2013

A machine learning paradigm based on sparse signal representation

Jie Yang
University of Wollongong

A Machine Learning Paradigm based on Sparse Signal Representation

A thesis submitted in partial fulfilment of the requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Jie Yang

School of Electrical, Computer and Telecommunications Engineering

January, 2013

Statement of Originality

I, Jie Yang, declare that this thesis, submitted in partial fulfillment of the requirements for the award of Doctor of Philosophy, in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is entirely my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Signed

Jie Yang

January, 2013

Contents

List of Figures

List of Tables

Acronyms

CS Compressed Sensing

ELM Extreme Learning Machines

FE Feature Extraction

FS Feature Selection

FOCUSS FOCal Undetermined System Solver

IMV Infinite Measurement Vector

KSVD K-Singular Value Decomposition

LC Linear Classifier

LDA Linear Discriminant Analysis

LS-SVM Least Squares Support Vector Machines

ML Machine Learning

MP Matching Pursuit

MSE Mean Squared Error

MLP Multilayer Perceptron

MMV Multiple Measurement Vector

MI Mutual Information

NN Neural Network

NRMSE Normalized Root Mean Square Error

OMP Orthogonal Matching Pursuit

PCA Principal Component Analysis

RPROP Resilient Back Propagation

SMV Single Measurement Vector

SVM Support Vector Machines

Abstract

Machine learning has been extensively investigated over the last three decades for its capability to learn mapping of functions from patterns. Nowadays, machine learning is routinely used in systems ranging from decision making to pattern recognition. However, with the emergence of ever-growing amount of information, machine learning techniques, such as neural networks and support vector machines, become unpractical or inefficient to solve large-scale problems. The main research of this thesis therefore focuses on analyzing the sparse signal representation algorithms and their application to machine learning.

Three major problems of machine learning are addressed in this thesis: structural (model) optimization, supervised training and feature selection. A variety of sparsity-based algorithms are proposed to tackle these problems. The proposed algorithms are capable of representing salient information using few elements, thereby saving memory storage and enhancing the generalization ability of the machine learning algorithms.

The first contribution is to optimize the structure of neural networks by finding a sparse representation for the network architecture. The proposed method starts with a very large network, and then a forward selection criterion is derived to identify important network parameters that minimize the residual output error. One advantage is that the algorithm makes no use of the problem-dependent

parameters, nor does it impose constraints on the network type.

The second contribution is the sparsity-based network training algorithm that extends the structural optimization method. The proposed training algorithm consists of two sub procedures: architecture optimization using sparse representation and weight update using dictionary learning. As a result, the algorithm is capable of training the network and optimizing the architecture simultaneously.

The sparsity-based training strategy is then applied to least squares support vector machine (LS-SVM). The model of LS-SVM is first reformulated as a sparse structure, and then the sparse representation is employed to reconstruct the learning model. The main advantage is that the proposed algorithm iteratively builds up the compact topology, while maintaining the training accuracy from the original large architecture.

The last contribution is to address the problem of dimensionality reduction via sparse signal representation. The presented algorithm regards the original feature set as the basis function, and selects discriminative features that minimize the residual error. The selected features have a direct correspondence to the performance requirement of the given problem. Experimentally, the proposed algorithm is tested with several benchmark classification tasks and a pedestrian detection problem. The results show that the sparsity-based algorithm achieves good classification accuracy and is also robust against variations in the number of training patterns.

Acknowledgments

Life is a remarkable journey. Completing a PhD is truly a memorable and interesting journey. The best and worst moments of my doctoral journey have been shared with many people.

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Salim Bouzerdoum for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. It has been such an honor and pleasure to work with him. I appreciate all his contribution of time, ideas, and funding to make my Ph.D. experience productive and stimulating.

I would also like to thank my co-supervisor Dr. Son Lam Phung for his encouragement and insightful comments. His enthusiasm and diligent attitude was contagious and motivational for me. I would like to acknowledge Dr. Fok Tivive for his friendship and assistance that has meant more to me than I could ever express. My sincere thanks also goes to Prof. Farzad Safaei and Prof. Jiangtao Xi, for their great technical support and encouragement.

The members of the Visual and Audio Signal Processing (VASP) group have contributed immensely to my personal and professional time. The group has been a source of friendship as well as good advice and collaboration. I am especially grateful for the best group in the world: Giang H. N. Le, Matthew Kitchener, Cher Hau Seng, Peiyao Li, Muhammad Nawaz, Wenbin Shao and Haoheng Zheng.

I could not complete my work without invaluable love and assistance from my parents: Mr. Hang Wen Yang and Mrs. Zhi Zhen Ke. Thank you for supporting me spiritually throughout my life. I would also like to mention the people who has often crossed my mind, people who in my life has shined warmth of love or a spark of light. Thank you for your turning up and self-giving support.

Life is a remarkable journey. As time flies, it carries on sadness and happiness, failure and success. So enjoy the journey.

Chapter 1

Introduction

Chapter contents

Over the past few decades, research on machine learning has received much attention from the signal processing and computer science communities. Although there have been some fundamental and groundbreaking advances in this field, many problems remain unsolved to date, such as processing the high-dimensional data. The thesis presented here proposes an investigation into the machine learning field which exploits the new paradigm of sparse representation to improve the generalization ability of machine learning algorithms. This chapter first introduces the background of machine learning and sparse representation. Second, the motivation for this research in machine learning is presented. Then the novel contributions of the thesis are summarized, followed by the thesis organization.

1.1 Machine learning

Machine learning has become very popular in recent years. Given training data samples, the machine learning algorithm is used to establish the potential model between the observed input and output samples. With these training samples, a machine learning algorithm will adjust its internal parameters to produce a function that approximates the implicit relationship between the input and output data.

There are various kinds of machine learning techniques, such as neural networks [1–4], support vector machines [5], [6], least squares support vector machines [7–9], decision tree [10–13], etc. Typically, the design of machine learning algorithms comprises two major phases: training phase and test phase. During training, a mapping model is induced based on the training patterns. The test phase is used for evaluation of the generalization ability of the machine learning algorithm.

Due to their learning capability from data, machine learning algorithms have been applied for problem-solving and decision-making in many areas. They can be employed either in data mining programs to detect network intrusion or fraudulent credit card transactions [14], [15], or in speech recognition systems to recognize spoken keywords [16], [17], or in computer version applications to track a moving object [18], [19], etc. In this thesis, we employ the new paradigm of sparse signal representation to the design of robust machine learning algorithms.

1.2 Sparse signal representation

Recently, a significant research effort has been devoted to finding the compact or sparse representation for target signals, and to enhancing the processing ability for large-scale data. Sparse representation indicates that a signal can be decomposed into a linear combination of a few elementary signals. The resulting sparse

solution only consists of few nonzero elements that are capable of representing the majority information conveyed by the target signal.

Sparse representation has several advantages, including the fact that it encourages an efficient model providing a simple interpretation for the target data. The inferred sparse coefficients also often have biological/physical meaning, revealing the hidden structure from the data. Sparse representation has therefore become a very important tool for many applications, such as signal reconstruction [20], [21], analog-to-information conversion [22], [23], radar imaging [24–27], and audio processing [28], [29].

There are mainly three sparse models employed in various applications. The single measurement vector (SMV) model is used to represent one-dimensional signals. For more complex applications that require multiple measurements, the multiple measurement vector (MMV) model is applied to enforce the same sparsity profile across several channels. The third sparse model is the infinite measurement vector (IMV) model, in which an infinite set of jointly sparse vectors (with nonzero elements occurring in the same location set) is considered.

So far the sparse representation is widely applied to the topic of signal reconstruction. Its main objective is to find the compact or sparse representation for target signal. The thesis presented here offers an alternative for developing novel algorithms using the sparse representation to solve complex and high-dimensional machine learning problems.

1.3 Motivations of the research

Research in developing novel machine learning algorithms is motivated by two major pitfalls: solving high-dimensional problems and addressing the structural optimization.

The amount of information generated by acquisition devices is always huge and ever-growing. For instance, thousands of images and transaction sale records

are generated every single second. This typically leads to gigabytes of data that exceeds the processing capacity of the most sophisticated machine learning algorithms. When solving large-scale or high-dimensional problems, the learning techniques such as neural networks or support vector machines become inefficient due to the computational complexity and memory storage requirements. Meanwhile, before applying any specific machine learning algorithm, one is faced with the question of determining the most suitable model, which is also known as model selection. For instance, in neural network, the network topology must be built ahead by fixing the number of hidden neurons and layers. Given the same problem, the performance obtained using different network structures may greatly differ from each other. However, there is not a theoretical formula giving clear insight for how to choose the optimal model. The model selection is typically solved by trial-and-error experiments or cross validation, but this process is computationally demanding. Hence, designing advanced machine learning algorithms that address the aforementioned problems is the primary motivation of this thesis. In general, the following problems need addressing:

1. What is the best criterion to describe the saliency or significance of the information?

2. How to eliminate unimportant elements from the learning model with minimal influence on the algorithm performance?

3. How to accelerate the algorithm convergence?

1.4 Novel research contributions

This thesis aims to address the machine learning problems using sparse representation. The key contributions are summarized as follows:

- *Neural network pruning using sparse representation*. A new technique for optimizing the structure of feed-forward neural networks is presented. The

important network parameters are selected by finding the sparse representation for the network structure.

- *Neural network training using sparse representation*. Develop a sparse-based supervised learning algorithm that combines sparse representation and network training. The advantage is that the proposed method generates a sparse network architecture and minimizes the training error simultaneously.

- *A sparse training algorithm for least squares support machine machines (LS-SVM)*. To improve the solution sparsity of the LS-SVM model, a sparse training algorithm is proposed by minimizing the output error and the model structure simultaneously.

- *Feature selection using sparse representation*. A sparse-based feature selection algorithm is proposed which aims to find fewer but more discriminative features.

1.5 Thesis organization

This thesis consists of seven chapters, with the current chapter presenting the introduction. Below is the general outline and structure of the remainder of the thesis.

- **Chapter 2** gives a comprehensive literature review of sparse representation and its extension compressed sensing. Three sparse models are presented, namely SMV, MMV and IMV. Accordingly, a review of sparsity measures and numerical algorithms for solving the sparse representation problem is given. Dictionary learning methods used in the sparse representation and compressed sensing are also presented.

- **Chapter 3** presents the application of the sparse representation to optimize the structure of neural networks. A brief review about feed-forward neural network and conventional pruning algorithms is given. Then the sparse-based pruning algorithm is proposed by establishing the link between the structural optimization and the sparse representation.

- **Chapter 4** proposes a novel supervised learning model to train the neural network and generate a sparse architecture simultaneously. The dictionary learning strategy is also employed to achieve the optimal trade-off between the model performance and architecture complexity.

- **Chapter 5** addresses the problem of training least squares support vector machines. Important support vectors are selected iteratively based on the similarity between support vectors and the residual error. The results show that the proposed method achieves comparable performance with typically a much sparser model.

- **Chapter 6** presents a novel algorithm for feature selection. This chapter first presents a brief review of existing feature reduction algorithms. Then the feature selection is reformulated as the sparse representation model for original features. The proposed algorithm is tested on benchmark classification tasks and a pedestrian detection problem.

- **Chapter 7** summarizes the research activities. This chapter also provides the concluding remarks and suggestions for future research directions.

1.6 Publications

The research undertaken in this thesis and during my PhD studies has resulted in the following publications:

1. J. Yang, A. Bouzerdoum, F. H. C. Tivive, M. G. Amin, "Multiple measurement

vector model and its application to through-the-wall radar imaging," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2011)*, pp. 2672 - 2675, 2011.

2. J. Yang, A. Bouzerdoum, M. G. Amin, "Multi-view through-the-wall radar imaging using compressed sensing," *European Signal Processing Conference (EUSIPCO 2010)*, pp. 1429 - 1433, 2010.

3. J. Yang, A. Bouzerdoum, S. L. Phung, "A particle swarm optimization algorithm based on orthogonal design," *IEEE Congress on Evolutionary Computation (CEC 2010)*, pp. 1-7, 2010.

4. J. Yang, A. Bouzerdoum, S. L. Phung, "Dimensionality reduction using compressed sensing and its application to a large-scale visual recognition task," *International Joint Conference on Neural Networks (IJCNN 2010)*, pp. 1-8, 2010.

5. J. Yang, A. Bouzerdoum, S. L. Phung, "A training algorithm for sparse LS-SVM using compressive sampling," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2010)*, pp. 2054-2057, 2010.

6. J. Yang, A. Bouzerdoum, S. L. Phung, "A new approach to sparse image representation using MMV and K-SVD," *Advanced Concepts for Intelligent Vision Systems (ACIVS 2009)*, pp. 200-209, 2009.

7. J. Yang, A. Bouzerdoum, S. L. Phung, "A neural network pruning approach based on compressive sampling," *International Joint Conference on Neural Networks (IJCNN 2009)*, pp. 3428-3435, 2009.

Chapter 2

Sparse Signal Representation and Compressed Sensing

Chapter contents

2.1 Introduction

Over the past decade, sparse (signal) representation has drawn increasing attention in the signal processing community [30–38]. The reason is that many natural signals have a *sparse* or *compressible* representation in a special domain, such as Fourier, discrete cosine transform (DCT) or wavelet domain. A signal is called *sparse* if it admits a transform domain representation in which most coefficients are zero. Similarly, a signal is said to be *compressible* if it can be closely approximated by a sparse signal. Finding the sparse signal representation allows the design of high performance algorithms for source separation [33], data compression [39], and data fusion [40].

Compressed sensing (CS), a special case of sparse signal representation, is proposed to reconstruct signals when fewer measurements are available than the dimension of the signal [41–46]. Traditionally, a signal is reconstructed using Shannon-Nyquist sampling theorem. That is, an analogue signal can be reconstructed perfectly if it is sampled at a rate at least twice the highest frequency present in the signal [47, 48]. However, for some signals, such as audio signals or images, the sampling rate can be very high, which may result in a very large number of samples. To address this problem, compressed sensing is applied to reconstruct a signal using far fewer measurements than required by the Shannon-Nquist sampling theorem.

In this chapter, we review the key theories and algorithms that form the basis of sparse representation and compressed sensing. The chapter is organized as follows. Section 2.2 focuses on the concept of sparse signal representation. Section 2.3 gives an overview of compressed sensing and introduces three main sparse models, namely *single measurement vector* (SMV), *multiple measurement vector* (MMV) and *infinite measurement vector* (IMV). A review of existing sparsity measures and numerical algorithms for solving the sparse representation is also presented. Section 2.4 presents several dictionary learning methods used in the

sparse representation and compressed sensing. Finally, Section 2.5 presents a summary of the chapter.

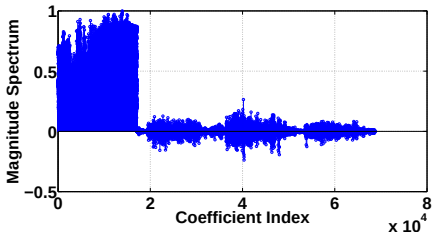
2.2 Sparse representation

Sparse representation aims to decompose a signal using a linear combination of relatively few basis elements. It offers a rigorous mathematical framework to analyze high-dimensional data: few coefficients are capable of representing the majority information from the target signals. The next example illustrates the power of sparse signal representation.

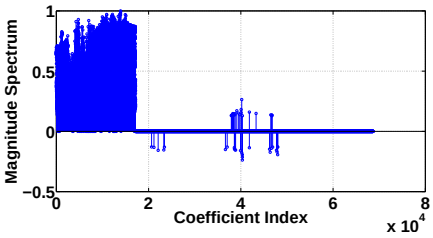
Example 1 Consider the "Lena" image of size 256×256 pixels and its discrete wavelet transform (DWT). The DWT analyzes the image at different frequency bands with different resolutions, and decomposes it into a coarse approximation and detail information. Without loss of generality, we apply the Haar wavelet to perform the wavelet transform.

(a) Lena image

(b) Reconstructed image



(c) DWT transform coefficients



(d) Reduced coefficients

Figure 2.1: Sparse representation for a signal: (a) original "Lena" image; (b) reconstructed image using 30% wavelet coefficients with large magnitudes; (c)DWT coefficients of the original image; (d) reduced DWT Coefficients.

The original "Lena" image is presented in Fig. 2.1 (a), and its corresponding wavelet transform coefficients are given in Fig. 2.1 (c). Although the total number of the wavelet transform coefficients is over 68,644, the majority have relatively small magnitudes. Insignificant coefficients can be removed without degrading the image quality. For instance, Fig. 2.1 (b) shows the reconstructed image using only 30% of the wavelet coefficients. Accordingly, the reduced coefficients after removing 70% of the coefficients with small magnitudes are shown in Fig. 2.1 (d); clearly the reconstructed image is visually close to the original image. This example shows that when a natural signal is highly compressible, it can be compressed efficiently while maintaining the salient information.

Consider a signal $\mathbf{s} \in \mathbb{R}^N$ and an orthonormal basis $\Psi = [\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_N]$, where $\boldsymbol{\psi}_n \in \mathbb{R}^N$, for $n = 1, 2, \cdots, N$. Then the signal $\mathbf{s}$ can be expressed as follows:

$$\mathbf{s} = \sum_{n=1}^{N} x_n \boldsymbol{\psi}_n \quad \text{or} \quad \mathbf{s} = \Psi \mathbf{x}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the weight vector. The element x_n can be obtained from the inner product between $\mathbf{s}$ and $\boldsymbol{\psi}_n$: $x_n = \langle \mathbf{s}, \boldsymbol{\psi}_n \rangle$. Furthermore, we assume that the underlying solution $\mathbf{x}$ has a compact or sparse structure. When only K coefficients from $\mathbf{x}$ are nonzero, the vector $\mathbf{x}$ is referred to as K-sparse. In this case, the N-dimension signal $\mathbf{s}$ can be perfectly represented using only K coefficients. That is, only a small subset of the coefficients is sufficient to capture the signal information. As a result, the sparse structure results in efficient data compression, fast transmission, and accurate statistical analysis [32–37, 49].

2.3 Compressed sensing

A novel sensing (sampling) paradigm, known as *compressive sampling* or *compressed sensing* (CS), has been developed as an extension of the sparse representation [42, 50–56]. The CS model allows a target signal to be recovered from fewer

measurements than required by the Shannon-Nyquist theorem. Note that CS can be regarded as a special case of sparse representation when fewer measurements are available than the number of signal samples to be recovered. According to the number of measurement vectors or channels, the sparse representation framework is categorized into three models.

1. *Single measurement vector* (SMV) model is applied when only one measurement vector is available [50, 57];

2. *Multiple measurement vector* (MMV) model considers more than one measurement vector simultaneously, where the solution is a two-dimensional array [58–60];

3. *Infinite measurement vector* (IMV) model consists of an infinite set of jointly sparse vectors that share the indices of nonzero elements [61].

In the past decade, these models have attracted a great amount of research effort, resulting in many exciting new applications. We refer the readers to [43, 51, 61–63] for a more detailed discussion on CS and its wide applications.

2.3.1 Single measurement vector model

In SMV model, the aim is to recover a sparse signal $\mathbf{x} \in \mathbb{R}^N$ from few linear measurements $\mathbf{y} \in \mathbb{R}^M$, where $M << N$. If a signal $\mathbf{s}$ is not sparse, it can usually be represent by a sparse vector $\mathbf{x}$ using a transformation $\Psi \in \mathbb{R}^{N \times N}$:

$$\mathbf{s} = \Psi \mathbf{x}. \tag{2.2}$$

Given a linear measurement matrix $\Phi \in \mathbb{R}^{M \times N}$, the measurement vector is given by

$$\mathbf{y} = \mathcal{D}\mathbf{x}, \tag{2.3}$$

where $\mathcal{D} = \Phi\Psi$ is known as the dictionary. The SMV model can then be expressed as

$$P: \quad \min S(\mathbf{x}) \quad \text{subject to} \quad \mathbf{y} = \mathcal{D}\mathbf{x}, \tag{2.4}$$

where $S(\mathbf{x})$ denotes a sparsity measure. The SMV model is also illustrated in Fig. 2.2.

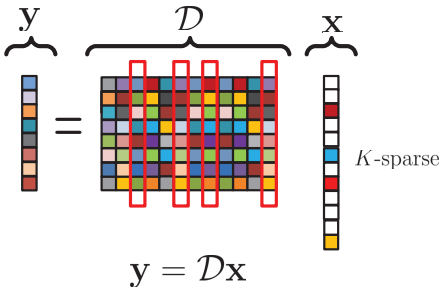


Figure 2.2: The SMV model aims to minimize a vector sparsity. In the sparse signal $\mathbf{x} \in \mathbb{R}^N$, only K entries are nonzero. Thus, the signal is K-sparse. The rectangular areas from the dictionary $\mathcal{D}$ are associated with nonzero coefficients from $\mathbf{x}$.

One simple strategy for solving Eq. (2.4) is to minimize the l_0-norm of $\mathbf{x}$, i.e., $S(\mathbf{x}) = \|\mathbf{x}\|_0$. The l_0 pseudo-norm is the cardinality or number of nonzero elements in $\mathbf{x}$. Thus Eq. (2.4) is rewritten as

$$P_0: \quad \min \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathcal{D}\mathbf{x}. \tag{2.5}$$

Next we discuss the situation when a unique solution to Eq. (2.5) exists. First, the notion of a *matrix spark* is introduced [64].

Definition 1 *Given an arbitrary matrix A, its matrix spark $\sigma(A)$ is defined as the smallest integer such that there exist σ columns of matrix A that are linearly dependent.*

Based on the matrix spark, Donoho *et al*. proved the following theorem that guarantees the existence of a unique solution of Problem P_0 [65].

Theorem 1 *Problem P_0 given in Eq. (2.5) admits a unique solution if there exists a sparse vector $\mathbf{x}^*$ satisfying the following conditions: $\mathbf{y} = \mathcal{D}\mathbf{x}^*$ and $\|\mathbf{x}^*\|_0 < \sigma(\mathcal{D})/2$, where $\sigma(\mathcal{D})$ is the spark of $\mathcal{D}$.*

The solution uniqueness depends on the dictionary $\mathcal{D}$ and the solution sparsity, rather than the measurement vector $\mathbf{y}$. However, the calculation of K sparse coefficients is NP-hard because the solution requires an exhaustive enumeration of all $\binom{K}{N}$ possible locations for nonzero elements in $\mathbf{x}$. More importantly, the computational complexity of Problem P_0 grows exponentially with N, the dimension of $\mathbf{x}$.

To address the computational issue, the l_1-based optimization is introduced. Problem P_0 in (2.5) is reformulated as

$$P_1 : \quad \min \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathcal{D}\mathbf{x}, \tag{2.6}$$

where $\|\mathbf{x}\|_1$ denotes the l_1-norm of $\mathbf{x}$. Since the optimization of (2.6) is no longer an NP-hard problem, the calculation can be found in linear time. Furthermore, if the solution of Problem P_0 is sufficiently sparse, it is identical with the solution of Problem P_1. Before describing a theorem about the equivalence of the two solutions, the definition of *mutual coherence* is first presented [43].

Definition 2 *The mutual coherence $\mu(\mathcal{D})$ of a dictionary $\mathcal{D}$ with N columns (atoms) is defined as the maximum normalized cross-correlation between the atoms of $\mathcal{D}$:*

$$\mu(\mathcal{D}) = \max_{1 \le i,j \le N, i \ne j} \frac{\left|\mathbf{d}_i^T \mathbf{d}_j\right|}{\|\mathbf{d}_i\| \left\|\mathbf{d}_j\right\|}, \tag{2.7}$$

where $\mathbf{d}_i$ is the i-th column vector of $\mathcal{D}$.

The equivalence of solutions between problems of (2.5) and (2.6) is established by the following theorem [52].

Theorem 2 *Given a dictionary $\mathcal{D}$ and the measurement vector $\mathbf{y}$, Problems P_0 and P_1 have exactly the same solution $\mathbf{x}^*$, if*

$$\mathbf{y} = \mathcal{D}\mathbf{x}^* \quad \textit{and} \quad \|\mathbf{x}^*\|_0 < \frac{1 + \mu(\mathcal{D})^{-1}}{2},$$

where $\mu(\mathcal{D})$ is the mutual coherence of the dictionary $\mathcal{D}$.

Furthermore, to better analyze the solution from Eq. (2.6), the restricted isometry property (RIP) constraint is introduced [66].

Definition 3 *Given $0 < \epsilon < 1$, the dictionary $\mathcal{D}$ obeys the RIP constraint given any K-sparse vector $\mathbf{x}$ if*

$$(1-\epsilon)\left\|\mathbf{x}\right\|_2^2 \leq \left\|\mathcal{D}\mathbf{x}\right\|_2^2 \leq (1+\epsilon)\left\|\mathbf{x}\right\|_2^2.$$

An alternative explanation of the RIP constraint is that any combination of column vectors from $\mathcal{D}$ is nearly orthogonal. Note that $\mathcal{D}$ can not be orthogonal since it has more columns than rows. Furthermore, the results from [41, 66, 67] show that a dictionary $\mathcal{D} \in \mathbb{R}^{M\times N}$ obeys the RIP constraint with high probability if $M \geq O(K\log(N/K))$, where K is the number of nonzero elements. Candes *et al*. proved the following theorem [67].

