elements of $z$ are nonnegative, and $\|z\|_1$ is the $\ell_1$-norm of $z$, the summation of the absolute value of each element in $z$. After the optimization, the index of the only nonzero element in $z$ indicates which cluster the vector $x$ belongs to.
The sparsity level of $z$ in Equation (2.2) is constrained to be one, implying a zero-order approximation for the original signals, resulting in large quantization errors for continuous signals. In order to alleviate the quantization error problem, we can relax the cardinality constraints. Suppose the signal is $k$-sparse, i.e., the signal can be well approximated by at most $k$ element bases given some dictionary $B$; then the sparse representation of the signal can be recovered by

$$\min_z \|x - Bz\|_2^2$$

s.t. $\|z\|_0 \leq k.$

(2.3)

Unfortunately, the above optimization is no longer convex and finding the exact solution is NP-hard. Recent advances in sparse representation theories reveals that Equation (2.3) can be well approximated by $\ell_1$-norm convex relaxation under broad conditions. Therefore, we can approximate the above sparse representation problem by

$$\min_z \|x - Bz\|_2^2 + \lambda \|z\|_1,$$

(2.4)

for some chosen regularization parameter $\lambda$, which is convex and efficient to solve. Given the dictionary model, Equation (2.4) finds the sparse representations that capture the intrinsic dimension of the signal, and based on that we extract our image feature representation for various recognition tasks in Section 2.2.3. In Section 2.2.4, we will describe how to learn a dictionary $B$ for our tasks.

### 2.2.3 Hierarchical Spatial Max Pooling

Given the spatially structured image representation $X = [Y_{011}, Y_{111}, Y_{121}, ..., Y_{441}]$, we transform the local descriptors falling into each spatial block into sparse representations with respect to a given dictionary $B$ as

$$Z_{ij}^0 = \arg \min_z \|Y_{ij}^0 - Bz\|_2^2 + \lambda \|z\|_1.$$

(2.5)

Correspondingly, we have the spatially structured sparse codes

$$Z = [Z_{11}^0, Z_{11}^1, Z_{12}^1, ..., Z_{44}^2],$$

(2.6)
where $Z^s_{ij}$ consists of the sparse representations as its columns for the local descriptors in $X^s_{ij}$. Our hierarchical spatial max pooling feature is then defined as

$$\beta = \bigcup_{s=0}^{2} \left[ \bigcup_{i,j=1}^{s+1} [\beta^s_{ij}] \right],$$  

(2.7)

where $\bigcup$ is the vector concatenation operator, and $\beta^s_{ij}$ is defined as

$$\beta^s_{ij} = \max(|Z^s_{ij}|),$$  

(2.8)

where the max function is to extract the maximum value for each row of a matrix. Equation (2.8) merely means the vector $\beta^s_{ij}$ contains the maximum absolute values from each row of the sparse representation matrix $Z^s_{ij}$. Figure 2.1 illustrates the whole framework of our spatial hierarchical feature pooling based on sparse representations of the local descriptors. Based on sparse coding, we first transform each local descriptor into the corresponding sparse representation given the over-complete dictionary, from which max pooling is performed over increasingly larger spatial regions to summarize feature vectors with different levels of translation invariance and discriminative power. The final feature representation combining the max pooling features from different spatial blocks across multiple spatial scales achieves a trade-off between discriminative power and translation invariance.

**Figure 2.1:** Illustration for the hierarchical spatial max pooling over sparse representations of the local descriptors for feature extraction.
2.2.4 Dictionary Training for Local Descriptors

Our feature extraction algorithm is built on the sparse representations of the local descriptors, and therefore, the over-complete dictionary promoting such representations is important. Such dictionaries can be chosen as mathematically predefined ones, or can be adapted to the signals of interest through learning [72]. Empirically, adaptive dictionaries usually generate better performances in practice [10]. Representative dictionary learning algorithms include MOD [73], K-SVD [12], Fields of Experts (FoE) [74], and others [75]. In our framework, we use the sparse coding algorithm in [75] for learning the over-complete dictionary for local image descriptors. Specifically, we randomly sample a large set of local descriptors $X = [x_1, x_2, ..., x_N]$ from our training images from all classes, from which we minimize the following energy function to train our dictionary $B$:

$$
\min_{\{z_i\}, B} \sum_{i=1}^{N} \{\|x_i - Bz_i\|_2^2 + \lambda \|z_i\|_1\}
$$

$$
\text{s.t. } \|B(:, k)\|_2 \leq 1, \ \forall k,
$$

(2.9)

where the norm constraints on each column of the dictionary $B$ is to avoid trivial solutions.\textsuperscript{1} The optimization problem in Equation (2.9) is not jointly convex in $B$ and $\{z_i\}_{i=1}^{N}$ simultaneously, but is convex in one, once the other is fixed. Therefore, the optimization is naturally done in an alternative coordinate descent fashion between $B$ and $\{z_i\}_{i=1}^{N}$. Fixing dictionary $B$, the inference for $\{z_i\}_{i=1}^{N}$ is a set of Lasso [76] problems that can be solved efficiently by many existing packages, e.g., [77]. Fixing the set of sparse representations, optimization for $B$ is simply a quadratically constrained quadratic programming problem. At each alternating step, the objective function value is reduced, and therefore, the algorithm will guarantee to converge to a local minimum [75], similar to the $K$-means algorithm. In practice, we do not find that our algorithm is sensitive to the local minimums.

\textsuperscript{1}The objective function value can always be reduced by increasing the column norms of $B$ and pushing the norm of $z_i$ to be arbitrarily small.
2.2.5 Experiments

In this section, we evaluate our generic hierarchical sparse coding framework for various image recognition tasks, including object recognition (Caltech-101 [78] and Caltech-256 [79]), scene classification (15 Scenes [38]), action recognition (TRECVID 2008 Surveillance video [80]), face recognition (CMU PIE [81] and CMU Multi-PIE [82]), gender recognition (FRGC 2.0 [83]), and handwritten digit recognition (MNIST [84]). In our experiments, we in particular implemented and compared with two classes of SPM methods,

1. KSPM: the popular nonlinear kernel SPM that uses spatial pyramid histograms and chi-square or intersection kernel;

2. LSMP: the simple linear SPM that uses a linear kernel on spatial pyramid histograms.

Our implementations use a single descriptor type, the SIFT descriptor [62], for the first three tasks and raw image patches for the remaining three tasks. The SIFT descriptors extracted from $16 \times 16$ pixel image patches are densely sampled from each image on a grid with step size of 8 pixels. Empirically, we find that using descriptors from more than one scale (e.g., $24 \times 24$ and $32 \times 32$) and denser sampling (e.g., step size of 4 pixels) typically helps to improve the performance. Instead of focusing on these engineering details, we simply choose the above settings to fairly compare with [38]. The dictionary size for Caltech-101, Caltech-256, and 15 Scenes are fixed as $1024$, and $256$ for TRECVID 2008 Surveillance Video. The raw image patches are densely sampled from the image on a regular grid with step size of 1 pixel. These raw patches are pre-normalized to be unit vectors before sparse coding. The patch size is chosen as $8 \times 8$ for CMU PIE, Multi-PIE, and FRGC, and $12 \times 12$ for MNIST. The dictionary size is $128$ for CMU PIE, Multi-PIE, FRGC, and $256$ for MNIST. The above parameters are set empirically, without searching for the optimal settings.

Following the common benchmarking procedures, we repeat the experimental process 10 times with independent random split of training and testing to obtain reliable results. The average of per-class recognition rates are recorded for each run. We report our final results by the mean and standard deviation of the recognition rates.
Object Recognition The Caltech-101 dataset contains 101 classes (including animals, vehicles, flowers, etc.) with high shape variability. Figure 2.2 shows some example images of the dataset. The number of images per category varies from 31 to 800. Most images have medium resolutions, i.e., about $300 \times 300$ pixels. We followed the common experiment setup for Caltech-101, training on 15 and 30 images per category and testing on the rest. Detailed comparison results are shown in Table 2.1. As shown, our sparse coding scheme outperforms linear SPM by more than 14 percent, and even outperform the nonlinear SPM [38] by a large margin (about 11% for 15 training and 9% for 30 training per category). One work that needs to be mentioned is the Kernel Codebooks [40], where the authors assigned each descriptor into multiple bins instead of using hard assignment. This scheme generally improves their baseline SPM by $5 \sim 6$ percent.\(^2\) However, their method is still based on nonlinear kernels.

Table 2.1: Classification rate (%) comparison on Caltech-101.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>15 training</th>
<th>30 training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al. [37]</td>
<td>59.10 ± 0.60</td>
<td>66.20 ± 0.50</td>
</tr>
<tr>
<td>KSPM [38]</td>
<td>56.40</td>
<td>64.40 ± 0.80</td>
</tr>
<tr>
<td>NBNN [39]</td>
<td>65.00 ± 1.14</td>
<td>70.40</td>
</tr>
<tr>
<td>ML+CORR [85]</td>
<td>61.00</td>
<td>69.60</td>
</tr>
<tr>
<td>KC [40]</td>
<td>–</td>
<td>64.14 ± 1.18</td>
</tr>
<tr>
<td>KSPM</td>
<td>56.44 ± 0.78</td>
<td>63.99 ± 0.88</td>
</tr>
<tr>
<td>LSPM</td>
<td>53.23 ± 0.65</td>
<td>58.81 ± 1.51</td>
</tr>
<tr>
<td>ScSPM</td>
<td>67.0 ± 0.45</td>
<td>73.2 ± 0.54</td>
</tr>
</tbody>
</table>

The Caltech-256 dataset holds 29,780 images falling into 256 categories with much higher intra-class variability and higher object location variability compared

\(^2\)Because the codebook baseline scores are lower, the improved absolute performance obtained by the kernel codebooks is not as high as may be obtained with a better baseline.
with Caltech-101. Each category contains at least 80 images. We tried our algorithm on 15, 30, 45, and 60 training images per category. The results are shown in Table 2.2. For all the cases, our ScSPM outperforms LSPM by more than 15 percent, and outperforms our own KSPM by more than 4 percent. In the cases of 45 and 60 training images per category, KSPM was not tried due to its very high computation cost for training.

Table 2.2: Classification rate (%) comparison on Caltech-256 dataset.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>15 training</th>
<th>30 training</th>
<th>45 training</th>
<th>60 training</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSPM [79]</td>
<td>–</td>
<td>34.10</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>KC [40]</td>
<td>–</td>
<td>27.17 ± 0.46</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>KSPM</td>
<td>23.34 ± 0.42</td>
<td>29.51 ± 0.52</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>LSPM</td>
<td>13.20 ± 0.62</td>
<td>15.45 ± 0.37</td>
<td>16.37 ± 0.47</td>
<td>16.57 ± 1.01</td>
</tr>
<tr>
<td>ScSPM</td>
<td>27.73 ± 0.51</td>
<td>34.02 ± 0.35</td>
<td>37.46 ± 0.55</td>
<td>40.14 ± 0.91</td>
</tr>
</tbody>
</table>

Scena Classification  We also tried our algorithm on the 15 Scenes dataset compiled by several researchers [86, 87, 38]. This dataset contains 4485 images falling into 15 categories, with the number of images in each category ranging from 200 to 400. The 15 categories vary from living room and kitchen to street and industrial 2.3. Following the same experiment procedure of Lazebnik et al. [38], we took 100 images per class for training and used the remaining ones for testing. The detailed comparison results are shown in Table 2.3. In this experiment, our implementation of kernel SPM was not able to reproduce the results reported in [38], probably due to the SIFT descriptor extraction and normalization process. Following our own baseline, the Linear ScSPM algorithm again achieves much better performance than KSPM and KC [40].

Table 2.3: Classification rate (%) comparison on the 15 Scenes dataset.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSPM [38]</td>
<td>81.40 ± 0.50</td>
</tr>
<tr>
<td>KC [40]</td>
<td>76.67 ± 0.39</td>
</tr>
<tr>
<td>KSPM</td>
<td>76.73 ± 0.65</td>
</tr>
<tr>
<td>LSPM</td>
<td>65.32 ± 1.02</td>
</tr>
<tr>
<td>ScSPM</td>
<td>80.28 ± 0.93</td>
</tr>
</tbody>
</table>
Action Recognition in Surveillance Video  This time, we tried our algorithm on the large-scale data of 2008 TRECVID Surveillance Event Detection Evaluation, sponsored by the National Institute of Standards and Technology (NIST). The data are 100 hours of surveillance videos, 10 hours each day, from London Gatwick International Airport. NIST defined 10 classes of events to detect, and provided 50 hours of annotated videos for training, and the remaining 50 hours of video for testing. The proposed algorithm was one of the main components in a system participating in three tasks of the evaluation, i.e., detecting CellToEar, ObjectPut, and Pointing, and was among the top performers. Some sample frames of these events are shown in Figure 2.4. In addition to the event duration annotated by NIST, we manually marked the locations of persons performing the three events of interest.

The tasks are extremely challenging in two aspects: (1) The human subjects have a large degree of variation in viewpoints and appearances, and are always in highly crowded and cluttered environments; (2) The detection system has to process 9 millions of $720 \times 576$ frames; hence, the computation load is far beyond most of the research efforts known from the literature. To make the computation affordable, our system took a simple frame-based approach: we first used a human detector to detect human subjects on each frame, and then applied classifiers on each detected region to further detect the events of interest. For each of the three events, we trained a binary classifier.
Since the training videos were recorded on five different days, we used five-fold cross validation to develop and evaluate our method, where each fold corresponded to one day. In total, we got 2114, 2172, and 8725 positive examples of `CellToEar`, `ObjectPut`, and `Pointing`, respectively, and about 200,000 negative examples (only a small subset!) in the training set. Each example was a cropped image containing a detected human subject with the annotated event, resized into a $100 \times 100$ image. For each example, we extracted SIFT descriptors for every $16 \times 16$ patch on a grid of stepsize 8. The codebook sizes of both VQ and SC were set to be 256. Nonlinear SVM does not work on such a large-scale training set; therefore, we only compared the two linear methods, ScSPM and LSPM. Due to the extremely unbalanced class distribution, we used ROC curves, as well as the AUC (area under the ROC curve) scores to evaluate the performance. The average AUC results over five folds are shown in Table 2.4. Typically, the SVM training on about 200,000 examples with 5376-dimensional features was finished in 5 minutes.

### Table 2.4: AUC comparison on TRECVID 2008 surveillance video.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>CellToEar</th>
<th>ObjectPut</th>
<th>Pointing</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSPM</td>
<td>0.688</td>
<td>0.714</td>
<td>0.744</td>
</tr>
<tr>
<td>ScSPM</td>
<td>0.744</td>
<td>0.773</td>
<td>0.769</td>
</tr>
</tbody>
</table>

**Face Recognition** For face recognition, we test our algorithm on CMU PIE [81] and CMU Multi-PIE [82]. The images are normalized to $32 \times 32$ for the CMU
The CMU PIE consists of 41,368 images of 68 people, each person under 13 poses, 43 different illumination conditions, and with 4 different expressions. Figure 2.5 shows some example images from this dataset. We use the same subset of the database as in [88, 29] for fair comparison. The subset only contains five near-frontal poses (C05, C07, C09, C27, C29) and all the images are under different illuminations and expressions. Therefore, there are 170 images for each individual. A random subset of $p$ ($p = 30, 50, 70, 90, 130$) images per person are selected as the training set and the rest of the database is considered as the testing set. The classification result is shown in Table 2.5. As shown, our sparse coding based algorithm significantly outperforms S-LDA [29], which reports as a state-of-the-art performance algorithm on this database.

![Figure 2.5: Example images from CMU PIE.](image)

<table>
<thead>
<tr>
<th>Training</th>
<th>30</th>
<th>50</th>
<th>70</th>
<th>90</th>
<th>130</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>7.90</td>
<td>4.80</td>
<td>4.00</td>
<td>3.40</td>
<td>2.90</td>
</tr>
<tr>
<td>R-LDA [88]</td>
<td>5.00</td>
<td>3.90</td>
<td>3.50</td>
<td>3.20</td>
<td>3.00</td>
</tr>
<tr>
<td>S-LDA [29]</td>
<td>3.60</td>
<td>2.50</td>
<td>2.10</td>
<td>1.80</td>
<td>1.60</td>
</tr>
<tr>
<td>ScSPM</td>
<td>0.81</td>
<td>0.26</td>
<td>0.22</td>
<td>0.11</td>
<td>0.037</td>
</tr>
</tbody>
</table>

The second experiment on face recognition is conducted on the large-scale CMU Multi-PIE database [82]. The database contains 337 subjects across simultaneous variations in pose, expression, and illumination. Some example images
are shown in Figure 2.6. In order to compare with [17] fairly, we use the same experiment settings for face recognition. Of these 337 subjects, 249 subjects present in Session 1 are used as the training set. Sessions 2, 3, and 4 are used as testing. The remaining 88 subjects are considered “outliers” or invalid images in [17] for face verification. But in this work we neglect them and only care about face recognition. For the training set, [17] only included seven frontal extreme illuminations, taken with neutral expression. We use exactly the same training set. For the test set, all 20 illuminations from Sessions 2 to 4 are used, which were recorded at distinct times over a period of several months. The dataset is challenging due to the large number of subjects and natural variations in subject appearance over time.

![Example images from CMU Multi-PIE.](image)

Figure 2.6: Example images from CMU Multi-PIE.

### Table 2.6: Face recognition error (%) on large-scale Multi-PIE.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Session 2</th>
<th>Session 3</th>
<th>Session 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>50.6</td>
<td>55.7</td>
<td>52.1</td>
</tr>
<tr>
<td>NN</td>
<td>32.7</td>
<td>33.8</td>
<td>37.2</td>
</tr>
<tr>
<td>NS</td>
<td>22.4</td>
<td>25.7</td>
<td>26.6</td>
</tr>
<tr>
<td>SR</td>
<td>8.6</td>
<td>9.7</td>
<td>9.8</td>
</tr>
<tr>
<td>ScSPM</td>
<td>5.4</td>
<td>9.0</td>
<td>7.5</td>
</tr>
</tbody>
</table>

Table 2.6 shows our results compared with those reported in the [17] for Linear Discriminant Analysis (LDA)[33], Nearest Neighbor (NN), Nearest Subspace (NS)[89], and Sparse Representation (SR). LDA, NN, and NS are used as the baseline algorithms in [17]. The SR algorithm, unifying face alignment and face recognition in the same framework, performs much better compared to those baseline algorithms, reporting the top classification accuracy on this dataset. To
compare with the SR algorithm, we make two noteworthy comments:

1. The linear combination model of SR is known to be good at handling illuminations. The training set is chosen to minimize its size.

2. The SR algorithm directly models the sparse representation as the classifier, which is highly nonlinear. Our model simply uses a linear SVM trained by \textit{one-vs.-all}, dividing the feature space into 249 parts.

And yet, our sparse coding based strategy remarkably reduces the error rates compared with SR.

\textbf{Gender Recognition}  Our gender recognition experiment is conducted on the FRGC 2.0 dataset \cite{83}. This dataset contains 568 individuals, totalling 14714 face images under various lighting conditions and backgrounds. Besides person identities, each image is annotated with gender and ethnicity. For gender recognition, we fix 114 persons’ 3014 images (randomly chosen) as the test set, and the remaining 451 individuals’ 11700 images as our training images. Comparisons are performed with the state-of-the-art algorithms on FRGC in the same experiment setting as reported in Table 2.7. Note that Convolutional Neural Network (CNN) is very complicated to train and is known to perform well on such tasks.

Table 2.7: Classification error (%) comparison with state-of-the-art gender recognition algorithms in the literature on FRGC.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (RBF)</td>
<td>8.6</td>
</tr>
<tr>
<td>CNN \cite{90}</td>
<td>5.9</td>
</tr>
<tr>
<td>ScSPM</td>
<td>6.8</td>
</tr>
</tbody>
</table>

\textbf{Handwritten Digit Recognition}  We also test our algorithm on the benchmark MNIST handwritten digit image dataset \cite{84}. The database consists of 70,000 handwritten digits, of which 60,000 digits are modeled as training and 10,000 as testing. The digits have been size-normalized and centered in a fixed-size image. Figure 2.7 shows some example images of the database. The supervised training optimization process converges quickly and we stop at five iterations. Table 2.8 shows the performance comparisons with other methods reported on the dataset.
“L1 sparse coding” and “Local coordinate coding” methods denote the holistic sparse coding scheme on the entire image with trained compact dictionaries, with the latter enforcing locality constraints. Our patch-based hierarchical model performs much better than the above holistical methods.

Table 2.8: Classification error (%) comparison with state-of-the-art algorithms in the literature on MNIST.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (RBF)</td>
<td>1.41</td>
</tr>
<tr>
<td>L1 sparse coding (linear SVM)</td>
<td>2.02</td>
</tr>
<tr>
<td>Local coordinate coding (linear SVM) [69]</td>
<td>1.90</td>
</tr>
<tr>
<td>Deep belief network</td>
<td>1.20</td>
</tr>
<tr>
<td>CNN [90]</td>
<td>0.82</td>
</tr>
<tr>
<td>ScSPM (linear SVM)</td>
<td>0.98</td>
</tr>
</tbody>
</table>

2.3 Efficient Highly Over-Complete Sparse Coding Using a Mixture Model

The previous hierarchical max pooling model based on sparse coding achieves very promising results on several benchmark datasets. Empirical studies show that using a larger dictionary for sparse coding to map the data into a higher-dimensional space will generate superior classification performance. However, the computation of both training and testing for the hierarchical model can be prohibitively heavy if the dictionary is highly over-complete. Although a non-linear regressor can be applied for fast inference [91], the dictionary training is
Figure 2.8: A simplified schematic illustration of the image encoding process using the mixture sparse coding scheme. (a) local descriptor extraction; (b) mixture modeling in the descriptor space; (c) sparse coding and feature pooling. Within each mixture, a small dictionary for sparse coding can be applied, thus speeding up the coding process.

still computationally challenging. Motivated by the LCC work in [69] that suggested sparse coding should be local with respect to the dictionary, we propose an efficient sparse coding scheme with highly over-complete dictionaries using a mixture model. The model is derived via a variational approach, and the coding speed can be improved approximately at the rate of the number of mixtures. Figure 2.8 illustrates the simplified version of the image encoding process. Our mixture model allows a much smaller dictionary to describe well the descriptors in each mixture, and thus the sparse coding computation can be effectively boosted.

To make a concrete argument, we show the ScSPM computation time for encoding one image as well as the performance (using the average precision measure (AP)) for dictionaries of different sizes on the PASCAL VOC 2007 dataset [92]. Thirty thousand local descriptors are extracted from the image. As shown, the performance keeps increasing as the dictionary size increases, while the computation time also increases, approximately linearly. In our experiment, training a dictionary beyond size 8192 is almost infeasible. The local coordinate coding (LCC) work [69] suggests that the sparse coding should be local and the bases far away from the current encoding point can be discarded. This motivates our local sparse coding scheme induced by a mixture model, where local sparse coding within each mixture can be very fast (refer to Figure 2.8). For comparison, using 1024 mixtures with dictionary size 256 for each mixture, the effective dictionary size is $1024 \times 256 = 262,144$, and our proposed approach can process one image in about one minute.