Theorem 3 *Suppose that $\mathcal{D}$ from (2.6) obeys the RIP constraint, and $\mathbf{x}_K$ is the best K-sparse approximation of the underlying solution $\mathbf{x}^0$. There exists some constant c_0 such that the solution $\mathbf{x}^*$ of (2.6) obeys*

$$\left\|\mathbf{x}^* - \mathbf{x}^0\right\|_2 \leq c_0 \frac{\left\|\mathbf{x}^0 - \mathbf{x}_K\right\|_1}{\sqrt{K}}, \tag{2.8}$$

and

$$\left\|\mathbf{x}^* - \mathbf{x}^0\right\|_1 \leq c_0 \left\|\mathbf{x}^0 - \mathbf{x}_K\right\|_1. \tag{2.9}$$

According to Theorem 3, if a dictionary obeys the RIP constraint and the underlying solution $\mathbf{x}^0$ is exactly K-sparse (i.e., $\mathbf{x}^0 = \mathbf{x}_K$), then the l_1-based model generates the accurate sparse solution: $\mathbf{x}^* = \mathbf{x}^0$. On the other hand, even if $\mathbf{x}^0$ is not exactly K-sparse, the reconstruction error is constrained by $\mathbf{x}^0$ and its best K-sparse approximation $\mathbf{x}_K$. Furthermore, the calculation of $\mathbf{x}^*$ is based on solving a convex problem, which requires no prior information about the underlying solution. If the additive measurement noise is taken into account, Eq. (2.6) is rewritten as

follows:

$$P_{1,\eta}: \quad \min \|\mathbf{x}\|_1 \quad \text{subject to} \quad \left\|\mathbf{y} - \mathcal{D}\mathbf{x}\right\|_2 < \eta, \tag{2.10}$$

where η bounds the amount of additional noise. Candes *et al*. further proved the following theorem given the additive noise η [67].

Theorem 4 *When $\mathcal{D}$ satisfies the RIP constraint, there exist some positive constants c_1 and c_2 such that the solution $\mathbf{x}^*$ to Problem $P_{1,\eta}$ obeys:*

$$\left\|\mathbf{x}^* - \mathbf{x}^0\right\|_2 \leq c_1 \frac{\left\|\mathbf{x}^0 - \mathbf{x}_K\right\|_1}{\sqrt{K}} + c_2\eta, \tag{2.11}$$

where $\mathbf{x}_K$ is the best K-sparse approximation of the underlying solution $\mathbf{x}^0$.

More discussion can be found in [41, 66, 67].

2.3.1.1 Geometric interpretation

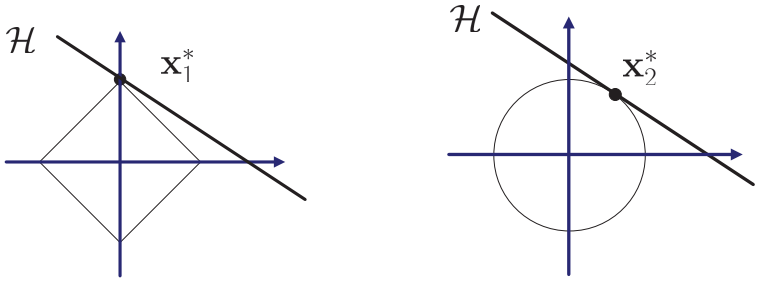
We now explain the reason that the l_1-based minimization is an effective way for solving the problem of sparse representation, rather than the l_2-based reconstruction. The conventional l_2-norm method takes the energy minimization into account, in which the least squares method is applied to generate the solution:

$$P_2: \quad \min \|\mathbf{x}\|_2 \quad \text{subject to} \quad \mathbf{y} = \mathcal{D}\mathbf{x}. \tag{2.12}$$

The l_2-based solution is usually given by: $\hat{\mathbf{x}} = \mathcal{D}^T \left(\mathcal{D}\mathcal{D}^T\right)^{-1} \mathbf{y}$.

Figure 2.3 presents geometric illustrations of the l_1 and l_2-based minimization. The straight line $\mathcal{H}$ expresses the linear constraint $\mathbf{y} = \mathcal{D}\mathbf{x}$. The l_1 ball in the $\mathbb{R}^2$ space is a hyperplane that is "pointy along the axes" with a certain radius [68] (see Fig. 2.3 (a)). When the l_1-based optimization is considered, the problem becomes that of determining the positions of points from the l_1 ball reaching the line $\mathcal{H}$. For the l_1 ball, the intersection occurs at the point near the coordinate axes, as shown in Fig. 2.3 (a). The intersection point denotes the K-sparse solution vector.

On the other hand, the l_2 ball (see Fig. 2.3 (b)), considering the standard Euclidean distance, is spherical and isotropic. Thus the l_2-norm minimization aims to find the closest points from the $\mathcal{H}$ line to the origin. As a result, the generated solution is not sparse; it has two non-zero components in Fig. 2.3 (b).



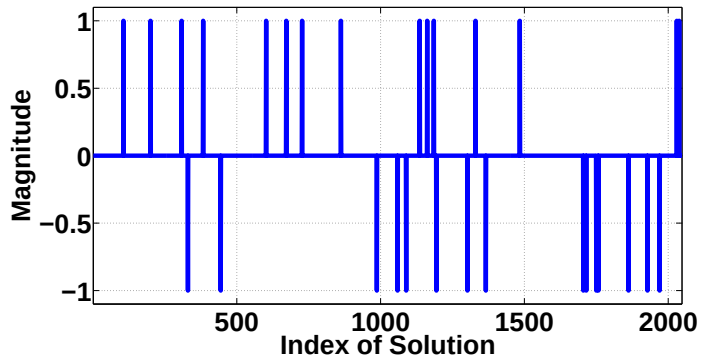
(a) l_1 ball (b) l_2 ball

Figure 2.3: Geometry illustration for the l_1 and l_2-based minimization. (a) Visualization of the l_1 minimization. (b) Visualization of the l_2 minimization with the hypersphere l_2 ball.

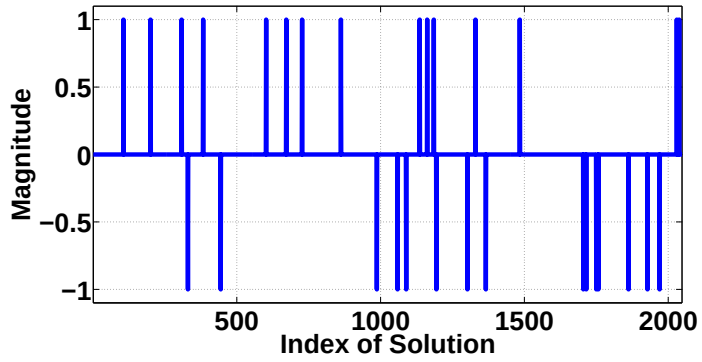
Example 2 To better compare the performance between the l_1- and l_2-based solutions, a sparse signal $\mathbf{x}$ was generated with length equal to 2048. Thirty spikes with the value 1 were initialized at random indices, while the remaining elements were set to zero. Furthermore, a random dictionary was generated and the number for sampling measurements was set to 300. Both l_1 and l_2-based models were employed and the results are shown in Fig. 2.4. As observed from Fig. 2.4 (c), the l_2-based method fails to obtain the sparse solution, whereas the l_1-based method succeeds in recovering the sparse solution (Fig. 2.4 (b)).

2.3.1.2 Numerical algorithms for SMV

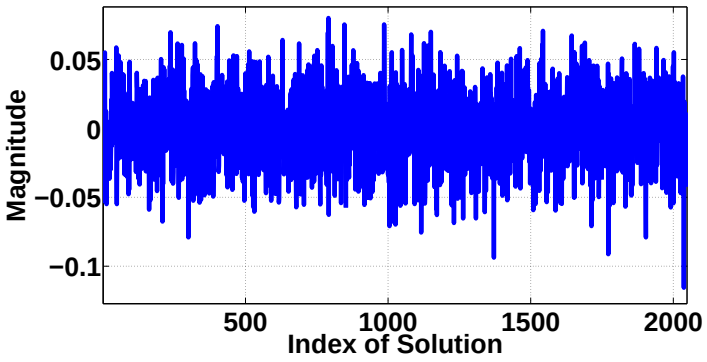
In this section, we discuss practical algorithms for finding the sparse solution of Problems P_0 and P_1. Various pursuit algorithms have been developed which are divided into two main groups: greedy algorithms and non-convex local optimization approaches. Greedy methods include *orthogonal matching pursuit* (OMP) [50], *matching pursuit* (MP) [57], and *stage-wise weak gradient pursuits* (SWGP) [69]. They first measure the similarity between the residual error and the dictionary atoms, and then select the atom that minimizes the residual error at each iteration.



(a) Original signal



(b) Signal reconstructed by l_1-based minimization



(c) Signal reconstructed by l_2-based minimization

Figure 2.4: Reconstructed signals using the l_1 and l_2-based minimization. A dictionary was randomly initialized with 2048 atoms.

Among greedy methods, the OMP algorithm is the most commonly used algorithm. It starts from an empty set and adds a new atom iteratively for the sparse representation. Then, the sparse solution is calculated using the least squares method or orthogonal projections. More details about the OMP algorithm can be found in [50, 62]. In particular, Tropp *et al*. proved the convergence property of the OMP algorithm [50].

Theorem 5 *Assume that an arbitrary K-sparse signal* $\mathbf{x} \in \mathbb{R}^N$ *(*$N \geq K$*) and a random* $M \times N$ *linearly independent matrix* $\mathcal{D}$*. If* $M \geq cK \log(N/\delta)$*, then OMP recovers* $\mathbf{x}$ *with the probability exceeding* $1 - \delta$*, where* $\delta \in (0, 0.36)$ *and c is a positive constant.*

Non-convex local methods, on the other hand, offer an alternative for solving the SMV model using re-weighted minimum norm. One of the most popular convex relaxation-based techniques is the FOCUSS algorithm. It employs Lagrange multiplier method for solving an l_p-norm ($p \leq 1$) minimization problem. The problem is reformulated as minimizing the Lagrangian function $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ given by

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \|\mathbf{x}\|_p^p + \boldsymbol{\lambda}^T (\mathbf{y} - \mathcal{D}\mathbf{x}), \tag{2.13}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier vector. The solution that minimizes $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ satisfies the following conditions:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = p \prod \mathbf{x}^T \mathbf{x} - \mathcal{D}^T \boldsymbol{\lambda} = 0 \quad \text{and} \quad \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{y} - \mathcal{D}\mathbf{x} = 0. \tag{2.14}$$

Apart from numerical algorithms for solving the sparse representation, the SMV model has been extensively used for various applications, such as analog-to-information conversion [22, 23], radar imaging [24–27], and audio processing [28, 29]. Although the SMV model achieves quantitatively favorable performance, further research requires the extension by considering multiple measurements. One reason is that the solution sparsity can be better reinforced when multiple

measurements are available with the same set of constraints. In the next subsection, we will introduce the multiple measurement vector model.

2.3.2 Multiple measurement vector model

The SMV model discussed in Section 2.3.1 aims to solve the sparse representation for a vector, whereas the multiple measurement vector (MMV) model is to find a joint-sparse representation for several signals, or a sparse matrix representation [59, 60]. Originally, the MMV model was motivated by a neuromagnetic inverse problem that arises in magnetoencephalography (MEG) [70].

Given the measurement matrix $Y \in \mathbb{R}^{M\times L}$ and the dictionary $\mathcal{D} \in \mathbb{R}^{M\times N}$ ($M << N$), the MMV model seeks to find a sparse matrix $X \in \mathbb{R}^{N\times L}$ by solving the following problem:

$$Q: \quad \min S(X) \quad \text{subject to} \quad Y = \mathcal{D}X, \tag{2.15}$$

where $S(X)$ is the matrix sparsity. Different choices for $S(X)$ are provided later. Obviously, when $L = 1$, Eq. (2.15) becomes the SMV model given in (2.4). Thus, the MMV model is considered as an extension of the SMV model by addressing the sparse representation for multiple SMV problems simultaneously. The MMV model is also described graphically in Fig. 2.5.

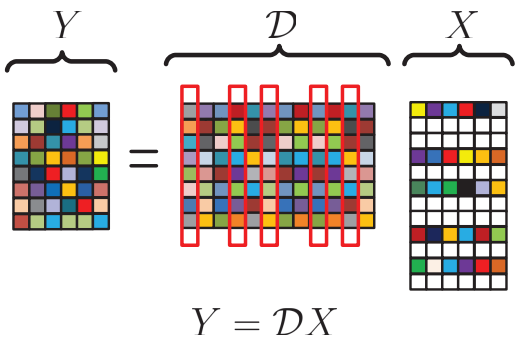


Figure 2.5: The MMV model aims to minimize a matrix sparsity. Multiple measurements are available which share the same set of constraints. The rectangular areas from the dictionary $\mathcal{D}$ are associated with nonzero rows from the sparse matrix X.

In the following, we discuss the MMV model with different sparsity measures. Then numerical algorithms are discussed, followed by a comparison between the SMV and MMV model.

The matrix sparsity is measured in terms of the number of nonzero rows. A non-zero row must have at least one non-zero entry. Let $\mathbf{r}_i$ be the i-th row of the matrix X, i.e., $X = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N]$. Let $\mathbf{v}$ be a column vector whose i-th element ($i = 1, 2, \ldots, N$) is given by:

$$v_i = \|\mathbf{r}_i\|_p \,, \tag{2.16}$$

where $\|\cdot\|_p$ denotes the vector p-norm ($0 \leq p \leq 1$). The sparsity of the matrix X can be expressed in terms of the sparsity of the vector $\mathbf{v}$,

$$S(X) = \|\mathbf{v}\|_0 \,. \tag{2.17}$$

The MMV model is then to find a sparse $N \times L$ matrix X by solving the following problem [59]:

$$Q_0 : \quad \min \ \|\mathbf{v}\|_0 \quad \text{subject to} \quad Y = \mathcal{D}X. \tag{2.18}$$

The principle of the MMV problem is that columns from the solution matrix X possess the same sparsity profile. That is, the columns of X share the indices of nonzero elements. The solution to Problem P_0 in Eq. (2.5) is a special case of (2.18) if only one measurement vector is available ($L = 1$). Again, solving (2.18) requires enumerating any possible subsets from all N rows, which is an NP-hard problem.

Note that in the SMV model, the solution equivalence between l_0 and l_1 minimization was established, which shows an efficient replacement for the l_0-based minimization. If the solution vector is sufficiently sparse, the result obtained from the l_1-norm minimization is identical with the solution from the l_0-norm minimization. Jie *et al*. extended the solution equivalence from SMV to MMV model [59]. Given an arbitrary matrix, the l_1-based sparsity measure for a matrix

is defined by

$$\text{Relax}(X) = \|\mathbf{v}\|_1 . \tag{2.19}$$

Note that the major difference between $S(X)$ and $\text{Relax}(X)$ is that we consider l_1-norm for each row $\mathbf{r}_i$ rather than l_0-norm. Therefore, Problem Q_0 is rewritten as follows:

$$Q_1 : \quad \min \ \text{Relax}(X) \quad \text{subject to} \quad Y = \mathcal{D}X. \tag{2.20}$$

The following theorem establishes the solution equivalence between the models (2.18) and (2.20) [59].

Theorem 6 *Given a dictionary $\mathcal{D}$, Problems Q_0 and Q_1, given in (2.18) and (2.20), have exactly the same sparse solution X if*

$$S(X) < \frac{1 + \mu(\mathcal{D})^{-1}}{2},$$

where $\mu(\mathcal{D})$ is the mutual coherence of dictionary $\mathcal{D}$.

The above theorem indicates that the solution equivalence depends on the dictionary and the matrix sparsity, rather than the measurement matrix Y.

2.3.2.1 Numerical algorithms for MMV

The recovery algorithms for the SMV model, such as the OMP and FOCUSS, are extended to the MMV model by making various implementations. In this section, we briefly discuss two main groups of algorithms developed for MMV: forward selection algorithms and regularization methods.

The forward selection algorithms make use of a small subset of atoms (columns from the dictionary) to represent the measurement matrix. Examples of forward selection algorithms are M-BMP and M-OMP [60] (also known as OMP-MMV [59]). At each iteration, only one atom is selected according to the similarity between the atom and the residual signal. Each selected atom corresponds to a nonzero row from the solution matrix X. The algorithms then employ different

projection or optimization methods to calculate the sparse solution. The forward selection iteration continues until a predefined criterion is satisfied.

Let E_t denote the the residual signal or reconstruction error at the t-th iteration ($E_0 = Y$) and $\mathcal{S}_t$ be the set of selected atoms from $\mathcal{D}$ ($\mathcal{S}_0 = \emptyset$, i.e., the empty set). The basic procedures for M-BMP and M-OMP algorithms are summarized in Algorithms 1 and 2, respectively.

Algorithm 1: M-BMP Algorithm to solve the MMV model.

input : A dictionary $\mathcal{D}$, the residual error E_{t-1} and maximal iteration T;
output: Selected atoms $\mathcal{S}_T$ and the residual error E_T;
for $t = 1$ **to** T **do**
  Compute $A_{t-1} = \mathcal{D}^T E_{t-1}$;
  Select from $\mathcal{D}$ *the column (or atom)* $\mathbf{d}_i^t$, *which corresponds to the row with the largest magnitude in* A_{t-1};
  Set $\mathcal{S}_t = \mathbf{d}_i^t$;
  Update $E_t = E_{t-1} - \mathcal{S}_t (\mathcal{S}_t)^T E_{t-1}$;
end

Algorithm 2: M-OMP Algorithm to solve the MMV model.

input : A dictionary $\mathcal{D}$, the residual error E_{t-1} and maximal iteration T;
output: Sparse solution X_T and the residual error E_T;
for $t = 1$ **to** T **do**
  Compute $A_{t-1} = \mathcal{D}^T E_{t-1}$;
  Select from $\mathcal{D}$ *the column (or atom)* $\mathbf{d}_i^t$, *which corresponds to the row with the largest magnitude in* A_{t-1};
  $\mathcal{S}_t = \mathcal{S}_{t-1} \bigcup \mathbf{d}_i^t$;
  $E_t = Y - \mathcal{S}_t X_t$, *where* $X_t = [\mathcal{S}_t^T \mathcal{S}_t]^{-1} \mathcal{S}_t^T Y$;
end

Another class of numerical algorithms is proposed in which the l_p-norm

$(0 < p \leq 1)$ is considered; When $p = 0$, Eq. (2.16) becomes a direct measure of the matrix sparsity (number of nonzero rows). The M-FOCUSS algorithm is proposed to minimize the following sparsity measure [60]:

$$J^{(p,q)}(X) = \sum_{i=1}^{N} \left(\|\mathbf{r}_i\|_p \right)^q, \quad 0 < p \leq 1, q \geq 1,$$

where $\mathbf{r}_i$ denotes the i-th row of the solution matrix X. The procedure of M-FOCUSS is given in Algorithm 3.

Algorithm 3: M-FOCUSS Algorithm to solve the MMV model.

input : A dictionary $\mathcal{D}$, the residual error E_{t-1}, and maximal iteration T;
output: Sparse solution X_T and the residual error E_T;
for $t = 1$ **to** T **do**
  Compute $X_{t-1} = \mathcal{D}^T E_{t-1}$;
  $W_t = \text{diag}\left(\left(\sum_{j=1}^{L} \left(x_{ij}^{t-1} \right)^2 \right)^{\frac{1-p/2}{2}} \right)$, *where* x_{ij}^{t-1} *denotes the element in the i-th row and the j-th column from* $(X_{t-1})^T$, *and* diag *represents a diagonal matrix*;
  $Q_t = (Z_t)^T \left(Z_t (Z_t)^T + \lambda I \right)^{-1} Y$, *where* $Z_t = DW_t$, $\lambda > 0$, *and I is the identity matrix*;
  $E_t = Y - \mathcal{D}X_t$ *and* $X_t = W_t Q_t$;
end

Cotter *et al*. have evaluated the performance of the M-BMP, M-OMP, and M-FOCUSS algorithms using a series of simulations [60]. Their experimental results indicate that M-FOCUSS performs better than the forward selection methods in terms of signal reconstruction. As for the computational complexity, the forward selection methods are more affordable than the regularization-based methods that are required to optimize all elements in the matrix solution at each iteration.

2.3.2.2 Comparison with SMV

Although the only difference between the SMV and MMV model is the number of measurements, there is no evidence that the solution for the MMV model can be obtained by solving the SMV model column-by-column. If the recovery algorithm from the SMV model is applied to solve the MMV model, the location for nonzero elements may be different in each solution vector. By contrast, the MMV approach considers all the measurement vectors simultaneously. Therefore, the solution vectors from the MMV model share the same sparsity profile: columns from the solution matrix share the indices of nonzero elements.

Eldar *et al*. proposed a measurement constraint, called the block Restricted Isometry Property (or block-RIP), to analyze the performance of the MMV model [71]. When the dictionary $\mathcal{D} \in \mathbb{R}^{M \times N}$ satisfies the block-RIP constraint, the sparse matrix X can be recovered using approximately $\mathcal{O}(-K \log(K/N))$ measurements, where $S(X) \leq K$. For the SMV model, if all L solution vectors are arranged as a composite column vector, then this $K \times L$-sparse vector requires $\mathcal{O}(-KL \log(K/N))$ measurements. Compared to the SMV model, the MMV model requires L times fewer measurements to compute the sparse solution. In other words, given the same amount of measurements, the MMV-based algorithm has a higher probability to achieve the better solution than the SMV model.

2.3.3 Infinite measurement vector model

Recently, the MMV model has been extended to an infinite set of jointly sparse vectors with nonzero elements occurring in a common location set. The resulting model, termed infinite measurement vector (IMV) model, consists of countable (single or multiple elements) or uncountable measurements (such as a continuous interval) [61]. Obviously, the MMV model is a special case of the IMV model. Given a dictionary $\mathcal{D} \in \mathbb{R}^{M \times N}$ with $M << N$, the IMV model is formulated as

follows:

$$\mathbf{y}(\lambda) = \mathcal{D}\mathbf{x}(\lambda), \quad \lambda \in \Lambda, \tag{2.21}$$

where Λ denotes an infinite set. Figure 2.6 shows the IMV model. In the IMV model, a jointly sparse solution is K-sparse in the sense that each column $\mathbf{x}(\lambda)$ is a K-sparse vector and nonzero elements from all columns are condensed in the same rows.

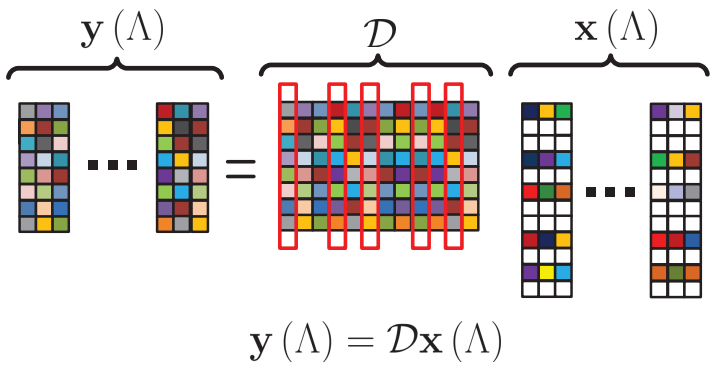


Figure 2.6: The IMV model consists of countable (single or multiple elements) or uncountable measurements. The solution is an infinite set of jointly sparse vectors with nonzero elements occurring in a common location set. The rectangular areas from the dictionary $\mathcal{D}$ represent the selected atoms, which are associated with nonzero rows.

Table 2.1 summarizes the relationships among three sparse models: SMV, MMV and IMV. The major difference among the three sparse models comes from the joint sparsity prior or number of measurements.

Table 2.1: Three sparse models and relevant properties.

Model	Linear System	Solution Dimension
SMV	$\mathbf{y} = \mathcal{D}\mathbf{x}$	1-D vector
MMV	$Y = \mathcal{D}X$	2-D matrix
IMV	$\mathbf{y}(\lambda) = \mathcal{D}\mathbf{x}(\lambda)$	2-D matrix with infinite columns

With more columns than rows, there is an infinite number of solutions to the underdetermined model of (2.21). In particular, the sparsity constraint is introduced so that a unique solution exists. The result turns out to be similar to that of the SMV and MMV models: a sufficient sparsity is the necessary condition for the solution uniqueness [61].

Theorem 7 *The* $\hat{\mathbf{x}}(\Lambda)$ *is the unique solution to the optimization of Eq. (2.21), if* $\hat{\mathbf{x}}(\Lambda)$ *is a K-sparse solution satisfying* $\mathbf{y}(\Lambda) = \mathcal{D}\hat{\mathbf{x}}(\Lambda)$*, and*

$$K \leq \frac{\sigma(\mathcal{D}) + \dim(\operatorname{span}(\mathbf{y}(\Lambda))) - 1}{2},$$

where $\sigma(\mathcal{D})$ *is the matrix spark for the dictionary* $\mathcal{D}$*,* $\operatorname{span}(\mathbf{y}(\lambda))$ *represents the subspace of minimal dimension containing the set* $\mathbf{y}(\Lambda)$*, and* $\dim(\cdot)$ *returns the dimension of its argument.*

Next, a robust recovery method is developed to solve the IMV model, in which the location for nonzero elements is the key. Once the indices of nonzero rows are identified, the IMV problem can be translated into a finite MMV model. In other words, there is an explicit MMV counterpart with the same nonzero location set for any IMV problem. The result is summarized by the theorem below [61].

Theorem 8 *Suppose that* $\hat{\mathbf{x}}(\Lambda)$ *is the K-sparse solution to Eq. (2.21). Then* $\hat{\mathbf{x}}(\Lambda)$ *has the same nonzero-location set with the matrix* U*, which satisfies the following linear system:*

$$V = \mathcal{D}U \quad \text{subject to} \quad VV^T = \int_{\lambda\in\Lambda} \mathbf{y}(\lambda)\,\mathbf{y}^T(\lambda)\,d\lambda.$$

Theorem 8 provides an alternative method for finding the locations of nonzero elements. Thus the infinite structure of the IMV model is converted to a specific MMV model.

The IMV model is especially useful to address problems with a large jointly-sparse matrix. For instance, the IMV model can be used in radar [72] or array processing [73] to simulate the sensor discretization. Furthermore, consider the sampled-delay Doppler spread function for the underwater acoustic sensing, the discrete snapshots can be replaced by a continuous estimate using the IMV model [74].

2.4 Dictionary learning

The performance of the sparse representation relies heavily on the dictionary structure [43, 46, 75–79]. In the previous section, the dictionary $\mathcal{D} \in \mathbb{R}^{M \times N}$ is randomly initialized and fixed during the sparse representation. Intuitively, a randomly generated dictionary may not obtain an accurate reconstruction. In this section, the dictionary learning techniques are discussed.

Another way to consider the mutual coherence (presented in Eq. (2.7)) is from the Gram matrix $\mathcal{G}$. A new dictionary $\widetilde{\mathcal{D}}$ is first obtained by normalizing all the atoms of the original dictionary $\mathcal{D}$. The mutual coherence is given by the largest magnitude of the off-diagonal entries of the Gram matrix $\mathcal{G} = \widetilde{\mathcal{D}}^T \widetilde{\mathcal{D}}$. A dictionary consisting of more similar atoms has a higher $\mu(\mathcal{D})$ value, whereas a dictionary with smaller $\mu(\mathcal{D})$ has a higher probability to generate a sparser and more accurate solution. To optimize the dictionary structure, a variety of methods are proposed which are grouped into three categories [80]: *clustering-based*, *Learning with specific structure*, and *probabilistic methods*.

1. Clustering-based learning algorithms are based on vector quantization (VQ) that is similar to the *K*-means clustering [75, 81–93]. The implicit assumption is that each measurement can be represented by one single atom, thereby converting the dictionary learning to a *K*-means clustering.

2. Learning with specific structure algorithms separate the dictionary into two parts: one part is fixed during the sparse representation, while the other part is adaptable [43, 46, 94–96].

3. Probabilistic algorithms reformulate the problem of dictionary optimization into a parameter-search model, in which the dictionary is optimized to minimize the reconstruction error [54, 97, 98].

2.4.1 Clustering-based learning algorithms

The clustering-based learning algorithms are based on vector quantization (VQ) method. The measurements are grouped to train the dictionary such that the reconstruction error to a given atom is minimal. The implicit assumption here is that each measurement can be represented by a single atom, which converts the learning procedure to a K-means clustering. A generalization of the K-means algorithm for dictionary learning, called the KSVD algorithm, has been firstly proposed in [75]. Then various KSVD-based methods were developed [82–87]. In KSVD, the atoms and their relevant coefficients are updated via the *singular value decomposition* (SVD) at each iteration; thus, the KSVD algorithm not only finds a sparse solution, but also updates simultaneously the dictionary $\mathcal{D}$.

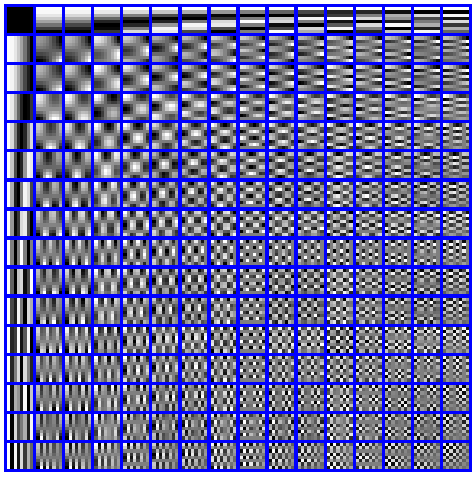
Consider an $M \times L$ matrix Y comprising L measurement vectors, and a dictionary $\mathcal{D} \in \mathbb{R}^{M \times N}$ to be optimized. In the KSVD method, each column $\mathbf{y}_i$ of Y is extracted and the traditional SMV method is applied on the pair $(\mathbf{y}_i, \mathcal{D})$ to solve the following problem:

$$\min \|\mathbf{x}_i\|_p \quad \text{subject to} \quad \mathbf{y}_i = \mathcal{D}\mathbf{x}_i, \tag{2.22}$$

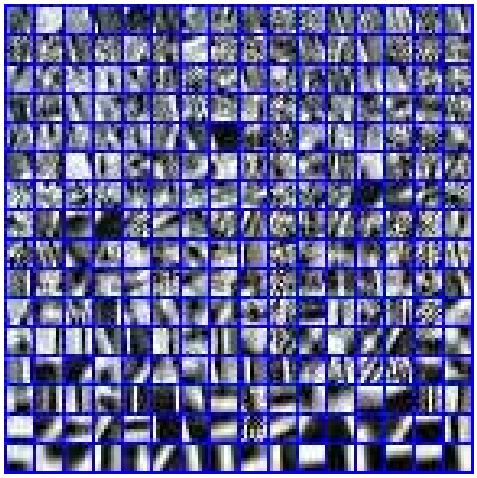
where $\|\cdot\|_p$ denotes the vector p-norm ($0 \leq p \leq 1$). The updating strategy from the KSVD method is shown in Algorithm 4.

Example 3 For comparison purposes, Fig. 2.7 illustrates the dictionary atoms from the discrete cosine transform (DCT) and the KSVD method extracted from the "Boat" image. The overcomplete DCT dictionary is established by sampling the cosine wave with different frequencies, as shown in Fig. 2.7 (b). Compared to the DCT dictionary, the atoms from the KSVD-based dictionary consist of more image details (Fig. 2.7 (c)). Experimental results presented in [75] show that the KSVD-based method leads to a better performance in terms of the image reconstruction.

(a) Original Image



(b) DCT-based dictionary



(c) KSVD-based dictionary

Figure 2.7: Resulting dictionary atoms using the "Boat" image. It can be observed that the learned dictionary consists of more image details, which leads to a better image representation.

Algorithm 4: KSVD-based Dictionary Optimization.

input : The initial dictionary $\mathcal{D}$, measurement matrix Y, and sparse solution X.

output: Optimized dictionary $\mathcal{D}$.

repeat

Find one atom $\mathbf{d}_i$ *from* $\mathcal{D}$ *that is associated with a nonzero row* ($\mathbf{r}_i$) *from* X;

Calculate the error without $\mathbf{d}_i$: $E_s = Y - \mathcal{D}X + \mathbf{d}_i\,\mathbf{r}_i$;

Apply the SVD operation on E_s *to update* $\mathbf{d}_i$ *and* $\mathbf{r}_i$:

$\{U, \Sigma, V\} = SVD(E_s)$, *where* U *and* V *are orthogonal matrices, and* Σ *is a diagonal matrix of the same dimension as* E_s;

Take the first column from U *and* $\Sigma \times V$ *to replace* $\mathbf{d}_i$ *and* $\mathbf{r}_i$, *respectively*;

until *all non-zero rows in* X *have been processed*;

Overall, the clustering-based algorithms outperform the traditional sparse representation techniques without dictionary optimization. However, they are time-consuming as the computational complexity grows exponentially with increasing number of measurements and size of the dictionary.

2.4.2 Learning with specific structure algorithms

The dictionary learning algorithms can also be optimized according to specific structure to satisfy predefined criteria, such as dictionary coherence. For instance, one can optimize the dictionary so that it gets close to an equiangular tight frame (ETF). In [46], a dictionary is learned based on a Gammatone generating function. The method optimizes a dictionary to achieve a minimal dictionary coherence, which tiles the time-frequency plane more uniformly than the original Gammatone filter bank.

However, the optimization using dictionary coherence is time-consuming because it considers all elements from $\mathcal{D}$. To address the computational issue, a heuristic approach is proposed in [43] based on the *averaged mutual coherence*.

Definition 4 *For a dictionary $\mathcal{D}$ with N atoms, its τ-averaged mutual coherence $\mu_\tau(\mathcal{D})$ considers elements of the Gram matrix $\mathcal{G}$ of the dictionary that are above the threshold τ:*

$$\mu_\tau(\mathcal{D}) = \frac{\sum_{1\leq i,j\leq N,\ i\neq j}\left(\left|\mathbf{g}_{ij}\right| \geq \tau\right)\left|\mathbf{g}_{ij}\right|}{\sum_{1\leq i,j\leq N,\ i\neq j}\left(\left|\mathbf{g}_{ij}\right| \geq \tau\right)}, \tag{2.23}$$

where $\mathbf{g}_{ij}$ is the element from the i-th row and j-th column of the Gram matrix $\mathcal{G}$.

If $\tau = 0$, $\mu_\tau(\mathcal{D})$ is the average value of the absolute entries of $\mathcal{G}$. Note that $\mu_\tau(\mathcal{D}) \leq \mu(\mathcal{D})$; for more relationship between $\mu_\tau(\mathcal{D})$ and $\mu(\mathcal{D})$, readers are referred to [43]. Then, one part of the dictionary $\mathcal{D}$ is fixed and the updating strategy is applied on the other part to minimize the dictionary coherence. More precisely, given the dictionary $\mathcal{D} = \Phi\Psi$, the sensing matrix Ψ is fixed during the sparse representation and Φ is the selection matrix to be optimized. The dictionary optimization is achieved by a "shrink" operation over Φ so that $\mu_\tau(\Phi\Psi)$ is minimized. Algorithm 5 shows the procedure for the averaged mutual coherence-based dictionary optimization.

Algorithm 5: Optimization for the selection matrix Φ.

input : Sensing matrix Ψ and threshold τ;
output: Optimized selection matrix $\widehat{\Phi}$;
Initialize the selection matrix Φ randomly.
Calculate the dictionary $\mathcal{D} = \Phi\Psi$ and obtain the Gram matrix $\mathcal{G}$.
Update $\mathcal{G}$ by shrinking its elements according to the threshold τ to generate $\widehat{\mathcal{G}}$.
Apply the SVD operation to ensure the rank of $\widehat{\mathcal{G}}$ is equal to K and calculate the squared root of $\widehat{\mathcal{G}}$, i.e., $\mathcal{U}\mathcal{U} = \widehat{\mathcal{G}}$.
Obtain the selection matrix: $\widehat{\Phi} = \arg\min \|\mathcal{U} - \Phi\Psi\|_2$.

Empirical results from [43] show that the performance of the sparse representation is improved using the shrink-based learning strategy. In other words, with the same recovery algorithm, the optimized dictionary leads to a much sparser

solution. The reason is that when the dictionary coherence is reduced, more independent atoms are involved. Therefore, fewer atoms are required and the performance of the sparse representation is enhanced.

2.4.3 Probabilistic algorithms

The probabilistic algorithms maximize the likelihood that measurements have efficient and sparse representations given a redundant dictionary [54, 97, 98]. Under the assumption that the primary visual area in the human cortex probably follows a sparse model, the *maximum likelihood* (ML) algorithm was firstly developed in [54]. Given a single measurement $\mathbf{y}$, the probabilistic-based method assumes that

$$\mathbf{y} = \mathcal{D}\mathbf{x} + \mathbf{e}, \tag{2.24}$$

where $\mathbf{e}$ represents the reconstruction error, or measurement noise. Then with L examples ($Y = \left\{\mathbf{y}_i\right\}_{i=1}^{L}$), probabilistic algorithms aim to maximize the likelihood function $P(Y|\mathcal{D})$. To simplify the problem, two assumptions are introduced. The first is that the measurements are drawn independently, therefore we have:

$$P(Y|\mathcal{D}) = \prod_{i=1}^{L} P(\mathbf{y}_i|\mathcal{D}). \tag{2.25}$$

The second assumption is that the noise can be modeled as a Gaussian zero-mean noise. Then the ingredients of the likelihood function could be computed as follows:

$$P(\mathbf{y}_i|\mathcal{D}) = \int P(\mathbf{y}_i, \mathbf{x}_i|\mathcal{D})d\mathbf{x} = \int P(\mathbf{y}_i|\mathbf{x}_i, \mathcal{D})d\mathbf{x}. \tag{2.26}$$

The dictionary learning problem is solved by iterating between two steps. In the first step, we look for the sparse coefficients, also known as *sparse coding*, based on the dictionary [54]. It can be solved by the convex optimization for each $\mathbf{y}_i$. The second step is the learning step which keeps the coefficients from the first

step constant, while optimizing the dictionary as follows:

$$\mathcal{D} = \arg\min_{\mathcal{D}} \sum_{i=1}^{L} P(\mathbf{y}_i, \mathbf{x}_i, |\mathcal{D}) = \arg\min_{\mathcal{D}} \sum_{i=1}^{L} \left\| \mathcal{D}\mathbf{x}_i - \mathbf{y}_i \right\|_2, \tag{2.27}$$

where L is the number of measurements.

Similar to the ML algorithm, the *method of optimal directions* (MOD), was presented in [98]. The main contribution of the MOD method is the simple way of updating the dictionary. Assume that the matrix X_t is the sparse solution at the t-th iteration. In the MOD algorithm, the dictionary is updated as follows:

$$\mathcal{D}_{t+1} = YX_t \left(X_t\right)^T \cdot \left(\left(X_t\right)\left(X_t\right)^T\right)^{-1}. \tag{2.28}$$

In [97], another probabilistic algorithm was proposed, termed *Maximum A-Posteriori Probability* (MAP). However, rather than optimizing the likelihood function $P(Y|\mathcal{D})$, the posterior $P(\mathcal{D}|Y)$ is used. Firstly, the overall mean square error is expressed in the matrix form:

$$E_t = Y - \mathcal{D}X_t.$$

Then at the t-th iteration, the dictionary is updated as follows:

$$\mathcal{D}_{t+1} = \mathcal{D}_t + \eta E_t X_t + \eta \left(X_t E_t \mathcal{D}_t\right) \mathcal{D}_t, \tag{2.29}$$

where η is a small constant. The last term from Eq. (2.29) allows different columns to have different norm values [97]. Despite the dictionary variety, the MAP algorithm provides slower training compared to the MOD method.

In both the MOD and MAP methods, the entire measurements Y and reconstruction error E_t is considered. Therefore, the dictionary optimization is time-consuming, when more measurements are employed. More recently, a fast dictionary online training algorithms have been proposed [99]. The online al-

gorithms consider only a subset of the measurements iteratively. The dictionary repeats updating the atoms until all training measurements have been used. This is quite beneficial in terms of computational complexity in practical applications.

2.5 Conclusion

In this chapter, we have briefly reviewed the concept of the sparse representation and compressed sensing. Three sparse models have been discussed based on the number of available measurements. We have also reviewed numerical algorithms to recover the sparse solution. In addition, the dictionary learning methods, which aim to optimize the dictionary structure, have been discussed briefly.

One promising property for the sparse model is that only few nonzero elements are required to represent the majority information from the target signal. Sparse representation and compressed sensing therefore have increasingly become recognized as providing extremely high performance for different applications. In the rest of the thesis, we will focus on the applications of sparse signal representation to machine learning problems. The next chapter introduces a pruning algorithm for feed-forward neural networks based on sparse representation.

Chapter 3

Neural Network Pruning using Sparse Representation

Chapter contents

3.1 Introduction

Neural networks (NNs) have been employed in many applications, such as time series prediction [3], pattern recognition [4, 18], decision making [100], modeling [101], and clustering [102], to name few. The success of their applications has motivated a growing interest in developing efficient training methods with fast convergence and good generalization ability. However, the performance of the network training algorithm depends on the network structure. For instance, using the same training method, a large structure may result in a fast convergence to a local minimum, but exhibits poor generalization performance because of overfitting the training samples. On the other hand, a too small network architecture may not be able to find the proper fit to the data. A variety of algorithms have been developed to optimize the network structure, including *network pruning* [103–107], *network construction* [108, 109], *network regularization* [110–112] and *evolutionary computation*-based optimization [113, 114]. Some of these algorithms will be reviewed in Section 3.2.4.

In this chapter, a new algorithm for network pruning based on sparse representation is presented. The proposed method, termed *sparse representation pruning* (SRP), is capable of creating a sparse network structure with good generalization ability. To the best of our knowledge, it is the first attempt to solve the structural optimization problem for NNs using sparse representation. The key properties of SRP are as follows:

1. Network structural optimization is done by iteratively selecting important elements (hidden neurons) that minimize the residual error. By contrast, traditional pruning algorithms eliminate redundant elements from the original architecture, which could be time-consuming particularly for large networks.

2. The SRP-based structural optimization method does not require the smoothness of the objective function, in contrast to some other techniques which depend on the assumption of smoothness [105–107].

The remainder of the chapter is organized as follows. Section 3.2 presents a brief review of feed-forward neural networks and existing network pruning algorithms. Section 3.3 describes the proposed SRP algorithm and establishes the link between the structural optimization and the sparse representation. Section 3.4 discusses implementation issues and presents experimental results, followed by concluding remarks in Section 3.5.

3.2 Feed-forward neural networks

Neural networks have been inspired by biological observations of the human brain function [115]. Over the past several decades, neural networks have evolved into powerful computation systems, which are able to learn complex nonlinear input-output relationship from data. There are two general types of network topology: *feedback* or *recurrent neural networks* and *feed-forward networks*. In recurrent networks, the connections between neurons can form directed cycles. Feed-forward neural networks, on the other hand, process the information only in the forward direction, from the input layer to the output layer. In this chapter, we focus on feed-forward networks. Subsections 3.2.1 and 3.2.2 present the basic concept of the perceptron neuron and feed-forward network structure, respectively. Then, Subsections 3.2.3 and 3.2.4 briefly describe conventional training and pruning algorithms developed for feed-forward neural networks.

3.2.1 Perceptron neuron

The perceptron is the basic information processing unit in neural networks. It takes input signals, combines them linearly, and then applies a nonlinearity to

produce the output signal. The general structure of a perceptron with Q inputs is shown in Fig. 3.1.

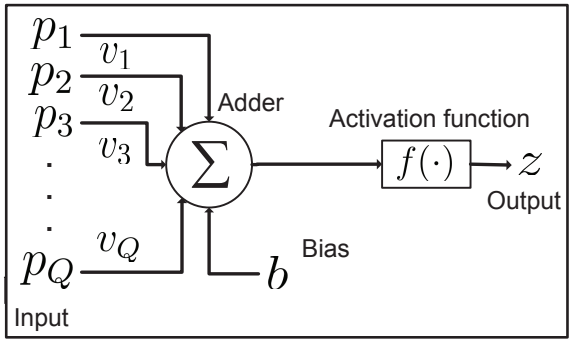


Figure 3.1: The perceptron model.

Each input component receives a signal p_i, which is transmitted along a synapse or connection link characterized by a weight v_i. An adder is employed to sum all the weighted input signals $v_i p_i$. Another input b, known as the bias, is also introduced to adjust the input signals. An activation function $f(\cdot)$, also known as transfer function, is used to control the amplitude range of the neuron output and provide a non-linear mapping from input to output. There are a number of activation function types, including the linear, sigmoid, logistic sigmoid and hyperbolic tangent functions [115]. The response of the perceptron to an input vector $\mathbf{p}$ can be expressed as

$$z = f\left(\mathbf{v}^T\mathbf{p} + b\right), \tag{3.1}$$

where $\mathbf{v}$ is the weight vector, b is the bias, T denotes the transpose operator, and $f(\cdot)$ is the activation function.

3.2.2 Network structure

A multilayer perceptron (MLP) or multilayer feed-forward neural network consists of a set of input neurons, one or more hidden layers of perceptrons, and an output layer of linear or sigmoid type neurons. Each neuron is connected to all

neurons in the succeeding layer. Signals only propagate through the network in the feed-forward direction.

For simplicity, we consider a network architecture with one hidden layer. Figure 3.2 shows the general structure of a fully connected network containing one hidden layer. On the left hand side, the input layer receives signals from the external environment. In the middle is the hidden layer, which receives signals from the input layer and sends its output signals to the output layer. The output layer processes the signals received from the hidden layer and produces the network response.

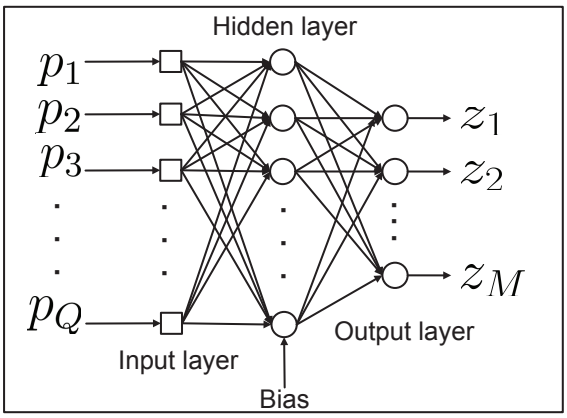


Figure 3.2: A feed-forward neural network with one input, hidden, and output layer.

Let $\mathbf{p} = \left[p_1, p_2, p_3, ..., p_Q\right]^T$ denote the input vector to a neural network with N hidden units and M output neurons. The output vector of the hidden layer $\mathbf{x}$ can be expressed as

$$\mathbf{x} = f\left(V\mathbf{p} + \mathbf{b}_1\right), \tag{3.2}$$

where $f(\cdot)$ is the activation function, $\mathbf{b}_1$ is the bias vector, and V is an $N \times Q$ weight matrix between the input and hidden layers. The i-th ($i = 1, 2, \cdots, N$) row of V represents the weights of the i-th hidden neuron. The final output of the network is given by

$$\mathbf{z} = g\left(W\mathbf{x} + \mathbf{b}_2\right), \tag{3.3}$$

where $g(\cdot)$ is the output activation function, $W \in \mathbb{R}^{M \times N}$ comprises the weight

vectors between the hidden and output layers, and $\mathbf{b}_2$ is the bias vector of the output neurons.

3.2.3 Network training

During network training, the actual network outputs are compared with the desired outputs or targets from the training patterns. The difference between the actual and desired outputs is treated as the network error that is later propagated back to adjust the network free parameters, i.e., weights and biases. That is, the training process adjusts the network free parameters so as to make the actual outputs of the network closer to the targets or desired outputs. There are a number of error criteria that can be minimized during training, the most common of which is the mean squared error (MSE):

$$\mathbb{E} = \frac{1}{L}\sum_{i=1}^{L}\left(\mathbf{y}_i - \mathbf{z}_i\right)^2, \tag{3.4}$$

where L is the number of training patterns, and $\mathbf{y}_i$ and $\mathbf{z}_i$ are, respectively, the i-th desired and actual outputs. To minimize the error in Eq. (3.4), different training algorithms are applied, such as *first order* [116–119], *second order* [119, 120], *hybrid methods* [121–123] and *evolutionary computation*-based algorithms [113, 124, 125].

Next, we briefly describe the *resilient back-propagation* (RPROP) training algorithm in [117], which is used in the experimental result section of this chapter. The RPROP algorithm is one of the most commonly used first-order methods. Let $\mathbb{E}(t)$ be the MSE between the actual and desired output at the t-th iteration. Each network parameter is updated according to the rule

$$\xi(t+1) = \xi(t) - \operatorname{sign}\left(\frac{\partial \mathbb{E}(t)}{\partial \xi(t)}\right)\Delta(t), \tag{3.5}$$

where $\xi(t)$ is the parameter vector at the t-th iteration containing all adaptable weights and biases from the network, $\Delta(t)$ is the step-size, and $\operatorname{sign}(\cdot)$ is the

signum function.

3.2.4 Network pruning

Before training the neural network, one is faced with the question of determining a suitable network structure. However, there exists no systematic way of choosing the optimum network structure. A large network architecture may result in poor generalization, even if it obtains high accuracy on the training data. A small network, on the other hand, takes a long time to train, and may not have sufficient free parameters to model the problem at hand. Therefore, determining the optimal network structure is one of the most important issues in neural network design. Sometimes trial-and-error techniques or cross validation methods, which are time-consuming, are used to search for the optimal structure before training the network. Often, however, more systematic methods are employed to optimize the neural network structure without degrading the network performance. They can be broadly categorized as follows:

- *network pruning*, which is based on a saliency analysis of each element (weights or hidden neurons) [103, 104, 106, 126–129];
- *network construction*, which begins with a small network and incrementally adds hidden neurons during the training process [130, 131];
- *evolutionary algorithms*, which are based on evolutionary search strategy such as genetic algorithms [132].

In this thesis, we mainly focus on network pruning algorithms. The general framework for network pruning can be described as follows:

1. set up a large neural network architecture and train it with any learning method (e.g. back-propagation algorithm), until the stopping criterion is met;

2. compute the significance for all elements and eliminate the least important ones;

3. retrain the pruned network; if the change in outputs between the original and pruned network is small enough, then go to step 2; otherwise stop.