The proposed approach extends our previous sparse coding based model by using a mixture of small over-complete dictionaries to model the descriptor space. A variational approach is applied to learn the model parameters.
Table 2.9: The relationships between the dictionary size and the computation time as well as the performance on the PASCAL VOC 2007 validation dataset. The computation time reported is an approximate time needed for encoding one image (in minutes).

<table>
<thead>
<tr>
<th>Dictionary Size</th>
<th>512</th>
<th>2048</th>
<th>8192</th>
<th>32,768</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation Time</td>
<td>1.5 min</td>
<td>3.5 min</td>
<td>14 min</td>
<td>N/A</td>
</tr>
<tr>
<td>Performance</td>
<td>45.3%</td>
<td>50.2%</td>
<td>53.2%</td>
<td>N/A</td>
</tr>
</tbody>
</table>

2.3.1 The Model

We describe the image local descriptor space using a $K$-mixture model, where the local distribution of each mixture is further governed by an over-complete dictionary. Let $X = \{x_n\}_{n=1}^N$ be the $N$ independent and identically distributed observation points, and $z = \{z_n\}_{n=1}^N$ be the corresponding $N$ hidden variables, where $z_n \in \{1, 2, ..., K\}$ is a random variable indicating the mixture assignments. Denote the mixture model parameters as $\Theta = \{B, w\}$, where $B = \{B_k\}_{k=1}^K$ is the set of over-complete dictionaries, where $B_k \in \mathbb{R}^{d \times D}$, and $w = \{w_k\}_{k=1}^K$ is the set of prior weights for the mixtures. We desire to learn the model by maximizing the likelihood

$$P(X|\Theta) = \prod_{n=1}^N P(x_n|\Theta) = \prod_{n=1}^N \sum_{z_n=1}^K w_{z_n} p(x_n|B_{z_n})$$

(2.10)

where we let

$$p(x_n|B_{z_n}) = \int p(x_n|B_{z_n}, \alpha_{z_n})p(\alpha_{z_n}^n|\sigma)d\alpha_{z_n}$$

(2.11)

be the marginal distribution of a latent-variable model with a Laplacian prior $p(\alpha_{z_n}^n|\sigma)$ on the latent variable $\alpha_{z_n}^n$, and $p(x_n|B_{z_n}, \alpha_{z_n}^n)$ is modeled as a zero-mean isotropic Gaussian distribution regarding the representation error $x_n - B_{z_n} \alpha_{z_n}^n$.

Learning the above model requires to compute the posterior $P(z|X, \Theta)$. However, under this model, this distribution is infeasible to be computed in a close form. Note that approximation can be used for the marginal distribution $p(x_n|B_{z_n})$ (introduced later in Equation (2.15)) in order to compute the posterior. This requires evaluating the mode of the posterior distribution of the latent variable for each data point, which, however, is computationally too slow. We thus develop a
fast variational approach, where the posterior \( p(z_n | x_n, \Theta) \) is approximated by

\[
q(z_n = k | x_n, \Lambda) = \frac{x_n^T A_k x_n + b_k^T x_n + c_k}{\sum_{k'} x_n^T A_{k'} x_n + b_{k'}^T x_n + c_{k'}}
\] (2.12)

where \( \Lambda = \{ (A_k, b_k, c_k) \} \), \( A_k \) is a positive definite matrix, \( b_k \) is a vector, and \( c_k \) is a scalar. For computational convenience, we assume \( A_k \) to be diagonal. \( \Lambda \) is a set of free parameters, determining the mixture partition in the descriptor space.

Then the learning problem can be formulated as

\[
\min_{\Theta, \Lambda} \sum_{n=1}^{N} \sum_{z_n=1}^{K} \left[ -q(z_n | x_n, \Lambda) \log p(x_n, z_n | \Theta) + q(z_n | x_n, \Lambda) \log q(z_n | x_n, \Lambda) \right]
\] (2.13)

which minimizes an upper bound of the negative log-likelihood \( -\sum_{i=1}^{N} \log p(x_i | \Theta) \) of the model [93].

2.3.2 The Learning Algorithm

The learning problem in Equation (2.13) can be cast into a standard variational EM algorithm, where we optimize \( \Lambda \) to push down the upper bound to approximate the negative log-likelihood at E-step, and then update \( \Theta \) in the M-step to maximize the approximated likelihood. Let the first term in the object be formulated into

\[
\sum_{n=1}^{N} \sum_{z_n=1}^{K} g(z_n | x_n, \Lambda) \log p(x_n, z_n | \Theta)
\] (2.14)

Note that the marginal distribution \( p(x_n | B_{z_n}) \) is difficult to evaluate due to the integration. We then simplify it by using the mode of the posterior distribution of \( \alpha_n \):

\[
- \log p(x_n | B_{z_n}) \approx \min_{\alpha_n} \left\{ -\log p(x_n | B_{z_n}, \alpha_n) - \log p(\alpha_n | \sigma) \right\}
\]

\[
= \min_{\alpha_n} \| x_n - B_{z_n} \alpha_n \|^2 + \lambda \| \alpha_n \|_1
\] (2.15)

which turns the integration into a standard sparse coding (or Lasso) problem. We then have the following updates rules for learning the model:
1. Optimize $\Lambda$

$$
\min_{\Lambda} \sum_{n=1}^{N} \sum_{z_n=1}^{K} \left\{ q(z_n|x_n, \Lambda) \left[ -\log p(x_n|B_{z_n}) - \log w_{z_n} + \log q(z_n|x_n, \Lambda) \right] \right\}
$$

(2.16)

2. Optimize $B$

$$
\min_{B} - \sum_{n=1}^{N} \sum_{z_n=1}^{K} q(z_n|x_n, \Lambda) \log p(x_i|B_{z_n})
$$

(2.17)

where each column of the dictionaries $\{B_k\}_{k=1}^{K}$ is constrained to be of unit $\ell_2$-norm. The optimization is a quadratically constrained quadratic programming problem.

3. Optimize $w$

$$
\min_{w} - \sum_{n=1}^{N} \sum_{z_n=1}^{K} q(z_n|x_n, \Lambda) \log w_{z_n}
$$

s.t. $$\sum_{z_n=1}^{K} w_{z_n} = 1
$$

(2.18)

which always leads to $w_{z_n} = \frac{1}{N} \sum_{n=1}^{N} q(z_n|x_n, \Lambda)$ using the Lagrange multiplier.

By alternatively optimizing over $\Lambda$, $B$, and $w$, we are guaranteed to find a local minimum for the problem in Equation (2.13). Note that $B = [B_1, B_2, ..., B_K] \in R^{d \times KD}$ is the effective highly over-complete dictionary ($KD \gg d$) to learn for sparse coding. The above mixture sparse coding model leverages the learning complexity by training $B_k (k = 1, 2, ..., K)$ separately and independently in Step 2 given the posteriors from Step 1. On the other hand, since we specify all the mixture dictionaries $B_k$ to be of the same size, their fitting abilities for each data mixture will affect the mixture model parameters in Step 1, and thus the mixture weights in Step 3. Therefore, the above training procedure will efficiently learn the highly over-complete dictionary $B$, while ensuring that the mixture dictionaries can fit each data mixture equally well.
2.3.3 Practical Implementation

The above iterative optimization procedures can be very fast with proper initialization for $\Lambda$, $B$, and $w$. We propose to initialize the model parameters by the following:

1. Initialize $\Lambda$ and $w$: fit the data $X$ into a Gaussian Mixture Model (GMM) with $K$ mixtures. The covariance matrix of each mixture is constrained to be diagonal for computational convenience.

$$ p(X|\nu, \Sigma, w) = \prod_{n=1}^{N} \sum_{k=1}^{K} v_k N(x_n | \mu_k, \Sigma_k). \tag{2.19} $$

The above Gaussian Mixture Model can be trained with the standard EM algorithm. Initialize $A_k$, $b_k$, $c_k$, and $w_k$ with $\Sigma_k^{-1}$, $-2\Sigma_k^{-1} \mu_k$, $\mu_k^T \Sigma_k^{-1} \mu_k$, and $v_k$, respectively.

2. Initialize $B$: Sample the data $X$ into $K$ clusters $\{X_k\}_{k=1}^{K}$, according to the posteriors of the data points calculated from the above GMM. Train the corresponding over-complete dictionaries $\{B_k^0\}_{k=1}^{K}$ for those clusters using standard sparse coding. Initialize $B$ with this trained set of dictionaries.

In practice, we can further simplify the above optimization procedure by updating the model parameters in just one round. We first fit the data into a Gaussian Mixture Model, constraining the mixtures to have equal weights. Then we sample the data for each mixture according to their posterior probabilities and train the dictionaries separately for each mixture. Since the equally weighted mixtures partition the descriptors into clusters of approximately the same size, the mixture dictionaries $\{B_k^0\}_{k=1}^{K}$ can thus roughly model them equally well (the mixture dictionaries have the same size $K$).

2.3.4 Image Coding

Similarly to the case of the previous ScSPM algorithm, suppose we have a set of local descriptors $S = [x_1, x_2, \ldots, x_S]$ extracted from an image or its sub-region, the set-level feature is defined on the laten variables (sparse codes) $\{\alpha_n^{z_n}\}$. Specifically, the local descriptors are first assigned to multiple mixtures according to the
posteriori, and then the sparse codes are extracted with the corresponding dictionaries. We pool these sparse codes using a weighted average within each mixture and stack them into a super-vector:

\[
fs = [\sqrt{w_1\mu_{1}\alpha}; \sqrt{w_2\mu_{2}\alpha}; \ldots; \sqrt{w_K\mu_{K}\alpha}] \tag{2.20}
\]

where

\[
\mu_{k}\alpha = \frac{\sum_{n=1}^{N} q(z_n = k|x_n, \Lambda)\alpha^n}{\sum_{n=1}^{N} q(z_n = k|x_n, \Lambda)} \tag{2.21}
\]

is the weighted average of the sparse codes with their posteriors for the \(k\)-th mixture. The super-vector feature representation Equation (2.20) has several characteristics that are not immediately obvious:

- The feature constructed in Equation (2.20) is based on the locally linear model assumption, and thus is well fitted to linear kernels.
- The square root operator on each weight \(w_k\) corresponds to the linearity of the feature.
- In practice, the posteriors \(\{p(z_n = k|x_n, \Lambda)\}_{k=1}^{K}\) are very sparse, \(i.e.,\) each data point will be assigned to only one or two mixtures. Therefore, Equation (2.21) is very fast to evaluate.
- The effective dictionary size of the sparse coding is \(K \times D\). However, in our mixture sparse coding model, the nonlinear coding only involves dictionaries of size \(D\), improving the computation approximately \(K\) times (typically we choose \(K \geq 1024\)).
- Weighted average pooling instead of max pooling (Equation (2.7)) is used due to the locality of the sparse codes.\(^3\)

Again, to incorporate the spatial information, we make use of the philosophy of the spatial pyramid [38] to divide the image into multiple sub-regions over different configurations. The final image feature is then built by concatenating all the super-vectors extracted from these spatial sub-regions.

\(^3\)Actually, weighted average pooling the non-zero sparse codes better fits the theory. The results we report here are based on simple weighted average.
2.3.5 Experiment Results

We evaluate the proposed model on the PASCAL Visual Object Classes Challenge (VOC) datasets. The goal of this challenge is to recognize objects from a number of visual object classes in realistic scenes (i.e., not pre-segmented objects). It is fundamentally a supervised learning problem in that a training set of labeled images is provided. In total, there are 20 object classes collected:

- **Person**: person
- **Animal**: bird, cat, cow, dog, horse, and sheep
- **Vehicle**: aeroplane, bicycle, boat, bus, car, motorbike, and train
- **Indoor**: bottle, chair, dining table, potted plant, sofa, and tv/monitor

Two main competitions for the PASCAL VOC challenge are organized:

- **Classification**: for each of the 20 classes, predicting presence/absence of an example of that class in the test image.
- **Detection**: predicting the bounding box and label of each object from the 20 target classes in the test image.

In this section, we apply our model for the classification task to both PASCAL VOC Challenge 2007 and 2009 datasets for convenient comparison.

The PASCAL VOC 2007 dataset [92] consists of 9,963 images and PASCAL VOC 2009 [94] collects even more, 14,743 images in total. Both datasets are split into 50% for training/validation and 50% for testing. The distributions of images and objects by class are approximately equal across the training/validation and test sets. These images range between indoor and outdoor scenes, close-ups and landscapes, and strange viewpoints. These datasets are extremely challenging because all the images are daily photos obtained from Flicker where the size, viewing angle, illumination, appearance, and poses of the objects vary significantly, with frequent occlusions. Figure 2.9 shows some example images for the 20 classes from the PASCAL VOC 2007 dataset.

The classification performance is evaluated using the average precision (AP) measure, the standard metric used by the PASCAL challenge, which computes the area under the Precision/Recall curve. The higher the score, the better the performance. In the following, we explain some implementation details of our algorithm.
Local Descriptor. In our experiments, we only use single descriptor type HoG as the local descriptors, due to its computational advantage over SIFT via integral image. These descriptors are extracted from a regular grid with step size 4 pixels on the image plane. At each location, three scales of patches are used for calculating the HoG descriptor: $16 \times 16$, $24 \times 24$, and $32 \times 32$. As a result, approximately 30,000 local descriptors are extracted for each image. We then reduce the descriptor dimension from 128 to 80 with PCA.

Mixture Modeling. For the VOC 07 dataset, $K = 1024$ mixtures are used and the size of the dictionary $D$ for each mixture is fixed to be 256. Therefore, the effective dictionary size is $1024 \times 256 = 262144$. Recall from Table 2.9 that working directly on a dictionary of this size is impossible. Using our mixture model, we only need to perform sparse coding on dictionaries of size 256, with little extra effort of computing the posterior for each descriptor, thus reducing the computation time for encoding one image below one minute. For the VOC 09 dataset, we increase the mixture number to 2048. $K$ and $D$ are chosen empirically, balancing the performance and computational complexity.

Spatial Pyramid Structure. A spatial pyramid is employed to encode the spatial information of the local descriptors. As suggested by the competition winner
system of VOC 2007 [95], we also use the same spatial pyramid shown in Figure 2.10 for both datasets. A total of 8 spatial blocks are defined, and we extract a super-vector by Equation (2.20) from each block and concatenate them with equal weights.

![Spatial pyramid structure](image)

Figure 2.10: Spatial pyramid structure used in both PASCAL VOC 2007 and 2009 datasets.

**Feature Normalization.** Since our feature is based on the linear model assumption, we use Linear Discriminant Analysis (LDA) to sphere the features, and then linear SVM or Nearest Centroid is applied for classification. In practice, we always observe some improvements from this normalization step.

We present the classification results on the two datasets in this section. The precisions for each object class and the average precision (AP) are given by comprehensive comparisons. For the VOC 2007 dataset, the results we have are obtained by training on the training set and testing on the validation set. We report our results in Table 2.10, where the results of the competition winner system of VOC 2007 [95] and a recently proposed algorithm LLC [96] on the validation set are also provided as references. As the detailed results for Winner’07 and LLC are not available, we only cite their APs. Note that the Winner’07 system uses multiple descriptors in addition to dense SIFT descriptors, and the multiple kernel weights are also optimized for best performance. The LLC algorithm, similar to our system, only employs a single kernel based on a single descriptor. In both cases, our algorithm outperforms Winner’07 and LLC by a significant margin of about 5% in terms of AP.

Table 2.11 shows our results and comparisons with the top systems in VOC 2009. In this table, we compare with Winner’09 system (from the NEC-UIUC team), and two honorable mention systems UVAS (from University of Amsterdam and University of Surrey) and CVC (from Computer Vision Centre Barcelona). The Winner’09 system obtains its results by combining the detection scores from
Table 2.10: Image classification results on the PASCAL VOC 2007 validation dataset.

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>aero</th>
<th>bicyc</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winner’07</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LLC [96]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td>78.5</td>
<td>61.6</td>
<td>53.0</td>
<td>69.8</td>
<td>31.69</td>
<td>62.2</td>
<td>81.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winner’07</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LLC [96]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td>59.3</td>
<td>50.3</td>
<td>75.4</td>
<td>72.9</td>
<td>82.1</td>
<td>26.1</td>
<td>36.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winner’07</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>54.2</td>
</tr>
<tr>
<td>LLC [96]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>55.1</td>
</tr>
<tr>
<td>Ours</td>
<td>60.5</td>
<td>55.9</td>
<td>41.8</td>
<td>55.7</td>
<td>81.6</td>
<td>56.3</td>
<td>59.6</td>
</tr>
</tbody>
</table>

an object detector. The UVAS system employs multiple kernel learning over multiple descriptors. The CVC system not only makes use of the detection results, but also unites multiple descriptors. Yet, the results of our algorithm are close to those of the Winner’09 system, and improves by a notable margin over the honorable mention systems.
Table 2.11: Image classification results on the PASCAL VOC 2009 dataset. Our results are obtained based on a single local descriptor without combining detection results.

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>aero</th>
<th>bicyc</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winner’09</td>
<td>88.0</td>
<td>68.6</td>
<td>67.9</td>
<td>72.9</td>
<td>44.2</td>
<td>79.5</td>
<td>72.5</td>
</tr>
<tr>
<td>UVAS</td>
<td>84.7</td>
<td>63.9</td>
<td>66.1</td>
<td>67.3</td>
<td>37.9</td>
<td>74.1</td>
<td>63.2</td>
</tr>
<tr>
<td>CVC</td>
<td>83.3</td>
<td>57.4</td>
<td>67.2</td>
<td>68.8</td>
<td>39.9</td>
<td>55.6</td>
<td>66.9</td>
</tr>
<tr>
<td>Ours</td>
<td>87.7</td>
<td>67.8</td>
<td>68.1</td>
<td>71.1</td>
<td>39.1</td>
<td>78.5</td>
<td>70.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winner’09</td>
<td>57.5</td>
<td>59.0</td>
<td>72.6</td>
<td>72.3</td>
<td>85.3</td>
<td>36.6</td>
<td>56.9</td>
</tr>
<tr>
<td>UVAS</td>
<td>54.7</td>
<td>53.5</td>
<td>68.1</td>
<td>70.6</td>
<td>85.2</td>
<td>38.5</td>
<td>47.2</td>
</tr>
<tr>
<td>CVC</td>
<td>47.2</td>
<td>47.3</td>
<td>67.7</td>
<td>66.8</td>
<td>88.8</td>
<td>40.2</td>
<td>46.6</td>
</tr>
<tr>
<td>Ours</td>
<td>53.3</td>
<td>59.2</td>
<td>71.6</td>
<td>70.6</td>
<td>84.0</td>
<td>30.9</td>
<td>51.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Winner’09</td>
<td>70.8</td>
<td>59.5</td>
<td>53.6</td>
<td>57.9</td>
<td>85.9</td>
<td>68.0</td>
<td>66.5</td>
</tr>
<tr>
<td>UVAS</td>
<td>64.0</td>
<td>57.1</td>
<td>46.2</td>
<td>49.3</td>
<td>83.2</td>
<td>68.1</td>
<td>62.1</td>
</tr>
<tr>
<td>CVC</td>
<td>63.7</td>
<td>50.8</td>
<td>34.9</td>
<td>49.4</td>
<td>79.4</td>
<td>71.5</td>
<td>59.7</td>
</tr>
<tr>
<td>Ours</td>
<td>70.7</td>
<td>57.4</td>
<td>51.7</td>
<td>55.9</td>
<td>85.9</td>
<td>66.7</td>
<td>64.6</td>
</tr>
</tbody>
</table>

2.4 Conclusion

In this chapter, we propose a novel generic image feature representation for image classification by max pooling over sparse codes of the local descriptors from different spatial blocks across multiple spatial scales. We first train an over-complete dictionary for the local descriptors, based on which we transform each local descriptor into a sparse code via $\ell_1$-norm minimization. Due to the sparsity nature of these sparse codes, max pooling is performed to extract the robust image statistics information for image recognition. Furthermore, max pooling is applied over increasingly larger spatial regions to achieve different levels of translation-invariance. We apply this generic feature representation to various image classification tasks, including object recognition, scene classification, action recognition, face recognition, gender recognition, and handwritten digit recognition. On all these tasks, our feature representation can achieve state-of-the-art performance on the corresponding benchmarks datasets with simple linear SVM.
CHAPTER 3

SINGLE IMAGE SUPER-RESOLUTION VIA SPARSE REPRESENTATION

3.1 Introduction

Super-resolution (SR) image reconstruction is currently a very active area of research, as it offers the promise of overcoming some of the inherent resolution limitations of the low-cost imaging sensors (e.g., cell phone or surveillance cameras), and allows better utilization of the growing capability of high-resolution displays (e.g., high-definition LCDs). Such resolution-enhancing technology may also prove to be essential in medical imaging and satellite imaging where diagnosis or analysis from low-quality images can be extremely difficult. Conventional approaches to generating a super-resolution image normally require multiple low-resolution images of the same scene, which are aligned with sub-pixel accuracy. The SR task is cast as the inverse problem of recovering the original high-resolution image by fusing the low-resolution images, based on reasonable assumptions or prior knowledge about the observation model that maps the high-resolution image to the low-resolution ones. The fundamental reconstruction constraint for SR is that the recovered image, after applying the same generation model, should reproduce the observed low-resolution images. However, SR image reconstruction is generally a severely ill-posed problem because of the insufficient number of low-resolution images, ill-conditioned registration, and unknown blurring operators, and therefore, the solution from the reconstruction constraint is not unique. Various regularization methods have been proposed to further stabilize the inversion of this ill-posed problem, such as [97, 98, 99].

However, the performance of these reconstruction-based super-resolution algorithms degrades rapidly when the desired magnification factor is large or the number of available input images is small. In these cases, the result may be overly smooth, lacking important high-frequency details [100]. Another class of SR approach is based on interpolation [101, 102, 103]. While simple inter-
polation methods such as bilinear or bicubic interpolation tend to generate overly smooth images with ringing and jagged artifacts, interpolation by exploiting the natural image priors will generally produce more favorable results. Dai et al. [102] represented the local image patches using the background and foreground descriptors, and reconstructed the sharp discontinuity between the two. Sun et al. [103] explored the gradient profile prior for local image structures and applied it to super-resolution. Such approaches are effective in preserving the edges in the zoomed image. However, they are limited in modeling the visual complexity of the real images. For natural images with fine textures or smooth shading, these approaches tend to produce watercolor-like artifacts.

A third category of SR approach is based on machine learning techniques, which attempt to capture the co-occurrence prior between low-resolution and high-resolution image patches. Freeman et al. [50] proposed an example-based learning strategy that applies to generic images where the low-resolution to high-resolution prediction is learned via a Markov Random Field (MRF) solved by belief propagation. Sun et al. [52] extended this approach by using the Primal Sketch priors to enhance blurred edges, ridges and corners. Nevertheless, the above methods typically require enormous databases of millions of high-resolution and low-resolution patch pairs, and are therefore computationally intensive. Chang et al. [53] adopt the philosophy of Locally Linear Embedding (LLE) [56] from manifold learning, assuming similarity between the two manifolds in the high- and low-resolution patch spaces. Their algorithm maps the local geometry of the low-resolution patch space to the high-resolution one, generating the high-resolution patch as a linear combination of its neighbors. Using this strategy, more patch patterns can be represented using a smaller training database. However, using a fixed number $K$ neighbors for reconstruction often results in blurring effects due to over- or under-fitting. In our previous work [68], we proposed a method for adaptively choosing the most relevant reconstruction neighbors based on sparse coding, avoiding the over- or under-fitting problems of [53] and producing superior results. However, directly seeking the sparse representation over a large randomly sampled image patch database is too time-consuming.