The network pruning algorithms can be further classified into two categories: weight pruning and hidden neuron pruning. Examples of weight pruning algorithms include *Optimal Brain Damage* (OBD) [126], *Optimal Brain Surgeon* (OBS) [127], and *Magnitude-based pruning* (MAG) [128]. On the other hand, hidden neuron pruning algorithms include *Skeletonization* (SKEL) [103], *non-contributing units* (NC) [104] and *Extended Fourier Amplitude Sensitivity Test* (EFAST) [106]. In SKEL, the saliency of a neuron is measured by the training error when the neuron is removed. The NC method searches for non-contributing hidden neurons by checking their effect on the network output. As for the EFAST algorithm, the importance of neurons is evaluated using an amplitude sensitivity test, which is a variance-based global sensitivity analysis method.

In short, conventional pruning algorithms attempt to estimate the sensitivity of the network elements (weights or hidden neurons). Thus, the significant elements which minimize the training error are maintained, while others with the least contribution are removed. The pruning-based methods benefit from the fast convergence of the initial large network to the local minimum. In spite of this, the pruning-based methods are time-consuming with large computational complexity. In Section 3.4, we will compare the existing SKEL, NC and EFAST methods with the proposed SRP algorithm.

3.3 Proposed SRP algorithm

In this section, we formulate the network structural optimization problem as a multiple measure vector (MMV) model. First, the MMV model is briefly de-

scribed. Second, network pruning is formulated as a linear inverse problem and cast as an MMV problem.

3.3.1 MMV model

The MMV model processes several measurement vectors simultaneously and produces a matrix solution [59, 60]. Consider the measurement matrix $Y \in \mathbb{R}^{M \times L}$, comprising L measurement vectors, and a known dictionary $\mathcal{D}$ containing N atoms. The MMV model aims to minimize the matrix sparsity of the solution:

$$\min \; S(X) \quad \text{subject to} \quad Y = \mathcal{D}X, \tag{3.6}$$

where $X \in \mathbb{R}^{N \times L}$ is the matrix solution, and $S(X)$ denotes the measure of the matrix sparsity. The principle of the MMV problem is that all the columns from the solution matrix X possess the same sparsity profile; that is, the columns of X share the indices of nonzero elements. Let $\mathbf{r}_i$ be the i-th row of the matrix X. Furthermore, let us define a vector $\mathbf{s}$ of length N whose i-th element s_i is given by:

$$s_i = \|\mathbf{r}_i\|_p \,, \tag{3.7}$$

where $\|\cdot\|_p$ denotes the vector $p-$norm $(0 \leq p \leq 1)$. One simple strategy to measure the matrix sparsity is to use the l_0 pseudo-norm of the vector $\mathbf{s}$:

$$S(X) = \|\mathbf{s}\|_0 \,. \tag{3.8}$$

However, the l_0-based optimization is computationally expensive because an exhaustive enumeration is required in terms of all possible locations of nonzero rows in X. In addition, the computational complexity grows exponentially with the size of the measurement matrix. Therefore, rather than using the l_0 pseudo-norm, we replace it with the l_1 norm. The MMV problem can then be stated

as

$$\min\ S(X) = \|\mathbf{s}\|_1 \quad \text{subject to} \quad Y = \mathcal{D}X. \tag{3.9}$$

In [59], the authors proved that the minimization problem in (3.9) is equivalent to that based on (3.8) when the sparsity of the solution X is sufficiently small.

3.3.2 Sparse representation based pruning

Neuron pruning algorithms optimize the network structure by eliminating the least important hidden neurons from the original network. Furthermore, removing neurons from the hidden layer is equivalent to removing the connection weights from the hidden neurons to the output layer. Next we formulate the neuron pruning problem as an MMV model.

Consider a three-layer fully-connected network with one hidden layer, an input layer and an output layer. Suppose the initial network architecture consists of Q inputs, N hidden neurons and M outputs. Let $P = \left[\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_L\right]$ be a matrix containing L training patterns and $Y = \left[\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_L\right]$ be the desired output matrix; each pair ($\mathbf{p}_i$, $\mathbf{y}_i$) forms an input vector and the corresponding desired output vector. Moreover, let $X \in \mathbb{R}^{N\times L}$ denote the output matrix of the hidden layer, in which the i-th row represents the output from the i-th hidden neuron, $i = 1, 2, ..., N$, and $Z \in \mathbb{R}^{M\times L}$ denote neural network output matrix corresponding to the input matrix P. Equations (3.2) and (3.3) can be expressed in matrix form as follows:

$$\begin{aligned} X &= f(VP + B_1), \\ \text{and} \quad Z &= g(WX + B_2). \end{aligned} \tag{3.10}$$

where V is the weight matrix between the input and hidden layers, W comprises the weight vectors between the hidden and output layers, B_1 is the bias matrix having the bias vector $\mathbf{b}_1$ as its columns, and B_2 is the bias matrix of the output layer with columns containing the bias vector $\mathbf{b}_2$. Without loss of generality,

we further assume that $g(\cdot)$ is a linear activation function. Note that if $g(\cdot)$ is an invertible function, we can transform the output neurons to linear units by applying the inverse function $g^{-1}(\cdot)$. In this case, the actual output of the network can simply be expressed as

$$Z = WX + B_2. \tag{3.11}$$

Give the desired output matrix Y, Eq. (3.11) can also be rewritten as:

$$Y = Z + E = WX + B_2 + E, \tag{3.12}$$

where $E = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_L]$ is the network error matrix; $\mathbf{e}_i$ is the error between the actual output $\mathbf{z}_i$ and the desired output $\mathbf{y}_i$.

The objective of the pruning process is to reduce the number of hidden neurons. Removing the i-th hidden neuron is equivalent to setting its output to zero, i.e., the i-th row of X becomes a null vector. Therefore, the pruning process is equivalent to minimizing the number of nonzero rows in X, or the matrix sparsity. The structural optimization problem then can be cast as the following MMV model:

$$\min\ S(X) \quad \text{subject to} \quad \left\|\tilde{Y} - WX\right\|_2 \leq \epsilon, \tag{3.13}$$

where $S(X)$ is a matrix sparsity measure, $\tilde{Y} = Y - B_2$, W is the weight matrix between the hidden and output layers, and ϵ is a noise bound.

Comparing Eq. (3.13) with the MMV model of Eq. (3.6), we can see that W serves as the dictionary $\mathcal{D}$ and the matrix $\tilde{Y}$ plays the role of the measurements. Thus, the network structural optimization can be regarded as finding the sparse representation for the matrix X. As a result, by selecting important hidden neurons, the output matrix X becomes a sparse matrix in which most rows have zero values.

To solve the MMV model, we apply the M-OMP algorithm presented in [60] due to its simplicity and efficiency. The M-OMP algorithm starts from a null solu-

tion and iteratively adds one nonzero element. In addition, the M-OMP algorithm will not select the same atom twice. Based on the above properties of M-OMP, at the t-th iteration, the SRP algorithm selects t different hidden neurons from the original network. The outputs from selected neurons are then accumulated to approximate the desired output. Thus, the representation error is expected to decrease when more neurons are involved.

After minimizing the sparsity of the matrix X, we also need to update the network parameters between the input and hidden layer. Let $\widehat{V} = [V, \quad \mathbf{b}_1]$ be the matrix of weights and biases of the hidden layer, and $\widehat{P}$ be the augmented input matrix:

$$\widehat{P} = \begin{bmatrix} P \\ \mathbf{1}^T \end{bmatrix},$$

where $\mathbf{1}^T$ is the L-dimensional row vector whose elements are equal to 1. The output matrix of the hidden layer can then be rewritten as

$$X = f\left(VP + B_1\right) = f(\widehat{VP}), \tag{3.14}$$

Assuming the activation function of the hidden layer ($f(\cdot)$) is invertible, then the matrix $\widehat{V}$ can obtained by solving the following least squares problem:

$$\widehat{V} = \arg\min_{\widehat{V}} \left\| f^{-1}(X) - \widehat{VP} \right\|_2. \tag{3.15}$$

where $f^{-1}(\cdot)$ represents the inverse of the activation function $f(\cdot)$.

The proposed framework for the structural optimization, termed sparse representation pruning (SRP), is then summarized in Algorithm 6. The proposed algorithm is similar with the conventional pruning-based approaches in that both the SRP and pruning-based methods require network training in the first place. The trained network provides a candidate structure for the subsequent pruning. The difference between the SRP and conventional pruning-based algorithms

comes from the optimization strategy. The proposed SRP algorithm selects important elements (considered as forward selection) whereas the traditional pruning algorithms adopt the backward elimination strategy to remove redundant hidden neurons. Furthermore, the termination criterion employed by the SRP method is predefined using the error from the validation data set. That is, the SRP-based pruning is terminated if the validation error increases for three successive iterations. This simple strategy works based on the assumption that an increase in error from the validation set indicates the start of overfitting.

Algorithm 6: SRP algorithm for structural optimization.

input : A trained network and maximum iteration T;

output: The sparse matrix X^*;

$\mathcal{D} = W$, $E(0) = \widehat{Y} = Y - B_2$;

for $t = 1$ **to** T **do**

  Update X by solving the MMV model in Eq. (3.13):

  $(X(t), E(t)) = M\text{-}OMP(\mathcal{D}, E(t-1), t)$,

  where M-OMP denotes Algorithm 2 from Chapter 2;

  Update the weight matrix V using Eq. (3.15);

  Evaluate the updated network using validation data set;

  if *the predefined termination condition is met* **then**

    break;

  end

end

$X^* = X(t)$;

3.4 Experimental results

This section presents the experimental results and comparison of the proposed SRP approach with other conventional neuron pruning methods. The proposed method is tested on various problems from the Proben [133] and UCI [134] bench-

mark data sets. The data sets, network architecture, and the evaluation criterion are presented in Subsection 3.4.1. The performance of the SRP on these data sets is evaluated in Subsection 3.4.2, and the comparison results are presented in Subsection 3.4.3.

3.4.1 Experimental methods

Eight benchmark problems for classification and function approximation, from Proben [133] and UCI [134] benchmark data sets, are chosen for experimental evaluation. The problems are selected so as to ensure a good coverage of classification and regression problems with binary and continuous variables and different numbers of input and output attributes. Table 3.1 presents details of the data sets.

Table 3.1: Data sets used for classification and function approximation benchmarking. The data sets "Building" and "Flare" consist of two sub problems with the same number of inputs and outputs.

Data set	Type	Input	Output	Training	Validation	Test
Cancer	Classification	9	2	350	175	174
Card	Classification	51	2	345	173	172
Diabetes	Classification	8	2	384	192	192
Building1	Approximation	14	3	2104	1052	1052
Building2	Approximation	14	3	2104	1052	1052
Flare1	Approximation	24	3	533	267	266
Flare2	Approximation	24	3	533	267	266
Hearta	Approximation	35	1	152	76	75

Each data set is partitioned into three subsets: a training set, a validation set, and a test set. The training set is used to train and optimize the network architecture. The validation set is used for the stopping criterion and the test set is used for evaluation of the generalization ability of the pruned network. The size of the training, validation and test sets in all cases is 50%, 25%, and 25%, respectively.

In all experiments, a fully-connected feed-forward network is employed with an input layer, one hidden layer, and an output layer. The activation function of the

hidden layer is set to tangent sigmoid function. The output neuron uses the linear function as activation function. The network is initialized with random weights in the range $[-0.1\ ,\ 0.1]$, and it is trained with the resilient back-propagation algorithm (RPROP) [117]. The training parameters are set as follows:

- the maximum number of training iterations is 500;
- the minimum performance gradient is 10^{-6};
- the learning rate is 0.01.

The network training terminates when either the maximum number of iterations is reached or the performance gradient falls below 10^{-6}.

The generalization performance is evaluated using the classification error for the classification problems, and the normalized root-mean-square error (NRMSE) for the function approximation problems. The NRMSE is calculated as follows:

$$\text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{L}\left(y_{ij}-z_{ij}\right)^2}{\sum_{i=1}^{M}\sum_{j=1}^{L}y_{ij}^2}}, \tag{3.16}$$

where M is the number of outputs, L is the number of training patterns, z_{ij} is the i-th actual output corresponding to the j-th input pattern, and y_{ij} is the desired output. The resulting structure is measured by the number of remaining hidden neurons.

3.4.2 Performance analysis of SRP

In this subsection, the performance of SRP is evaluated on classification and regression problems. First, we consider how the generalization ability of the original large network is improved using the SRP algorithm. We also investigate the effect of the initial network size on the structure of the pruned network and its generalization performance. A large network is more prone to converging to a local minimum than a smaller network. However, the network with a small size

may not generalize very well. The purpose is then to find out the robustness of the proposed algorithm to various network sizes. The number of initial hidden neurons (N) is set to 32, 64, 128, and 256. The initial networks are then trained by the RPROP method with training parameters as presented in Subsection 3.4.1.

Table 3.2 shows the average classification error and NRMSE of the original network over 30 independent runs. From the results presented in this table, it can be clearly observed that as the size of the hidden layer increases, the generalization ability of the network reduces. The average classification error rate (over all data sets) for $N = 256$ increases by 4.94% compared to the average error rate for $N = 32$ hidden neurons. Table 3.3 presents the related classification error rates and the NRMSE of the proposed SRP algorithm. Clearly the pruned network using SRP outperforms the original trained network by achieving lower classification and regression errors on the test sets. In particular, the generalization ability of the large networks ($N = 256$) is improved greatly after applying SRP. Furthermore, the performance across all initial network sizes is comparable; that is, the SRP algorithm yields similar performance for all initial network sizes. For example, the difference of average classification rate over all data sets between the worst ($N = 64$) and the best ($N = 256$) is less than 0.61%. Table 3.4 shows the number of remaining neurons after applying the SRP algorithm. The large initial networks provide more hidden neurons for the proposed algorithm to select from. The final networks, however, contain similar number of hidden neurons. This proves the robustness of the proposed SRP method to the initial network size; that is, the size of pruned structure obtained from the SRP algorithm is not affected by the initial network size. Table 3.5 shows the computation time of the SRP algorithm for different initial network sizes. The SRP method requires slightly more pruning time for bigger initial network size. However, the difference of average computation time is less than 0.28 second. This is because the SRP algorithm selects hidden neurons instead of eliminating the redundant neurons. As a result, the computational complexity depends on the number of hidden

neurons to be selected instead of the number of hidden neurons to be removed.

Table 3.2: Classification and regression error on the test set from the original trained networks, using the conventional RPROP method.

Data set	Classification Error (%)			
	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Cancer	**1.94±0.85**	2.91±1.42	5.54±2.57	8.86±2.21
Card	**19.42±1.96**	20.52±2.31	22.14±3.27	24.10±4.06
Diabetes	**26.11±3.99**	26.58±3.11	29.48±3.86	29.37±4.23
Average	**15.83±2.27**	16.67±2.28	19.05±3.23	20.77±3.50
Data set	NRMSE			
	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Building1	**0.234±0.072**	0.274±0.083	0.336±0.085	0.441±0.132
Building2	**0.141±0.081**	0.161±0.098	0.192±0.120	0.240±0.143
Flare1	**1.491±0.197**	2.061±0.742	2.880±0.990	4.239±1.710
Flare2	**1.618±0.606**	2.330±0.995	3.255±0.837	4.748±1.282
Hearta	**0.719±0.178**	0.879±0.426	1.188±0.755	1.769±1.002
Average	**0.841±0.227**	1.141±0.469	1.570±0.557	2.287 ± 0.854

Table 3.3: Classification and regression error on the test set using the SRP-based algorithm with initial networks of different sizes.

Data set	Classification Error (%)			
	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Cancer	1.27±0.33	**1.23±0.41**	1.35±0.37	1.26±0.21
Card	15.25±2.31	15.85±1.73	15.23±2.38	**15.12±2.11**
Diabetes	24.52±3.13	24.13±2.12	23.27±3.23	**23.12±2.11**
Average	13.68±1.92	13.77±1.42	13.28±1.99	**13.16±1.48**
Data set	NRMSE			
	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Building1	0.233±0.025	**0.206±0.033**	0.244±0.028	0.235±0.031
Building2	0.163±0.032	**0.162±0.022**	**0.162±0.022**	**0.162±0.022**
Flare1	0.985±0.103	0.980±0.123	**0.941±0.135**	0.944±0.083
Flare2	1.025±0.201	0.990±0.201	0.996±0.198	**0.988±0.182**
Hearta	0.443±0.011	0.435±0.015	0.428±0.021	**0.415±0.011**
Average	0.569±0.074	0.557±0.079	0.554±0.081	**0.548±0.065**

Table 3.4: Average number of remaining neurons for the SRP-based algorithm based on initial networks of different sizes.

Data set	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Cancer	2.5±0.8	**2.3±0.7**	2.6±0.3	2.4±0.2
Card	**6.1±0.9**	6.4±0.8	6.8±0.6	6.6±0.8
Diabetes	4.8±0.3	**4.7±0.5**	4.9±0.3	5.2±0.4
Building1	**13.9±1.3**	14.4±1.8	15.8±1.7	15.2±1.3
Building2	**18.9±2.1**	19.1±2.0	19.0±2.3	19.2±2.1
Flare1	8.8±0.9	8.7±0.8	**8.7±0.7**	9.0±0.8
Flare2	**9.3±0.6**	9.5±0.7	9.8±0.6	9.9±0.8
Hearta	**5.9±0.5**	6.1±0.6	6.2±0.6	6.3±0.5
Average	**8.8±0.9**	8.9±1.0	9.2±0.9	9.2±0.9

Table 3.5: Summary of average pruning time (second) for the SRP-based algorithm based on initial networks of different sizes.

Data set	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Cancer	0.032±0.021	0.035±0.012	0.042±0.011	**0.031±0.010**
Card	**0.031±0.012**	0.047±0.018	0.047±0.021	0.056±0.022
Diabetes	**0.031±0.025**	0.032±0.010	0.047±0.011	0.063±0.012
Building1	**0.728±0.121**	0.848±0.201	0.828±0.103	0.878±0.087
Building2	**2.406±0.501**	2.512±0.751	2.922±0.831	3.359±0.981
Flare1	0.031±0.012	**0.031±0.011**	0.047±0.021	0.054±0.013
Flare2	**0.047±0.021**	0.063±0.022	0.071±0.023	0.094±0.017
Hearta	**0.031±0.011**	0.035±0.021	0.041±0.013	0.040±0.023
Average	**0.292±0.090**	0.450±0.131	0.506±0.129	0.572±0.146

Based on the results presented in Tables 3.2 to 3.5, we can conclude that the performance of the original network is significantly improved after applying the SRP algorithm, compared to existing training algorithm. Although slightly more hidden neurons are selected with bigger initial network, the SRP method leads to similar generalization performance.

3.4.3 Comparison with traditional pruning algorithms

In this subsection, the proposed SRP algorithm is compared with conventional pruning algorithms. Three existing neuron-pruning methods are used for comparison: SKEL [103], NC [104], and EFAST [106]. To make the comparison fair, we implement existing neuron-pruning methods using the same stopping criterion. In other words, the pruning is terminated if the validation error increases for three successive iterations. All the networks are initialized with 128 hidden nodes.

Table 3.6 presents the classification and approximation errors of the proposed SRP algorithm and conventional pruning methods. The proposed SRP approach achieves the lowest error for five of the eight data sets. In the other three ("Building1", "Building2", and "Flare2") data sets, SRP achieves the second smallest error. Figure 3.3 presents the Receiver Operating Characteristic (ROC) curves of the SRP algorithm and those of existing methods for the three classification problems.

Tables 3.7 and 3.8 show, respectively, the number of remaining neurons and pruning time of the SRP and traditional pruning algorithms. For the pruned structure, SRP employs slightly more hidden neurons (9.2) than the EFAST method (6.7) for solving eight problems. However, the average network structure obtained from SRP is much smaller than NC (84.2) and SKEL (73.9), respectively. Moreover, SRP requires the least time to converge. The reason is that SRP selects the most significant elements from the trained network. This method can be regarded as a forward selection instead of backward elimination, hence fast convergence is

expected. By contrast, NC, SKEL, and EFAST methods spend much longer time on pruning unimportant hidden neurons.

Table 3.6: Classification and approximation error rate on the test sets from various pruning algorithms.

Data set	Classification error (%)			
	NC	SKEL	EFAST	SRP
Cancer	2.56±1.23	2.83±1.13	2.53±1.33	**1.35±0.37**
Card	19.62±2.43	15.81±3.22	18.21±2.91	**15.23±2.38**
Diabetes	28.11±2.13	26.51±2.43	24.59±2.61	**23.27±3.23**
Average	16.76±1.93	15.05±2.26	15.11±2.28	**13.28±1.99**
	NRMSE			
Building1	**0.104±0.021**	0.104±0.022	0.839±0.101	0.244±0.028
Building2	0.520±0.081	0.519±0.085	**0.155±0.024**	0.162±0.022
Flare1	2.055±0.913	1.953±0.816	1.400±0.502	**0.941±0.135**
Flare2	1.985±0.893	1.453±0.906	**0.912±0.502**	0.996±0.198
Hearta	2.871±1.024	1.132±0.521	0.896±0.303	**0.428±0.021**
Average	1.514±0.586	1.032±0.470	0.840±0.286	**0.554±0.081**

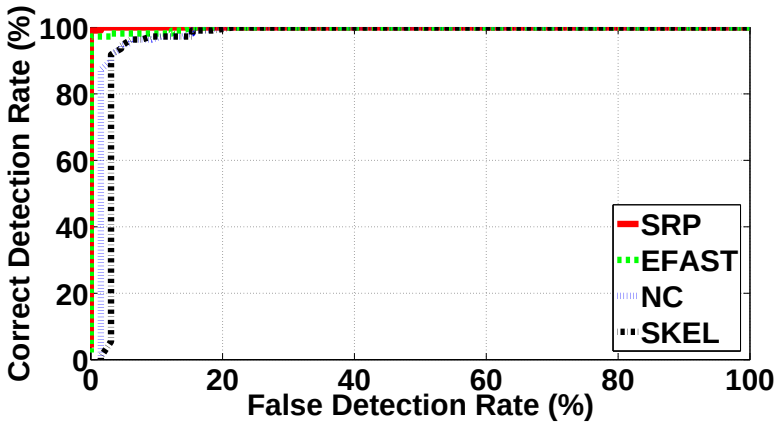
Table 3.7: Remaining neurons for different pruning methods.

Data set	NC	SKEL	EFAST	SRP
Cancer	102.3±10.0	105.6±8.9	6.8±1.2	**2.6±0.3**
Card	22.0±2.5	**4.3±1.9**	11.2±2.3	6.8±0.6
Diabetes	15.8±2.3	5.3±1.8	7.2±1.4	**4.9±0.3**
Building1	121.8±3.7	122.2±2.6	**6.3±1.5**	15.8±1.7
Building2	124.3±2.1	125.2±1.5	**7.2±1.5**	19.0±2.3
Flare1	127.1±0.9	127.0±1.0	**5.5±1.3**	8.7±0.7
Flare2	117.2±2.8	83.8±12.5	**4.2±2.3**	9.8±0.6
Hearta	43.2±5.8	17.6±3.2	**4.9±2.0**	6.2±0.6
Average	84.2±3.8	73.9±4.2	**6.7±1.7**	9.2±0.9

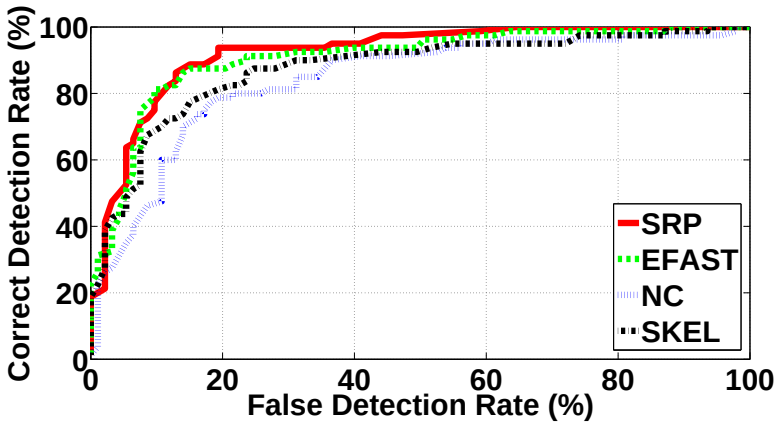
3.5 Conclusion

In this chapter, we have presented a novel method for neural network structural optimization using the multiple measurement vector (MMV) model. The new method leads to a good generalization performance and a compact architecture.

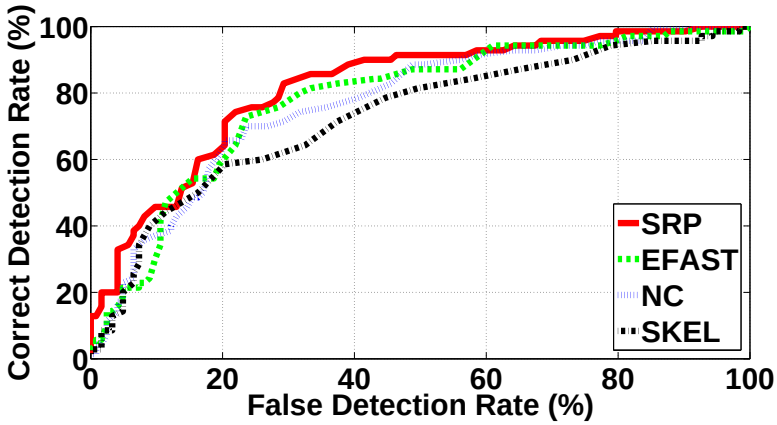
The link between the optimization of the network structure and the MMV model was established. The proposed method was employed to find the sparse



(a) "Cancer" data set



(b) "Card" data set



(c) "Diabetes" data set

Figure 3.3: Receiver Operating Characteristic (ROC) for pruning algorithms.

Table 3.8: Pruning time for various pruning methods.

Data set	NC	SKEL	EFAST	SRP
Cancer	92.5±23.0	12.6±5.9	8.8±3.2	**0.04±0.01**
Card	420.0±122.3	63.3±21.9	18.2±5.3	**0.05±0.02**
Diabetes	321.8±72.1	45.3±7.8	10.2±3.4	**0.05±0.01**
Building1	201.8±53.7	22.2±8.6	26.3±8.5	**0.83±0.10**
Building2	62.3±22.1	21.5±3.5	25.2±5.4	**2.92±0.83**
Flare1	98.5±18.8	7.0±1.8	2.5±1.3	**0.05±0.02**
Flare2	117.2±2.8	83.8±12.5	4.2±2.3	**0.07±0.02**
Hearta	43.2±5.8	17.6±3.2	4.9±2.0	**0.04±0.01**
Average	169.7±40.1	34.2±8.2	12.5±3.9	**0.51±0.13**

representation of the network parameters by selecting the most significant hidden neurons, instead of eliminating the redundant ones, thereby overcoming overfitting and leading to quick convergence. The proposed method was evaluated on a diverse set of classification and function approximation problems. Experimental results show that the proposed SRP algorithm achieves better generalization ability and faster convergence compared to existing neuron pruning algorithms.

Chapter 4

Neural Network Training using Sparse Representation

Chapter contents

4.1 Introduction

Supervised learning has a long history in machine learning [117–119, 121–123, 135–143]. The fundamental challenge in developing a supervised learning algorithm is the trade-off between the generalization performance and the model complexity [105–107]. When the supervised learning algorithm fails to converge to the global minima, other techniques, such as architecture optimization methods, are employed to compensate for the network performance [126, 127, 130–132].

The sparse representation pruning (SRP) method presented in Chapter 3 improves the network generalization performance by optimizing network architecture. The initial network is first trained and then the hidden neurons are selected based on their significance. One disadvantage, however, is that the SRP method requires initial network training, which is time-consuming, especially for a large-sized network.

In this chapter, an extension of the SRP algorithm is proposed to train the initial network and optimize the structure simultaneously. This algorithm is conceptually related to the SRP model. Most importantly, using a dictionary learning strategy, the proposed algorithm supports more efficient computation than the typical prune-after-training method. To the best of our knowledge, this proposed algorithm, termed *Sparse Neural Network* (SNN), is the first attempt to train NNs based on the sparse representation paradigm. The novelty of the proposed algorithm is summarized as follows:

1. The SNN framework achieves a compact structure by selecting significant hidden neurons, instead of connecting all hidden neurons to the output

nodes. At each training iteration, the hidden neuron that reduces the residual error the most is selected. Therefore, the training accuracy improves continually as more hidden neurons are added to the selection.

2. To obtain a more parsimonious network architecture, the dictionary learning algorithm, based on the singular value decomposition, is introduced. The algorithm aims to maximize the diversity of the outputs from all hidden neurons, thereby improving performance of the sparsity-based supervised learning algorithm.

The rest of the chapter is organized as follows. Section 4.2 gives a brief account of traditional supervised learning algorithms. Section 4.3 details the sparsity-based training algorithm. In this section, the concept of the architecture sparsity is formalized and the problem of network training is reformulated as a sparse representation model. Then the dictionary-learning based algorithm is proposed to optimize the output from hidden neurons. Section 4.4 discusses the implementation issues and evaluates the performance of the proposed supervised learning method, followed by concluding remarks given in Section 4.5.

4.2 Neural network training algorithms

The neural network training refers to the process of tuning the free parameters so that the neural network yields a particular response to a specific input pattern. Network learning is necessary when the intrinsic relationship between inputs and outputs is unknown. After training, the network learns to recognize certain patterns and gives the correct output response. There are three major learning paradigms: *supervised learning*, *unsupervised learning*, and *reinforcement learning*.

1. Supervised learning simulates the learning ability of animals that tend to learn from examples [117–119, 121–123, 135–137]. There exists an external

"supervisor" with knowledge of the environment, in the form of input-output examples. The difference between the actual network output and a desired output is referred to as the error signal. The error is further minimized to find out the relationship between the input and output examples.

2. In addition to supervised learning, higher animals and in particular humans are capable of independently learning from input patterns only. In this process, there are no external "supervisors" or examples to be learned. In unsupervised learning the neural network is designed to discover correlations and hidden relationships from input patterns [144–149]. Discovering such relations allows the machine to make correct decisions based on the input data and to successfully forecast future outputs.

3. In reinforcement learning, the network receives a reinforcement signal based on the actions taken [150–153]. If a positive reinforcement is received, then the probability of the taken action is strengthened reinforced; otherwise, the tendency to produce that action is weakened.

In the following, we focus on supervised learning algorithms for neural networks. The purpose of these learning algorithms is inferring an input-output function from labeled training data. They have been studied extensively and employed for a wide spectrum of applications. In supervised learning, each training pattern consists of an input object and a desired output. The training algorithm analyzes the relationship between the input and output patterns to produce a mapping function. This mapping could be a classifier (in classification) or a regression function (in function approximation). A cost function is also introduced to measure the performance of the neural network. The commonly used cost function is the mean squared error (MSE), defined as

$$\mathbb{E} = \frac{1}{L} \sum_{i=1}^{L} \left(\mathbf{y}_i - \mathbf{z}_i \right)^2 , \tag{4.1}$$

where L is the number of training patterns, and $\mathbf{y}_i$ and $\mathbf{z}_i$ are, respectively, the desired and actual outputs corresponding to the i-th input pattern. Therefore, training a neural network model is equivalent to optimizing the mapping function of n variables, where n is the total number of network parameters (weights and biases).

4.2.1 First order methods

The first-order methods analyze the first two (constant and linear) terms of the Taylor series expansion of the cost function (4.1). These algorithms, where the local gradient is used to determine the direction of minimization, are known as steepest or gradient descent methods. The *back-propagation* (BP) method is applied to propagate the error from the output layer to the preceding layers, thereby allowing the computation of the error gradient with respect to the input and hidden layer parameters [116]. This BP algorithm is regarded as one of the most significant breakthroughs in neural network training. Let $\mathbb{E}(t)$ be the MSE between the actual and desired output at the t-th iteration, and $\boldsymbol{\xi}(t)$ be the parameter vector containing all adaptable weights and biases at the t-th iteration. The gradient $\nabla\mathbb{E}(t)$ of the error $\mathbb{E}(t)$ with respect to $\boldsymbol{\xi}(t)$ is computed as

$$\nabla\mathbb{E}(t) = \frac{\partial\mathbb{E}(t)}{\partial\boldsymbol{\xi}(t)}. \tag{4.2}$$

The update rule of the basic BP algorithm could be written as

$$\boldsymbol{\xi}(t+1) = \boldsymbol{\xi}(t) - \alpha\nabla\mathbb{E}(t), \tag{4.3}$$

where α is the learning rate (or step size). The problem with the standard first-order back-propagation method is the inefficient computation due to the slow convergence. Many improvements have been made to overcome the slow convergence of the BP algorithm, such as the *resilient back-propagation* (RPROP)

method [117, 118].

Furthermore, if the learning rate α is too small, the convergence of the training algorithm may be very slow. On the other hand, having α too large may result in the weights oscillating during iterations. A dynamic learning rate is more preferred to overcome the need for trial-and-error methods to select the learning rate. The method presented in [154] increases the learning rate when the output error decreases, whereas the learning rate is reduced if the error increases. In [155], the learning rate is adjusted to accelerate the convergence using the direction cosine of the error derivative vector. Another fast training algorithm is presented in [156], in which a linear system of equations for the output layer is developed. This system is then solved using the modified *Gram Schmidt* and *delta rule* algorithm to achieve a fast training.

4.2.2 Second order methods

In contrast to first-order methods, second-order algorithms use the Hessian matrix to perform better estimation of both step size and direction. In other words, the quadratic term from the Taylor expansion of the network error is also taken into account. *Newtons Method* is one of the basic second order algorithms, which updates the network parameters based on

$$\begin{array}{c} \nabla\mathbb{E}_1 + \frac{\partial^2 E}{\partial \xi_1^2}\Delta\xi_1 + \frac{\partial^2 E}{\partial \xi_1 \partial \xi_2}\Delta\xi_2 + \cdots + \frac{\partial^2 E}{\partial \xi_1 \partial \xi_n}\Delta\xi_n = 0 \\ \nabla\mathbb{E}_2 + \frac{\partial^2 E}{\partial \xi_2 \partial \xi_1}\Delta\xi_1 + \frac{\partial^2 E}{\partial \xi_2^2}\Delta\xi_2 + \cdots + \frac{\partial^2 E}{\partial \xi_2 \partial \xi_n}\Delta\xi_n = 0 \\ \cdots \\ \nabla\mathbb{E}_n + \frac{\partial^2 E}{\partial \xi_n \partial \xi_1}\Delta\xi_1 + \frac{\partial^2 E}{\partial \xi_n \partial \xi_2}\Delta\xi_2 + \cdots + \frac{\partial^2 E}{\partial \xi_n^2}\Delta\xi_n = 0 \end{array}, \tag{4.4}$$

where n is the total number of network parameters, $\nabla\mathbb{E}_i$ and ξ_i, respectively, represent the i-th element from gradient vector $\nabla\mathbb{E}$ and parameter vector $\boldsymbol{\xi}$, and

$\Delta\xi_i$ is the change of ξ_i. Equation (4.4) can also be written in matrix form:

$$-\nabla\mathbb{E} = \begin{bmatrix} \frac{\partial^2 E}{\partial \xi_1^2} & \frac{\partial^2 E}{\partial \xi_1 \partial \xi_2} & \cdots & \frac{\partial^2 E}{\partial \xi_1 \partial \xi_n} \\ \frac{\partial^2 E}{\partial \xi_2 \partial \xi_1} & \frac{\partial^2 E}{\partial \xi_2^2} & \cdots & \frac{\partial^2 E}{\partial \xi_2 \partial \xi_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial \xi_n \partial \xi_1} & \frac{\partial^2 E}{\partial \xi_n \partial \xi_2} & \cdots & \frac{\partial^2 E}{\partial \xi_n^2} \end{bmatrix} \begin{bmatrix} \Delta\xi_1 \\ \Delta\xi_2 \\ \cdots \\ \Delta\xi_n \end{bmatrix}, \tag{4.5}$$

where the square matrix of partial derivatives of E is the Hessian matrix:

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial \xi_1^2} & \frac{\partial^2 E}{\partial \xi_1 \partial \xi_2} & \cdots & \frac{\partial^2 E}{\partial \xi_1 \partial \xi_n} \\ \frac{\partial^2 E}{\partial \xi_2 \partial \xi_1} & \frac{\partial^2 E}{\partial \xi_2^2} & \cdots & \frac{\partial^2 E}{\partial \xi_2 \partial \xi_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial \xi_n \partial \xi_1} & \frac{\partial^2 E}{\partial \xi_n \partial \xi_2} & \cdots & \frac{\partial^2 E}{\partial \xi_n^2} \end{bmatrix}. \tag{4.6}$$

The optimal change in the weight vector is then computed as

$$\Delta\boldsymbol{\xi} = -H^{-1}\nabla\mathbb{E}. \tag{4.7}$$

However, the calculation of the Hessian matrix and its inverse is always computationally expensive, thereby leading to approximation methods being investigated. The *Quasi-Newton* methods have been proposed in [120], which avoid the direct computation of the Hessian inverse H^{-1} by iteratively computing a matrix Q_t, such that

$$\lim_{t \to +\infty} Q_t = H^{-1}. \tag{4.8}$$

The weights are updated using the following rule:

$$\boldsymbol{\xi}(t+1) = \boldsymbol{\xi}(t) - Q_t \nabla\mathbb{E}(t). \tag{4.9}$$

The disadvantage of Quasi-Newton methods is that the storage requirement of Q_t is proportional to the square of the number of free parameters. The *conjugate*

gradient (CG) methods are introduced due to their extremely moderate storage requirements [119, 157, 158]. The CG algorithms minimize the error using the search direction that is conjugate to the previous directions. Generally, the CG-based algorithms provide faster convergence than basic back-propagation technique and sometimes faster than Quasi-Newton methods.

4.2.3 Hybrid methods

Second order methods are far superior in terms of learning time when compared to first-order back-propagation methods. However, the convergence of network training cannot be always guaranteed because they are more likely to get stuck in local minima. Combining the training speed of Newton (second order) algorithm and the stability of BP (first order) algorithm, *Levenberg Marquardt* (LM) algorithm is proposed as one of the most efficient algorithms for training small and medium sized networks [121]. The LM algorithm approximates the Hessian matrix as follows:

$$H^* \approx J^{\mathrm{T}} J + \mu I, \tag{4.10}$$

where μ is called the combination coefficient, I is the identity matrix, and J is the Jacobian matrix. This Jacobian matrix can be computed as

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial \xi_1} & \frac{\partial e_{11}}{\partial \xi_2} & \cdots & \frac{\partial e_{11}}{\partial \xi_n} \\ \frac{\partial e_{12}}{\partial \xi_1} & \frac{\partial e_{12}}{\partial \xi_2} & \cdots & \frac{\partial e_{12}}{\partial \xi_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{LM}}{\partial \xi_1} & \frac{\partial e_{LM}}{\partial \xi_2} & \cdots & \frac{\partial e_{LM}}{\partial \xi_n} \end{bmatrix},$$

where L is the number of training patterns, M is the number of output dimensions, and e_{ij} is the training error at the j-th output when applying the i-th training pattern. The update rule of the LM algorithm is given by

$$\boldsymbol{\xi}(t+1) = \boldsymbol{\xi}(t) - \left(J^{\mathrm{T}}(t) J(t) + \mu I\right)^{-1} J(t)\, \mathbf{e}(t), \tag{4.11}$$

where $J(t)$ is the Jacobian matrix at the t-th iteration, and the error vector $\mathbf{e}(t)$ has the form

$$\mathbf{e}(t) = [e_{11}, e_{12}, \cdots, e_{LM}]^{\mathrm{T}}. \tag{4.12}$$

Another hybrid training algorithm is proposed in [123] by combining a new calculation of the Jacobian matrix with second-order learning methods. The training algorithm in [159] simplifies network training using the forward-only computation. One major feature is that it allows arbitrarily connected network structure. Therefore, more complex network architectures can be efficiently trained, leading to the reduction of the computational complexity.

4.2.4 Other methods

So far the above training methods are used for a fixed network structure. In other words, the number of layers or hidden neurons is determined before training. However, as mentioned in Section 3.2.4, there exists no systematic way of choosing the optimum network structure. Having a too large network size can result in overfitting, whereas a too small structure may not allow the network to reach the desired performance level. To dynamically tune the network structure while training, adaptive-structure training methods are employed, such as *Bayesian regularization* (BR) [160, 161], *standard constructive approach* (SCA) [162] and *evolutionary optimization*-based (EO) methods [113, 124, 125, 163]. These approaches train and optimize the network architecture simultaneously.

The BR method updates the network parameters according to Levenberg-Marquardt algorithm [160, 161]. In BR, the output errors and the network parameters are included in the cost function. Note that a network weight or bias is removed if its value is reduced to zero. Therefore, by minimizing the cost function, the BR algorithm achieves the minimal topology and trains the network simultaneously.

The SCA method starts with an empty topology [162]. During the training,

SCA dynamically determines the size of the network by adding extra hidden neurons, as long as the output error keeps decreasing. The SCA approach stops adding new neurons when the error starts increasing.

A few other methods have attempted to combine evolutionary computation with network training [113, 124, 125, 163]. EO methods are powerful search algorithms based on the mechanism of natural selection. They simultaneously consider many points in the search space so as to increase the chance of convergence to the global minimum. In EO-based training, each element from the chromosome represents a network parameter (either weight or basis). The evolutionary strategy is used to search the best combination of the chromosomes.

4.3 Sparsity-based network training

In this section, a neural network training algorithm based on sparse representation and dictionary learning is proposed. The proposed training method, referred to as SNN for short, generates a sparse network architecture and minimizes the training error simultaneously. Subsection 4.3.1 introduces the general framework of the proposed network training method. Subsection 4.3.2 describes the optimization strategy that selects the important hidden neurons from an initial network. Subsection 4.3.3 presents a dictionary learning-based approach to determine the optimum network parameters.

4.3.1 SNN Problem formulation

Consider a three-layer fully-connected network with one hidden layer, an input layer and an output layer. Suppose the initial network architecture consists of Q inputs, N hidden neurons and M outputs. Let $V \in \mathbb{R}^{N\times Q}$ and $W \in \mathbb{R}^{M\times N}$ denote the weight matrices of the hidden layer and the output layer, respectively, where the rows of the matrix represent the weight vectors of the individual neurons. Suppose we have L pairs $(\mathbf{p}_i, \mathbf{y}_i)$ of input vectors and their corresponding desired

output patterns. Let $P = \left[\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_L\right]$ be the input matrix and $Y = \left[\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_L\right]$ be the desired output matrix of the training patterns. The output matrix of the hidden layer can be expressed as

$$X = f\left(VP + B_1\right), \tag{4.13}$$

where $f(\cdot)$ is the activation function of the hidden layer, and B_1 is the bias matrix having the bias vector $\mathbf{b}_1$ as its columns. The i-th row of the matrix X represents the outputs of the i-th hidden neuron due to the input patterns P. Let $Z \in \mathbb{R}^{M \times L}$ denote the neural network output matrix, corresponding to the input matrix P:

$$Z = g\left(WX + B_2\right), \tag{4.14}$$

where $g(\cdot)$ is the activation function of output layers, and B_2 is the bias matrix of the output layer containing the bias vector $\mathbf{b}_2$ as its columns. Again, if $g(\cdot)$ is an invertible function, we can always replace the output neurons with linear units by applying the inverse function $g^{-1}(\cdot)$. Therefore, without loss of generality, we hereafter assume that $g(\cdot)$ is a liner activation function. The actual output of the network can then be expressed as

$$Z = WX + B_2. \tag{4.15}$$

Based on the above formulation, a new training algorithm is proposed which consists of two stages: Architecture optimization and Weight update.

Stage one: Architecture optimization. The proposed model optimizes the network structure during the training process. The SNN framework considers the network topology as a sparse structure, then the sparse representation algorithm is employed to reconstruct the network structure. The hidden neurons are evaluated in an iterative fashion based on their contribution to the desired output. Only the neurons that contribute most to the minimization of the residual error are selected

from the candidate neuron pool.

Stage two: Weight update. Architecture optimization aims to select important hidden neurons from a candidate neuron pool. The neuron selection is equivalent to finding the sparse representation for all existing hidden neurons. Furthermore, the weight matrix between the hidden and output layer is regarded as the dictionary in the sparse representation. To improve the performance of the sparse representation, a dictionary learning method is introduced to update the output-layer weights. This is done by maximizing the diversity of the outputs from hidden neurons.

In the following two subsections, we describe the architecture optimization and dictionary learning-based approach to train the neural network.

4.3.2 Architecture optimization

The architecture optimization stage aims to find a compact topology with the minimal number of hidden neurons. This is done by selecting important neurons from the existing network while ignoring the rest. Note that removing a hidden neuron from the initial network is equivalent to setting the output from this hidden neuron to zero. The architecture optimization method is similar with the SRP method presented in Chapter 3.

Given the desired output matrix Y, the bias matrix of the output layer B_2, and the weight matrix between the hidden and output layer W, Eq. (4.15) can be rewritten as

$$Y = Z + E = WX + B_2 + E, \tag{4.16}$$

where $E = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_L]$ is the network error matrix; that is, the column vector $\mathbf{e}_i$ represents the error between the actual output $\mathbf{z}_i$ and the desired output $\mathbf{y}_i$. The architecture optimization process is then cast as solving the following multiple

measurement vector (MMV) problem:

$$\min\ S(X) \quad \text{subject to} \quad \left\|\tilde{Y} - WX\right\|_2 \leq \epsilon, \tag{4.17}$$

where $S(X)$ is a measure of matrix sparsity, $\tilde{Y} = Y - B_2$, X is the output matrix from hidden neurons, and ϵ is the bound on the network error. We apply the M-OMP algorithm presented in [60] to solve the MMV model. After selecting the important hidden neurons from the network, the matrix X becomes a sparse matrix where most of the rows are zero. Let $\widehat{V}$ denote a matrix containing the weights and the biases of the hidden layer:

$$\widehat{V} = [V, \quad \mathbf{b}_1]\,. \tag{4.18}$$

Let $\widehat{P}$ be the augmented input matrix:

$$\widehat{P} = \begin{bmatrix} P \\ \mathbf{1}^T \end{bmatrix}, \tag{4.19}$$

where $\mathbf{1}$ is a L-dimensional vector whose elements are equal to 1. The weight and bias between the input and hidden layer can then be adjusted by minimizing the following least squares criteria:

$$\widehat{V} = \arg\min_{\widehat{V}} \left\| f^{-1}(X) - \widehat{VP} \right\|_2. \tag{4.20}$$

where $f^{-1}(\cdot)$ represents the inverse of the activation function $f(\cdot)$.

4.3.3 Weight update

During the architecture optimization, the proposed algorithm generates a compact architecture by selecting the most significant hidden neurons from the original network. The neuron selection is equivalent to finding the sparse representation

of the matrix X. Note that the weight matrix W (see Eq. (4.17)) serves as the dictionary $\mathcal{D}$ in the sparse representation. To improve the performance of the sparse representation, a novel dictionary learning algorithm, based on KSVD, is introduced to update the weight matrix W.

The KSVD algorithm, one of the most popular algorithms for the dictionary learning, is used to update the atoms through an SVD-based iterative method [75]. The traditional KSVD method assumes that the dictionary $\mathcal{D}$ is unknown so the method optimizes $\mathcal{D}$ as well as the sparse representation. Thus, the optimization problem becomes

$$(\mathcal{D}^*, X^*) = \arg \min_{\mathcal{D},X} \{S(X) + \|Y - \mathcal{D}X\|_2\}. \tag{4.21}$$

The interested reader is referred to [75] for more details on the KSVD algorithm. Despite the efficiency of training the dictionary, the KSVD method is not suitable for high dimensional signals. For instance, it relies on a pursuit algorithm to calculate the sparse coefficients for each column of the solution. Furthermore, KSVD only updates one atom in the dictionary at each iteration. All these properties lead to a high computational complexity.

Since the proposed SNN method is based on the MMV model, nonzero elements in the solution are clustered in a few rows. As a result, we are able to update more than one atom in the dictionary at each iteration. To aid the explanation, we firstly define an operator called *svds*. By the singular value decomposition, an arbitrary matrix $A \in \mathbb{R}^{M\times N}$ can be written as

$$A = \mathcal{U}\Sigma\mathcal{V}, \tag{4.22}$$

where $\mathcal{U} \in \mathbb{R}^{M\times M}$ and $\mathcal{V} \in \mathbb{R}^{N\times N}$ are orthogonal matrices, and Σ is a diagonal matrix of the same dimension as A. Given a positive integer k, we define the *svds*

operator as

$$\text{svds}(A, k) = \{\mathcal{U}_k, \Sigma_k, \mathcal{V}_k\}, \tag{4.23}$$

where $\mathcal{U}_k$ is the first k columns in $\mathcal{U}$, Σ_k is a diagonal matrix of size $k \times k$, and $\mathcal{V}_k$ is the first k columns in $\mathcal{V}$. Overall, the strategy for updating the dictionary and the corresponding coefficients matrix X, termed *multi-KSVD* (mKSVD), is shown in Algorithm 7.

Algorithm 7: mKSVD-based dictionary update strategy.

input : The signal matrix Y, the initial dictionary $\mathcal{D}(0)$, the corresponding coefficients matrix $X(0)$, the number of update atoms k and the number of maximal iterations T;

output: An optimized dictionary $\mathcal{D}^*$ and the coefficients X^*;

for $t = 1$ **to** T **do**

  repeat

    Select k non-zero rows from $X(t-1)$*:*

    $$X_s = \{\mathbf{x}_i | \mathbf{x}_i \in X(t-1) \text{ and } \mathbf{x}_i \neq 0\}.$$

    Record k columns from $\mathcal{D}(t-1)$ *that are associated with* X_s*:*

    $$\mathcal{D}_s = \{\mathbf{d}_i \in \mathcal{D}(t-1) \mid \text{row } \mathbf{x}_i \in X_s\},$$

    where $\mathbf{d}_i$ *is the i-th column from* $\mathcal{D}(t-1)$*.*

    Calculate the error due to $\mathcal{D}_s$*:*

    $$E_s = Y - \mathcal{D}(t-1)X(t-1) + \sum_{\mathbf{d}_i \in \mathcal{D}_s} \mathbf{d}_i \, \mathbf{x}_i.$$

    Apply the svds operator to E_s *to compute* X_s *and* $\mathcal{D}_s$*:*

    $$\{\mathcal{U}_k, \Sigma_k, \mathcal{V}_k\} = \text{svds}(E_s, k), \text{ and } \mathcal{D}_s = \mathcal{U}_k,\ X_s = \Sigma_k \mathcal{V}_k.$$

  until *all non-zero rows in* $X(t-1)$ *have been processed*;

end

Return $\mathcal{D}^* = \mathcal{D}(T)$*, and* $X^* = X(T)$*.*

As observed from Algorithm 7, when $k = 1$, the mKSVD method becomes the traditional KSVD algorithm, in which only one atom is updated at each iteration. Thus the mKSVD approach is regarded as a general case for the KSVD method. More importantly, mKSVD updates several atoms (hidden neurons)

simultaneously, which accelerates the convergence of the algorithm. The mKSVD algorithm is then employed in the SNN algorithm to optimize the dictionary W. This is done by replacing the signal matrix Y, the initial dictionary $\mathcal{D}(0)$, and the corresponding coefficients matrix $X(0)$ from Algorithm 7 with $\tilde{Y}$, W, and X from Eq. (4.17), respectively.