While the approaches mentioned above were proposed for generic image super-resolution, specific image priors can be incorporated when tailored to SR applications for specific domains such as human faces. This face hallucination problem was addressed in the pioneering work of Baker and Kanade [104]. However, their gradient pyramid-based prediction does not directly model the face prior, and the
pixels are predicted individually, causing discontinuities and artifacts. Liu et al. [105] proposed a two-step statistical approach integrating the global PCA model and a local patch model. Although the algorithm can generate good results, the holistic PCA model tends to yield results that look like the average face and the probabilistic local patch model is complicated and computationally demanding. Liu et al. [106] proposed a new approach based on Tensor Patches and residue compensation. While this algorithm adds more details to the face, it also introduces more artifacts.

This chapter focuses on the problem of recovering the super-resolution version of the given single low-resolution image. Similarly to the aforementioned learning-based methods, we will rely on patches from the input image. However, instead of working directly with the image patch pairs sampled from high- and low-resolution images [68], we learn a compact representation for these patch pairs to capture the co-occurrence prior, which significantly improves the speed of the algorithm. Our approach is motivated by recent results in sparse signal representation, which suggest that the linear relationships among high-resolution signals can be accurately recovered from their low-dimensional projections [4, 5]. Although the super-resolution problem is very ill-posed, making precise recovery impossible, the image patch sparse representation demonstrates both effectiveness and robustness in regularizing this inverse problem. To be more precise, let $D \in \mathbb{R}^{n \times K}$ be an over-complete dictionary of $K$ atoms ($K > n$), and suppose a high-resolution signal $x \in \mathbb{R}^n$ can be represented as a sparse linear combination with respect to $D$. That is, the signal $x$ can be written as $x = D\alpha_0$ where where $\alpha_0 \in \mathbb{R}^K$ is a vector with very few ($\ll n$) nonzero entries. In practice, we might only observe a small set of measurements $y$ of $x$:

$$y = Lx = LD\alpha_0, \quad (3.1)$$

where $L \in \mathbb{R}^{k \times n}$ with $k < n$ is a projection matrix. In our super-resolution context, $x$ is a high-resolution image (patch), while $y$ is its low-resolution counterpart (or features extracted from it). If the dictionary $D$ is over-complete, the equation $x = D\alpha$ is underdetermined for the unknown coefficients $\alpha$. The equation $y = LD\alpha$ is even more dramatically underdetermined. Nevertheless, under mild conditions, the sparsest solution $\alpha_0$ to this equation will be unique. Furthermore, if $D$ satisfies an appropriate near-isometry condition, then for a wide variety of matrices $L$, any sufficiently sparse linear representation of a high-resolution
Figure 3.1: Reconstruction of a feline face with magnification factor 2. Left: result by our method. Right: the original image. There is little noticeable difference visually even for such a complicated texture. The RMSE for the reconstructed image is 5.92 (only the local patch model is employed).

image patch $x$ in terms of the $D$ can be recovered (almost) perfectly from the low-resolution image patch $[5, 107]$.\(^1\) Figure 3.1 shows an example that demonstrates the capabilities of our method derived from this principle. The image of the feline face is blurred and downsampled to half of its original size in both dimensions. Then we zoom the low-resolution image to the original size using the proposed method. Even for such a complicated texture, sparse representation recovers a visually appealing reconstruction of the original signal.

Recently, sparse representation has been successfully applied to many other related inverse problems in image processing, such as denoising [10] and restoration [11], often improving on the state-of-the-art. For example in [10], the authors use the K-SVD algorithm [12] to learn an over-complete dictionary from natural image patches and successfully apply it to the image denoising problem. In our setting, we do not directly compute the sparse representation of the high-resolution patch. Instead, we will work with two coupled dictionaries, $D_h$ for high-resolution patches, and $D_l$ for low-resolution ones. The sparse representation of a low-resolution patch in terms of $D_l$ will be directly used to recover the corresponding high-resolution patch from $D_h$. We obtain a locally consistent solution by allowing patches to overlap and demanding that the reconstructed high-resolution patches agree on the overlapped areas. In this chapter, we try to learn the two over-complete dictionaries in a probabilistic model similar to [75].

\(^1\)Even though the structured projection matrix defined by blurring and downsampling in our SR context does not guarantee exact recovery of $\alpha_0$, empirical experiments indeed demonstrate the effectiveness of such a sparse prior for our SR tasks.
To enforce that the image patch pairs have the same sparse representations with respect to $D_h$ and $D_l$, we learn the two dictionaries simultaneously by concatenating them with proper normalization. The learned compact dictionaries will be applied to both generic image super-resolution and face hallucination to demonstrate their effectiveness.

Compared with the aforementioned learning-based methods, our algorithm requires only two compact learned dictionaries, instead of a large training patch database. The computation, mainly based on linear programming or convex optimization, is much more efficient and scalable, compared with [50, 52, 53]. The online recovery of the sparse representation uses the low-resolution dictionary only—the high-resolution dictionary is used to calculate the final high-resolution image. The computed sparse representation adaptively selects the most relevant patch bases in the dictionary to best represent each patch of the given low-resolution image. This leads to superior performance, both qualitatively and quantitatively, compared to the method described in [53], which uses a fixed number of nearest neighbors, generating sharper edges and clearer textures. In addition, the sparse representation is robust to noise as suggested in [10], and thus our algorithm is more robust to noise in the test image, while most other methods cannot perform denoising and super-resolution simultaneously.

The remainder of this chapter is organized as follows. Section 3.2 details our formulation and solution to the image super-resolution problem based on sparse representation. Specifically, we study how to apply sparse representation for both generic image super-resolution and face hallucination. In Section 3.3, we discuss how to learn the two dictionaries for the high- and low-resolution image patches, respectively. Various experimental results in Section 2.2.5 demonstrate the efficacy of sparsity as a prior for regularizing image super-resolution.

3.2 Image Super-Resolution from Sparsity

The single-image super-resolution problem asks: given a low-resolution image $Y$, recover a higher-resolution image $X$ of the same scene. Two constraints are modeled in this work to solve this ill-posed problem: (1) reconstruction constraint, which requires that the recovered $X$ should be consistent with the input $Y$ with respect to the image observation model; and (2) sparsity prior, which assumes that the high-resolution patches can be sparsely represented in an appropriately chosen
over-complete dictionary, and that their sparse representations can be recovered from the low-resolution observation.

**Reconstruction Constraint**  The observed low-resolution image $Y$ is a blurred and downsampled version of the high-resolution image $X$:

$$Y = SHX$$  \hspace{1cm} (3.2)

Here, $H$ represents a blurring filter, and $S$ the downsampling operator.

Super-resolution remains extremely ill-posed, since for a given low-resolution input $Y$, infinitely many high-resolution images $X$ satisfy the above reconstruction constraint. We further regularize the problem via the following prior on small patches $x$ of $X$:

**Sparsity Prior**  The patches $x$ of the high-resolution image $X$ can be represented as a sparse linear combination in a dictionary $D_h$ trained from high-resolution patches sampled from training images:

$$x \approx D_h\alpha$$  \hspace{1cm} (3.3)

The sparse representation $\alpha$ will be recovered by representing patches $y$ of the input image $Y$, with respect to a low-resolution dictionary $D_l$ co-trained with $D_h$. The dictionary training process will be discussed in Section 3.3.

We apply our approach to both generic images and face images. For generic image super-resolution, we divide the problem into two steps. First, as suggested by the sparsity prior in Equation (3.3), we find the sparse representation for each local patch, respecting spatial compatibility between neighbors. Next, using the result from this local sparse representation, we further regularize and refine the entire image using the reconstruction constraint in Equation (3.2). In this strategy, a local model from the sparsity prior is used to recover lost high-frequency components for image details. The global model from the reconstruction constraint is then applied to remove possible artifacts from the first step and make the image more consistent and natural. The face images differ from the generic images in that the face images have more regular structures and thus reconstruction constraints in the face subspace can be more effective. For face image super-resolution, we reverse the above two steps to make better use of the global face
structure as a regularizer. We first find a suitable subspace for human faces and apply the reconstruction constraints to recover a medium-resolution face image. We then recover the local details using the sparsity prior for image patches.

3.2.1 Generic Image Super-Resolution from Sparsity

Local Model from Sparse Representation Similarly to the patch-based methods mentioned previously, our algorithm tries to infer the high-resolution image patch for each low-resolution image patch from the input. For this local model, we have two dictionaries $D_h$ and $D_l$, which are trained to have the same sparse representations for each high- and low-resolution image patch pair. We subtract the mean pixel value for each patch, so that the dictionary represents image textures rather than the absolute intensities. In the recovery process, the mean value for each high-resolution image patch is then predicted by its low-resolution version.

For each input low-resolution patch $y$, we find a sparse representation with respect to $D_l$. The corresponding high-resolution patch bases $D_h$ will be combined according to these coefficients to generate the output high-resolution patch $x$. The problem of finding the sparsest representation of $y$ can be formulated as

$$\min \|\alpha\|_0 \quad \text{s.t.} \quad \|F D_l \alpha - F y\|_2^2 \leq \epsilon,$$

(3.4)

where $F$ is a (linear) feature extraction operator. The main role of $F$ in Equation (3.4) is to provide a perceptually meaningful constraint.\footnote{Traditionally, one would seek the sparsest $\alpha \text{ s.t. } \|D_l \alpha - y\|_2 \leq \epsilon$. For super-resolution, it is more appropriate to replace this 2-norm with a quadratic norm $\|\cdot\|_{F^T F}$ that penalizes visually salient high-frequency errors.} on how closely the coefficients $\alpha$ must approximate $y$. We will discuss the choice of $F$ in Section 3.3.

Although the optimization problem in Equation (3.4) is NP-hard in general, recent results [108, 109] suggest that as long as the desired coefficients $\alpha$ are sufficiently sparse, they can be efficiently recovered by instead minimizing the $\ell_1$-norm\footnote{There are also some recent works showing certain non-convex optimization problems can produce superior sparse solutions to the $\ell_1$-norm convex problem, e.g., [110] and [111].} as follows:

$$\min \|\alpha\|_1 \quad \text{s.t.} \quad \|F D_l \alpha - F y\|_2^2 \leq \epsilon.$$  

(3.5)
The Lagrange multiplier offers an equivalent formulation

$$\min_{\alpha} \|FD_l\alpha - Fy\|_2^2 + \lambda\|\alpha\|_1, \tag{3.6}$$

where the parameter $\lambda$ balances sparsity of the solution and fidelity of the approximation to $y$. Notice that this is essentially a linear regression regularized with $\ell_1$-norm on the coefficients, known in the statistical literature as the Lasso problem [76].

Solving Equation (3.6) individually for each local patch does not guarantee the compatibility between adjacent patches. We enforce compatibility between adjacent patches using a one-pass algorithm similar to that of [51].\footnote{There are different ways to enforce compatibility. In [53], the values in the overlapped regions are simply averaged, which will result in blurring effects. The greedy one-pass algorithm [51] is shown to work almost as well as the use of a full MRF model [50]. Our algorithm, not based on the MRF model, is essentially the same by trusting partially the previously recovered high-resolution image patches in the overlapped regions.} The patches are processed in a raster-scan order in the image, from left to right and top to bottom. We modify Equation (3.5) so that the super-resolution reconstruction $D_h\alpha$ of patch $y$ is constrained to closely agree with the previously computed adjacent high-resolution patches. The resulting optimization problem is

$$\min_{\alpha} \|\alpha\|_1 \quad \text{s.t.} \quad \|FD_l\alpha - Fy\|_2^2 \leq \epsilon_1, \quad \|PD_h\alpha - w\|_2^2 \leq \epsilon_2, \tag{3.7}$$

where the matrix $P$ extracts the overlapping region between the current target patch and the previously reconstructed high-resolution image patches, and $w$ contains the values of the previously reconstructed high-resolution image patches in the overlapping region. The constrained optimization (3.7) can be similarly reformulated as

$$\min_{\alpha} \|\tilde{D}\alpha - \tilde{y}\|_2^2 + \lambda\|\alpha\|_1, \tag{3.8}$$

where $\tilde{D} = \begin{bmatrix} FD_l \\ \beta PD_h \end{bmatrix}$ and $\tilde{y} = \begin{bmatrix} Fy \\ \beta w \end{bmatrix}$. The parameter $\beta$ controls the tradeoff between matching the low-resolution input and finding a high-resolution patch that is compatible with its neighbors. In all our experiments, we simply set $\beta = 1$. Given the optimal solution $z$ to Equation (3.8), the high-resolution patch can be reconstructed as $x = D_hz$. 

4 There are different ways to enforce compatibility. In [53], the values in the overlapped regions are simply averaged, which will result in blurring effects. The greedy one-pass algorithm [51] is shown to work almost as well as the use of a full MRF model [50]. Our algorithm, not based on the MRF model, is essentially the same by trusting partially the previously recovered high-resolution image patches in the overlapped regions.
Algorithm 1 Super-Resolution via Sparse Representation

1: **Input:** training dictionaries \(D_h\) and \(D_l\), a low-resolution image \(Y\).
2: **For** each \(3 \times 3\) patch \(y\) of \(Y\), taken starting from the upper-left corner with 1 pixel overlap in each direction,
   - Compute the mean pixel value \(m\) of patch \(y\).
   - Solve the optimization problem with \(\tilde{D}\) and \(\tilde{y}\) defined in Equation (3.8):
     \[\min_{\alpha} \| \tilde{D}\alpha - \tilde{y} \|_2^2 + \lambda \| \alpha \|_1.\]
   - Generate the high-resolution patch \(x = D_hz\). Put the patch \(x + m\) into a high-resolution image \(X_0\).
3: **End**
4: Using gradient descent, find the closest image to \(X_0\) which satisfies the reconstruction constraint:
   \[X^* = \arg \min_X \| SHX - Y \|_2^2 + c \| X - X_0 \|_2^2.\]
5: **Output:** super-resolution image \(X^*\).

Enforcing Global Reconstruction Constraint  Notice that Equations (3.5) and (3.7) do not demand exact equality between the low-resolution patch \(y\) and its reconstruction \(D_l\alpha\). Because of this, and also because of noise, the high-resolution image \(X_0\) produced by the sparse representation approach may not satisfy the reconstruction constraint (3.2) exactly. We eliminate this discrepancy by projecting \(X_0\) onto the solution space of \(SHX = Y\) by computing
\[X^* = \arg \min_X \| SHX - Y \|_2^2 + c \| X - X_0 \|_2^2.\] (3.9)
The solution to this optimization problem can be efficiently computed using gradient descent. The update equation for this iterative method is
\[X_{t+1} = X_t + \nu[H^T S^T (Y - SHX_t) + c(X - X_0)],\] (3.10)
where \(X_t\) is the estimate of the high-resolution image after the \(t\)-th iteration, \(\nu\) is the step size of the gradient descent.

We take result \(X^*\) from the above optimization as our final estimate of the high-resolution image. This image is as close as possible to the initial super-resolution \(X_0\) given by sparsity, while respecting the reconstruction constraint. The entire super-resolution process is summarized in Algorithm 1.
Global Optimization Interpretation  The simple SR algorithm outlined in the previous two subsections can be viewed as a special case of a more general sparse representation framework for inverse problems in image processing. Related ideas have been profitably applied in image compression and image denoising [10], [11]. In addition to placing our work in a larger context, these connections suggest means of further improving the performance, at the cost of increased computational complexity.

Given sufficient computational resources, one could in principle solve for the coefficients associated with all patches simultaneously. Moreover, the entire high-resolution image $X$ itself can be treated as a variable. Rather than demanding that $X$ be perfectly reproduced by the sparse coefficients $\alpha$, we can penalize the difference between $X$ and the high-resolution image given by these coefficients, allowing solutions that are not perfectly sparse, but better satisfy the reconstruction constraints. This leads to a large optimization problem:

$$
X^* = \arg \min_{X,\{\alpha_{ij}\}} \left\{ \|SHX - Y\|^2_2 + \lambda \sum_{i,j} \|\alpha_{ij}\|_0 + \gamma \sum_{i,j} \|D_h \alpha_{ij} - P_{ij}X\|^2_2 + \tau \rho(X) \right\}.
$$

(3.11)

Here, $\alpha_{ij}$ denotes the representation coefficients for the $(i, j)_{th}$ patch of $X$, and $P_{ij}$ is a projection matrix that selects the $(i, j)_{th}$ patch from $X$. The penalty function $\rho(X)$ encodes additional prior knowledge about the high-resolution image. This function may depend on the image category or may take the form of a generic regularization term (e.g., Huber MRF, total variation, bilateral total variation).

Algorithm 1 can be interpreted as a computationally efficient approximation to Equation (3.11). The sparse representation step recovers the coefficients $\alpha$ by approximately minimizing the sum of the second and third terms of Equation (3.11). The sparsity term $\|\alpha_{ij}\|_0$ is relaxed to $\|\alpha_{ij}\|_1$, while the high-resolution fidelity term $\|D_h \alpha_{ij} - P_{ij}X\|^2_2$ is approximated by its low-resolution version $\|FD_l \alpha_{ij} - Fy_{ij}\|^2_2$.

Notice, that if the sparse coefficients $\alpha$ are fixed, the third term of Equation (3.11) essentially penalizes the difference between the super-resolution image $X$ and the reconstruction given by the coefficients: $\sum_{i,j} \|D_h \alpha_{ij} - P_{ij}X\|^2_2 \approx \|X_0 - X\|^2_2$. Hence, for small $\gamma$, the gradient descent optimization step of Algorithm 1 approximately minimizes the sum of the first and third terms of Equation (3.11).
Algorithm 1 does not, however, incorporate any prior besides sparsity of the representation coefficients – the term $\rho(X)$ is absent in our approximation. In Section 3.4 we will see that sparsity in a relevant dictionary is a strong enough prior that we can already achieve good super-resolution performance. Nevertheless, in settings where further assumptions on the high-resolution signal are available, these priors can be incorporated into the global reconstruction step of our algorithm.

### 3.2.2 Face Super-Resolution from Sparsity

Face image resolution enhancement is usually desirable in many surveillance scenarios, where there is always a large distance between the camera and the objects (people) of interest. Unlike the generic image super-resolution discussed earlier, face images are more regular in structure and thus should be easier to handle. Indeed, for face super-resolution, we can deal with lower resolution input images. The basic idea is first to use the face prior to zoom the input to a reasonable medium resolution, and then to employ the local sparsity prior model to recover details. To be precise, the solution is also approached in two steps: (1) global model: use reconstruction constraint to recover a medium high-resolution face image, but the solution is searched only in the face subspace; and (2) local model: use the local sparse model to recover the image details.

**Non-Negative Matrix Factorization** In face super-resolution, the most frequently used subspace method for modeling the human face is Principal Component Analysis (PCA), which chooses a low-dimensional subspace that captures as much of the variance as possible. However, the PCA bases are holistic, and tend to generate smooth faces similar to the average face. Moreover, because principal component representations allow negative coefficients, the PCA reconstruction is often hard to interpret.

Even though faces are objects with lots of variance, they are made up of several relatively independent parts such as eyes, eyebrows, noses, mouths, checks, and chins. Non-Negative Matrix Factorization (NMF) [112] seeks a representation of the given signals as an additive combination of local features. To find such a
part-based subspace, NMF is formulated as the following optimization problem:

$$\begin{align*}
\arg \min_{U, V} & \|X - UV\|_2^2 \\
\text{s.t.} & \quad U \geq 0, V \geq 0,
\end{align*}$$

(3.12)

where $X \in \mathbb{R}^{n \times m}$ is the data matrix, $U \in \mathbb{R}^{n \times r}$ is the basis matrix, and $V \in \mathbb{R}^{r \times m}$ is the coefficient matrix. In our context here, $X$ simply consists of a set of pre-aligned high-resolution training face images as its column vectors. The number of the bases $r$ can be chosen as $n \ast m/(n + m)$ which is smaller than $n$ and $m$, meaning a more compact representation. It can be shown that a local optimum of Equation (3.12) can be obtained via the following updating rules:

$$\begin{align*}
V_{ij} & \leftarrow V_{ij} \frac{(U^T X)_{ij}}{(U^T U V)_{ij}} \\
U_{ki} & \leftarrow U_{ki} \frac{(X V^T)_{ki}}{(U V V^T)_{ki}},
\end{align*}$$

(3.13)

where $1 \leq i \leq r$, $1 \leq j \leq m$, and $1 \leq k \leq n$. The obtained basis matrix $U$ is often sparse and localized.

**Two-Step Face Super-Resolution** Let $X$ and $Y$ denote the high- and low-resolution faces, respectively. $Y$ is obtained from $X$ by smoothing and downsampling as in Equation (3.2). We want to recover $X$ from the observation $Y$. Here, we assume $Y$ has been pre-aligned to the training database by either manually labeling the feature points or with some automatic face alignment algorithm such as the method used in [105]. We can achieve the optimal solution for $X$ based on the Maximum a Posteriori (MAP) criteria,

$$X^* = \arg \max_X p(Y | X) p(X).$$

(3.14)

Here, $p(Y | X)$ models the image observation process, bearing the following form,

$$p(Y | X) = \frac{1}{Z} \exp\left(-\frac{\|SHUc - Y\|_2^2}{2\sigma^2}\right)$$

(3.15)

where $Z$ is a normalization factor, $H$ is the blurring operator, $S$ is the downsampling operator, $U$ is the basis matrix for the face images, $c$ is the representation coefficient for the test face, and $\sigma$ is the variance of the Gaussian noise in the
observation $Y$. A prior $p(X)$ on the underlying high-resolution face image $X$ is used, typically in the exponential form $p(X) = \exp(-t\rho(X))$. Using the rules in Equation (3.13), we can obtain the basis matrix $U$, which is composed of sparse bases. Let $\Omega$ denote the face subspace spanned by $U$. Then in the subspace $\Omega$, the super-resolution problem in Equation (3.14) can be formulated using the reconstruction constraints as:

$$ c^* = \arg \min_c \|SHUc - Y\|_2^2 + \eta\rho(Uc) \quad \text{s.t. } c \geq 0, \quad (3.16) $$

where $\rho(Uc)$ is a prior term regularizing the high-resolution solution, $c \in \mathbb{R}^{r \times 1}$ is the coefficient vector in the subspace $\Omega$ for estimating the high-resolution face, and $\eta$ is a parameter used to balance the reconstruction fidelity and the penalty of the prior term. Here, we simply use a generic image prior requiring that the solution be smooth. Let $\Gamma$ denote a matrix performing high-pass filtering. The final formulation for optimization (3.16) is

$$ c^* = \arg \min_c \|SHUc - Y\|_2^2 + \eta\|\Gamma Uc\|_2 \quad \text{s.t. } c \geq 0. \quad (3.17) $$

The medium-resolution image $\hat{X}$ is approximated by $Uc^*$. The prior term in optimization (3.17) suppresses the high-frequency components, resulting in an over-smooth solution image. We rectify this with the local patch model based on sparse representation mentioned earlier in Section 3.2.1. The complete framework of our algorithm is summarized in Algorithm 2.
Algorithm 2 Face Hallucination via Sparse Representation

1: **Input:** sparse face basis matrix $U$, training dictionaries $D_h$ and $D_l$, a low-resolution aligned face image $Y$.