4.4 Experimental results

In this section, we evaluate the performance of the SNN algorithm based on several function approximation and classification problems. The details of the data sets, network architecture, and the evaluation criterion are presented in Subsection 4.4.1. The effect of the initial network size on the performance of the SNN method is evaluated in Subsection 4.4.2. The effect of the dictionary learning is analyzed in Subsection 4.4.3. The comparison results between the proposed algorithm and traditional training methods are then presented in Subsection 4.4.4.

4.4.1 Experimental setup

Several benchmark problems from Proben1 [133] and UCI [134] data sets are considered. The problems are selected so as to ensure a good coverage of classification and regression problems with binary and continuous variables and different numbers of input and output attributes. Table 4.1 presents details of the data sets.

Each data set is partitioned into three subsets: a training set, a validation set, and a test set. The training data is used to train and optimize the network architecture, the validation set is used for the stopping criterion, and the test set is used for the evaluation of the generalization performance. The sizes of the training, validation and test sets in all cases are 50%, 25%, and 25%, respectively.

In all experiments, a feed-forward network is employed with one hidden layer. The activation function between the input and hidden layers is the standard tangent sigmoid function; the output neurons have a linear activation function.

Table 4.1: Data sets used for classification and function approximation benchmarking. A partitioning into training, validation, and test set is also given for each data set.

Data set	Task type	Input	Output	Training	Validation	Test
Cancer	Classification	9	2	350	175	174
Card	Classification	51	2	345	173	172
Diabetes	Classification	8	2	384	192	192
Building1	Approximation	14	3	2104	1052	1052
Building2	Approximation	14	3	2104	1052	1052
Flare1	Approximation	24	3	533	267	266
Flare2	Approximation	24	3	533	267	266
Hearta	Approximation	35	1	152	76	75

Table 4.2: Parameters for various training algorithms.

Algorithms	Training parameters
SNN	• the number for updated atoms is set to $k = 3$; • the maximal iteration for dictionary learning is $T = 10$.
RPROP	• the maximum training iteration is 300; • the minimum performance gradient is 10^{-6}; • the learning rate is 0.01.
BR	• the maximum training iteration is 300; • the Marquardt adjustment rate is 0.005.
SCA	• the number of hidden neurons from the initial network is set to 1; • one extra hidden neuron will be added iteratively.
EO	• the population size is 100; • the crossover and mutation probability is 0.7 and 0.32, respectively.

Initial network parameters are chosen randomly in the range [-0.1 0.1] using the uniform distribution. Four conventional training algorithms are implemented for comparison with SNN: RPROP [117], Bayesian regularization algorithm (BR) [160], standard constructive approach (SCA) [162] and evolutionary optimization-based (EO) methods [163]. Their parameters are presented in Table. 4.2. To make the comparison fair, the network training is terminated if the validation error increases for three successive times or the maximum training iteration is reached.

The network performance is evaluated by processing the data sets for 30 independent runs. The classification error is used to evaluate performance for the classification problems, while the normalized root-mean-square error (NRMSE) is used for function approximation problems (see Eq. (3.16)). As for the network structure, we compare the number of remaining hidden neurons.

4.4.2 Initial network size

In this subsection, we analyze the robustness of the SNN method to different network sizes. A relatively large network is more prone to converging to a local minimum than a smaller network. However the large network also takes longer training time. As a result, we evaluate the proposed algorithm based on various initial network sizes. The number of initial hidden neurons (N) is set to 32, 64, and 128.

Tables 4.3 and 4.4 show the average NRMSE and classification error rate of the proposed training algorithm based on different initial network sizes. These results indicate that larger network sizes give slightly better generalization performance. For instance, the average NRMSE on the test sets is 0.515, 0.483 and 0.470 for initial network sizes 32, 64 and 128, respectively. Meanwhile, The best regression performance on the test set is always obtained with $N = 128$.

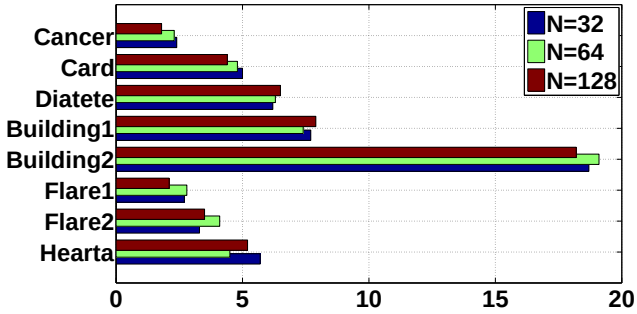
The resulting average network sizes are presented in Fig. 4.1 (a), and the average training time is shown in Fig. 4.1 (b). In terms of the number of remaining hidden neurons, the SNN method generates similar network size, independent of the initial network size. For instance, the proposed algorithm selects on average 6.46, 6.35, and 6.40 neurons for initial network sizes 32, 64 and 128, respectively. On the other hand, slightly longer training time is required for larger initial network sizes. When $N = 128$, the SNN method requires 10.1% extra time on average to complete training, compared to $N = 32$. Based on these simulation results, we can conclude that the larger initial network size leads to longer training time and better generalization capability; however, final network size is insensitive to the initial network size. This is because the SNN algorithm selects hidden neurons instead of eliminating the redundant neurons. As a result, the final network structure depends on the number of hidden neurons to be selected instead of the number of hidden neurons to be removed.

Table 4.3: Regression NRMSE obtained from the SNN algorithm with initial structures of different sizes.

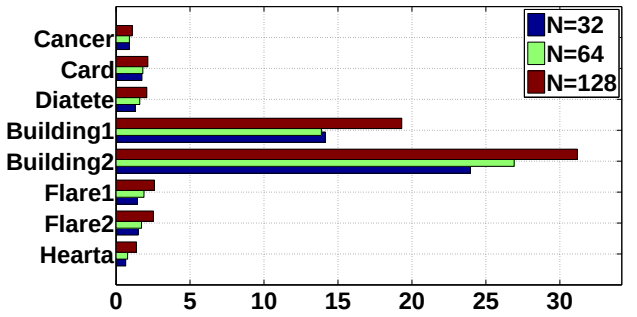
Data set	Training error (NRMSE)		
	$N = 32$	$N = 64$	$N = 128$
Building1	**0.061±0.012**	0.067±0.010	0.062±0.012
Building2	**0.133±0.025**	0.134±0.011	0.134±0.021
Flare1	0.889±0.046	0.885±0.057	**0.875±0.063**
Flare2	1.068±0.124	0.983±0.106	**1.005±0.122**
Hearta	**0.461±0.106**	0.494±0.092	0.479±0.023
Average	0.522±0.063	0.513±0.055	**0.511±0.048**
Data set	Test error (NRMSE)		
	$N = 32$	$N = 64$	$N = 128$
Building1	0.192±0.021	0.194±0.022	**0.168±0.022**
Building2	0.134±0.022	0.134±0.011	**0.123±0.011**
Flare1	0.887±0.105	0.786±0.143	**0.784±0.103**
Flare2	0.964±0.108	0.905±0.122	**0.875±0.102**
Hearta	0.400±0.074	0.399±0.062	**0.399±0.012**
Average	0.515±0.066	0.483±0.072	**0.470±0.050**

Table 4.4: Classification error rate obtained from the SNN algorithm as a function of initial network size.

Data set	Training error (%)		
	$N = 32$	$N = 64$	$N = 128$
Cancer	4.15±1.25	4.02±1.21	**3.94±2.33**
Card	12.42±3.43	**12.13±2.13**	12.96±4.25
Diabetes	21.23±3.84	22.61±3.62	**19.61±3.21**
Average	12.60±2.84	12.92±2.32	**12.17±3.26**
Data set	Test error (%)		
	$N = 32$	$N = 64$	$N = 128$
Cancer	**0.62±0.23**	1.05±0.46	0.72±0.58
Card	12.44±2.15	**12.12±1.29**	12.99±2.71
Diabetes	21.67±3.22	21.52±3.14	**21.22±3.32**
Average	11.58±1.87	**11.56±1.63**	11.64±2.20



(a) Number of remaining hidden neurons



(b) Training time (s)

Figure 4.1: Summary of resulting average network size and training time using the SNN algorithm with initial networks of different sizes.

4.4.3 Dictionary learning

In Chapter 3, a pruning-like algorithm (called SRP) was proposed which selects the most important hidden neurons iteratively. The major difference between SRP and SNN is that SNN adopts a dictionary learning strategy. To better understand the effect of dictionary learning on the performance of the trained network, we compare SNN with the SRP algorithm using the data sets from Table 4.1. The implementation of the SRP algorithm is given in Chapter 3. The network is initialized with 128 hidden neurons in both cases.

Tables 4.5 and 4.6 present the regression NRMSE and classification error rate for the SRP and SNN algorithms, respectively. In all cases, the test error of SNN is lower than that of SRP, which shows a consistent improvement in performance. Overall, the average classification accuracy of SNN is improved by 2.05% (on the training sets) and 1.64% (on the test sets) compared to the performance of the SRP

method. Again, the performance of sparse representation depends heavily on the employed dictionary. When the dictionary is optimized, a sparser and more accurate solution is obtained. Similarly, when the dictionary learning algorithm is applied in SNN to optimize network weights, it leads to a more compact network structure.

Table 4.5: Summary of the regression NRMSE of SNN against SRP for the approximation problems.

Data set	Training error (NRMSE)	
	SRP	SNN
Building1	0.104±0.038	**0.062±0.012**
Building2	0.151±0.045	**0.134±0.021**
Flare1	0.917±0.105	**0.875±0.063**
Flare2	**1.002±0.138**	1.005±0.122
Hearta	**0.478±0.026**	0.479±0.023
Average	0.530±0.070	**0.511±0.048**
Data set	Test error (NRMSE)	
	SRP	SNN
Building1	0.244±0.028	**0.168±0.022**
Building2	0.162±0.022	**0.123±0.011**
Flare1	0.941±0.135	**0.784±0.103**
Flare2	0.996±0.198	**0.875±0.102**
Hearta	0.428±0.021	**0.369±0.012**
Average	0.554±0.081	**0.464±0.050**

Next we compare SNN with SRP in terms of the resulting structure. As shown in Fig. 4.2, for seven out of eight data sets, SNN utilizes fewer hidden neurons than SRP. In SRP the weight matrix is fixed during the structural optimization, whereas, this weight matrix (dictionary) is optimized in SNN to maximize the diversity of the outputs from all hidden neurons, thereby improving performance of the sparsity-based supervised learning algorithm.

4.4.4 Comparison with conventional training methods

In this subsection, the performance of the proposed algorithm is compared with those of existing methods that train and optimize the network structure simul-

Table 4.6: Summary of the classification error rate of SNN against SRP.

Data set	Training error (%)	
	SRP	SNN
Cancer	**3.72±1.40**	3.94±2.33
Card	18.61±4.14	**12.96±4.25**
Diabetes	20.33±3.81	**19.61±3.21**
Average	14.22±3.12	**12.17±3.26**
Data set	Test error (%)	
	SRP	SNN
Cancer	1.35±0.37	**0.72±0.58**
Card	15.23±2.38	**12.99±2.71**
Diabetes	23.27±3.23	**21.22±3.32**
Average	13.28±1.99	**11.64±2.20**

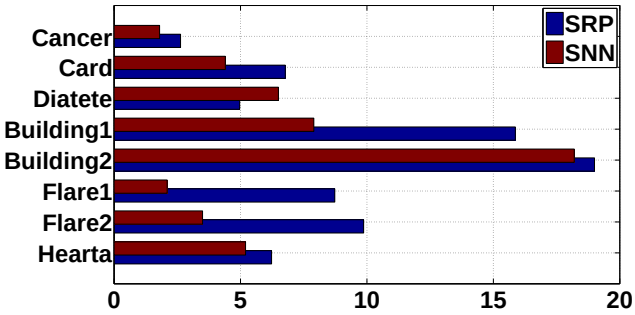


Figure 4.2: Average resulting network structures obtained from the SRP and SNN method.

taneously. Three conventional training algorithms are selected to compare with SNN: Bayesian regularization (BR) [160], standard constructive approach (SCA) [162] and evolutionary optimization-based (EO) methods [163]. The parameters used for these algorithms are given in Table 4.2. These approaches train and minimize the network architecture simultaneously. For BR, EO, and SNN, the networks are initialized with 128 hidden neurons; SCA starts from a network with only one hidden neuron. We also compare the training and test performance of SNN with those of traditional training algorithms RPROP and LM, which use a fixed network structure.

The average NRMSE over 30 runs from regression benchmarking problems are summarized in Table 4.7. Clearly the proposed approach achieves better

generalization ability in comparison to other methods. Although the SNN method obtains slightly worse training accuracy, it achieves the best performance on the test data sets for three out of five problems, namely, the Building1, Flare1, and Flare2. On the remaining two data sets, the SNN achieves the second smallest test error; however, the difference of 0.001 in NRMSE is not significant.

Table 4.7: Regression NRMSE for the proposed SNN method and traditional training algorithms for function approximation.

Data set	Training error (NRMSE)			
	BR	SCA	EO	SNN
Building1	0.073±0.012	0.084±0.031	0.079±0.021	**0.062±0.012**
Building2	**0.098±0.024**	0.154±0.017	0.165±0.057	0.134±0.021
Flare1	**0.875±0.052**	0.875±0.053	0.886±0.063	0.875±0.063
Flare2	**1.002±0.132**	1.027±0.180	1.123±0.200	1.005±0.122
Hearta	0.336±0.022	**0.328±0.028**	0.338±0.025	0.479±0.023
Average	**0.477±0.048**	0.494±0.309	0.518±0.073	0.511±0.048
Data set	Test error (NRMSE)			
	BR	SCA	EO	SNN
Building1	0.314±0.025	0.297±0.019	0.307±0.020	**0.168±0.022**
Building2	**0.122±0.017**	0.156±0.050	0.161±0.045	0.123±0.011
Flare1	0.852±0.120	0.871±0.108	0.869±0.123	**0.784±0.103**
Flare2	0.912±0.120	0.881±0.106	0.905±0.186	**0.875±0.102**
Hearta	**0.368±0.016**	0.400±0.011	0.374±0.031	0.369±0.012
Average	0.514±0.060	0.521±0.059	0.523±0.081	**0.464±0.050**

Table 4.8 reports the average classification error from various training algorithms over 30 runs. We find that the proposed algorithm generates the best results on the test data. The average test accuracy from the BR, SCA, EO, and SNN method generates 12.42%, 12.87%, 12.32%, and 11.64% respectively. Table 4.9 presents the size of final network obtained from BR, SCA, EO and SNN. The SCA obtains the most compact architecture, while BR, EO and SNN requires extra 122.5, 25.6 and 2.0 hidden neurons on average. That is, the proposed algorithm achieves the second smallest network structure. However, the proposed SNN method achieves much better test accuracy than SCA. By contrast, both BR and EO perform worse than SCA in terms of the average generalization ability and

more hidden neurons.

Table 4.8: Summary of the classification error (%) of SNN against existing algorithms. The BR algorithm fails to solve the "Card" problem because it runs out of the memory (labeled with the × symbol).

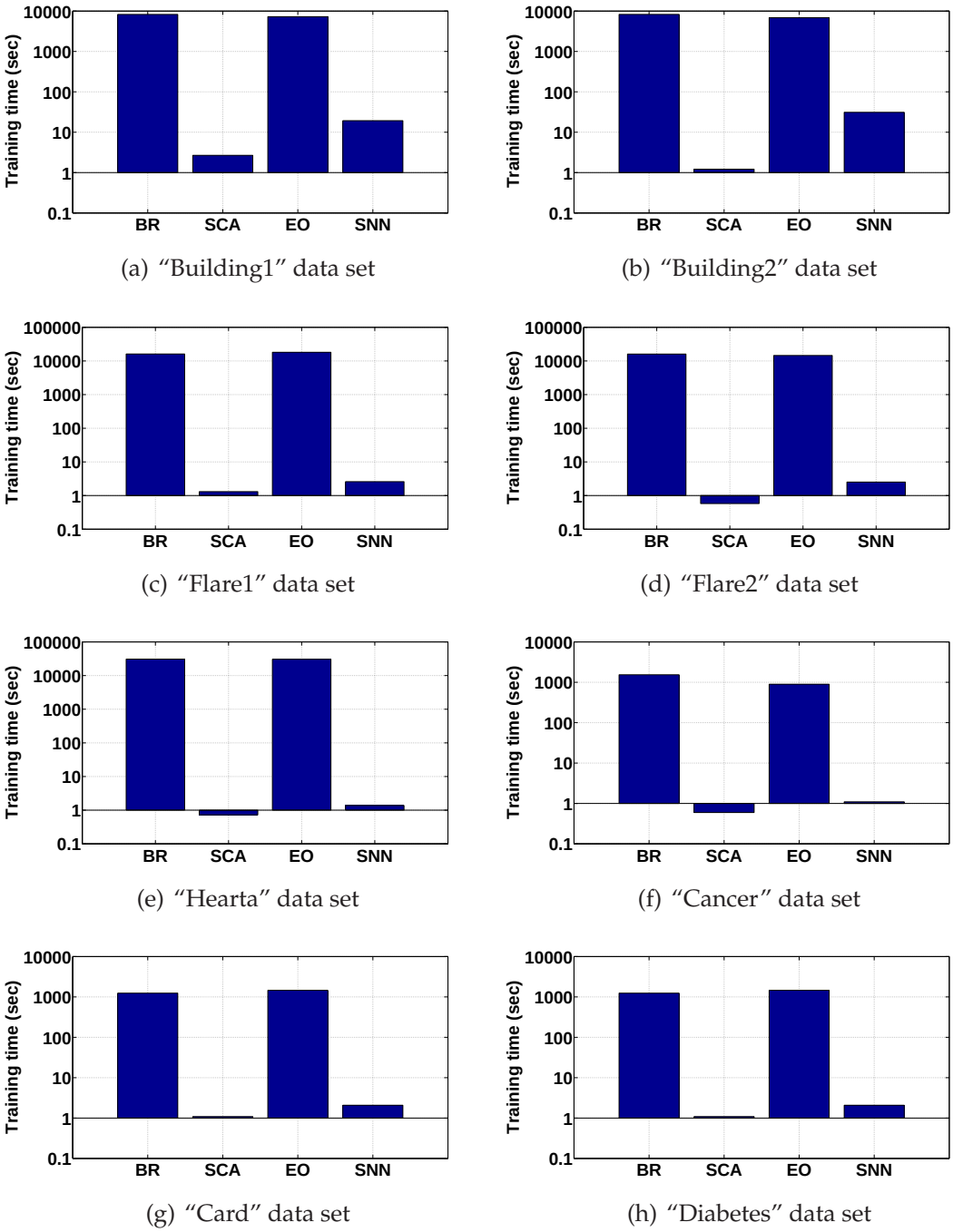
Algorithms		Cancer	Card	Diabetes	Average
BR	Training	5.15±2.85	12.35±3.23	**19.27±4.29**	12.25±3.46
	Test	2.14±1.57	13.35±1.68	21.76±2.38	12.42±1.88
SCA	Training	3.95±3.05	**11.02±5.86**	23.08±4.27	12.68±4.39
	Test	2.32±1.86	13.39±2.68	22.90±4.10	12.87±2.88
EO	Training	4.95±2.03	13.47±3.58	25.86±4.23	14.76±3.28
	Test	2.13±0.93	**12.58±1.85**	22.25±4.68	12.32±2.49
SNN	Training	**3.94±2.33**	12.96±4.25	19.61±3.21	**12.17±3.26**
	Test	**0.72±0.58**	12.99±2.71	**21.22±3.32**	**11.64±2.28**

Table 4.9: Remaining hidden neurons for the conventional training and proposed SNN methods.

Data sets	BR	SCA	EO	SNN
Building1	126.0±2.0	9.0±1.3	23.0±2.3	**7.9±1.0**
Building2	126.0±1.9	**6.5±1.0**	24.5±3.0	18.2±0.3
Flare1	126.8.0±1.2	4.6±2.3	22.6±4.9	**2.1±0.7**
Flare2	127.0±1.0	**2.0±0.9**	24.8±6.5	3.5±1.0
Hearta	127.3±0.7	**2.2±0.7**	25.0±5.9	5.2±1.5
Cancer	126.0±2.0	2.2±0.5	32.2±4.5	**1.8±0.3**
Card	127.2±0.8	**1.9±0.3**	43.9±7.8	4.4±1.3
Diabetes	127.6±0.4	**4.6±1.0**	42.6±6.7	6.5±2.1
Average	126.7±1.3	**4.2±1.0**	29.8±5.2	6.2±1.0

We now consider the computational time for various training algorithms, see Fig. 4.3. On average, SNN spends 21.2 seconds to train the eight problems, which is significantly better than BR and EO (14920.4 seconds and 14467.5 seconds, respectively). The BR and EO-based algorithms require the longest time to converge because they operate on single weights. Therefore, more computational time is spent on pruning insignificant weights to remove the hidden neurons. By contrast, SNN selects the most significant hidden neurons from the trained network. This method can be regarded as a forward selection instead of backward elim-

ination, thereby saving computation time. Although the SCA method spends less time on training the networks, the proposed SNN algorithm achieves better generalization ability.



(a) "Building1" data set (b) "Building2" data set

(c) "Flare1" data set (d) "Flare2" data set

(e) "Hearta" data set (f) "Cancer" data set

(g) "Card" data set (h) "Diabetes" data set

Figure 4.3: Training time obtained from different algorithms for the regression and classification problems.

Comparison with RPROP and LM. The RPROP and LM algorithms are commonly used to train a fixed network structure. In the following we compare the

performance of SNN to that of a network of the same size trained with RPROP and LM. In other words, the final network size obtained with SNN is used to initialize a new network, which is then trained with RPROP and LM.

Tables 4.10 and 4.11 show the average performances of SNN algorithm against conventional RPROP and LM methods, respectively, over 30 independent runs. On average, the SNN method achieves lower training accuracy compared to the RPROP and LM methods; however, it has a better generalization accuracy on all but two data sets. One reason is that the RPROP and LM methods overfit the training data, and thus their test accuracy are reduced on the test set. Overall, the results show that the proposed algorithm not only achieves better generalization ability compared to RPROP and LM training methods, but also optimizes the network structure simultaneously.

Table 4.10: Summary of the regression NRMSE of SNN against the RPROP and LM methods using the same network size.

Data set	Training error (NRMSE)		
	RPROP	LM	SNN
Building1	0.203±0.057	0.078±0.025	**0.062±0.012**
Building2	0.134±0.066	**0.101±0.053**	0.134±0.021
Flare1	0.880±0.561	**0.867±0.038**	0.875±0.063
Flare2	0.889±0.441	**0.847±0.088**	1.005±0.122
Hearta	**0.416±0.105**	0.445±0.025	0.479±0.023
Average	0.504±0.246	**0.468±0.046**	0.511±0.048
Data set	Test error (NRMSE)		
	RPROP	LM	SNN
Building1	0.273±0.107	0.177±0.082	**0.168±0.022**
Building2	0.138±0.078	**0.113±0.043**	0.123±0.011
Flare1	0.907±0.485	0.908±0.145	**0.784±0.103**
Flare2	1.035±0.991	0.971±0.138	**0.875±0.102**
Hearta	0.408±0.092	0.409±0.091	**0.369±0.012**
Average	0.552±0.351	0.516±0.100	**0.464±0.050**

Next, we briefly explain the reasons behind the superiority of the SNN algorithm. At first, SNN employs the architecture optimization (using sparse representation) to select significant hidden neurons that minimize the residual output

Table 4.11: Summary of classification error rate of SNN against conventional RPROP and LM methods.

Data set	Training error (%)		
	RPROP	LM	SNN
Cancer	**3.27±1.43**	3.31±0.98	3.94±2.33
Card	11.48±3.45	**10.60±2.36**	12.96±4.25
Diabetes	22.13±5.32	**19.27±4.31**	19.61±3.21
Average	12.29±3.40	**11.06±2.55**	12.17±3.26
Data set	Test error (%)		
	RPROP	LM	SNN
Cancer	**0.69±0.31**	0.91±0.42	0.72±0.58
Card	13.89±2.58	15.20±3.25	**12.99±2.71**
Diabetes	23.98±5.81	22.22±7.61	**21.22±3.32**
Average	12.85±2.90	12.78±3.76	**11.64±2.20**

error. The neuron selection can be regarded as a forward selection compared to the backward elimination in the classical pruning methods. The proposed selection strategy therefore leads to a more compact structure and affordable computation. Secondly, the dictionary learning method is further introduced to optimize the network weights, so that the performance of the sparse representation is improved. Thus, the proposed SNN method obtains a better or competitive performance compared to existing training methods.