2: **Find** a smooth high-resolution face $\hat{X}$ from the subspace spanned by $U$ through:
   - Solve the optimization problem in (3.17):
     $$\arg\min_c \|SHUc - Y\|_2^2 + \eta \|\Gamma Uc\|_2 \quad s.t. \quad c \geq 0.$$  
   - $\hat{X} = Uc^*$.

3: **For** each patch $y$ of $\hat{X}$, taken starting from the upper-left corner with 1 pixel overlap in each direction,
   - Compute and record the mean pixel value of $y$ as $m$.
   - Solve the optimization problem with $\tilde{D}$ and $\tilde{y}$ defined in (3.8):
     $$\min_{\alpha} \|\tilde{D}\alpha - \tilde{y}\|_2^2 + \lambda \|\alpha\|_1.$$  
   - Generate the high-resolution patch $x = D_h\alpha^* + m$. Put the patch $x$ into a high-resolution image $X^*$.

4: **Output:** super-resolution face $X^*$.

3.3 Learning the Dictionary Pair

In the previous section, we discussed regularizing the super-resolution problem using a sparse representation prior, which states that each pair of low- and high-resolution image patches has the same sparse representations with respect to the two dictionaries $D_h$ and $D_l$. A straightforward way to obtain two such dictionaries is to directly sample image patch pairs from the training images, which preserves the correspondence between the low- and high-resolution patch items [68]. However, such a strategy will result in large dictionaries and hence expensive computation. This section focuses on learning a much more compact dictionary pair for speeding up the algorithm.

3.3.1 Single Dictionary Training

Sparse coding is the problem of finding sparse representations of the signals with respect to an over-complete dictionary $D$. The dictionary is usually learned from a set of training examples $X = [x_1, x_2, ..., x_t]$. Generally, it is hard to learn a compact dictionary which guarantees that the sparse representation of Equa-
tion (3.4) can be recovered from $\ell_1$-minimization in Equation (3.5). Fortunately, many sparse coding algorithms proposed previously suffice for practical applications. In this work, we focus on the following formulation:

$$D = \arg \min_{D, Z} \|X - DZ\|^2_2 + \lambda \|Z\|_1$$

subject to $\|D(:, k)\|^2_2 \leq 1, k = 1, 2, \ldots, K$. (3.18)

where the $\ell_1$-norm on $Z$ is to enforce sparsity, and the $\ell_2$-norm constraints on the columns of $D$ remove the scaling ambiguity.\(^5\) This particular formulation has been studied extensively [113, 75, 114]. Formulation (3.18) is not convex in both $D$ and $Z$, but is convex in one of them with the other fixed. The optimization performs in an alternating manner over $Z$ and $D$:

1. Initialize $D$ with a Gaussian random matrix, with each column unit normalized.

2. Fix $D$, update $Z$ by

$$Z^* = \arg \min_Z \|X - DZ\|^2_2 + \lambda \|Z\|_1,$$

which can be solved efficiently through linear programming.

3. Fix $Z$, update $D$ by

$$D = \arg \min_D \|X - DZ\|^2_2$$

subject to $\|D(:, k)\|^2_2 \leq 1, k = 1, 2, \ldots, K$. (3.20)

which is a Quadratically Constrained Quadratic Programming (QCQP) that is ready to be solved in many optimization packages.

4. Iterate between step (2) and (3) until convergence. In our implementation, we used a MATLAB package developed in [75].

3.3.2 Joint Dictionary Training

Given the sampled training image patch pairs $\mathcal{P} = \{X^h, Y^l\}$, where $X^h = [x_1, x_2, \ldots, x_n]$ is the set of sampled high-resolution image patches and $Y^l = [y_1, y_2, \ldots, y_m]$ is the set of sampled low-resolution image patches.

\(^5\)Note that without the norm constraints, the cost can always be reduced by dividing $Z$ by $c > 1$ and multiplying $D$ by $c > 1$. 

50
Therefore, the corresponding set of low-resolution image patches (or features), our goal is to learn dictionaries for low- and high-resolution image patches, so that the sparse representation of the low-resolution patch is the same as that of the corresponding high-resolution patch. This is a difficult problem, due to the ill-posed nature of the super-resolution problem. The individual sparse coding problems in the low- and high-resolution image patch spaces are

\[ D_l = \arg \min_{\{D_l, Z\}} \| Y^l - D_l Z \|_2^2 + \lambda \| Z \|_1, \]  

(3.21)

and

\[ D_h = \arg \min_{\{D_h, Z\}} \| X^h - D_h Z \|_2^2 + \lambda \| Z \|_1, \]  

(3.22)

respectively. We combine these two objectives, forcing the low- and high-resolution image patches to share the same sparse codes, and formulate

\[
\min_{\{D_h, D_l, Z\}} \frac{1}{N} \| X^h - D_h Z \|_2^2 + \frac{1}{M} \| Y^l - D_l Z \|_2^2 \\
+ \lambda \left( \frac{1}{N} + \frac{1}{M} \right) \| Z \|_1, \]  

(3.23)

where \( M \) and \( N \) are the dimensions of the low- and high-resolution image patches, respectively. Here, \( 1/M \) and \( 1/N \) balance the two cost terms of Equations (3.21) and (3.22). We can rewrite Equation (3.23) as

\[
\min_{\{D_h, D_l, Z\}} \| X_c - D_c Z \|_2^2 + \lambda \left( \frac{1}{N} + \frac{1}{M} \right) \| Z \|_1, \]  

(3.24)

or equivalently

\[
\min_{\{D_h, D_l, Z\}} \| X_c - D_c Z \|_2^2 + \hat{\lambda} \| Z \|_1, \]  

(3.25)

where

\[
X_c = \begin{bmatrix} \frac{1}{\sqrt{N}} X^h \\ \frac{1}{\sqrt{M}} Y^l \end{bmatrix}, \quad D_c = \begin{bmatrix} \frac{1}{\sqrt{N}} D_h \\ \frac{1}{\sqrt{M}} D_l \end{bmatrix}. \]  

(3.26)

Thus we can use the same learning strategy in the single dictionary case for training the two dictionaries for our super-resolution purpose. Note that since we are using features from the low-resolution image patches, \( D_h \) and \( D_l \) are not simply related by a linear transform anymore; otherwise, the training process of Equation (3.25) will depend on the high-resolution image patches only (for detail, refer
Figure 3.2: The high-resolution image patch dictionary trained by Equation (3.25) using 100,000 high- and low-resolution image patch pairs sampled from the a generic training image database. In total, 512 dictionary atoms are learned with each atom of size $9 \times 9$. to Section 3.3.3). Figure 3.2 shows the dictionary learned by Equation (3.25) for generic images.\(^6\) The learned dictionary demonstrates basic patterns of the image patches, such as orientated edges, instead of raw patch prototypes, due to its compactness.

3.3.3 Feature Representation for Low-Resolution Image Patches

In Equation (3.4), we use a feature transformation $F$ to ensure that the computed coefficients fit the most relevant part of the low-resolution signal for super-resolution, and hence have a more accurate prediction for the high-resolution image patch reconstruction. Typically, $F$ is chosen as some kind of high-pass filter. This is reasonable from a perceptual viewpoint, since people are more sensitive to the high-frequency content of the image. The high-frequency components of the low-resolution image are also arguably the most important for predicting the lost high-frequency contents in the target high-resolution image.

In the literature, researchers have suggested extracting different features for the low-resolution image patch in order to boost the prediction accuracy. Freeman et al. [50] used a high-pass filter to extract the edge information from the low-

\(^6\) We omit the dictionary for the low-resolution image patches because we are training on image features instead the patches.
resolution input patches as the feature. Sun et al. [52] used a set of Gaussian derivative filters to extract the contours in the low-resolution patches. Chang et al. [53] used the first- and second-order gradients of the patches as the feature representation. In this work, we also use the first- and second-order derivatives as the feature for the low-resolution patch due to its simplicity and effectiveness. The four 1-D filters used to extract the derivatives are

\[ f_1 = [-1, 0, 1], \quad f_2 = f_1^T, \]
\[ f_3 = [1, 0, -2, 0, 1], \quad f_4 = f_3^T. \]

(3.27)

Applying these four filters yields four feature vectors for each patch, which are concatenated into one vector as the final representation of the low-resolution patch. In our implementation, the four filters are not applied directly to the sampled low-resolution image patches; instead, we apply the four filters to the training images. Thus, for each low-resolution training image, we obtain four gradient maps, we extract four patches from these gradient maps at each location, and then concatenate them to get the feature vector. Therefore, the feature representation for each low-resolution image patch also encodes its neighboring information, which is beneficial for promoting compatibility among adjacent patches in the final super-resolution image.

In practice, we find that it works better to extract the features from the upsampled version of the low-resolution image instead of the original one. That is, we first upsample the low-resolution image by a factor of two using bicubic interpolation, and then extract gradient features from it. Since we know all the upsampling ratios, it is easy to track the correspondence between high-resolution image patches and the upsampled low-resolution image patches both for training and testing. Because of the way of extracting features from the low-resolution image patches, the two dictionaries \( D_h \) and \( D_l \) are not simply linearly related, making the joint learning process in Equation (3.25) more necessary.

\footnote{We choose the factor of two mainly for dimension considerations. For example, if we work on 3 × 3 patches in the low-resolution image, by upsampling the image by the ratio of two, the final feature for the nine dimensional low-resolution patch will be 6 × 6 × 4 = 144.}
3.4 Experimental Results

In this section, we first demonstrate the super-resolution results obtained by applying the above methods on both generic and face images. We then move on to discuss various influential factors for the proposed algorithm including dictionary size, noise with inputs, and the global reconstruction constraints.

In our experiments, we magnify the input low-resolution image by a factor of three for generic images and four for face images, which is commonplace in the literature of single frame super-resolution. In generic image super-resolution, for the low-resolution images, we always use $3 \times 3$ low-resolution patches (upsampled to $6 \times 6$), with overlap of one pixel between adjacent patches, corresponding to $9 \times 9$ patches with overlap of three pixels for the high-resolution patches. In face super-resolution, we choose the patch size as $5 \times 5$ pixels for both low- and high-resolution face images. For color images, we apply our algorithm to the illuminance channel only, because humans are more sensitive to illuminance changes. We therefore interpolate the color layers (Cb, Cr) using plain bicubic interpolation. We evaluate the results of various methods both visually and qualitatively in Root Mean Square Error (RMSE). Even though RMSE is a common criterion in image processing for recovery, it is not quite reliable for rating visual image quality [115], as we will see in the following parts. Note that since we only work on the illuminance channel, the RMSE reported is carried out only on the illuminance channel.

3.4.1 Single Image Super-Resolution

Generic Image Super-resolution We apply our methods to generic images such as flowers, human faces, and architectures. The two dictionaries for low- and high-resolution image patches are trained from 100,000 patch pairs randomly sampled from natural images collected from the Internet. We preprocess these images by cropping out the texture regions and discarding the smooth parts. Unless otherwise explicitly stated, we always fix the dictionary size as 1024 in all our experiments, which is a balance between computational complexity and image quality (Section 3.4.2 will examine the effects of different dictionary sizes). In the super-resolution algorithm Equation (3.8), the choice of $\lambda$ depends on the level of noise.

---

8 Other authors prepare the training patches by extracting the image edges and sample patches around the edge regions to get the patch primitives.
in the input image, which we will discuss further in Section 3.4.3. For generic low-noise images, we always set $\lambda = 0.1$ in all our experiments, which generally yields satisfactory results.

Figures 3.3 and 3.4 compare the outputs of our method with those of the neighborhood embedding method (NE) proposed in [53]. The NE method is similar to ours in the sense that both methods use the linear combination weights derived from the low-resolution image patch to generate the underlying high-resolution image patch. Unlike our method, the NE method uses a fixed $k$-nearest neighbors to find the reconstruction supports directly from sampled training patches and does not include a dictionary training phase. To make a fair comparison, we use the same 100,000 training patch pairs for the NE method, and we try different $k$’s to get the most visually appealing results. Using a compact dictionary pair, our method is much faster and yet generates sharper results. As the reconstructed images show in Figures 3.3 and 3.4, there are noticeable differences in the texture of the leaves, the fuzz on the leaf stalk, and the freckles on the girl’s face. In the captions of both figures, we list the RMSEs in parentheses following each method. As shown, our method can achieve lower RMSE than both bicubic interpolation and NE. An interesting observation is that, although NE generates visually more appealing results than bicubic, its RMSE is actually higher than bicubic, indicating that RMSE is not a reliable criterion for visual image quality.

55
In Figure 3.5, we compare our method with several more state-of-the-art methods on an image of the Parthenon used in [102], including back projection (BP) [116], NE [53], and the recently proposed method based on a learned soft edge prior (SE) [102]. The result from back projection has many jaggy effects along the edges. NE generates sharp edges in places, but blurs the texture on the temple’s facade. The SE method overall gives a decent reconstruction, but introduces undesired smoothness that is not present in our result. We also give the RMSEs for all the methods in parentheses in the caption. Again, besides best visual quality, our method achieves the lowest RMSE among these methods as well.

**Face Super-Resolution** In this part, we evaluate our proposed super-resolution algorithm on frontal view human faces. The experiments are conducted on the face database FRGC Ver 1.0 [117]. All these high-resolution face images were aligned by an automatic alignment algorithm using the eye positions, and then cropped to the size of 100 × 100 pixels. To obtain the face subspace $\Omega$ spanned by $U$, we selected 540 face images as the training, covering both genders, different races, varying ages and different facial expressions (Figure 3.6). These high-resolution face images are blurred and downsampled to the size of 25 × 25 pixels to form the low-resolution counterparts. To prepare the coupled dictionaries needed for our sparse representation algorithm, we also sample 100,000 patch pairs from the training images and learn the dictionaries of size 1024. Thirty new face images
(from people not in the training set) are chosen as our test cases, which are blurred and downsampled to the size of $25 \times 25$ pixels in the same procedure as preparing the training set. These low-resolution input faces are aligned by manually labeling the eyeball positions.

Figure 3.6: Example training faces for the face super-resolution algorithm. The training images cover faces of both genders, different ages, different races, and various facial expressions.

As mentioned earlier, face image super-resolution can handle more challenging tasks than generic image super-resolution due to the regular face structure. Indeed, it is not an easy job to zoom an $25 \times 25$ low-resolution face image by four times with the method for generic image super-resolution. First, the down-sampling process loses so much information that it is difficult to predict well a $12 \times 12$ high-resolution patch given only a $3 \times 3$ image patch. Second, the resolution of the face image is so low that the structures of the face that are useful for super-resolution inference (such as corners and edges) collapse into only a couple of pixels. The two-step approach for face super-resolution, on the other hand, can compensate for the lost information in the first step using the redundancy of the face structures by searching a solution in the face subspace respecting the reconstruction constraints. The local model from sparse representation then can be further employed to enhance the edges and textures to achieve sharper results. In Figure 3.7, we compare the proposed two-step approach with the direct sparse representation method for generic images. Since the resolution of the input face image is so low, a direct application of the generic approach does not generate satisfying results.

In our experiments with face images, we also set $\lambda = 0.1$ for the sparsity regularization. We compare our algorithm with bicubic interpolation [101] and BP [116]. The results are shown in Figure 3.8, which indicate that our method can generate much higher resolution faces. Column (d) shows the intermediate results from the NMF global modeling and column (e) demonstrates the results after local sparse modeling. Comparing the two columns, the local sparse modeling further
Figure 3.7: The comparison between the two-step face super-resolution algorithm with the generic image super-resolution algorithm applied to low-resolution face images. From left to right: input image, super-resolution result using our two-step approach, and super-resolution result using the generic approach. The two-step face super-resolution algorithm generates visually much better results.

enhances the edges and textures, and also reduces RMSE.

3.4.2 Effects of Dictionary Size

The above experimental results show that the sparsity prior for image patches is very effective in regularizing the otherwise ill-posed super-resolution problem. In those results, we fix the dictionary size to be 1024. Intuitively, larger dictionaries should possess more representation power (in the extreme, we can use the sampled patches as the dictionary directly as in [68]), and thus may generate a more accurate approximation, at the cost of increased computation cost. In this section, we evaluate the effects of the dictionary size on generic image super-resolution. From the sampled 100,000 image patch pairs, we train four dictionaries of size 256, 512, 1024, and 2048, and apply them to the same input images. We also use the 100,000 image patches directly as the dictionary for comparison. The results are evaluated both visually and quantitatively in RMSE.

Figure 3.9 shows the reconstructed results for the Lena image using dictionaries of different sizes. While there are not many visual differences for the results using different dictionary sizes from 256 to 2048 and the whole sampled patch set, we indeed observe the reconstruction artifacts will gradually diminish with larger dictionaries. The visual observation is also supported by the RMSEs of the recovered images. In Table 3.1, we list the RMSEs of the reconstructed images for different dictionaries. As shown in the table, using larger dictionaries...
Figure 3.8: Results of our algorithm compared to other methods and the corresponding average RMSEs. From left to right columns: (a) low-resolution input; (b) bicubic interpolation (RMSE: $8.024$); (c) back projection (RMSE: $7.474$); (d) global NMF modeling followed by bilateral filtering (RMSE: $10.738$); (e) global NMF modeling and Sparse Representation (RMSE: $6.891$); (f) Original.

will generate smaller RMSEs, and all of them are smaller than those by bicubic interpolation. However, the computation is approximately linear to the size of the dictionary; larger dictionaries will result in heavier computation. Figure 3.10 shows the computation time in seconds with “Girl” as the test image. The algorithm was written in MATLAB without optimization for speed, and was ran on a laptop of Core duo @ 1.83 G with 2 G memory. The computation time with the entire training set as the dictionary [68] is almost an hour, much slower than our current solution with trained compact dictionaries. In practice, one chooses the appropriate dictionary size as a trade-off between reconstruction quality and computation. We find that dictionary size 1024 can yield decent outputs, while allowing fast computation.
Table 3.1: The RMSEs of the reconstructed images using dictionaries of different sizes, and using the raw image patches directly from which the dictionaries are trained.

<table>
<thead>
<tr>
<th>Images</th>
<th>Bicubic</th>
<th>D256</th>
<th>D512</th>
<th>D1024</th>
<th>D2048</th>
<th>Raw Patches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Girl</td>
<td>5.912</td>
<td>5.606</td>
<td>5.603</td>
<td>5.491</td>
<td>5.473</td>
<td>5.483</td>
</tr>
<tr>
<td>Flower</td>
<td>3.530</td>
<td>3.266</td>
<td>3.271</td>
<td>3.212</td>
<td>3.164</td>
<td>3.139</td>
</tr>
</tbody>
</table>

Figure 3.9: The effects of dictionary size on the super-resolution reconstruction of Lena. From left to right: dictionary size 256, 512, 1024, 2048, and the whole training patch set, respectively.

Figure 3.10: The computation time on the “Girl” image with dictionaries of different sizes (in seconds).
3.4.3 Robustness to Noise

Most single input super-resolution algorithms assume that the input images are clean and free of noise, an assumption which is probably to be violated in real applications. To deal with noisy data, previous algorithms usually divide the recovery process into two disjoint steps: first denoising and then super-resolution. However, the results of such a strategy will depend on the specific denoising technique, and any artifacts during the denoising step on the low-resolution image will be kept or even magnified in the latter super-resolution step. Here we demonstrate that by formulating the problem in our sparse representation model, our method is much more robust to noise, and thus can handle image super-resolution and denoising simultaneously. Note that in Equation (3.6) the parameter $\lambda$ depends on the noise level of the input data; the noisier the data, the larger the value of $\lambda$ should be. Figure 3.11 shows how $\lambda$ influences the reconstructed results given the same clean input image. The larger $\lambda$, the smoother the result image gets. This is obvious by formulating Equation (3.8) into a Maximum a Posterior (MAP) problem:

$$\alpha^* = \arg \max P(\alpha) \cdot P(\tilde{y}|\alpha, \tilde{D}).$$  (3.28)

where

$$P(\alpha) = \frac{1}{2b} \exp\left(-\frac{\|\alpha\|_1}{b}\right)$$

$$P(\tilde{y}|\alpha, \tilde{D}) = \frac{1}{2\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\|\tilde{D}\alpha - \tilde{y}\|_2^2\right),$$  (3.29)

where $b$ is the variance of the Laplacian prior on $\alpha$, and $\sigma$ is the variance of the noise assumed on the data $\tilde{y}$. Taking the negative log likelihood in Equation (3.28), we get the exact optimization problem in Equation (3.8), with $\lambda = \sigma^2/b$. Suppose the Laplacian variance $b$ is fixed; then the noisier the data ($\sigma^2$ is larger), the larger the value $\lambda$ should be. On the other hand, given the input image, the larger we set the value $\lambda$, the noisier the model will assume the data to be, and thus the algorithm will tend to generate smoother results.

To test the robustness of our algorithm to noise, we add different levels of Gaussian noise to the low-resolution input images. The standard deviation of the Gaussian noise ranges from 4 to 10. The regularization parameter $\lambda$ is empirically set to be one tenth of the standard deviation. In Figure 3.12, we show the results of our algorithm applied to the Liberty statue image with different levels of
Table 3.2: The RMSEs of the reconstructed images from different levels of noisy inputs.

<table>
<thead>
<tr>
<th>Noise Levels / Gaussian σ</th>
<th>0</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicubic</td>
<td>9.873</td>
<td>10.423</td>
<td>11.037</td>
<td>11.772</td>
</tr>
<tr>
<td>Our method</td>
<td><strong>8.359</strong></td>
<td><strong>9.240</strong></td>
<td><strong>10.454</strong></td>
<td><strong>11.448</strong></td>
</tr>
</tbody>
</table>

Figure 3.11: The effects of $\lambda$ on the recovered image given the clean input. From left to right, $\lambda = 0.01, 0.05, 0.1, 0.2, 0.3$. The larger $\lambda$ is, the smoother the result image gets. Note that the results are generated from the local sparse model only.

Gaussian noise. For comparisons, we also show the results of using bicubic and NE [53]. As expected, the results of bicubic are both noisy and blurry. For NE, we tuned the number of neighbors for different levels of noise in order to get the best visual results. As shown in Figure 3.12, the NE method is good at preserving edges, but fails to distinguish the signal from noise, and therefore generates results with magnified noise. Our algorithm is capable of performing denoising and super-resolution simultaneously. Table 3.2 shows the RMSEs of the reconstructed images from different levels of noisy data. In terms of RMSE, our method outperforms both bicubic interpolation and NE in all cases.

3.4.4 Effects of Global Constraints

The global reconstruction constraint enforced by Equation (3.9) is employed to refine the local image patch sparse model, ensuring the recovered high-resolution image to be consistent with its low-resolution observation as a whole. In our experiments, we observe that the sparsity prior is very effective and contributes the most for recovering the missing high-frequency information, while the global constraint in the second step reduces RMSE by removing some minor artifacts, which are hardly seen from the first step. Table 3.3 shows the RMSEs of the results from
Figure 3.12: Performance evaluation of our proposed algorithm on noisy data. Noise level (standard deviation of Gaussian noise) from left to right: 0, 4, 6, and 8. Top row: input images. Middle row: recovered images using NE [53] (k = 13, 12, 9, 7). Bottom row: recovered images using our method (λ = 0.1, 0.4, 0.6, 0.8).
Table 3.3: The global constraint in the second step further refines the results from local sparse model in the first step and reduces RMSEs.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Flower</th>
<th>Girl</th>
<th>Parthenon</th>
<th>Lena</th>
<th>Statue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Model</td>
<td>3.365</td>
<td>5.669</td>
<td>12.247</td>
<td>6.775</td>
<td>8.902</td>
</tr>
</tbody>
</table>

the local sparse model only and the local model combined with the global model. The RMSEs of bicubic interpolation are again given as the reference. As shown, the local sparse model can achieve better RMSEs than bicubic interpolation, and the global constraint further reduces the RMSEs.

3.5 Conclusion

We address the single image super-resolution problem based on sparse recovery. Research on image statistics suggests that image patches can be well-represented as a sparse linear combination of elements from an appropriately chosen over-complete dictionary. Inspired by this observation, we seek a sparse representation for each patch of the low-resolution input, and then use the coefficients of this representation to generate the high-resolution output. Theoretical results from compressed sensing suggest that under mild conditions, the sparse representation can be correctly recovered from the downsampled signals. By jointly training two dictionaries for the low- and high-resolution image patches, we can enforce the similarity of sparse representations between the low-resolution and high-resolution image patch pairs with respect to their own dictionaries. Therefore, the sparse representation of a low-resolution image patch can be applied to the high-resolution image patch dictionary to generate a high-resolution image patch. The learned dictionary pair is a more compact representation of the patch pairs, reducing the computational cost substantially compared with previous approaches. The effectiveness of such a sparsity prior is demonstrated for both general image super-resolution and the special case of face hallucination. In both cases, our algorithm generates high-resolution images that are competitive or even superior in quality to those produced by other similar SR methods. In addition, the local sparse modeling of our approach is robust to noise, and therefore,
the proposed algorithm can handle super-resolution with noisy inputs in a unified framework.
In this chapter, we present a novel procedure to construct a robust and datum-adaptive $\ell_1$-graph, which is applicable to many graph-orientated algorithms in machine learning. The graph construction utilizes the overall contextual information instead of only pairwise Euclidean distances as done conventionally. The neighboring samples of a datum and the corresponding ingoing edge weights are simultaneously derived by solving an $\ell_1$-norm optimization problem, where each datum is reconstructed by a linear combination of the remaining samples and a noise item, with the objective of minimizing the $\ell_1$-norm of both reconstruction coefficients and data noise. Compared with the conventional graphs constructed by $k$-nearest-neighbor and $\epsilon$-ball methods, the $\ell_1$-graph has the following three advantages.

1. The $\ell_1$-graph is robust owing to the overall contextual $\ell_1$-norm formulation and the explicit consideration of data noise. Figure 4.1(a) shows the graph robustness comparison between our $\ell_1$-graph and the $k$-nearest-neighbor graph.

2. The sparsity of the $\ell_1$-graph is automatically determined instead of manually set as in the $k$-nearest-neighbor and $\epsilon$-ball methods.

3. The $\ell_1$-graph is datum-adaptive. As shown in Figure 4.1(b), the number of neighbors selected by $\ell_1$-graph is adaptive for each datum, which is valuable for applications with unevenly distributed data.

This $\ell_1$-graph is then employed in Section 4.2 to instantiate a series of graph-oriented algorithms for various machine learning tasks, e.g., data clustering, subspace learning, and semi-supervised learning. Owing to the above three advantages over classical graphs, our $\ell_1$-graph brings consistent performance gain in all these tasks as detailed in Section 4.3.
Figure 4.1: Robustness and adaptiveness comparisons for neighbors selected by our $\ell_1$-graph and the $k$-nn graph. (a) Illustration of basis samples (1st row), reconstruction coefficient distribution in the $\ell_1$-graph (left), samples to reconstruct (middle, with added noises from the third row onward), and similarity distribution of the $k$-nearest neighbors selected with Euclidean distance (right) in the $k$-nn graph. Here the horizontal axes indicate the index number of the training samples. The vertical axes of the left column indicate the reconstruction coefficient distribution for all training samples in sparse coding, and those of the right column indicate the similarity value distribution of $k$-nearest neighbors. Note that the number in parenthesis is the number of neighbors changed compared with results in the second row, and our $\ell_1$-graph shows much more robustness to image noises. (b) Neighboring sample comparison between our $\ell_1$-graph and the $k$-nn graph. The red bars indicate the numbers of the neighbors selected by our $\ell_1$-graph automatically and adaptively. The green bars indicate the numbers of kindred samples among the neighbors selected by $\ell_1$-graph. And the blue bar indicate the numbers of kindred samples within the $k$-nearest neighbors measured by Euclidean distance in $k$-nn graph. The results are obtained on the USPS digit database [118], and the horizontal axis indicates the index of the reference samples to reconstruct. For better viewing, please see original enlarged color pdf file.
4.1 Rationales on the $\ell_1$-graph

For a general data clustering or classification problem, the training sample set is assumed to be represented as a matrix $X = [x_1, x_2, \ldots, x_N]$, where $x_i \in \mathbb{R}^m$, $N$ is the sample number and $m$ is the feature dimension. For supervised learning problems, the class label of the sample $x_i$ is then assumed to be $l_i \in \{1, 2, \ldots, N_c\}$, where $N_c$ is the total number of classes.

4.1.1 Motivations

Motivated by the recent advances in sparse coding [108, 119, 16], our $\ell_1$-graph is proposed to overcome the limitations of the classical graph construction methods [57, 56] in modeling high-dimensional data. The graph construction process includes both neighborhood selection and graph edge weight assignment, which are assumed in this work to be unsupervised, i.e., without harnessing any data label information.

The approaches of $k$-nearest neighbors and $\epsilon$-ball are very popular for graph construction in the literature. Both of them determine the neighboring samples based on pairwise Euclidean distance, which is, however, very sensitive to data noise; one noisy feature may dramatically change the graph structure especially in the high-dimensional space. Also, when the data are not evenly distributed, the $k$-nearest neighbors of a datum may involve faraway inhomogeneous data if $k$ is set too large, and the $\epsilon$-ball may involve only single isolated datum if $\epsilon$ is set too small. Moreover, the optimum of $k$ (or $\epsilon$) is datum-dependent, and one single global parameter may result in unreasonable neighborhood structure for certain datums.

The research on sparse coding or sparse representation has a long history. Recent research shows that sparse coding appears to be biologically plausible as well as empirically effective for image processing and pattern classification [16]. Olshausen et al. [113] employed the Bayesian models and imposed the $\ell_1$-norm prior for deducing the sparse representation, and Wright et al. [16] proposed to use sparse representation for face recognition. In this work, beyond the sparse coding for individual test datums, we are interested in the overall behavior of the whole sample set for sparse representation, and then present the general concept of the $\ell_1$-graph, followed by its applications in various machine learning tasks, e.g., data clustering, subspace learning, and semi-supervised learning.
4.1.2 Robust Sparse Representation

There has been much interest in computing sparse linear representation with respect to an over-complete dictionary of the basis elements. Suppose we have an under-determined system of linear equations: \( x = D\alpha \), where \( x \in \mathbb{R}^m \) is the vector to be approximated, \( \alpha \in \mathbb{R}^n \) is the vector for the unknown reconstruction coefficients, and \( D \in \mathbb{R}^{m \times n} (m < n) \) is the over-complete dictionary with \( n \) bases. Generally, a sparse solution is more robust and facilitates the consequent identification of the test sample \( x \). This motivates us to seek the sparsest solution to \( x = D\alpha \) by solving the following optimization problem:

\[
\min_{\alpha} \|\alpha\|_0, \quad \text{s.t.} \quad x = D\alpha,
\]

where \( \| \cdot \|_0 \) denotes the \( \ell_0 \)-norm. As we mentioned before, it is well known that this sparsest representation problem is NP-hard. However, recent results [108, 2] show that if the solution is sparse enough, the sparse representation can be recovered by the following convex \( \ell_1 \)-norm minimization which can be efficiently solved [120],

\[
\min_{\alpha} \|\alpha\|_1, \quad \text{s.t.} \quad x = D\alpha.
\]

In practice, there typically exists noise on certain elements of \( x \), and a natural way to recover these elements and provide a robust estimation of \( \alpha \) is to formulate

\[
x = D\alpha + \zeta = \begin{bmatrix} D & I \end{bmatrix} \begin{bmatrix} \alpha \\ \zeta \end{bmatrix},
\]

where \( \zeta \in \mathbb{R}^m \) is the noise term. Then by setting \( B = \begin{bmatrix} D & I \end{bmatrix} \in \mathbb{R}^{m \times (m+n)} \) and \( \alpha' = \begin{bmatrix} \alpha \\ \zeta \end{bmatrix} \), we can solve the following \( \ell_1 \)-norm minimization problem with respect to both reconstruction coefficients and data noise,

\[
\min_{\alpha'} \|\alpha'\|_1, \quad \text{s.t.} \quad x = B\alpha',
\]

This optimization problem is convex and can be transformed into a general linear programming problem. There exists a globally optimal solution, and the optimization can be solved efficiently using many available \( \ell_1 \)-norm optimization toolboxes like [77].
Algorithm 3 \(\ell_1\)-Graph Construction

1: \textbf{input:} The sample data set denoted as the matrix \(X = [x_1, x_2, \ldots, x_N]\), where \(x_i \in \mathbb{R}^m\).
2: \textbf{for} \(i = 1, 2, \ldots, N\) \textbf{do}
3: \hspace{1em} Solve the \(\ell_1\)-norm minimization
4: \hspace{2em} \(z^i = \arg \min_{\alpha^i} \|\alpha^i\|_1 + \beta \|\alpha^i\|_2^2\), \text{ s.t. } x_i = B^i \alpha^i, \quad (4.5)\)
5: \hspace{1em} where \(B^i = [x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_N, I]\) with \(I\) being the identity matrix.
6: \textbf{end for}
7: \textbf{output:} similarity matrix \(W\).

4.1.3 \(\ell_1\)-graph Construction

For graph construction, the basic procedure is to determine the link or edge similarity between one data sample and the remaining data samples. We formulate this procedure as one of finding the sparse representation of a current data sample in terms of the remaining data samples. The sparse representation coefficients are directly used to determine the neighborhood and edge weights in a simultaneous manner. Therefore, our \(\ell_1\)-graph summarizes the overall behavior of the whole data set via sparse representation. The construction process is formally stated in Algorithm 3.

Figure 4.2 shows illustrations of the \(\ell_1\)-graphs based on data from the YALE-B face database [121] and the USPS digit database [118]. An interesting observation from Figure 4.2 is that, besides being robust to noise and datum-adaptive for neighbor selection, the \(\ell_1\)-graph has the potential to connect more kindred samples, and hence may potentially convey more discriminative information, which is valuable for its applications in data clustering, subspace learning, and semi-supervised learning. Taking the face image as an example, the intuition behind the observed discriminating power of \(\ell_1\)-graph is that, if one expects to reconstruct a face image with all other face images as the bases, the most efficient way in terms of the number of relevant bases is to use similar images or images from the same subject, which naturally leads to a sparse solution and coincides with the empirical observations in [16] for robust face recognition with sparse representa-
Figure 4.2: Visualization comparison of (a) the $\ell_1$-graph and (b) the $k$-nn graph, where the $k$ for each datum is automatically selected in the $\ell_1$-graph. The thickness of the edge line indicates the value of the edge weight (Gaussian kernel weight for the $k$-nn graph). For ease of display, we only show the graph edges related to the samples from two classes, and in total 30 classes from the YALE-B database are used for graph construction. (c) Illustration of the positions of a reference sample (red), its kindred neighbors (yellow), and its inhomogeneous neighbors (blue) selected by (i) the $\ell_1$-graph and (ii) the $k$-nn graph based on samples from the USPS digit database [118]. For better viewing, please see the original enlarged color pdf file.

Discussions

1. The formulation in Equation (4.4) is based on the assumption that the feature dimension, $m$, is reasonably high; otherwise, the sparsity of the noise will make no sense.
2. In implementation, the data normalization, i.e., $\|x_i\|_2 = 1$, is critical for deriving semantically reasonable coefficients.
3. The $k$-nn graph is flexible in terms of the selection of the similarity or distance measurement, but the optimality is heavily data dependent. In this work, we use the most conventional Euclidean distance for selecting the $k$-nearest neighbors.
4. For certain extreme cases, e.g., if we simply duplicate each sample and generate another new dataset of double size, the $\ell_1$-norm minimization may only connect one of those duplicated pairs, and thus fail to convey valuable information. To avoid such problems, in practice we add one small $\ell_2$-norm regularization on the sparse solution in Equation (4.5), known as the Elastic net [122].
5. It should be noted that the $\ell_1$-graph construction procedure can be easily extended to the kernel space, because computing the sparse representation only involves dot products in the constraints. Algorithm 4 states this procedure. In this work, we mainly focus
Algorithm 4 $\ell_1$-Graph Construction in Kernel Space

1: **input:** The sample data set denoted as the matrix $X = [x_1, x_2, ..., x_N]$, and a kernel function $\kappa$.

2: **for** $i = 1, 2, ..., N$ **do**

3: Solve the $\ell_1$-norm minimization

\[
\begin{align*}
    z_i &= \arg \min_{\alpha} \|\alpha\|_1 + \beta \|\alpha\|_2^2 \\
    \text{s.t.} & \quad \kappa(x_i, x_i) + \alpha^T \kappa(B^i, B^i)\alpha - 2\kappa(x_i, B^i)\alpha = 0, \tag{4.6}
\end{align*}
\]

where $B^i = [x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_N, I]$ with I being the identity matrix.

4: Assign

\[
W_{ij} = \begin{cases} 
    \left|z^i(j)\right| & i \neq j \\
    1 & i = j 
\end{cases}
\]

5: **end for**

6: **output:** similarity matrix $W$.

on the $\ell_1$-graph in the original space in the following experiments.

4.2 Learning with the $\ell_1$-graph

An informative graph is critical for those graph-oriented learning algorithms. Similarly to the classical graphs constructed by the $k$-nearest-neighbor or $\epsilon$-ball methods, the $\ell_1$-graph can be integrated with various learning algorithms for various tasks, e.g., data clustering, subspace learning, and semi-supervised learning. In this section, we briefly introduce how to benefit from the $\ell_1$-graph for these tasks.

4.2.1 Spectral Clustering with the $\ell_1$-graph

Data clustering is the classification of samples into different groups, or, more precisely, the partition of samples into subsets, such that the data samples within each subset are similar to each other. Spectral clustering [54] is among the most popular algorithms for this task, but there exists one parameter $\delta$ controlling the similarity between a data pair with a Gaussian kernel in [54]. Intuitively, since the contribution of one sample to the reconstruction of another sample is a good indicator of the similarity between these two samples, we decide to use the reconstruction coefficients to constitute the similarity graph for spectral clustering.
Because the weights of the graph are used to indicate the similarities between different samples, they should be assumed to be non-negative. Using the $\ell_1$-graph, the algorithm can automatically select the neighbors for each datum, and at the same time the similarity matrix is automatically derived from the calculation of these sparse representations. The detailed spectral clustering algorithm based on the $\ell_1$-graph is listed as follows.

1. Symmetrize the graph similarity matrix by setting the matrix $W = (W + W^T)/2$.

2. Set the graph Laplacian matrix $L = D^{-1/2}WD^{-1/2}$, where $D = [d_{ij}]$ is a diagonal matrix with $d_{ii} = \sum_j W_{ij}$.

3. Find $c_1, c_2, \cdots, c_K$, the eigenvectors of $L$ corresponding to the $K$ largest eigenvalues, and form the matrix $C = [c_1, c_2, \cdots, c_K]$ by stacking the eigenvectors in columns.

4. Treat each row of $C$ as a point in $\mathbb{R}^K$, and cluster them into $K$ clusters via the $K$-means method.

5. Finally, assign $x_i$ to the cluster $j$ if the $i$-th row of the matrix $C$ is assigned to the cluster $j$.

### 4.2.2 Subspace Learning with the $\ell_1$-graph

Similarly to the graph construction process in Locally Linear Embedding (LLE), the $\ell_1$-graph characterizes the neighborhood reconstruction relationship. In LLE, the graph is constructed by reconstructing each datum with its $k$-nearest neighbors or the samples within the $\varepsilon$-ball based on the $\ell_2$-norm. LLE and its linear extension, Neighborhood Preserving Embedding (NPE) [123], both rely on the global graph parameter ($k$ or $\varepsilon$). Following the idea of the NPE algorithm, our $\ell_1$-graph can be used to develop a subspace learning algorithm as follows.

The general purpose of subspace learning is to search for a transformation matrix $P \in \mathbb{R}^{m \times d}$ (usually $d \ll m$) for transforming the original high-dimensional datum into a low-dimensional one. The $\ell_1$-graph uncovers the underlying sparse reconstruction relationship of each datum, and it is desirable to preserve these reconstruction relationships in the dimensionality reduced feature space. Note that
in the dimensionality reduced feature space, the reconstruction capability is measured by the $\ell_2$-norm. Then finding the transformation matrix can be formulated as the following optimization

$$
\min_{P^T X X^T P = I} \sum_{i=1}^{N} \|P^T x_i - \sum_{j=1}^{N} W_{ij} P^T x_j\|^2, \quad (4.7)
$$

where $W_{ij}$ is determined by the constructed $\ell_1$-graph. This optimization problem can be solved with the generalized eigenvalue decomposition approach as

$$
X M X^T p_{m+1-j} = \lambda_j X X^T p_{m+1-j}, \quad (4.8)
$$

where $M = (I - W)^T (I - W)$, and $p_{m+1-j}$ is the eigenvector corresponding to the $j$-th largest eigenvalue $\lambda_j$ as well as the $(m + 1 - j)$-th column vector of the matrix $P$.

The derived transformation matrix is then used for dimensionality reduction as

$$
y_i = P^T x_i, \quad (4.9)
$$

where $y_i$ is the corresponding low-dimensional representation of the sample $x_i$ and finally the classification process is performed in this low-dimensional feature space with reduced computational cost.

### 4.2.3 Semi-Supervised Learning with the $\ell_1$-graph

As shown in Figure 4.1 and Figure 4.2, the $\ell_1$-graph is robust to data noises and is datum-adaptive; empirically, it also has the potential to convey more discriminative information compared with conventional graphs based on $k$-nearest neighbors or $\epsilon$-ball. These properties make the $\ell_1$-graph a good candidate for propagating the label information over the graph. Semi-supervised learning recently has attracted much attention, and has been widely used for both regression and classification purposes. The main idea of semi-supervised learning is to utilize the unlabeled data to improve the classification and generalization capability on the test data. Commonly, the unlabeled data are used as an extra regularization term for the objective functions of the traditional supervised learning algorithms.

In this work, the unlabeled data are used to enlarge the vertex number of the $\ell_1$-graph, which further enhances the robustness of the graph. Finally, the $\ell_1$-graph
based on both labeled and unlabeled data is used to develop a semi-supervised learning algorithm. Here, we take Marginal Fisher Analysis (MFA) [28] as an example for the supervised part in our semi-supervised learning. Similarly to the philosophy of [30], the objective for the $\ell_1$-graph based semi-supervised learning is defined as

$$\min_P \gamma S_c(P) + (1 - \gamma) \sum_{i=1}^{N} \|P^T x_i - \sum_{j=1}^{N} W_{ij} P^T x_j\|^2 / S_p(P),$$

where $\gamma \in (0, 1)$ is a threshold for balancing the supervised term and the $\ell_1$-graph regularization term, and the supervised part is defined as

$$S_c(P) = \sum_i \sum_{j \in \mathcal{N}_{k_1}^+(i)} \|P^T x_i - P^T x_j\|^2,$$

$$S_p(P) = \sum_i \sum_{(i,j) \in \mathcal{P}_{k_2}(l)} \|P^T x_i - P^T x_j\|^2.$$

$S_c$ indicates the intra-class compactness, which is represented as the sum of distances between each point and its neighbors of the same class. $\mathcal{N}_{k_1}^+(i)$ is the index set of the $k_1$-nearest neighbors of the sample $x_i$ in the same class. $S_p$ indicates the separability of different classes, which is characterized as the sum of distances between the marginal points and their neighboring points of different classes. $\mathcal{P}_{k_2}(l)$ is a set of data pairs that are the $k_2$-nearest pairs among the set $\{(i, j), l_i = l, l_j \neq l\}$. $W$ is the weight matrix of the $\ell_1$-graph. Similarly to Equation (4.7), the optimum can be obtained via the generalized eigenvalue decomposition method, and the derived projection matrix $P$ is then used for dimensionality reduction and the consequent data classification.

### 4.3 Experiments

In this section, we systematically evaluate the effectiveness of the $\ell_1$-graph in three learning tasks, namely, data clustering, subspace learning, and semi-supervised learning. For comparison purposes, the classical $k$-nearest-neighbor graph and $\epsilon$-ball graph with different graph weighting approaches are implemented as evaluation baselines. Note that for all algorithms based on the $k$-nearest-neighbor graph and the $\epsilon$-ball, the reported results use the best-tuned $k$ and $\epsilon$ among all
4.3.1 Data Sets

For all the experiments, three databases are tested. The USPS handwritten digit database [118] includes 10 classes (0-9 digit characters) and 11000 samples in total. We randomly select 200 samples of each digit character for the experiments, and all of these images are normalized to the size of $32 \times 32$ pixels. The forest covertype database [124] was collected for predicting forest cover type from cartographic variables. It includes seven classes and 581,012 samples in total. We randomly select 100 samples for each type in the following experiments. The Extended YALE-B database [121] contains 38 individuals and around 64 near-frontal view face images under different illuminations per individual, where each image is manually cropped and normalized to the size of $32 \times 32$ pixels. All the images were taken against a dark homogeneous background with the subjects in an upright and frontal position.