4.5 Conclusion

We have presented a novel training algorithm for neural networks to train the network and optimize its structure simultaneously. The method is characterized formally by the concept of sparse representation, termed sparse neural network (SNN).

There are two major contributions in this chapter. First, the network structural optimization is formulated as a sparse representation problem. Only the significant neurons are selected to achieve a compact network structure. Second, the dictionary learning-based algorithm is employed to improve the performance

of the sparse representation. The experimental results obtained from function approximation and classification problems show that the proposed training algorithm is superior to conventional training methods in terms of generalization ability and computational cost.

Chapter 5

A Training Algorithm for Sparse LS-SVM

Chapter contents

5.1 Introduction

Support Vector Machines (SVMs) have received a great deal of attention since their introduction in 1995 [5, 164–168]. Along with kernel methods, SVMs are one of the most fascinating recent developments in classifier design. However, the computational complexity of SVMs scales quadratically with the number of training patterns. A variant of SVMs, known as *Least Squares Support Vector Machines* (LS-SVMs), were introduced in [7] to address the computational issue of SVMs. One advantage of LS-SVMs over SVMs is that the sensitive loss function, used with SVMs, is replaced by a set of equality constraints; thereby, the quadratic programming problem of SVMs is reduced to solving a system of linear equations [9]. Another advantage is that LS-SVMs involve tuning fewer parameters. The empirical studies presented in [8, 9] have shown that LS-SVMs are comparable to standard SVMs in terms of generalization performance. LS-SVMs are now considered as essential machine learning tools for regression and classification tasks.

The major drawback of LS-SVMs is the solution sparsity, in which a great number of support vectors (SVs) are required in the model. The support vectors are typically a small portion of training samples, used to construct the decision function. However, too many SVs will influence the training accuracy as well as the generalization ability, not to mention computation cost [8, 169, 170].

In this chapter, we present a sparse training algorithm for the LS-SVM model, termed *Sparse Least Squares Support Vector Machine* (SLS-SVM). The LS-SVM model is first reformulated as a sparse representation problem. The training process is then accomplished by iteratively finding important support vectors that minimize the residual error; a measurement matrix is also introduced to reduce the computational cost. The main advantage is that the proposed algorithm performs model training and SV selection simultaneously. By contrast, existing sparse training methods require full training of the LS-SVM model before finding im-

portant SVs, which leads to costly computation [8, 169, 170]. To the best of our knowledge, this is the first attempt to optimize the LS-SVM training based on sparse representation.

The remainder of the chapter is organized as follows. Section 5.2 gives a brief introduction about SVM and LS-SVM training process. Section 5.3 presents the proposed sparsity-based training approach. Section 5.4 compares the proposed algorithm with conventional training methods using function approximation and classification problems. Section 5.5 presents concluding remarks.

5.2 SVMs and LS-SVMs

In this section, we first introduce the formulation of the SVM-based optimization problem. Then we briefly review some basic work on LS-SVMs, as the extension to SVMs.

5.2.1 Support Vector Machines

As a supervised learning approach, the SVM algorithm has been demonstrated to perform well in various practical applications [5, 6]. The structure of an SVM is similar to that of a neural network. However, the procedure of obtaining the SVM structure is different from that of neural networks. On one hand, in neural networks, determining an appropriate structure, such as the number of hidden neurons, involves heuristic techniques and expensive cross validation. On the other hand, the process of constructing an SVM evolves from analytical methods. Hidden units in SVMs (known as the support vectors) are automatically determined from training.

The SVM model is fundamentally formulated to address two-class classification problems. The decision boundary is constructed by finding a hyperplane that achieves the maximum separation between two classes. Suppose that we have a training set consisting of N patterns $\{\mathbf{x}_i, z_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the i-th input

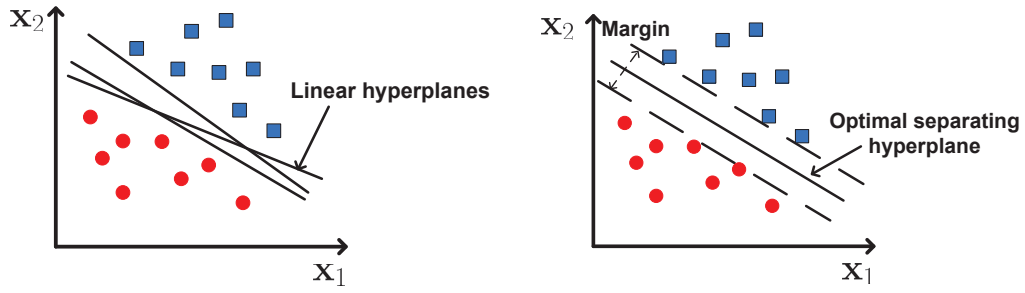
pattern and $z_i \in \{1, -1\}$ is the class label. Assume that the data is linearly separable in the input space, the decision function can be written as

$$\begin{cases} \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 & \text{for } z_i = 1, \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 & \text{for } z_i = -1, \end{cases} \tag{5.1}$$

or

$$z_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \tag{5.2}$$

where $\mathbf{w}$ is the wight vector, b is a bias term, and $\langle \mathbf{w}, \mathbf{x} \rangle$ is the dot product of the vectors $\mathbf{w}$ and $\mathbf{x}$.



(a) Hyperplanes (b) Optimal hyperplane (solid line)

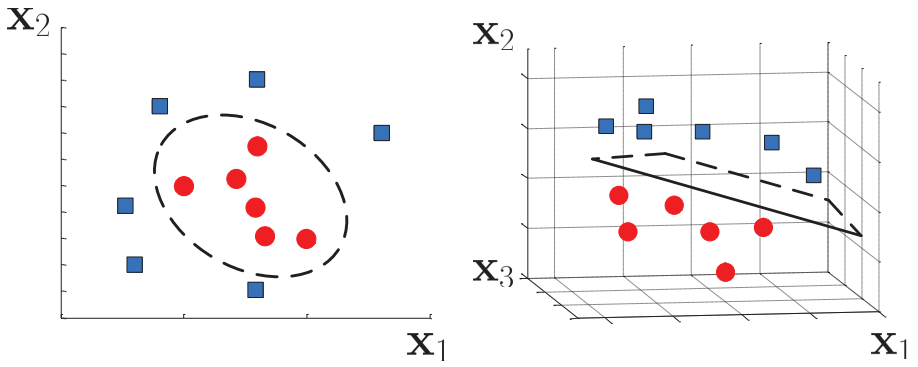
Figure 5.1: Separating hyperplanes for two-class problem in SVM. (a) the data can be separated by many linear hyperplanes. (b) optimal separating hyperplane achieves the maximum separation.

Note that there exist more than one hyperplanes that can separate the two classes (Fig. 5.1(a)); however, only one of them, termed *optimal separating hyperplane*, can maximize the margins between the two classes, see Fig. 5.1(b). Consequently, the SVM training is to find $\mathbf{w}$ and b that achieve the maximum separation:

$$\min \; J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad z_i(\langle \mathbf{w}, \mathbf{x} \rangle + b) \geq 1 \text{ for } i = 1, \ldots, N. \tag{5.3}$$

The parameter optimization in (5.3) is built on the assumption of linear separability of the training data. However, if the problem is not linearly separated in the input space, an SVM classifier with a linear decision boundary will have a poor generalization ability. To improve the classification accuracy, the data samples

are usually projected from the input space to a higher-dimensional space via a mapping function $\varphi(\cdot)$ (see Fig. 5.2). As a result, a non-linear decision boundary can be constructed for classification.



(a) Input space (b) Projected space

Figure 5.2: According to the mapping function $\varphi(\cdot)$, the original data is projected from the input space (a) to a higher-dimensional space (b).

SVMs have several important properties including the ability to model complex nonlinear decision boundaries in a wide variety of applications. However, one major drawback of SVMs is the intensive computational complexity when solving a quadratic program with inequality constraints, see Eq. (5.3). This limits SVMs from addressing large-scale problems. The next subsection reviews briefly LS-SVMs, which aim to address the computational issue of SVMs.

5.2.2 Least Squares Support Vector Machines

Least Squares Support Vector Machines (LS-SVMs) have been developed to achieve more affordable and faster training. In the LS-SVM model, the inequality constraints in (5.3) are replaced with equality constraints, and the unknown parameters are then obtained by solving the following problem:

$$\min \ \mathcal{J}(\mathbf{w}, b, \mathbf{e}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^{N} e_i^2,$$

$$\text{subject to} \quad z_i = \mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) + b + e_i, \quad i = 1, 2, ..., N,$$

where **e** is the error vector, and γ is a regularization parameter. This problem can be solved using the Lagrange multiplier method:

$$\min \ \mathcal{L}(\mathbf{w}, b, \mathbf{e}, \boldsymbol{\alpha}) = \mathcal{J}(\mathbf{w}, b, \mathbf{e}) + \sum_{i=1}^{N} \alpha_i [z_i - \mathbf{w}^{\mathrm{T}} \varphi(\mathbf{x}_i) - b - e_i], \tag{5.4}$$

where α_i $(i = 1, 2, ..., N)$ are the Lagrange multipliers, which may be positive or negative due to the equality constraints. According to the Karush-Kuhn-Tucker conditions, the minimization of Eq. (5.4) satisfies

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i), \\
&\frac{\partial \mathcal{L}}{\partial b} = 0 \longrightarrow \sum_{i=1}^{N} \alpha_i = 0, \\
&\frac{\partial \mathcal{L}}{\partial e_i} = 0 \longrightarrow \alpha_i = \gamma e_i, \quad i = 1, 2, ...N, \\
&\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \longrightarrow \mathbf{w}^T \varphi(\mathbf{x}_i) + b = z_i - e_i, \quad i = 1, 2, ...N.
\end{aligned} \tag{5.5}$$

Furthermore, the above conditions lead to a linear system of equations after eliminating **w** and **e**:

$$\left[\begin{array}{c|c} Q + \gamma^{-1} I_N & \mathbf{1}_N \\ \hline \mathbf{1}_N^T & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ b \end{array}\right] = \left[\begin{array}{c} \mathbf{z} \\ 0 \end{array}\right], \tag{5.6}$$

where Q is the kernel matrix with $Q_{ij} = \varphi(\mathbf{x}_i)^T \varphi\left(\mathbf{x}_j\right)$, the vector $\mathbf{1}_N$ is the N-dimensional vector whose elements are equal to 1, and I_N is the $N \times N$ identity matrix.

With the application of Mercer's theorem [171], there is no need to compute explicitly the mapping $\varphi(\cdot)$ as this can be done implicitly through the use of positive-definite kernel functions $\mathcal{K}(\cdot, \cdot)$, that is, $Q_{ij} = \mathcal{K}\left(\mathbf{x}_i, \mathbf{x}_j\right)$. The resulting

LS-SVM model for nonlinear function then becomes

$$z_i = \sum_{j=1}^{N} \alpha_j \mathcal{K}\left(\mathbf{x}_i, \mathbf{x}_j\right) + b + e_i, \quad i = 1, 2, ..., N. \tag{5.7}$$

The most used kernel functions are liner, polynomial and radial basis function (RBF) [172]. In this chapter the RBF function is employed which is expressed as

$$\mathcal{K}\left(\mathbf{x}, \mathbf{x}_i\right) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|_2^2}{\delta^2}\right), \tag{5.8}$$

where δ is a constant indicating the width of the RBF function.

To train the LS-SVM, the conjugate gradient (CG) algorithm was employed to solve the linear system (5.6) [119, 173, 174]. The disadvantage is that the computational complexity increases exponentially with the size of the linear system. The sequential minimization optimization (SMO) algorithm was proposed to speed up the calculation [6]. Apart from the computational cost on training, the LS-SVM method also requires a large number of support vectors (SVs), which may influence the training performance and the generalization capacity. Too many SVs result in poor generalization on the test data even if it can obtain high accuracy on the training data. Therefore, several optimization methods have been suggested to improve the sparseness of the LS-SVM model. Suykens *et al*. first proposed removing training samples that have the smallest absolute support values [8]. However, this method might eliminate training samples near the decision boundary, which has a negative influence on the training performance. An improved method was proposed in [169] where a reduced training set comprised of samples near the decision boundary is used to retrain the LS-SVM. In [170], support vectors were eliminated by minimizing the output error after few samples have been deleted. However, the method involves the inversion of a matrix that is often singular or near singular. Another sparse training method was proposed in [175], in which the support vectors are removed through regularization to accelerate the

training process. For more on sparse training algorithms, the interested reader is referred to the survey presented in [176].

5.3 Sparse learning algorithm for LS-SVM

In this section, we present a sparse learning algorithm for LS-SVM, termed *Sparse Least Squares Support Vector Machines (SLS-SVM)*, to address the solution sparsity and the computational complexity of LS-SVM. The main difference between SLS-SVM and existing techniques is that the proposed method minimizes the model structure while training the LS-SVM simultaneously. By contrast, most traditional methods need to train the model before finding the sparse support vectors. In Subsection 5.3.1, we present the sparse training model for LS-SVM. In Subsection 5.3.2, we introduce the measurement matrix to further reduce the computational cost.

5.3.1 Sparse LS-SVM training

The LS-SVM training algorithm aims to find the optimal parameters for the vector $[\boldsymbol{\alpha}^T, b]^T$ that satisfies Eq. (5.6). Note that setting a particular element of $\boldsymbol{\alpha}$ to zero is equivalent to removing the corresponding training sample or support vector. Therefore, the goal of finding a sparse LS-SVM model, within a given tolerance of accuracy, can be equated to minimizing the number of nonzero elements from the parameter vector $[\boldsymbol{\alpha}^T, b]^T$. For further discussion, we simplify Eq. (5.6) as follows. Let $\widehat{\mathbf{z}}$ be a composite vector containing the desired output:

$$\widehat{\mathbf{z}} = \begin{bmatrix} \mathbf{z} \\ 0 \end{bmatrix}. \tag{5.9}$$

The matrix Ψ is represented as

$$\Psi = \left[\begin{array}{c|c} Q + \gamma^{-1} I_N & \mathbf{1}_N \\ \hline \mathbf{1}_N^T & 0 \end{array}\right]. \tag{5.10}$$

The parameter vector $\mathbf{x}$ is represented as

$$\mathbf{x} = [\boldsymbol{\alpha}^{\mathrm{T}}, b]^{\mathrm{T}}. \tag{5.11}$$

Then the training problem for LS-SVM can be cast as follows:

$$\min S(\mathbf{x}) \quad \text{subject to} \quad \left\|\widehat{\mathbf{z}} - \Psi\mathbf{x}\right\|_2 < \epsilon, \tag{5.12}$$

where $S(\mathbf{x})$ denotes a sparsity measure, and ϵ bounds the amount of additional noise. One strategy for solving Eq. (5.12) is to minimize the l_1-norm of $\mathbf{x}$, i.e., $S(\mathbf{x}) = \|\mathbf{x}\|_1$. Now the sparse LS-SVM learning problem is converted to solving a single measurement vector (SMV) model. Training the LS-SVM model is then equivalent to finding a sparse representation for the parameter vector $\mathbf{x}$. Furthermore, only the support vectors associated with nonzero parameters are selected; the remaining support vectors do not affect the model in Eq. (5.6).

Any algorithm that can solve the SMV model can be employed for the SLS-SVM learning. Herein, we use the orthogonal matching pursuit (OMP) algorithm, which has been proven to be effective for solving the SMV model [50]. At each iteration, significant support vectors from the matrix Ψ will be selected according to the similarity between the support vectors and the residual signal.

By implementing the OMP algorithm, the convergence of the SLS-SVM approach is influenced by the number of training samples and orthogonality of training data. Suppose that we have K support vectors with $N > cK\log(N/\delta)$, where N is the number of training samples, the parameter $\delta \in (0, 0.36)$, and c is a

positive constant. According to Theorem 5 from Chapter 2, after K iterations, SLS-SVM is guaranteed to find the sparsest solution with a probability exceeding $1-\delta$. Note that the above claim is built on the success of pursuit algorithms, which depends on the number of training samples and orthogonality of training data, thus the convergence is not always guaranteed. However, the OMP method, which leads to a decreasing residual error, is known to perform reasonably well [50].

Next, we compare the computational complexity of the proposed algorithm with other sparse training methods [8, 169, 170, 175], shown in Table 5.1. For the proposed algorithm, at each iteration, the OMP algorithm performs a Cholesky factorization, which requires $O(K^2)$ operations, where K is the number of support vectors to be selected. Furthermore, SLS-SVM selects one support vector iteratively. The selection procedure can be regarded as a forward selection of K support vectors from the entire N training samples. Therefore, K loops are required to select K support vectors. The computational complexity for the SLS-SVM algorithm is $O(K^3)$.

Table 5.1: Computational complexity of various training algorithms for LS-SVM, where N is the number of training samples, K is the number of selected support vectors, and P is the number of support vectors to be removed.

	SLS-SVM	[8, 169, 170]	[175]
Loops	K	$N-K$	$N-K$
Complexity	$O(K^3)$	$O(N \times P^2)$	$O(N \times K^2)$

By contrast, other training algorithms train an LS-SVM using all N training examples [8, 169, 170]. Then, they apply pruning operation to selectively delete training examples until only K support vectors remain. This process is computationally expensive as each deletion requires $O(P^2)$ operations, where P ($N > P >> K$) is the number of training examples being compared. In addition, $N-K$ loops are required to achieve the sparse model. Therefore, the computational complexity is $O(N \times P^2)$.

The method in [175] trains LS-SVM using K training examples before adding another training example and updating the parameters. Although only $O(K^2)$

operations are required at each iteration, the method in [175] determines the important support vectors by removing $N - K$ samples. That is, $N - K$ loops are required to achieve the final sparse model. The total computational complexity then becomes $O(N \times K^2)$. Therefore, compared to conventional methods, faster convergence is expected from the proposed sparsity-based algorithm.

5.3.2 Training using compressed sensing

The proposed algorithm searches for a sparse representation for the LS-SVM model. At each iteration, the significant support vectors contributing the most to the desired output are selected. The problem with the minimization of Eq. (5.12) is that the entire training set with all N samples is considered, which leads to costly computation, especially with large training sets.

Compressed sensing (CS) has received considerable attention recently for its ability to perform data acquisition and compression simultaneously [66]. It can be used to reconstruct a sparse approximation of a compressible signal from far fewer measurements than required by the sampling theorem. This has the advantage of reducing the amount of the data acquisition and computational time.

Fewer measurements can be constructed by simply choosing a random *measurement matrix* in most cases, and the recovery of the sparse solution can still be achieved with high probability [41, 67]. In particular, if the measurement matrix contains only one nonzero element in each row and each column, then the measurement process is equivalent to selecting a subset of training data to perform the sparse representation. If the non-zero element in each column is equal to 1, this special measurement matrix is known as the *selection matrix*.

We now formulate the sparse LS-SVM training using fewer measurements. Given a linear measurement matrix $\Phi \in \mathbb{R}^{M\times(N+1)}$ ($M << N$), the measurement vector is given by

$$\mathbf{y} = \Phi\widehat{\mathbf{z}}, \tag{5.13}$$

where $\widehat{\mathbf{z}}$ is the augmented vector in Eq. (5.9). Given the matrix Ψ in Eq. (5.10), the optimization problem in (5.12) can be expressed as follows:

$$\min \ \|\mathbf{x}\|_1 \quad \text{subject to} \qquad \left\|\mathbf{y} - \mathcal{D}\mathbf{x}\right\|_2 < \epsilon, \tag{5.14}$$

where $\mathcal{D} = \Phi\Psi$ is the dictionary, and ϵ is a bound on the amount of noise. We should note that the dictionary $\mathcal{D}$ is of size $M \times (N+1)$, where $M << N$; this is much smaller than Ψ, which has size $(N+1)\times(N+1)$. In particular, if the selection matrix with $M = N + 1$ is employed, then all the original measurements will be selected. In this case, solving the CS problem in Eq. (5.14) is converted to that of Eq. (5.12).

5.4 Experimental results

This section presents the experimental results and comparisons of the proposed algorithm with conventional LS-SVM training techniques. The employed classification data sets and the evaluation of the training algroithm are presented in Subsection 5.4.1. The performance of the SLS-SVM is then evaluated in Subsection 5.4.2. The comparison results with conventional training algorithms are presented in Subsection 5.4.3.

5.4.1 Experimental methods

Six classification problems are chosen from UCI repository [134] for experimental evaluation (see Table 5.2). Each data set is partitioned into two subsets: a training set and a test set. The training set is used to train and optimize the LS-SVM model. The test set is used for the evaluation of the generalization performance of the model. The sizes of the training and test sets are 67% and 33%, respectively.

The performance of the training algorithms is evaluated using the classification accuracy. The resulting model structure is measured by the number of remaining

Table 5.2: Employed data set. A partitioning into training and test set is also given for each data set.

Data Set	Input	Training	Test
Cancer	9	466	233
Card	51	460	230
Diabetes	8	512	256
Liver	7	230	115
SPECT Heart	22	180	87
Chess	36	2137	915

support vectors, i.e.,

$$k = K/\left(N+1\right) \times 100\%, \tag{5.15}$$

where K is the number of selected support vectors, and N is the number of training samples. Thus, a larger value for k means that more support vectors are selected during training. For the proposed SLS-SVM algorithm, the percentage of selected measurements is denoted

$$m = M/\left(N+1\right) \times 100\%, \tag{5.16}$$

where M is the number of selected measurements.

5.4.2 Performance analysis of SLS-SVM

In this subsection, we first investigate the effect of the remaining support vectors on the generalization ability. Second, we test the performance of the SLS-SVM method with two types of measurement matrices and different numbers of selected measurements.

5.4.2.1 Support vectors

The number of support vectors is critical to the performance of the SLS-SVM algorithm. For instance, a large model with more support vectors may result in a fast convergence to a local minimum, but exhibits poor generalization performance because of overfitting. Meanwhile, a too small model may not be able to

find the proper fit to the data. Therefore, in this subsection we analyze how the number of support vectors impacts the performance of SLS-SVM. The remaining model structure is set to k = 20%, 30%, 50%, and 100%, and the percentage of measurements is fixed at m =100% with the selection matrix. Note that using the selection matrix with m =100% is equivalent to solving the sparse representation for Eq. (5.12).

Table 5.3 shows the performance of the proposed algorithm with respect to the remaining model structure. As can be observed, the proposed method achieves a better classification accuracy on the training sets with increasing number of support vectors. The performance on the training sets is 86.92%, 91.34%, 95.50%, and 99.83% for k = 20%, 30%, 50% and 100%, respectively. However, a larger k leads to a longer training time. In other words, when more support vectors are selected, higher computational cost is required to train the SLS-SVM algorithm.

On the other hand, the generalization ability of the trained model is not guaranteed to be improved with more support vectors. For instance, the average classification rate on the test set is 80.94%, 83.30%, 83.05%, and 82.79% for k = 20%, 30%, 50% and 100%, respectively. One reason is that the model with more support vectors overfits the training data, and thus its accuracy is reduced on the test sets. Overall, the results show that selecting fewer support vectors leads to better generalization performance and requires less computation cost.

5.4.2.2 Measurement matrix

A measurement matrix is employed in the proposed SLS-SVM algorithm, which is used to reduce the computational complexity. In this subsection, we analyze the robustness of the proposed algorithm to two types of the measurement matrices: a random measurement matrix and a selection matrix, with each row and each column in the latter containing only one non-zero element (equals to 1). Meanwhile we set the percentage of selected measurements to m = 20%, 40%, 70% and 100%. To make the comparison fair, the remaining model is fixed at k =30%.

Table 5.3: Summary of the classification accuracy (%) and training time (sec) from the proposed SLS-SVM method. Various numbers of selected support vectors are considered.

Data sets	k	20%	30%	50%	100%
Cancer	Training	98.71	99.62	**100**	**100**
	Test	**98.71**	**98.71**	**98.71**	**98.71**
	Time	**1.04±0.23**	2.08±0.75	7.03±3.11	54.68±23.58
Card	Training	83.51	88.33	92.81	**100**
	Test	**87.43**	86.52	85.71	85.71
	Time	**1.10±1.07**	2.06±1.78	8.25±1.05	69.47±22.12
Diabetes	Training	86.13	96.09	**100**	**100**
	Test	67.97	**70.70**	68.75	67.19
	Time	**1.16±0.58**	2.72±0.89	10.54±3.25	85.05±12.47
Liver	Training	73.48	79.57	90.00	**100**
	Test	67.83	**68.96**	**68.96**	**68.96**
	Time	**0.18±0.01**	0.29±0.02	0.59±0.12	4.32±1.01
SPECT Heart	Training	86.11	88.33	92.22	**100**
	Test	**87.78**	85.56	86.67	86.67
	Time	**0.07±0.01**	0.14±0.05	0.35±0.09	2.57±0.73
Chess	Training	93.56	96.07	97.98	**98.96**
	Test	75.90	89.36	**89.51**	**89.51**
	Time	**57.23±23.75**	550.76±35.79	1635.25±50.14	72780.98±420.19
Average	Training	86.92	91.34	95.50	**99.83**
	Test	80.94	**83.30**	83.05	82.79
	Time	**10.13±4.28**	93.01±6.55	277.01.25±9.23	12166.18±80.17

Tables 5.4 and 5.5 show the performances of the SLS-SVM algorithms with the random measurement and selection matrices, respectively.