4.3.2 Spectral Clustering with the $\ell_1$-graph

In this group of experiments, for a comprehensive evaluation, the $\ell_1$-graph based spectral clustering algorithm is compared with the spectral clustering based on the Gaussian kernel [54] graph, the LE-graphs (used in Laplacian eigenmaps [57] algorithm), the LLE-graphs (used in LLE [56]), and also the $K$-means clustering results based on the derived low-dimensional representations from Principal Component Analysis (PCA) [58]. And two metrics, the accuracy (AC) and the normalized mutual information (NMI) [125], are used for performance evaluation. Suppose that $x$ is the clustering result label vector and $y$ is the ground truth sample label vector; the AC measure is defined as

$$AC = \frac{\sum_{i=1}^{N} \delta(\hat{L}(i) - \text{map}_{(x,y)}(i))}{N}. \quad (4.12)$$

$N$ denotes the total number of samples. $\delta(x)$ is the indicator function; it equals 1 if and only if $x = 0$. $\text{map}_{(x,y)}$ is the best mapping function that permutes $x$ to match $y$. The Kuhn-Munkres algorithm can be used to obtain the best mapping [120].
The mutual information (MI) between two random variables \(x\) and \(y\) is defined as

\[
\text{MI}(x, y) = \sum_{y \in y} \sum_{x \in x} p(x, y) \log_2 \left( \frac{p(x, y)}{p(x)p(y)} \right),
\]

(4.13)

where \(p(x), p(y)\) denote the marginal probability distribution functions of \(x\) and \(y\), respectively, and \(p(x, y)\) is the joint probability distribution function of \(x\) and \(y\). Suppose \(H(x)\) and \(H(y)\) denote the entropies of \(p(x)\) and \(p(y)\). MI\((x, y)\) varies between 0 and \(\max(H(x), H(y))\). We can use normalized mutual information (NMI) as the second metric for the label vectors \(x\) and \(y\),

\[
\text{NMI}(x, y) = \frac{\text{MI}(x, y)}{\max(H(x), H(y))}.
\]

(4.14)

It is obvious that NMI takes values in \([0, 1]\). Unlike AC, NMI is invariant with the permutation of labels, i.e., NMI does not require a matching between \(x\) and \(y\) in advance.

The visualization comparison of the data clustering results (digit characters 1 ∼ 3 from the USPS database) based on the \(\ell_1\)-graph and those based on the LE-graph and \(K\)-means are depicted in Figure 4.3, which shows that the data are much better separated in our \(\ell_1\)-graph. The quantitative comparison results on clustering accuracy are listed in Tables 4.1 to 4.3 for these three databases, respectively. From the listed results, three observations can be made: (1) the clustering results from the \(\ell_1\)-graph based spectral clustering algorithm are consistently much better than those from other evaluated algorithms for both metrics; (2) \(k\)-nn-graph (LLE) based spectral clustering algorithm is relatively more stable compared with other ones; and (3) \(\epsilon\)-ball based algorithms are shown to be generally worse, in both accuracy and robustness, than the corresponding \(k\)-nn graph based algorithms, and thus for the consequent experiments, we only report the results from \(k\)-nn graph based algorithms. All the results listed in the tables are from the best-tuned algorithmic parameters, e.g., kernel parameter for G-graph, the number of neighboring samples \(k\) and \(\epsilon\) for LE-graphs and LLE-graphs, and the retained feature dimensions for PCA. To further compare the \(\ell_1\)-norm and \(\ell_2\)-norm in graph edge weight deduction, we show the clustering accuracies on the USPS dataset based on the \(\ell_1\)-graph and \(k\)-nn-graph (LLE) with variant \(k\) in Figure 4.4, which shows that \(\ell_1\)-graph is consistently better than the \(\ell_2\)-norm based graph construction for all \(k\)’s. The performance of the latter first increases, and then drops slowly after \(k\) is large enough.
Figure 4.3: Visualization of the data clustering results from (a) the $\ell_1$-graph, (b) the LE-graph, and (c) PCA and $K$-means for three clusters (handwritten digits 1, 2 and 3 in the USPS database). The coordinates of the points in (a) and (b) are obtained from the eigenvalue decomposition in the third step of Section 4.2.1. Different colors of the points indicate different digits. For better viewing, please see the color pdf file.

Figure 4.4: Comparison of clustering accuracies of the $\ell_1$-graph (red line, one fixed value) and ($k$-nn + LLE)-graphs (blue curve) with variant $k$’s on the USPS dataset and the cluster number $K=7$. It shows that $\ell_1$-norm is superior over $\ell_2$-norm in deducing informative graph weights.

4.3.3 Subspace Learning with $\ell_1$-graph

The experiments on classification based on subspace learning are also conducted on the above three databases. To make the comparisons fair, for all the evaluated algorithms we first apply PCA as preprocessing step by retaining 98\% of the energy.

To extensively evaluate the algorithmic performance on the USPS database, we randomly sampled 10, 20, 30, and 40 images from each digit as the training data. Similarly, for the forest covertype database, we randomly sampled 5, 10, 15, and 20 samples from each class as the training data. And for the Extended YALE-B database, we randomly sampled 10, 20, 30, 40, and 50 training images for each individual class. All the remaining data samples are used for testing purposes.
Table 4.1: Clustering accuracies measured by NMI and AC for spectral clustering algorithms based on the $\ell_1$-graph, the Gaussian kernel graph (G-graph), the LE-graphs, the LLE-graphs, and PCA+$K$-means on the USPS digit database. The cluster number $K$ indicates the number of classes used for experiments; we use the first $K$ classes in the database for the data clustering experiments.

<table>
<thead>
<tr>
<th>USPS Cluster #</th>
<th>Metric</th>
<th>$\ell_1$-graph</th>
<th>G-graph</th>
<th>LE-graph</th>
<th>LLE-graph</th>
<th>$K$-means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$k$-nn</td>
<td>$\epsilon$-ball</td>
<td>$k$-nn</td>
<td>$\epsilon$-ball</td>
<td></td>
</tr>
<tr>
<td>$K = 2$</td>
<td>NMI</td>
<td>1.000</td>
<td>0.672</td>
<td>0.858</td>
<td>0.627</td>
<td>0.636</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>1.000</td>
<td>0.922</td>
<td>0.943</td>
<td>0.918</td>
<td>0.917</td>
</tr>
<tr>
<td>$K = 4$</td>
<td>NMI</td>
<td>0.977</td>
<td>0.498</td>
<td>0.693</td>
<td>0.540</td>
<td>0.606</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.994</td>
<td>0.663</td>
<td>0.853</td>
<td>0.735</td>
<td>0.777</td>
</tr>
<tr>
<td>$K = 6$</td>
<td>NMI</td>
<td>0.972</td>
<td>0.370</td>
<td>0.682</td>
<td>0.456</td>
<td>0.587</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.991</td>
<td>0.471</td>
<td>0.739</td>
<td>0.594</td>
<td>0.670</td>
</tr>
<tr>
<td>$K = 8$</td>
<td>NMI</td>
<td>0.945</td>
<td>0.358</td>
<td>0.568</td>
<td>0.371</td>
<td>0.544</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.981</td>
<td>0.423</td>
<td>0.673</td>
<td>0.453</td>
<td>0.598</td>
</tr>
<tr>
<td>$K = 10$</td>
<td>NMI</td>
<td>0.898</td>
<td>0.346</td>
<td>0.564</td>
<td>0.424</td>
<td>0.552</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.873</td>
<td>0.386</td>
<td>0.578</td>
<td>0.478</td>
<td>0.537</td>
</tr>
</tbody>
</table>

Here we use the error rate to measure the classification performance, defined as

$$\text{error rate} = 1 - \frac{\sum_{i=1}^{N_t} \delta(\hat{y}_i - y_i)}{N_t}$$  \hspace{1cm} (4.15)$$

where $\hat{y}_i$ is the predicted sample label, $y_i$ is the ground truth sample label, $N_t$ is the total number of the testing samples, and $\delta(\cdot)$ again is the indicator function.

The best performance of each algorithm over all possible parameters, i.e., graph parameters and feature dimension retained, is reported along with the corresponding feature dimension. The popular unsupervised subspace learning algorithms PCA, NPE, and LPP, and the supervised Fisherfaces algorithm [33] are evaluated for comparison with the $\ell_1$-graph based subspace learning, which is unsupervised. For NPE and LPP, we used their unsupervised versions for fair comparison. For LPP, we use the cosine metric in graph construction to get a better performance.

The detailed comparisons on experimental results for classification are listed in Tables 4.4 to 4.6 for these three databases. From these results, we observe: (1) on the forest covertype and Extended YALE-B databases, $\ell_1$-graph based unsupervised subspace learning algorithm generally performs better than the supervised algorithm Fisherfaces, and on the USPS database, Fisherfaces shows a little better than the former; (2) the $\ell_1$-graph based subspace learning algorithm is much superior over all the other evaluated unsupervised subspace learning algorithms; and (3) NPE and LPP show to be better than PCA. Note that for all the classifi-
Table 4.2: Clustering accuracies measured by NMI and AC for spectral clustering algorithms based on the $\ell_1$-graph, the Gaussian kernel graph (G-graph), the LE-graphs, the LLE-graphs, and PCA+$K$-means on the forest covertype database.

<table>
<thead>
<tr>
<th>COV</th>
<th>Metric</th>
<th>$\ell_1$-graph</th>
<th>G-graph</th>
<th>LE-graph</th>
<th>LLE-graph</th>
<th>$K$-means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$k$-nn</td>
<td>$\epsilon$-ball</td>
<td>$k$-nn</td>
<td>$\epsilon$-ball</td>
<td>$k$-nn</td>
</tr>
<tr>
<td>$K = 3$</td>
<td>NMI</td>
<td>0.792</td>
<td>0.651</td>
<td>0.554</td>
<td>0.419</td>
<td>0.642</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.903</td>
<td>0.767</td>
<td>0.697</td>
<td>0.611</td>
<td>0.813</td>
</tr>
<tr>
<td>$K = 4$</td>
<td>NMI</td>
<td>0.706</td>
<td>0.585</td>
<td>0.533</td>
<td>0.534</td>
<td>0.622</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.813</td>
<td>0.680</td>
<td>0.608</td>
<td>0.613</td>
<td>0.782</td>
</tr>
<tr>
<td>$K = 5$</td>
<td>NMI</td>
<td>0.623</td>
<td>0.561</td>
<td>0.515</td>
<td>0.451</td>
<td>0.556</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.662</td>
<td>0.584</td>
<td>0.541</td>
<td>0.506</td>
<td>0.604</td>
</tr>
<tr>
<td>$K = 6$</td>
<td>NMI</td>
<td>0.664</td>
<td>0.562</td>
<td>0.545</td>
<td>0.482</td>
<td>0.602</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.693</td>
<td>0.585</td>
<td>0.564</td>
<td>0.523</td>
<td>0.632</td>
</tr>
<tr>
<td>$K = 7$</td>
<td>NMI</td>
<td>0.763</td>
<td>0.621</td>
<td>0.593</td>
<td>0.452</td>
<td>0.603</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.795</td>
<td>0.642</td>
<td>0.629</td>
<td>0.498</td>
<td>0.634</td>
</tr>
</tbody>
</table>

Figure 4.5: Visualization comparison of the subspace learning results. The first 10 basis vectors of (a) PCA, (b) NPE, (c) LPP, and (d) the $\ell_1$-graph are calculated from the face images in the YALE-B database.

cation experiments in this work, we used the classical nearest neighbor classifier [33, 123, 59] for fairly comparing the discriminating power of the derived subspaces from different algorithms. The visualization comparisons of the subspaces learned from the $\ell_1$-graph and those based on PCA, LPP, and NPE are depicted in Figure 4.5, from which we see that bases from PCA show to be most similar to real faces as it is motivated for direct data reconstruction.

4.3.4 Semi-Supervised Learning with $\ell_1$-graph

The semi-supervised learning is driven by the philosophy that the unlabeled data can also convey useful information for the learning process. We again use the three databases for evaluating the effectiveness of the semi-supervised algorithm based on the $\ell_1$-graph by comparing the classification performance with semi-supervised
Table 4.3: Clustering accuracies measured by NMI and AC for spectral clustering algorithms based on the $\ell_1$-graph, the Gaussian kernel graph (G-graph), the LE-graphs, the LLE-graphs, and PCA+$K$-means on the Extended YALE-B database. The G-graph performs poorly in this case; a possible explanation is that the illumination difference dominates the G-graph similarity.

<table>
<thead>
<tr>
<th>YALE-B</th>
<th>Metric</th>
<th>$\ell_1$-graph</th>
<th>G-graph</th>
<th>LE-graph</th>
<th>LLE-graph</th>
<th>$K$-means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$k$-nn</td>
<td>$\epsilon$-ball</td>
<td>$k$-nn</td>
<td>$\epsilon$-ball</td>
<td></td>
</tr>
<tr>
<td>$K = 10$</td>
<td>NMI</td>
<td>0.738</td>
<td>0.07</td>
<td>0.420</td>
<td>0.354</td>
<td>0.404</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.758</td>
<td>0.175</td>
<td>0.453</td>
<td>0.413</td>
<td>0.450</td>
</tr>
<tr>
<td>$K = 15$</td>
<td>NMI</td>
<td>0.759</td>
<td>0.080</td>
<td>0.494</td>
<td>0.475</td>
<td>0.438</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.762</td>
<td>0.132</td>
<td>0.464</td>
<td>0.494</td>
<td>0.440</td>
</tr>
<tr>
<td>$K = 20$</td>
<td>NMI</td>
<td>0.786</td>
<td>0.080</td>
<td>0.492</td>
<td>0.450</td>
<td>0.454</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.793</td>
<td>0.113</td>
<td>0.478</td>
<td>0.445</td>
<td>0.418</td>
</tr>
<tr>
<td>$K = 30$</td>
<td>NMI</td>
<td>0.803</td>
<td>0.090</td>
<td>0.507</td>
<td>0.417</td>
<td>0.459</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.821</td>
<td>0.088</td>
<td>0.459</td>
<td>0.383</td>
<td>0.410</td>
</tr>
<tr>
<td>$K = 38$</td>
<td>NMI</td>
<td>0.776</td>
<td>0.110</td>
<td>0.497</td>
<td>0.485</td>
<td>0.473</td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td>0.785</td>
<td>0.081</td>
<td>0.443</td>
<td>0.445</td>
<td>0.408</td>
</tr>
</tbody>
</table>

Table 4.4: USPS digit recognition error rates (%) for different subspace learning algorithms. Note that the numbers in the parentheses are the feature dimensions retained with the best accuracies.

<table>
<thead>
<tr>
<th>USPS</th>
<th>Unsupervised</th>
<th>Supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train #</td>
<td>PCA</td>
<td>NPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>37.21(17)</td>
<td>33.21(33)</td>
</tr>
<tr>
<td>20</td>
<td>30.59(26)</td>
<td>27.97(22)</td>
</tr>
<tr>
<td>30</td>
<td>26.67(29)</td>
<td>23.46(42)</td>
</tr>
<tr>
<td>40</td>
<td>23.25(25)</td>
<td>20.86(18)</td>
</tr>
</tbody>
</table>

Table 4.5: Forest cover recognition error rates (%) for different subspace learning algorithms.

<table>
<thead>
<tr>
<th>COV</th>
<th>Unsupervised</th>
<th>Supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train #</td>
<td>PCA</td>
<td>NPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>33.23(17)</td>
<td>28.80(6)</td>
</tr>
<tr>
<td>10</td>
<td>27.29(18)</td>
<td>25.56(11)</td>
</tr>
<tr>
<td>15</td>
<td>23.75(14)</td>
<td>22.69(16)</td>
</tr>
<tr>
<td>20</td>
<td>21.03(29)</td>
<td>20.10(10)</td>
</tr>
</tbody>
</table>

learning algorithms based on the Gaussian kernel graph, the LE-graph, and the LLE-graph. For all the semi-supervised learning algorithms, the supervised learning term is based on the Marginal Fisher Analysis (MFA) \cite{28} algorithm. And the classification error rate is used to measure the performances. For a fair comparison, the parameters $k_1$, $k_2$, and $\gamma$ in MFA are tuned for all proper combinations, and the results reported are based on the best parameter combination. The detailed
Table 4.6: Face recognition error rates (%) for different subspace learning algorithms on the Extended YALE-B database.

<table>
<thead>
<tr>
<th>YALE-B</th>
<th>Unsupervised</th>
<th>Supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCA</td>
<td>NPE</td>
</tr>
<tr>
<td>Train #</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>44.41(268)</td>
<td>23.41(419)</td>
</tr>
<tr>
<td>20</td>
<td>27.17(263)</td>
<td>14.62(317)</td>
</tr>
<tr>
<td>30</td>
<td>20.11(254)</td>
<td>9.40(485)</td>
</tr>
<tr>
<td>40</td>
<td>16.98(200)</td>
<td>5.84(506)</td>
</tr>
<tr>
<td>50</td>
<td>12.68(366)</td>
<td>3.78(488)</td>
</tr>
</tbody>
</table>

comparison experiment results for semi-supervised leaning algorithms based on different graphs, the original supervised algorithm, and the baseline of PCA, are shown in Tables 4.7 to 4.9. We have two observations: (1) the ℓ₁-graph based semi-supervised learning algorithm generally achieves the highest classification accuracies compared to those of the semi-supervised learning based on the traditional graphs; and (2) semi-supervised learning can generally bring accuracy improvements compared to the counterparts without harnessing extra information from the unlabeled data.

Table 4.7: USPS digit recognition error rates (%) for different semi-supervised, supervised and unsupervised learning algorithms. Note that the numbers in the parentheses are the feature dimensions retained with the best accuracies.

<table>
<thead>
<tr>
<th>USPS</th>
<th>Semi-supervised</th>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ℓ₁-graph</td>
<td>LLE-graph</td>
<td>LE-graph</td>
</tr>
<tr>
<td>Train #</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>25.11(33)</td>
<td>34.63(9)</td>
<td>30.74(33)</td>
</tr>
<tr>
<td>20</td>
<td>26.94(41)</td>
<td>41.38(39)</td>
<td>30.39(41)</td>
</tr>
<tr>
<td>30</td>
<td>23.25(49)</td>
<td>36.55(49)</td>
<td>27.50(49)</td>
</tr>
<tr>
<td>40</td>
<td><strong>19.17</strong>(83)</td>
<td>30.28(83)</td>
<td>23.55(83)</td>
</tr>
</tbody>
</table>
Table 4.8: Forest cover recognition error rates (%) for different semi-supervised, supervised and unsupervised learning algorithms. The numbers in the parentheses are the feature dimensions retained with the best accuracies.

<table>
<thead>
<tr>
<th>COV</th>
<th>Train #</th>
<th>Semi-supervised</th>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\ell_1)-graph</td>
<td>LLE-graph</td>
<td>LE-graph</td>
</tr>
<tr>
<td>5</td>
<td>22.50(9)</td>
<td>29.89(5)</td>
<td>25.81(7)</td>
<td>29.89(5)</td>
</tr>
<tr>
<td>10</td>
<td>17.45(10)</td>
<td>24.93(10)</td>
<td>27.74(8)</td>
<td>24.93(10)</td>
</tr>
<tr>
<td>20</td>
<td>15.00(8)</td>
<td>19.17(10)</td>
<td>17.38(9)</td>
<td>19.17(10)</td>
</tr>
<tr>
<td>30</td>
<td>12.26(8)</td>
<td>15.32(8)</td>
<td>13.81(10)</td>
<td>16.40(8)</td>
</tr>
</tbody>
</table>

Table 4.9: Face recognition error rates (%) for different semi-supervised, supervised and unsupervised learning algorithms on the Extended YALE-B database. The numbers in the parentheses are the feature dimensions retained with the best accuracies.

<table>
<thead>
<tr>
<th>YALE-B</th>
<th>Train #</th>
<th>Semi-supervised</th>
<th>Supervised</th>
<th>Unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(\ell_1)-graph</td>
<td>LLE-graph</td>
<td>LE-graph</td>
</tr>
<tr>
<td>5</td>
<td>21.63(51)</td>
<td>33.47(51)</td>
<td>33.47(51)</td>
<td>33.47(51)</td>
</tr>
<tr>
<td>10</td>
<td>9.56(61)</td>
<td>18.39(33)</td>
<td>18.39(33)</td>
<td>18.39(33)</td>
</tr>
<tr>
<td>20</td>
<td>5.05(57)</td>
<td>14.30(29)</td>
<td>11.26(53)</td>
<td>14.30(29)</td>
</tr>
<tr>
<td>30</td>
<td>2.92(73)</td>
<td>9.15(70)</td>
<td>7.37(71)</td>
<td>11.06(70)</td>
</tr>
</tbody>
</table>

4.4 Conclusion

In machine learning, the graph construction procedure essentially determines the potentials of those graph-oriented learning algorithms for image analysis. We address the graph construction problem as one of finding the sparse representation of each datum with respect to the dictionary composed of the remaining data samples. The sparse representation coefficients, which have been empirically shown to be informative for classification purposes, are used directly to determine the edge weights between the current datum and all the remaining data samples. Such a graph construction procedure is based on the assumption that natural high-dimensional signals lie in a union of low-dimensional linear subspaces. A series of new algorithms for various machine learning tasks, e.g., data clustering, subspace learning, and semi-supervised learning, are then derived based on this new graph. Compared with with the conventional \(k\)-nearest-neighbor graph and the \(\epsilon\)-ball graph, we demonstrate that our new graph possesses three advantages: (1) robustness to noise; (2) automatic sparsity; and (3) adaptive neighborhood selec-
tion. Extensive experiments on diverse real-world datasets show the consistent superiority of our new graph over those classical graphs in clustering, subspace learning, and semi-supervised learning tasks.
CHAPTER 5
BILEVEL SPARSE MODELING

5.1 Introduction

Signal processing and pattern recognition techniques commonly require meaningful data representations that capture relevant properties of signals, e.g., in compression, the representation should account for the essential content of signal with only a few coefficients. Representations with orthogonal and bi-orthogonal dictionaries were prevalent in signal processing techniques during the past decade, due to their mathematical simplicity and computational efficiency, e.g., wavelets for compression (JPEG2000) and denoising [126]. Despite their simplicity, these dictionaries are limited in expressive power, leading to the recent development of over-complete dictionaries, which have more elementary signal atoms than the signal dimension and thus offer the flexibility to represent a much wider range of signal phenomena [72].

Sparse and redundant data modeling seeks the representation of signals as linear combinations of a small number of atoms from a pre-specified dictionary. Recently, there is a fast increasing interest in dictionary training—using machine learning techniques to learn an adaptive over-complete dictionary directly from data. Most of these algorithms focus on \( \ell_0 \)- and \( \ell_1 \)-sparse penalty measures, which lead to simple formulations and allow the use of recently developed efficient sparse coding techniques. Examples include the Method of Optimal Directions (MOD) with the \( \ell_0 \)-sparsity measure proposed by Engan et al. [73], the more efficient K-SVD algorithm by Aharon et al. [12], an efficient formulation with the \( \ell_1 \)-sparsity measure by Lee et al. [75], and an online large-scale learning algorithm by Mairal et al. [127]. The main advantage of trained dictionaries is that they are adaptive to the signal of interest, which induces state-of-the-art performances on many signal recovery tasks, e.g., denoising [12], inpainting [128], and super-resolution [129].
Existing dictionary learning methods mainly focus on training a dictionary to efficiently represent the signal space sparsely, without considering the later tasks that are based on these sparse representations. In many applications and scenarios, the sparse representations, on which those algorithms are built, are inarguably important. Therefore, learning the dictionaries that promote such sparse representations is critically important. It is desirable to learn dictionaries, which not only can sparsely represent the signal space well, but also generate more meaningful sparse representations for the tasks to be performed. For example, our image classification work in Chapter 2, the super-resolution algorithm in Chapter 3, and compressive sensing are all based on the sparse representations or sparse codes of the signals in terms of some dictionaries. Indeed, supervised dictionary training for image classification has been explored by several previous works including [25, 65, 24] and ours [27], and remarkable improvements have been obtained. It is therefore expected that for other tasks, such as sparse recovery, which are based on sparse coding, training a task-specific dictionary would be also beneficial. In this chapter, we target a generic dictionary learning model for learning more informative sparse representations for different tasks. Specifically, we apply this generic dictionary learning model to image classification, super-resolution, and compressive sensing. We formulate the algorithm as a bilevel optimization problem, which can be solved by a justified stochastic gradient descent procedure.

In this chapter, we first briefly review the standard sparse coding algorithm in Section 5.2 to make the chapter self-explaining. Then in Section 5.3, we propose our generic bilevel sparse coding model of learning dictionaries for various tasks. The learning algorithm is formulated as a bilevel optimization problem, which we can solve efficiently using the justified stochastic gradient descent. In Section 5.4, we apply our generic model to three different applications, i.e., image classification based on pooling sparse codes, compressive sensing, and single image super-resolution, and demonstrate significant improvements over previous sparse coding approaches.

5.2 Sparse Coding in a Single Feature Space

In this section, we briefly review the sparse coding algorithm in a feature space for learning dictionaries adapted to the signal, which already appear several times in previous chapters. The goal of sparse coding is to represent an input signal $\mathbf{x} \in \mathbb{R}^d$
approximately as a weighted linear combination of a few elementary signals called basis atoms, often chosen from an over-complete dictionary \( D \in \mathbb{R}^{d \times K} \) \((d < K)\).

Sparse coding is the method to automatically discover such a good set of basis atoms. Concretely, given the training data \( \{x_i\}_{i=1}^N \), the problem of learning a dictionary for sparse coding, in its most popular form, is solved by minimizing the energy function that combines square reconstruction errors and \( \ell_1 \)-sparsity penalties on the representations:

\[
\min_{D, \{\alpha_i\}_{i=1}^N} \sum_{i=1}^N \frac{1}{2} \|x_i - D\alpha_i\|_2^2 + \lambda \|\alpha_i\|_1
\]

\[\text{s.t. } \|D(:, k)\|_2 \leq 1, \ \forall k \in \{1, 2, ..., K\}, \]

where \( D(:, k) \) is the \( k \)-th column of \( D \) and \( \lambda \) is a parameter controlling the sparsity penalty. The above optimization problem is convex with either \( D \) or \( \{\alpha_i\}_{i=1}^N \) fixed, but not with both. When \( D \) is fixed, inference for \( \{\alpha_i\}_{i=1}^N \) is known as the Lasso problem in the statistics literature; when \( \{\alpha_i\}_{i=1}^N \) are fixed, solving \( D \) becomes a standard quadratically constrained quadratic programming (QCQP) problem. A practical solution to Equation (5.1) is to alternatively optimize over \( D \) and \( \{\alpha_i\}_{i=1}^N \), and the algorithm is guaranteed to converge to a local minimum [75].

The above sparse coding algorithm in a single feature space only cares about the representation efficiency, regardless of the specific tasks to be performed. For many applications, such as single image super-resolution, compressive sensing, and texture transfer, learning the dictionary for sparse representations is critical. In the following section, we will discuss learning more meaningful dictionaries for specific tasks by a bilevel formulation.

### 5.3 Bilevel Sparse Coding for Specific Tasks

In this section, we propose our generic bilevel sparse coding model for specific tasks. To keep the model general, we do not specify the form of the cost function for now, which will be defined depending on the applications.
5.3.1 The Learning Model

Suppose we are interested in modeling the feature space $\mathcal{X} \in \mathbb{R}^d$, where the signals are sparse in its high-dimensional space, i.e., the signals have sparse representations in terms of some dictionary or basis. For many specific tasks, we can build the model based on these sparse representations, e.g., image classification or sparse recovery. Given the training samples $\{x_i\}_{i=1}^N$, our bilevel sparse coding model aims to learn a dictionary for these signals such that their sparse representations are more meaningful for the task to be performed. Concretely, we formulate our sparse coding model as a bilevel optimization problem:

$$
\min_{D, \Theta} \frac{1}{N} \sum_{i=1}^N F(z_i, D, \Theta)
$$

s.t. $z_i = \arg \min_\alpha \| \alpha \|_1$, s.t. $\| x_i - D \alpha \|_2^2 \leq \epsilon, \forall i,$

$$
\| D(k,:) \|_2 \leq 1, \forall k,
G(\Theta) \leq 0,
$$

where $D$ is the dictionary that can sparsely represent the signals in $\mathcal{X}$, $F$ is some smooth cost function, and $\Theta$ is the model parameter defined by the specific tasks. For example, $F(z_i, D, \Theta)$ can be defined as a regression task based on the sparse code $z_i$ in super-resolution and compressive sensing, or it can be defined for classification purposes, which we will discuss in detail later. It should be noted that the above formulation is not limited in modeling only a single feature space; it can also learn dictionaries in coupled feature spaces. For instance, we could easily develop a coupled sparse coding model as

$$
\min_{D_x, D_y} \frac{1}{N} \sum_{i=1}^N \| z_x - z_y \|_2^2
$$

s.t. $z_x^i = \arg \min_\alpha \| \alpha \|_1$, s.t. $\| x_i - D_x \alpha \|_2^2 \leq \epsilon_x, \forall i,$

$$
z_y^i = \arg \min_\alpha \| \alpha \|_1$, s.t. $\| y_i - D_y \alpha \|_2^2 \leq \epsilon_y, \forall i,$

$$
\| D_x(k,:) \|_2 \leq 1, \forall k,
\| D_y(k,:) \|_2 \leq 1, \forall k.
$$

The above coupled sparse coding model aims to learn two dictionaries for the coupled spaces $\mathcal{X}$ and $\mathcal{Y}$, such that for each signal pair $\{x_i, y_i\}$ tied by some
(unknown) mapping function, their sparse representations in terms of their own
dictionary are consistent. As many applications involve coupled feature spaces,
*e.g.*, low- and high-resolution patch spaces in super-resolution, and source and
target patch spaces in texture transfer, the above coupled sparse coding model
could potentially be useful.

### 5.3.2 The Learning Algorithm

The problem in Equation (5.2) is a bilevel optimization problem [130], where opti-
mization problems ($\ell_1$-norm minimizations in this case) appear in the constraints.
In our problem, the upper-level problem $F$ selects the dictionaries $D$, and the
lower-level $\ell_1$-norm minimization returns the sparse code $z$ to the upper-level $F$ in
order to evaluate the objective function value. Being generically non-convex and
non-differentiable, bilevel optimization programs are intrinsically difficult [130].
Even for the “simplest” instance, the linear-linear bilevel program was shown to
be NP-hard [131]. In this section, we develop an efficient optimization procedure
based on the first-order projected stochastic gradient descent, which turns out to
be quite effective in practice, as we will see in the experimental part.

**Formulation** A large class of approaches for solving the bilevel optimization
problem is based on the descent method [130]. In Problem (5.2), $z$ is the output
of the lower-level $\ell_1$-norm minimization based on $D$. Assuming that we can define
$z$ as an implicit function $z(D)$ of $D$ depending on the inputs $x_i$ and $y_i$, Problem
(5.2) may be viewed solely in terms of the upper-level variables $D$. Given a feasi-
ble point for $D$, the descent method makes an attempt to find a feasible (descent)
direction along which the upper-level objective decreases. The major issue about
the descent method is the availability of the gradient of the upper-level objective,
$\nabla F_D$, at a feasible point. Applying the chain rule, we have, whenever $\partial z_i/\partial D$
are well defined:

$$(\nabla F_i)_D = \frac{\partial F_i}{\partial D} + \frac{\partial F_i}{\partial z_i} \frac{\partial z_i}{\partial D},$$

where the function is evaluated at the current iteration. However, there is no ana-
lytical link between $z_i$ and $D$ for direct evaluation of $\partial z_i/\partial D$. In the following
section, we will see that the sparse code $z_i$ is almost differentiable with respect
to its depending dictionary $D$, and thus the gradients can be evaluated by implicit
differentiation [25, 27].

**Derivatives in the \( \ell_1 \)-norm minimization**  Note that the \( \ell_1 \)-norm minimization problem in Equation (5.2) can be equivalently reformulated as an unconstrained optimization problem for a properly chosen \( \lambda \)

\[
z = \arg \min_\alpha \| x - D\alpha \|_2^2 + \lambda \| \alpha \|_1, \tag{5.5}
\]

known as the Lasso in the statistics literature [132]. We denote \( \Lambda \) as the active set of the Lasso solution \( z \), i.e., \( \Lambda = \{ k : z(k) \neq 0 \} \), for the following discussion.

In order to compute the gradient of \( z \) with respect to \( D \), we first introduce the following lemmas.

**Lemma 1.** For a given response vector \( x \), there is a finite sequence of \( \lambda \)'s, \( \lambda_0 > \lambda_1 > \cdots > \lambda_K = 0 \), such that if \( \lambda \) is in the interval of \( (\lambda_m, \lambda_{m+1}) \), the active set \( \Lambda \) and sign vector \( \text{Sgn}(z_\Lambda) \) are constant with respect to \( \lambda \).

These characteristics of the Lasso solution have been shown by Efron et al. [132]. The active set changes at \( \{ \lambda_m \} \); hence, they are called transition points [133]. Any \( \lambda \in [0, \infty) \setminus \{ \lambda_m \} \) is called a nontransition point.

**Lemma 2.** \( \forall \lambda, z \) is a continuous function of \( D \) and \( x \).

Instead of a formal proof, we simply state that function \( f(x, \alpha, D) = \| x - D\alpha \|_2^2 + \lambda \| \alpha \|_1 \) is continuous in \( x \), \( \alpha \) and \( D \), and thus \( z \) is a continuous function of \( x \) and \( D \) [133], [134], which can be easily proven by contradiction.

**Lemma 3.** Fix any \( \lambda > 0 \), and \( \lambda \) is not a transition point for \( x \), the active set \( \Lambda \) and the sign vector \( \text{Sgn}(z_\Lambda) \) are locally constant with respect to both \( x \) and \( D \).

**Proof.** Fixing \( D \in \mathbb{R}^{d \times K} \), it is easy to show that \( \Lambda \) and \( \text{Sgn}(z_\Lambda) \) are locally constant with respect to \( x \), given that \( \lambda \) is not a transition point for \( x \). The proof based on Lemma 2 and the equiangular conditions [132] is given in [133]. Therefore, for the given signal \( x \in \mathbb{R}^d \), there exists a \( d \)-dimensional Ball(\( x \), \( \varepsilon \)) with center \( x \) and radius \( \varepsilon \), such that \( \Lambda \) and \( \text{Sgn}(z_\Lambda) \) are constant.

Now, fix the signal vector \( x \). Denote by Ball(\( D \), \( \varepsilon \)) the \( dK \)-dimensional ball with center \( D \) and radius \( \varepsilon \). Consider a perturbation \( E \) on the dictionary \( D \) such that \( D_\varepsilon = D + E \in \text{Ball}(D, \varepsilon) \). Its Lasso formulation for \( x \) is

\[
\min_\alpha \| x - D_\varepsilon \alpha \|_2^2 + \lambda \| \alpha \|_1, \tag{5.6}
\]
which we reformulate as

\[ \min_{\alpha} \left\| (x - E\alpha) - D\alpha \right\|_2^2 + \lambda \|\alpha\|_1. \quad (5.7) \]

Denote \( x_e = x - E\alpha \), which is \( x \) plus a perturbation \( -E\alpha \). Since \( \|\alpha\|_2^2 \leq B \) for some upper bound \( B \) and \( \|E\|_2^2 \leq \varepsilon^2 \), based on the Cauchy inequality, the perturbation vector \( \|E\alpha\|_2^2 \leq d\varepsilon^2 B \). Therefore, there exists a sufficiently small \( \varepsilon \), such that for any perturbation vector \( \|E\|_2 \leq \varepsilon \), we have \( \|E\alpha\|_2 \leq \varepsilon \), i.e., \( x_e \in \text{Ball}(x, \varepsilon) \) holds. Based on the local constancy property with respect to \( x \) in the above first step, we conclude that \( \Lambda \) and \( \text{Sgn}(z_\Lambda) \) are also locally constant with respect to \( D \).

For \( \lambda \) being a nontransition point, we have the equiangular conditions [132]:

\[ \frac{\partial \|x - Dz\|_2^2}{\partial z(k)} + \lambda \text{sign}(z(k)) = 0, \text{ for } k \in \Lambda, \quad (5.8) \]

\[ \left| \frac{\partial \|x - Dz\|_2^2}{\partial z(k)} \right| < \lambda, \text{ for } k \notin \Lambda. \quad (5.9) \]

Equation (5.8) is the stationary condition for \( z \) to be optimal, which links \( z \) and \( D \) analytically on the active set \( \Lambda \). We rewrite this condition as

\[ D_\Lambda^T D_\Lambda z_\Lambda - D_\Lambda^T x + \lambda \text{Sgn}(z_\Lambda) = 0, \quad (5.10) \]

where \( D_\Lambda \) consists of the columns of \( D \) in the active set \( \Lambda \). Based on Lemma 3, the active set \( \Lambda \) and sign vector \( \text{Sgn}(z_\Lambda) \) are constant in a local neighborhood of \( D \), and therefore, Equation (5.10) and Equation (5.9) hold for a sufficiently small perturbation of \( D \). Denoting \( \Omega \) as the nonactive set, we can now evaluate the full gradient of \( z \) with respect to \( D \) in three parts:

1. As \( z \) is a continuous function of \( D \), \( \Lambda \) and \( \text{Sgn}(z_\Lambda) \) are locally constant with respect to \( D \), we can apply implicit differentiation to Equation (5.10) to get the partial derivative \( z_\Lambda \) with respect to \( D_\Lambda \),

\[ \frac{\partial z_\Lambda}{\partial D_\Lambda} = (D_\Lambda^T D_\Lambda)^{-1} \left( \frac{\partial D_\Lambda^T x}{\partial D_\Lambda} - \frac{\partial D_\Lambda^T D_\Lambda}{\partial D_\Lambda} z_\Lambda \right). \quad (5.11) \]

\( D_\Lambda^T D_\Lambda \) is well conditioned for inverse if \( z \) is unique. In practice, we find that this is not a problem.
2. As $z_\Lambda$ is only connected with $D_\Lambda$, a perturbation on $D_\Omega$ would not change its value, and therefore, we have
\[
\frac{\partial z_\Lambda}{\partial D_\Omega} = 0. \tag{5.12}
\]

3. As $\Lambda$ and $\text{Sgn}(z_\Lambda)$ are constant for a small perturbation of $D$, $z_\Omega$ stays as zero, so we have
\[
\frac{\partial z_\Omega}{\partial D} = 0. \tag{5.13}
\]

In summary, based on the assumption that $\lambda$ is not a transition point, $\partial z / \partial D$ is very sparse and the only nonzero part is given by $\partial z_\Lambda / \partial D_\Lambda$, making it very efficient to evaluate in practice.

For $\lambda$ being a transition point, the partial derivative $\partial z_\Omega / \partial D$ in Equation (5.13) is not exactly 0 any more. However, we have the following lemma proved in [133].

**Lemma 4.** $\forall \lambda > 0$, $\exists$ a null set $\mathcal{N}_\lambda$ which is a finite collection of hyperplanes in $\mathbb{R}^d$. Then $\forall \mathbf{x} \in \mathbb{R}^d \setminus \mathcal{N}_\lambda$, $\lambda$ is not any of the transition points of $\mathbf{x}$.

Based on this lemma, for a reasonable assumption on the distribution of the input vectors $\mathbf{x}$, the chance that $\lambda$ is the transition point for $\mathbf{x}$ is low and thus is neglectable. On the other hand, from a practical point of view, even if we could not evaluate the full gradient, as long as we find a feasible direction from the partial derivative, we can still decrease the objective function value using the descent method.

**Stochastic Gradient Descent** As the gradient $\partial z_\Omega / \partial D$ is neglectable on expectation, we can evaluate $(\nabla F_i)_D$ in Equation (5.4) based on $\partial z_\Lambda / \partial D_\Lambda$ only for stochastic gradient descent, and convergence can still be achieved [135]. The optimization updating rule is simply
\[
D^{n+1} = D^n - r(n) \frac{\langle \nabla F_D \rangle}{\| \langle \nabla F_D \rangle \|_2}, \tag{5.14}
\]
with
\[
r(n) = \frac{r_0}{(n/N + 1)^p}, \tag{5.15}
\]
where $n$ is the cumulative counts of the data samples fed into the learning algorithm, $N$ is the total number of iterations, $p \leq 1$ controls the shrinkage rate of the step size for stochastic gradient descent, and $r_0$ is the initial learning rate.
Since we have the norm constraints on each dictionary atom, we project the updated dictionary back onto the unitary ball after each update. Furthermore, to ensure that the learned dictionaries can sparsely represent the data samples well, we add the reconstruction constraint \( \|x_i - Dz_i\|_2^2 \) to the cost function \( F \) as an additional regularization. As the optimization problem in Equation (5.2) is highly nonlinear and highly nonconvex, we can only expect this projected first-order stochastic gradient procedure to find a local minimum. In practice, we find that our algorithm is quite efficient and effective with proper initialization.

Up to now, the cost function \( F \) in Equation (5.2) is still not defined, which depends on specific applications. In the following section, we will discuss in detail three applications of the proposed bilevel sparse coding model, i.e., image classification, compressive sensing, and patch-wise single image super-resolution.

### 5.4 Applications

In this section, we apply our generic bilevel sparse coding model discussed above to three different tasks, i.e., image classification, compressive sensing, and single image super-resolution based on sparse recovery. We will show that our learning algorithm is both efficient and effective in practice.

#### 5.4.1 Supervised Dictionary Training for Image Classification

In Section 2.2 of Chapter 2, we proposed a generic feature representation based on max pooling over sparse codes of the local descriptors from different spatial blocks across multiple spatial scales. The dictionary that generates the sparse codes for the local descriptors is trained in a reconstructive and unsupervised manner as in Section 5.2, which is only concerned with reconstruction accuracy instead of classification. Recall from Section 2.2 that the image representation \( \beta \) can be regarded as an implicit function of the dictionary \( D \) and the input image \( X \), and therefore, we can apply our generic sparse coding model in Equation (5.2) for supervised dictionary training. To see this, we first review the feature representation extraction process in the following.

1. To encode the spatial information, we represent the image as a collection of
local image descriptors in a spatial pyramid structure:

$$X = [Y_{11}^0, Y_{11}^1, Y_{12}^1, \ldots, Y_{44}^2],$$ (5.16)

where $Y_{ij}^s$ is the collection of local descriptors from the $(i, j)$-th spatial block in the $s$-th spatial scale.

2. Given the dictionary $D$ for the local descriptor space, we can transform each local descriptor $x$ into the corresponding sparse code via $\ell_1$-norm minimization. Therefore, the sets of local descriptors are transformed into sets of sparse codes in the same spatial pyramid structure:

$$Z = [S_{11}^0, S_{11}^1, S_{12}^1, \ldots, S_{44}^2],$$ (5.17)

where $S_{ij}^s$ is the subset of sparse codes for descriptors falling into the $(i, j)$-th spatial block in the $s$-th spatial scale.

3. The final image feature representation is obtained from the max pooling features from each subset of sparse codes in the spatial pyramid.

$$\beta = \bigcup_{s=0}^{2} \bigcup_{i,j=1}^{2^s} [\beta_{ij}^s]$$ (5.18)

with

$$\beta_{ij}^s = \max(|S_{ij}^s|).$$ (5.19)

where $\bigcup$ means concatenation, and the “max” operation is performed along each row of $|S_{ij}^s|$.

Now we can introduce our supervised dictionary training algorithm based on our bilevel sparse coding model. Given the training images with labels $\{X_i, y_i\}_{i=1}^N$, where $X_i$ represents the $i$-th image and $y_i$ is its label, we can formulate the supervised dictionary learning algorithm in the following way fitting into our generic
sparse coding model:

$$\min_{D, w} \sum_{i=1}^{N} \ell(y_i, f(\beta_i, w)) + \gamma \|w\|^2_2$$

s.t. \( \beta_i = \text{pooling}(Z_i) \)

$$Z_i = \arg \min_{Z} \|X_i - DZ\|^2_2 + \lambda \|Z\|_1$$

\[ \|D(:,k)\|_2 \leq 1, \forall k, \] (5.20)

where \( \ell(\cdot) \) is a differentiable loss function and \( f \) is the prediction model depending on the parameter \( w \). In our work, we use the squared hinge loss function for \( \ell \) and a linear classification model for \( f \), but other choices of the loss function and prediction model can also be used.

For ease of presentation, we denote the objective function as \( E(D, w, \{X_i\}_{i=1}^{N}) \).

Minimizing the energy function in terms of \( D \) and \( w \), we are training our classifier together with the dictionary, and thus we can expect the dictionary learned will be more effective for classification. Once the dictionary \( D \) is fixed, optimization for \( w \) is simply training the classifier. Fixing the classifier \( w \), optimization for \( D \) follows the previous procedure developed for our bilevel sparse coding model. First, we try to find the gradient of \( E \) with respect to \( D \) using the chain rule,

$$\frac{\partial E}{\partial D} = \sum_{i=1}^{N} \frac{\partial \ell}{\partial D} = \sum_{i=1}^{N} \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial D} = \sum_{i=1}^{N} \frac{\partial \ell}{\partial f} \frac{\partial f}{\partial \beta_i} \frac{\partial \beta_i}{\partial D}$$

(5.21)

Computing the derivatives of the sparse representation matrix with respect to the dictionary \( D \) follows the implicit differentiation approach discussed in Section 5.3.2. Evaluating the derivative of \( \beta_i \) with respect to \( Z_i \) is a little tricky, as \( \beta_i \) is obtained by “max” pooling over \( Z_i \) in a spatial pyramid structure but the “max” operation is not differentiable. Nevertheless, in order to compute the derivative, we assume the indexes of the maximum elements do not change due to a small perturbation of the dictionary. In practice, we find that such an assumption works rather well with a proper initialization of the dictionary.

We apply our supervised dictionary learning model to face recognition, handwritten digit recognition, and gender recognition on previously reported benchmark datasets. In the following, we re-introduce these datasets and report the new
results in comparison with previous results based on the unsupervised dictionary training.

**CMU PIE** The database consists of 41,368 images of 68 people, each person under 13 poses, 43 different illumination conditions, and with 4 different expressions. We use a subset of the same database as in [88, 29] for fair comparison. The subset only contains five near-frontal views (C05, C07, C09, C27, C29) and all the images under different illuminations and expressions. Therefore, there are 170 images for each individual. A random subset of \( p \) (\( p = 30, 50, 70, 90, 130 \)) images per person is selected as the training set and the remaining of the database is considered as the test set.

![Cost function value](image1.png) ![Classification error](image2.png)

**Figure 5.1:** The supervised optimization process on CMU PIE for 10 iterations.

Figure 5.1 shows the optimization process of supervised training for 10 iterations for \( p = 50 \). For each iteration, we record its cost function value, and also evaluate the performance with the current learned dictionary on the testing set. As expected, the classification error decreases as the cost function value decreases. Table 5.1 shows the performance comparisons with the literature on this dataset. “Improvement” shows the improvement from the unsupervised dictionary to the supervised dictionary for our algorithm. As shown, both our unsupervised and supervised sparse coding algorithm significantly outperform S-LDA [29], reported as the state-of-the-art performance algorithm on this database, reducing the error rate by more than 10 times.

**CMU Multi-PIE** The second experiment on face recognition is conducted on the large-scale CMU Multi-PIE database [82]. The database contains 337 sub-
Table 5.1: Classification error (%) on CMU PIE database for different algorithms. “Improvement” shows the improvement from unsupervised sparse coding (U-SC) to supervised sparse coding (S-SC) in terms of reducing the error rate.

<table>
<thead>
<tr>
<th>Training</th>
<th>30</th>
<th>50</th>
<th>70</th>
<th>90</th>
<th>130</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>7.9</td>
<td>4.8</td>
<td>4.0</td>
<td>3.4</td>
<td>2.9</td>
</tr>
<tr>
<td>R-LDA [88]</td>
<td>5.0</td>
<td>3.9</td>
<td>3.5</td>
<td>3.2</td>
<td>3.0</td>
</tr>
<tr>
<td>S-LDA [29]</td>
<td>3.6</td>
<td>2.5</td>
<td>2.1</td>
<td>1.8</td>
<td>1.6</td>
</tr>
<tr>
<td>U-SC</td>
<td>0.81</td>
<td>0.26</td>
<td>0.22</td>
<td>0.110</td>
<td>0.037</td>
</tr>
<tr>
<td>S-SC</td>
<td>0.49</td>
<td>0.15</td>
<td>0.12</td>
<td>0.037</td>
<td>0.000</td>
</tr>
<tr>
<td>Improvement</td>
<td>39.5%</td>
<td>42.3%</td>
<td>45.5%</td>
<td>66.4%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Subjects across simultaneous variations in pose, expression, and illumination. Some example images are shown in Figure 2.6. In order to compare with [17] fairly, we use the same experiment settings for face recognition. Of these 337 subjects, 249 subjects present in Session 1 are used as the training set. Sessions 2, 3, and 4 are used as testing. The remaining 88 subjects are considered “outliers” or invalid images in [17] for face verification. But in this work we neglect them and only care about face recognition. For the training set, [17] only included seven frontal extreme illuminations, taken with neutral expression. We use exactly the same training set. For the test set, all 20 illuminations from Sessions 2 to 4 are used, which were recorded at distinct times over a period of several months. The dataset is challenging due to the large number of subjects and natural variations in subject appearance over time.

Table 5.2: Face recognition errors (%) on large-scale Multi-PIE. “Improvement” row shows the improvement due to supervised training.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Session 2</th>
<th>Session 3</th>
<th>Session 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>50.6</td>
<td>55.7</td>
<td>52.1</td>
</tr>
<tr>
<td>NN</td>
<td>32.7</td>
<td>33.8</td>
<td>37.2</td>
</tr>
<tr>
<td>NS</td>
<td>22.4</td>
<td>25.7</td>
<td>26.6</td>
</tr>
<tr>
<td>SR</td>
<td>8.6</td>
<td>9.7</td>
<td>9.8</td>
</tr>
<tr>
<td>U-SC</td>
<td>5.4</td>
<td>9.0</td>
<td>7.5</td>
</tr>
<tr>
<td>S-SC</td>
<td>4.8</td>
<td>6.6</td>
<td>4.9</td>
</tr>
<tr>
<td>Improvement</td>
<td>11.1%</td>
<td>26.7%</td>
<td>34.7%</td>
</tr>
</tbody>
</table>

Table 5.2 shows our results compared with those reported in the [17] for Linear Discriminant Analysis (LDA) [33], Nearest Neighbor (NN), Nearest Subspace (NS) [89], and Sparse Representation (SR). Our supervised sparse coding strategy
significantly reduces the error rates of SR by 41.9%, 32.0%, and 50.0% for sessions 2, 3, and 4, respectively.

**Handwritten Digit Recognition**  We also test our algorithm on the benchmark MNIST handwritten digit image dataset [84]. The database consists of 70,000 handwritten digits, of which 60,000 digits are modeled as training and 10,000 as testing. The digits have been size-normalized and centered in a fixed-size image. The supervised training optimization process converges quickly and we stop at five iterations. Table 5.3 shows the performance comparisons with other methods reported on the dataset. The supervised training reduces the error of the unsupervised model by 14.3% and achieves performance similar to Convolutional Neural Network (CNN) under the same condition, which is known to be the best algorithm on the MNIST dataset.

**Table 5.3:** Classification error (%) comparison with state-of-the-art algorithms in the literature on the MNIST dataset.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (RBF)</td>
<td>1.41</td>
</tr>
<tr>
<td>L1 sparse coding (linear SVM)</td>
<td>2.02</td>
</tr>
<tr>
<td>Local coordinate coding (linear SVM) [69]</td>
<td>1.90</td>
</tr>
<tr>
<td>Deep belief network</td>
<td>1.20</td>
</tr>
<tr>
<td>CNN [90]</td>
<td>0.82</td>
</tr>
<tr>
<td>U-SC (linear SVM)</td>
<td>0.98</td>
</tr>
<tr>
<td>S-SC (linear SVM)</td>
<td>0.84</td>
</tr>
<tr>
<td>Improvement</td>
<td>14.3%</td>
</tr>
</tbody>
</table>

**Gender Recognition**  Our gender recognition experiment is conducted on the FRGC 2.0 dataset [83]. This dataset contains 568 individuals, totalling 14714 face images under various lighting conditions and backgrounds. Besides person identities, each image is annotated with gender and ethnicity. For gender recognition, we fix 114 persons’ 3014 images (randomly chosen) as the test set, and the remaining 451 individuals’ 11700 images as our training images. Comparisons are performed with the state-of-the-art algorithms on FRGC in the same experiment setting as reported in Table 5.4. The supervised sparse coding strategy boosts the performance by 22.1% error reduction compared with the unsupervised version, outperforming the top performance algorithm CNN.
Table 5.4: Classification error (%) comparison with state-of-the-art gender recognition algorithms in the literature on the FRGC 2.0 dataset.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (RBF)</td>
<td>8.6</td>
</tr>
<tr>
<td>CNN [90]</td>
<td>5.9</td>
</tr>
<tr>
<td>Unsupervised Sparse Coding</td>
<td>6.8</td>
</tr>
<tr>
<td>Supervised Sparse Coding</td>
<td>5.3</td>
</tr>
<tr>
<td><strong>Improvement</strong></td>
<td><strong>22.1%</strong></td>
</tr>
</tbody>
</table>

5.4.2 Compressive Sensing

Compressive sensing is about acquiring a sparse signal in the most efficient way possible with the help of an incoherent projecting basis [2]. Unlike traditional sampling methods, compressive sensing provides a new framework for sampling signals in a multiplexed manner [4], and states that sparse signals can be exactly recovered from a number of linear projections of a dimension that is considerably lower than the number of samples required by the Shannon-Nyquist theorem. Compressive sensing relies on two fundamental principles: sparsity of the signal and incoherent sampling. Given $x \in \mathbb{R}^d$ is a $d$-dimensional signal, compressive sensing requires that the signal has a sparse representation in terms of some dictionary $D_x$. Let $\Phi$ be an $m \times d$ sampling matrix ($m \ll d$), such as $y = \Phi x$ is an $m$-dimensional vector of linear measurements of the underlying signal $x$. Compressive sensing requires the sensing matrix $\Phi$ and the sparse representation matrix $D_x$ be as incoherent as possible. The recovery of $x$ from its linear measurement $y$ can be done by $\ell_1$-norm minimization under conditions related to the sparsity of the signal and the incoherence of the signal,

$$z = \arg \min_{\alpha} \|\alpha\|_1, \text{ s.t. } y = \Phi D_x \alpha, \quad x = D_x \alpha. \quad (5.22)$$

Therefore, the choices of the sensing matrix $\Phi$ and the sparse representation matrix $D_x$ are both critical for the success of compressive sensing recovery, especially when only few measurements are available. A method for simultaneously learning the sensing matrix and the sparse representation matrix is proposed in [136] by reducing the mutual coherence of the dictionary. In practice, the sensing matrix is usually constrained by the hardware implementation, and therefore, in this dissertation we fix our sensing matrix, and try to optimize the sparse representation matrix $D_x$. Fitting into the generic bilevel sparse coding model, our
optimization over the dictionary $D_x$ can be formulated in the following:

$$\min_{D_x} \sum_{i=1}^{N} \|x_i - D_x z_i\|_2^2$$

s.t. $y_i = \Phi x_i, \forall i,$

$$z_i = \arg \min_\alpha \|\alpha\|_1, \text{ s.t. } \|y_i - D_y \alpha\|_2^2 \leq \epsilon, \forall i$$

$$D_y = \Phi D_x,$$

$$\|D_x (k, :\|_2 \leq 1, \forall k.$$ (5.23)

It is easy to see that the above model is a special case of the generic model in Equation (5.2) by defining $F = \|x_i - D_x z_i\|_2^2$ and optimizing only over $D_x$.

In this model, instead of improving the conditions from the compressive sensing theory as in [136], we directly optimize the dictionary $D_x$ to achieve low sparse recovery errors.

For experimental evaluation, we randomly sample 10,000 image patches of size $16 \times 16$ pixels for training and 5000 image patches of the same size for testing from medical images. We use the Haar wavelet basis as our baseline, which also serves as the initial dictionary for our optimization. For the sampling matrix, we use the Bernoulli random matrix with sampling rate at 10%, 15%, 20%, 25%, and 30% for the measurements. Figure 5.2 (a) draws how the objective function value drops with the optimization iterations for sampling rate at 10%, which shows that the algorithm converges very fast, typically in 10 iterations (faster than standard sparse coding). Even with only one iteration, the algorithm already provides a reasonable solution. Figure 5.2 (b) demonstrates the average recovery accuracy comparisons on the 5000 testing image patches between the Haar wavelet and our learned dictionary across different sampling rates. In all cases, our learned dictionary outperforms the Haar wavelet basis by a remarkable margin of around 5 dB. This is especially striking for small sampling rates, e.g., our algorithm dramatically reduces the RMSE from 31.8 to 16.8 for sampling rate at 10%. In Figure 5.3, we perform patch-wise sparse recovery on the whole “bone” image for sampling rate at 20%. The result from the Haar wavelet basis shows obvious blocky artifacts, while our result is much more accurate and informative.
Figure 5.2: (a) Objective function value vs. iteration number for 10% sampling rate. The optimization converges very fast. (b) Recovery accuracy comparisons on the test image patches in terms of PSNR for the Harr wavelet basis and our learned dictionary at different sampling rates. In all cases, our learning scheme improves the baseline by around 5 dB.

5.4.3 Single Image Super-Resolution

Image super-resolution is the class of techniques that construct a high-resolution image from one or several low-resolution observations [137]. Among all those techniques, the patch-based single image super-resolution approach is one of the promising approaches for many practical applications. Many previous example-based super-resolution methods [51] apply a non-parametric approach to the super-resolution problem with a large training patch set. Motivated by the recent compressive sensing theories [4], we formulate the problem as patch-wise sparse recovery in Section 3.2. In order to train a compact model, we proposed the joint sparse coding procedure to learn two dictionaries $D_x$ and $D_y$, for high- and low-resolution image patches, respectively, such that the sparse representation $z_y$ of a low-resolution image patch $y$ is the same as the sparse representation $z_x$ of the corresponding high-resolution image patch $x$. For any given testing low-resolution patch $y_i$, the algorithm first finds its sparse representation $z_i$ in terms of $D_y$ using the $\ell_1$-norm minimization, and then recover the underlying high-resolution image patch $x_i$ as $\hat{x}_i = D_x z_i$. In the following, we review this joint sparse coding algorithm and point out its inconsistency problem between learning and testing, which motivates our bilevel sparse coding model for super-resolution, which is described in the following.

**Joint Sparse Coding for Super-Resolution** In Section 3.3, we introduced our joint sparse coding for learning the high- and low-resolution image patch dic-
Figure 5.3: Recovery comparison on the “bone” image with 20% measurements. (a) Ground truth image. (b) Recovery with wavelet basis (22.8 dB). (c) Recovery with learned dictionary (27.6 dB). (d) Recovery PSNR comparison under different sampling rates.

tionaries for super-resolution. The algorithm itself is not constrained to super-resolution only. Formally, joint sparse coding aims to learn the dictionaries $D_x$ and $D_y$ for two given coupled feature spaces, $\mathcal{X}$ and $\mathcal{Y}$, tied by a certain mapping function $\mathcal{F}$, such that the sparse representation of $x_i \in \mathcal{X}$ in terms of $D_x$ should be the same as that of $y_i \in \mathcal{Y}$ in terms of $D_y$, where $y_i = \mathcal{F}(x_i)$. Accordingly, if $y_i$ is our observation signal, we can recover its underlying latent signal $x_i$ via its sparse representation in terms of $D_y$. The joint sparse coding model formulates the problem by generalizing the basic sparse coding scheme as follows:

$$
\min_{D_x, D_y, (\alpha_i)_{i=1}^N} \frac{1}{2} \sum_{i=1}^N \left( \|x_i - D_x \alpha_i\|_2^2 + \|y_i - D_y \alpha_i\|_2^2 \right) + \lambda \|\alpha_i\|_1,
$$

(5.24)

which basically requires that the resulting common sparse representation should reconstruct both $y_i$ and $x_i$ well. However, this joint sparse coding can only be claimed to be optimal in the concatenated feature space of $\mathcal{X}$ and $\mathcal{Y}$, but not in each feature space separately. To see this, we can group the first two reconstruction
errors together by denoting

\[ x_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \quad D = \begin{bmatrix} D_x \\ D_y \end{bmatrix}. \]  

(5.25)

Then Equation (5.24) reduces to a standard sparse coding problem in the concatenated feature space of \( X \) and \( Y \).

For single image super-resolution, we only observe the low-resolution signal \( y_i \), from which we want to find the common sparse representation for recovery of \( x_i \) by

\[ z_i = \min_{\alpha} \frac{1}{2} \| y_i - D_y \alpha_i \|_2^2 + \lambda \| \alpha_i \|_1. \]  

(5.26)

However, comparing with the training procedure in Equation (5.24), the term in the above optimization, \( \frac{1}{2} \| x_i - D_x \alpha_i \|_2^2 \), is missing because \( x_i \) is unknown. As a result of this inconsistency between training and testing, reconstruction accuracy for \( x_i \) is not guaranteed. In the following, our bilevel sparse coding model solves this inconsistency issue and shows remarkable recovery accuracy improvement.

**Bilevel Sparse Coding for Super-Resolution**

Let the signals of low-resolution image patches constitute the observation space \( Y \) and the high-resolution image patches constitute the latent space \( X \). We want to model the mapping between the two spaces by our bilevel sparse coding, and then use the learned dictionaries to recover high-resolution patch \( x \) for any given low-resolution patch \( y \). Following the single image super-resolution routine discussed in Section 3.2, the low-resolution \( y \) is represented by the gradient features of its interpolated version using bicubic interpolation. Different from compressive sensing, the mapping between high- and low-resolution image patches is no longer linear, but complicated and obscure. As a result, the high- and low-resolution dictionaries \( D_x \) and \( D_y \) are no longer linearly related, and have to be defined separately.

Suppose the dictionary \( D_x \) is given\(^2\) to sparsely represent high-resolution signals in \( X \). Our goal is to learn a “coupled” dictionary \( D_y \) over \( Y \), such that the sparse representation \( z \) of any \( y \in Y \) in terms of \( D_y \) can be used to recover its corresponding \( x \in X \) with dictionary \( D_x \) as \( \hat{x} = D_x z \). Formally, the optimization

\(^2\)Either by standard sparse coding or mathematical derivation.
for $D_y$ can be formulated in the following:

$$
\min_{D_y} \sum_{i=1}^{N} \left\| D_i z_i^y - x_i \right\|_2^2
$$

s.t. $z_i^y = \arg \min_{\alpha} \| \alpha \|_1$, s.t. $\| y_i - D_y \alpha \|_2^2 \leq \epsilon$, $\forall i$

$$
\| D_y(k, :) \|_2 \leq 1, \quad \forall k,
$$

(5.27)

where $\{x_i, y_i\}_{i=1}^N$ are training examples randomly sampled from the coupled patch spaces $\{X, Y\}$. Again, the above model is a special case of the bilevel sparse coding model in Equation (5.2) with $f = \| D_i z_i^y - x_i \|_2^2$ and optimization only over $D_y$.

To train the dictionary $D_y$, we sample 100,000 patches from high- and low-resolution image pairs to obtain the training data. The patch size is chosen as $5 \times 5$ to achieve sufficient sparsity while maintaining an affordable dictionary dimension. We use the dictionaries trained from joint sparse coding as the initialization for $D_x$ and $D_y$. The learning algorithm quickly converges in less than five iterations. We compare the results of our bilevel sparse coding model with those of the joint sparse coding model, because the joint sparse coding model already provides the state-of-the-art single super-resolution results.

The recovery accuracy of different sparse coding models is first evaluated on an independent validation set containing 100,000 image patch pairs. Figure 5.4 shows the pixel-wise mean square error (MSE) improvements using our coupled dictionary over the joint dictionary training method. It can be seen that our approach significantly reduces the recovery errors in all pixel locations. Figure 5.4 (b) shows the pixel-wise MSE pattern for the recovered high-resolution image patches using the proposed coupled dictionary training method.

In Figure 5.5, the super-resolution results on “Flower”, “Lena” and “Street” by magnification factor of two are shown for both dictionary training methods. Low-resolution patches are sampled from the test images on a regular grid with some overlapping for recovery. Usually, higher recovery accuracy can be achieved by increasing the amount of overlapping between adjacent patches, but at the expense of increased computation time. As shown, the results given by our method are free of artifacts even with 0, 1, and 2-pixel overlapping; on the contrary, the artifacts introduced by joint dictionary training are always visible.\(^3\) Quantitatively,

\(^3\)Actually, the artifacts of the joint dictionary training method will not disappear until 3-pixel
Figure 5.4: Left: percentages of pixel-wise MSE reduced by our coupled training method compared with joint dictionary training method. Right: MSE pattern for the recovered patches.

Figure 5.6 shows the recovery PSNRs on both “Flower” and “Lena” for different amounts of patch overlapping. For reference, the PSNRs of the bicubic interpolation are also plotted as horizontal dashed lines. In all cases, our method outperforms the other two substantially. More importantly, recovery using our coupled dictionary with 0-pixel patch overlapping can be approximately as good as the one given by joint dictionary with 3-pixel overlapping, reducing the computation time by more than six times. This advantage is especially critical for many real time applications and mobile devices.
Figure 5.5: Super-Resolution results up-scaled by magnification factor of 2, using joint dictionary training (top row) and bilevel sparse coding (bottom row), with 0, 1, and 2-pixel overlapping between adjacent patches for Flower, Lena, and Street image, respectively.

Figure 5.6: Recovery PSNRs using different dictionary training methods with different patch overlappings on the two example images. The bilevel sparse coding method with 0-pixel overlapping can achieve the same level of performance as joint training with 3-pixel overlapping.
5.5 Conclusion

In this chapter, we address the problem of learning meaningful dictionaries for sparse modeling in diverse specific tasks. We propose a generic task-driven sparse coding model formulated as a bilevel optimization problem. The bilevel program is intrinsically hard, which is solved by stochastic gradient descent. To calculate the gradient involving non-differentiable $\ell_1$-norm minimization, we prove that we can use implicit differentiation to find the partial derivatives from which we obtain a descent direction for optimization. We apply our generic learning model to three different tasks, including image classification, compressive sensing, and single image super-resolution. In all cases, our new dictionary learning model can generate more informative sparse representations for the tasks over previous reconstructive dictionary learning approaches.
CHAPTER 6

SUMMARY

In this dissertation, we have explored sparse modeling for various tasks in computer vision and machine learning to address their specific challenges, which we summarize in the following.

**Image Classification** We propose a generic image feature representation by max pooling over sparse codes of local image descriptors from different spatial blocks across multiple spatial scales. By modeling the local descriptors using sparse codes, and using max pooling to extract the robust statistics, our feature is very discriminative and fits linear kernels well, and thus is scalable to large-scale scenarios. The feature representation is robust to many image variations, e.g., translations, rotations, misalignment, and illuminations, and is consistent for diverse image classification tasks. We have applied this feature representation to object recognition, scene categorization, face recognition, gender recognition, human action recognition, and handwritten digit recognition. On all these tasks, our feature representation consistently achieved state-of-the-art performances on the benchmark datasets with a simple linear SVM classifier.

**Image Super-Resolution** Inspired by recent compressive sensing theories, we formulate the single image super-resolution problem as a stable sparse signal recovery problem. From a large number of training patches, we propose the joint dictionary training procedure to learn a much more compact model, which reduces computational costs while still achieving superior results both qualitatively and quantitatively compared with previous approaches. Furthermore, as theoretically and experimentally justified, our super-resolution algorithm is more robust to noise, which is important for practical applications. Compared with previous example-based single image super-resolution approaches, our new algorithm is more mathematically sound, more compact, more computationally efficient, and more robust to noise. The proposed framework also enriches the sparse recovery
theories and applications from the machine learning perspective.

**Graph Construction** In machine learning, graph construction is critical for all graph-oriented learning algorithms. We propose to find the neighbors and edge weights for graph construction simultaneously for one data sample by seeking its sparse representation in terms of the remaining data samples. Compared with conventional approaches, the new graph is more robust to noise, naturally sparse, adaptive in neighbor selection, and convey more informative information for clustering and recognition. Applying this new graph to many graph-based data analysis, including data clustering, subspace learning, and semi-supervised learning, we have achieved significant improvements over conventional graphs, which implies that sparse representation is a good principle for graph construction on the high-dimensional data with sparse structures.

**Dictionary Learning** For specific tasks based on sparse representations, the dictionary on which these sparse representations build is critical to success of those algorithms [12], [10], [128], [24], [27]. Classical dictionary learning algorithms mainly focus on learning dictionaries that can sparsely represent the signal space well, regardless of the tasks to be performed based on these representations. We address this problem by formulating a general formulation that connects the low-level dictionary learning with the high-level tasks via a bilevel optimization program, for which we develop an efficient stochastic gradient descent procedure to find a local optimum. We apply this general dictionary learning formulation to sparse coding based image classification, compressive sensing, and single image super-resolution, all leading to substantial improvements over the classical dictionary learning approaches.

Reviewing the literatures of signal processing, computer vision, and machine learning, data representation is always the first step toward an effective model for many difficult problems. We also see that sparsity has been playing an important role in one way or another for these models, even though it may not be an exact representation in the conventional reconstructive manner. Our work in this dissertation and other publications on sparse coding (see Chapter 7) have demonstrated that a simple sparsity prior on the representation is surprisingly effective for many applications. A closely related application with sparse coding for sparse representation is the recent research development on low-rank matrices, where the matrix is decomposed onto an orthogonal rank-1 matrix basis formed by the singular vec-
tors, which promotes a special sparse representation adapted to the matrix itself. The sparsity prior in this case is also very effective for many ill-posed matrix problems, such as matrix completion. While the current sparse coding algorithms are essentially assuming 1-D signals, imposing some desired properties (e.g., rank) on the 2-D dictionary atoms may lead to a much more meaningful representation for image signals, as suggested by the low-rank matrix theories. Looking into the future, sparsity will be one of the most important cues for understanding high-dimensional data.
CHAPTER 7

PUBLICATIONS DURING PH.D. STUDY


REFERENCES