Several observations can be made from these results. First, the proposed SLS-SVM method achieves similar generalization performance, independent of the type of the measurement matrix. For instance, with respect to various sizes of measurement matrix 20%, 40%, 70%, and 100%, the proposed algorithm generates on average 81.15%, 84.40%, 84.03%, and 83.93% classification accuracy on the test sets with the random measurement matrix; if the selection matrix is employed, the classification accuracy on the test sets becomes 79.23%, 83.67%, 84.23%, and 83.30%, respectively. The average difference of 0.72% classification rate on the test sets of the six problems is not significant.

Second, the SLS-SVM method achieves better training accuracy with larger m. Given the random measurement matrix, the average classification rate on the

Table 5.4: Average classification accuracy (%) and training time (sec) of the proposed SLS-SVM algorithm, as a function of the size of the random measurement matrix.

Data sets	m	20%	40%	70%	100%
Cancer	Training	96.57±3.25	97.00±2.25	**99.14±0.65**	99.00±0.32
	Test	**99.14±0.43**	**99.14±0.43**	**99.14±0.43**	**99.14±0.43**
	Time	**1.22±0.57**	1.67±0.75	2.06±0.68	2.10±0.87
Card	Training	82.17±6.16	85.83±2.51	87.17±1.57	**88.30±0.45**
	Test	83.04±5.27	86.52±1.23	86.52±0.96	**86.52±0.23**
	Time	**1.09±0.84**	1.57±0.78	1.92±0.78	2.05±1.52
Diabetes	Training	72.66±12.43	84.41±1.84	87.32±0.48	**90.96±0.17**
	Test	69.14±4.43	73.05±2.23	**73.05±1.35**	72.31±1.32
	Time	**1.48±0.52**	2.29±0.78	2.83±0.92	2.79±1.05
Liver	Training	75.87±5.10	77.87±2.52	**79.22±1.33**	78.75±0.90
	Test	70.43±1.90	70.43±0.73	70.43±0.68	**70.43±0.66**
	Time	**0.18±0.01**	0.20±0.03	0.21±0.01	0.29±0.02
SPECT Heart	Training	83.89±4.21	86.11±2.01	87.78±0.91	**88.22±0.51**
	Test	85.56±1.01	**87.78±0.08**	85.56±0.06	85.67±0.03
	Time	**0.11±0.05**	0.13±0.07	0.14±0.08	0.14±0.07
Chess	Training	78.38±12.57	87.55±7.89	94.45±5.73	**94.50±3.66**
	Test	79.57±7.63	89.51±5.82	89.51±3.33	**89.51±1.58**
	Time	**165.56±58.75**	223.89±23.56	378.68±56.89	578.69±25.97
Average	Training	81.59±7.29	86.46±3.17	89.18±1.78	**89.96±1.00**
	Test	81.15±3.45	**84.40±1.75**	84.03±1.13	83.93±0.71
	Time	**28.27±10.12**	38.29±4.38	64.30±9.89	97.68±4.91

training set is 81.59%, 86.46%, 89.18%, and 89.96% for m = 20%, 40%, 70% and 100%, respectively. Obviously, higher training accuracy requires more measurements. Selecting only a very small number of measurements may not provide sufficient information for training. In terms of the training time, more measurements require longer training time. When m =100%, the proposed method with the random measurement and selection matrix requires 71.06% and 72.15% extra time, respectively, to train the LS-SVM model compared to that of m = 20%. Overall, using fewer measurements, the proposed algorithm not only achieves the affordable computation, but also maintains the good generalization performance.

5.4.3 Comparison with sparse training algorithms

In this subsection, the proposed SLS-SVM algorithm is compared with three conventional sparse training algorithms, namely Pruning1 [8], Pruning2 [170],

Table 5.5: Summary of the classification accuracy (%) and training time (sec) for the SLS-SVM algorithm with respect to the size of the selection matrix.

Data sets	m	20%	40%	70%	100%
Cancer	Training	96.57±2.57	97.00±2.10	98.50±0.57	**99.62**
	Test	98.71±0.57	**99.14±0.17**	**99.14±0.17**	98.71
	Time	**0.92±0.57**	1.46±0.75	1.95±0.68	2.08±0.75
Card	Training	85.43±2.75	**89.78±0.79**	88.91±0.57	88.33
	Test	82.83±0.57	87.39±1.45	**87.83±0.66**	86.52
	Time	**1.10±0.84**	2.03±0.78	2.05±0.78	2.06±1.78
Diabetes	Training	75.78±5.84	82.62±2.68	91.02±1.87	**96.09**
	Test	68.27±2.15	72.27±0.33	**75.78±0.17**	70.70
	Time	**1.29±0.52**	2.03±0.78	2.59±0.92	2.72±0.89
Liver	Training	65.22±7.84	78.52±5.33	78.22±2.33	**79.57**
	Test	65.57±1.56	**69.57±0.92**	68.70±0.85	68.96
	Time	**0.16±0.01**	0.22±0.03	0.24±0.12	0.29±0.02
SPECT Heart	Training	84.44±4.84	88.89±3.33	**90.00±2.56**	88.33
	Test	83.89±3.56	**86.67±0.77**	85.56±1.01	85.56
	Time	**0.10±0.05**	0.13±0.07	0.13±0.08	0.14±0.05
Chess	Training	76.20±6.23	88.56±4.89	92.14±3.73	**96.07**
	Test	76.12±8.33	87.00±5.82	88.35±1.33	**89.36**
	Time	**155.56±30.33**	203.77±20.56	365.78±48.23	550.76±35.79
Average	Training	80.61±5.01	87.56±3.18	89.80±1.94	**91.34**
	Test	79.23±2.79	83.67±1.58	**84.23±0.70**	83.30
	Time	**26.52±5.39**	34.94±3.83	62.14±8.47	93.01±6.55

and Add [175]. Both Pruning 1 and Pruning2 methods first train an LS-SVM using all training examples. Then, different pruning strategies are employed to selectively delete training examples until a certain number of support vectors are left. The Add method presented in [175] trains LS-SVM on a small set of training examples before adding another training example and updating the parameters. To make a fair comparison, we implemented the SLS-SVM using the least number of support vectors achieved by its counterpart. The standard LS-SVM algorithm is also considered for benchmarking. Furthermore, in SLS-SVM, the percentage of selected measurements is set to m =40% with the random measurement matrix. Table 5.6 shows the remaining model structure, the training accuracy, and the test accuracy obtained with different methods.

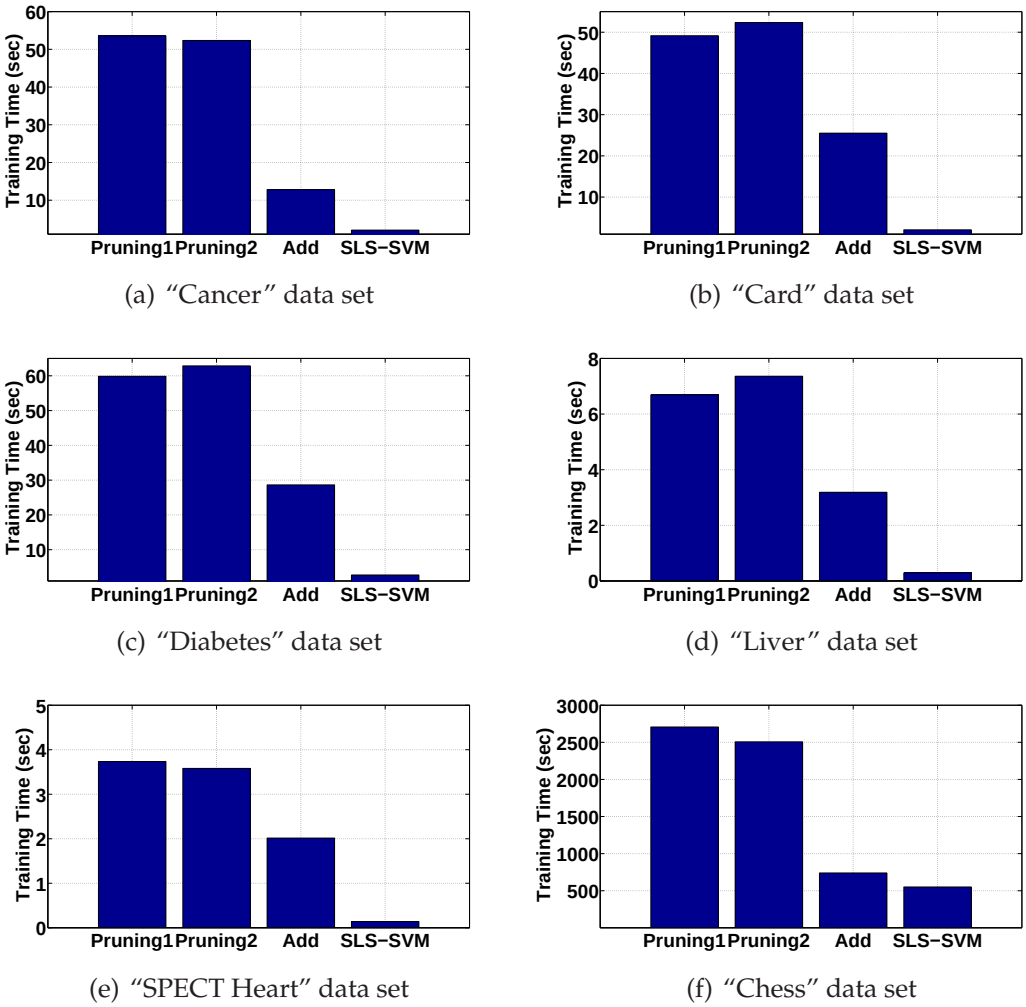
Compared to conventional training algorithms, the SLS-SVM achieves a significant improvement in terms of classification accuracy using the same number

Table 5.6: Summary of remaining support vectors and classification accuracy (%) for various sparse training algorithms of LS-SVM.

		LS-SVM	Pruning1[8]	Pruning2[170]	Add[175]	SLS-SVM
Cancer	k	100	33.00	37.00	**30.00**	**30.00**
	Training	96.35	98.35	**98.71**	97.77	97.00±2.26
	Test	98.71	94.57	95.26	97.00	**99.14±0.43**
Card	k	100	46.50	48.50	**32.20**	**32.20**
	Training	85.65	85.39	**86.01**	85.77	85.83±2.56
	Test	**86.96**	84.57	85.26	85.00	86.52±1.23
Diabetes	k	100	56.50	41.50	**30.20**	**30.20**
	Training	74.61	89.39	**92.14**	87.77	84.44±1.89
	Test	70.31	64.57	65.26	67.00	**73.05±2.27**
Liver	k	100	66.50	54.50	**21.70**	**21.70**
	Training	58.70	69.39	67.91	66.87	**75.87±5.08**
	Test	56.52	65.57	66.26	68.00	**70.43±1.88**
SPECT Heart	k	100	47.80	52.20	**23.90**	**23.90**
	Training	81.67	82.89	82.89	82.11	**85.11±2.51**
	Test	82.22	83.33	82.89	85.56	**86.67±0.67**
Chess	k	100	34.70	32.50	**29.20**	**29.20**
	Training	**89.03**	75.69	78.51	77.11	80.33±8.77
	Test	48.69	52.32	61.87	78.55	**89.18±3.77**
Average	k	100	47.50	44.37	**27.87**	**27.87**
	Training	81.00	83.58	84.48	83.40	**84.76±3.85**
	Test	73.90	74.16	76.13	80.19	**84.17±1.71**

of support vectors. On average, the SLS-SVM achieves a classification accuracy of 84.17% on the test sets from the six problems, which is much better than the accuracy of Pruning1 (74.16%), Pruning2 (76.13%) and Add (80.19%) methods. Furthermore, the SLS-SVM algorithm with average 27.87% support vectors outperforms the standard LS-SVM. The average accuracy of the proposed method is improved by 3.76% (on the training sets) and 10.27% (on the test sets) compared to the performance of the standard LS-SVM method. Overall, it is empirically confirmed that the proposed method obtains significant improvement compared to existing pruning methods and the standard LS-SVM algorithm.

We then investigated the computational complexity of all training algorithms. Figure 5.3 presents the average training time over 30 runs. The proposed algorithm spends 93.01 seconds on average to solve six problems, which is much better than the training time of Pruning1 (480.88 seconds), Pruning2(448.60 seconds) and Add (135.35 seconds) methods.



(a) "Cancer" data set

(b) "Card" data set

(c) "Diabetes" data set

(d) "Liver" data set

(e) "SPECT Heart" data set

(f) "Chess" data set

Figure 5.3: Training time obtained from different algorithms for the classification tasks.

There are two reasons for the slow convergence of conventional training methods. Firstly, existing methods require training the model before finding the sparse structure. Thus, it takes a long time, especially if a large number of training patterns or input attributes are involved. Secondly, conventional training methods apply the backward elimination to remove insignificant support vectors. By contrast, the proposed method is more efficient in terms of model training. Starting from the empty solution, SLS-SVM iteratively selects significant support vectors that minimize the residual output error. This method can be regarded as a forward selection instead of backward elimination, therefore a fast convergence is expected. Furthermore, the proposed approach employs a measurement matrix to

choose a small set of measurements to perform the sparse representation, rather than the entire training set. As a result, the SLS-SVM algorithm offers much quicker training time.

5.5 Conclusion

In this chapter, we have proposed a new LS-SVM training method based on sparse signal representation. The kernel matrix in LS-SVM is regarded as a dictionary in the sparse representation, hence the goal for training LS-SVM is converted to simply finding the sparse parameter for the classifier. A measurement matrix is also introduced to further reduce the computational complexity. The main difference between SLS-SVM and existing techniques is that the proposed method is capable of selecting important support vectors and training the LS-SVM model simultaneously. On the other hand, most traditional methods need to train the model before finding the support vectors. Consequently, the proposed training method leads to a quicker convergence and a much sparser structure.

Chapter 6

Feature Selection using Sparse Representation

Chapter contents

6.1 Introduction

In Chapters 3 and 4, we employed the sparse representation to optimize the structure of neural networks by selecting important hidden neurons. If we consider the sparse representation for the input pattern, then the process is equivalent to selecting the important input elements, i.e., feature selection. In this chapter, we discuss feature selection using sparse representation. The curse of dimensionality has attracted much research effort over the past decades. Many machine learning algorithms become unwieldy and computationally expensive if the number of features is too large. Determining the significant features is problem-dependent, which needs to balance the model performance and the number of required features. A large number of features not only results in overfitting and poor generalization, but also increases the computational overhead. On the other hand, a small set of features may not be sufficient for learning the task at hand due to lack of information.

Feature reduction approaches, therefore, are usually employed to reduce the input attributes without affecting performance; fewer but discriminative features could lead to less computational complexity and better generalization ability. Feature reduction methods can be broadly categorized as *dimensionality reduction* [177–186] and *feature selection* [187–197]. Traditionally, the dimensionality reduction method transforms the original data from the high-dimensional space to a space with fewer dimensions. One disadvantage is that some dimensionality reduction approaches involve eigenvalue decomposition which is extremely time-consuming. By contrast, feature selection algorithms aim to determine the significance of available features, thereby selecting a small set of informative features from the original data.

In this chapter, we propose a novel algorithm for feature selection based on sparse representation, termed herein *sparse feature selection* (SFS) algorithm. We consider the discriminative features as a subset of the full feature vector. The

aim is to find the sparse feature vector that explains the observations. To reduce computational complexity even further, a measurement matrix is introduced, which projects the input features onto a lower dimensional space.

The remainder of the chapter is organized as follows. Section 6.2 presents a brief review of existing feature reduction algorithms. Section 6.3 establishes the link between feature selection and sparse representation, and presents the SFS algorithm. Section 6.4 presents experimental results based on several classification problems, and compares the performance of the SFS algorithm with those of existing methods. Furthermore, the SFS algorithm is evaluated on a real word application of pedestrian detection. Section 6.5 presents concluding remarks.

6.2 Feature reduction

Feature reduction is a fundamental technique for large-scale data processing. Based on feature reduction, the salient information from the original data is maintained while the number of required features is reduced. Generally, feature reduction approaches are cast into two categories: dimensionality reduction and feature selection.

- Dimensionality reduction aims to extract features by projecting the original high-dimensional data to a lower-dimensional space through an algebraic transformation [177–181]. Examples of dimensionality reduction include *principal component analysis* (PCA) [178, 179] and *linear discriminant analysis* (LDA) [180, 181].

- Feature selection finds the most representative features from the original data using a binary transformation matrix. The most popular feature selection methods are *Laplacian Score* (LS) [187], *ReliefF* [188], *Simba* [189], and *Mutual Information*-based methods [190, 198].

This chapter is concerned with feature selection. However, for completeness dimensionality reduction methods are briefly reviewed in the next subsection.

6.2.1 Dimensionality reduction

In this subsection, we consider two most popular algorithms for dimensionality reduction, namely PCA and LDA. The PCA algorithm employs an orthogonal linear transformation to reduce the original data to a set of linearly uncorrelated variables. The input variables are converted to a new coordinate system in which the greatest variance comes from the first coordinate (known as the first principal component), the second greatest variance from the second coordinate, etc. Suppose that we have a data set consisting of Q patterns $\mathbf{p}_i$, $i = 1, \cdots, Q$. Let Σ be the covariance matrix, that is,

$$\Sigma = \frac{1}{Q} \sum_{i=1}^{Q} \left(\mathbf{p}_i - \overline{\mathbf{p}}\right)\left(\mathbf{p}_i - \overline{\mathbf{p}}\right)^{\mathrm{T}}, \tag{6.1}$$

where $\mathbf{p}_i$ is the i-th input and $\overline{\mathbf{p}}$ is the mean vector of the entire data set. The PCA algorithm aims to find a transformation matrix W by maximizing the following optimization criteria $J(W)$:

$$J(W) = trace\left(W^{\mathrm{T}} \Sigma W\right). \tag{6.2}$$

LDA is a single-stage supervised technique which finds linear projections of data so that the classes are maximally separable. Let n be the number of classes, and π_i be the prior probability of class i. Furthermore, let Σ_i denote the covariance matrix of the i-th class, and $\overline{\mathbf{p}}_i$ be the mean vector of class i. Then, the within-class scatter matrix is defined as

$$S_W = \sum_{i=1}^{n} \pi_i \Sigma_i. \tag{6.3}$$

In addition, the between-class scatter matrix S_B is defined as

$$S_B = \sum_{i=1}^{n} \pi_i \left(\overline{\mathbf{p}}_i - \overline{\mathbf{p}}\right)\left(\overline{\mathbf{p}}_i - \overline{\mathbf{p}}\right)^{\mathrm{T}}. \tag{6.4}$$

Thus, LDA is used to find the feature space in which

$$J(W) = \operatorname{trace}\left[\left(W^{\mathrm{T}} S_W W\right)^{-1}\left(W^{\mathrm{T}} S_B W\right)\right] \tag{6.5}$$

is maximized.

Dimensionality reduction has drawn increasing attention in the signal processing community since it offers an efficient model providing a simple interpretation for the original data. Therefore, it has become a very important tool for many applications, such as omics prediction [180], face recognition [199], and audio coding [200]. One limitation is that some methods, such as PCA and LDA, require the eigenvalue decomposition that is extremely time-consuming. In particular, if the dimension exceeds the number of training samples, LDA may encounter a singularity problem.

6.2.2 Feature selection

In contrast to dimensionality reduction methods, feature selection algorithms aim to detect the most representative features from the original data set. All the features are measured and ranked according to their significance; then features are selected based on their ranks.

Suppose we have Q pairs of input vectors $\mathbf{p}_i$ and corresponding output vectors $\mathbf{z}_i$, $i = 1, \cdots, Q$. Assuming the input vector $\mathbf{p}_i$ is multidimensional, let $\mathbf{f}_j$ denote a vector containing the j-th feature from all input patterns:

$$\mathbf{f}_j = \left[p_{1,j}, p_{2,j}, \ldots, p_{Q,j}\right]^{\mathrm{T}}, \tag{6.6}$$

where $p_{i,j}$ is the j-th component of $\mathbf{p}_i$. Feature selection is performed by assigning to each feature $\mathbf{f}_j$ its significance of relevance $R(\mathbf{f}_j)$. Several feature selection methods have been proposed to evaluate the significance of features, some of which are summarized below.

1. Laplacian Score (LS) method evaluates a feature based on its locality preserving power [187]. The LS method generates a nearest neighbour graph with Q nodes, with each node representing an input pattern. The significance of $\mathbf{f}_j$ is computed as follows:

$$R(\mathbf{f}_j) = \frac{\widetilde{\mathbf{f}}^{\mathrm{T}} M_l \widetilde{\mathbf{f}}}{\widetilde{\mathbf{f}}^{\mathrm{T}} (M_w + M_l) \widetilde{\mathbf{f}}}, \tag{6.7}$$

where

$$\widetilde{\mathbf{f}} = \mathbf{f}_j - \frac{\mathbf{f}_j^{\mathrm{T}} (M_w + M_l) \mathbf{1}}{\mathbf{1}^{\mathrm{T}} (M_w + M_l) \mathbf{1}}, \tag{6.8}$$

M_w is the weight matrix of the graph, M_l is the graph Laplacian matrix [201], and $\mathbf{1}$ is a column vector with all elements equal to 1.

2. ReliefF algorithm measures the importance of features according to the margin distance from training patterns [188]. In ReliefF method, the feature significance $R(\mathbf{f}_j)$, which is initialized as zero, is iteratively updated when a new training sample is considered. Assume that the newly added training sample is $\mathbf{p}_i$. Let $\mathbf{q}_i$ and $\mathbf{r}_i$ denote the nearest samples to the selected pattern $\mathbf{p}_i$ with the same and different label, respectively. Then the relevance of $\mathbf{f}_j$ is computed as

$$R(\mathbf{f}_j) = R(\mathbf{f}_j) + (p_{i,j} - q_{i,j})^2 - (p_{i,j} - r_{i,j})^2, \tag{6.9}$$

where $q_{i,j}$ and $r_{i,j}$ denote the relevant j-th feature from the $\mathbf{q}_i$ and $\mathbf{r}_i$ pattern, respectively.

3. Simba is another feature selection method based on the margin distance

[189]. The following equation is used to calculate $R(\mathbf{f}_j)$:

$$R(\mathbf{f}_j) = R(\mathbf{f}_j) + \frac{1}{2}\frac{\partial u(\mathbf{p}_i)}{\partial \mathbf{p}_i}\left(\frac{\left(p_{i,j} - q_{i,j}\right)^2}{\left\|\mathbf{p}_i - \mathbf{q}_i\right\|_2} - \frac{\left(p_{i,j} - r_{i,j}\right)^2}{\left\|\mathbf{p}_i - \mathbf{r}_i\right\|_2}\right)R(\mathbf{f}_j), \tag{6.10}$$

where $u(\cdot)$ is a predefined utility function. The commonly used utility functions are the linear, $u(\mathbf{p}_i) = \mathbf{p}_i$, and the sigmoid function, $u(\mathbf{p}_i) = 1/(1 + \exp(-\beta * \mathbf{p}_i))$, where β is a positive scalar defined by the user. Based on different utility functions, the Simba algorithm is cast into *Simba Linear* (SL) and *Simba Sigmoid* (SS) methods.

4. Mutual information (MI)-based feature selection approach is introduced in [190, 198]. Two algorithms have been proposed using the Max-Relevance (MAX_R) and Min-Redundancy (MIN_R) measures. First, the mutual information is introduced to measure the level of similarity between two discrete random vectors $\mathbf{f}$ and $\mathbf{z}$:

$$I(\mathbf{f}, \mathbf{z}) = \sum_{f \in \mathbf{f}} \sum_{z \in \mathbf{z}} p(f, z) \log\left(\frac{p(f, z)}{p(f)\, p(z)}\right), \tag{6.11}$$

where $p(f, z)$ is the joint probability function of $\mathbf{f}$ and $\mathbf{z}$, and $p(f)$ and $p(z)$ are the marginal probability functions of $\mathbf{f}$ and $\mathbf{z}$, respectively. Then the MAX_R measures the mutual information between the selected features and all the desired outputs:

$$\mathrm{MAX}_R = \frac{1}{|S|}\sum_{j \in S} I\left(\mathbf{f}_j, \mathbf{z}_i\right), \quad i = 1, 2, ..., Q, \tag{6.12}$$

where S is the set of indices of the selected features. Moreover, the MIN_R measure considers the mutual information between the j-th and k-th feature:

$$\mathrm{MIN}_R = \frac{1}{|S|}\sum_{j,k \in S} I\left(\mathbf{f}_j, \mathbf{f}_k\right). \tag{6.13}$$

Two methods have been proposed to optimize Max-Relevance and Min-Redundancy simultaneously. One of them is the *mutual information difference* (MID) method that searches features that maximize the difference between MAX_R and MIN_R,

$$\mathrm{MID} = (\mathrm{MAX}_R - \mathrm{MIN}_R)\,.$$

Another method is called *mutual information quotient* (MIQ), which selects important features by maximizing the ration between MAX_R and MIN_R,

$$\mathrm{MIQ} = (\mathrm{MAX}_R/\mathrm{MIN}_R)\,.$$

6.3 Sparsity-based feature selection

In this section, we describe the proposed feature selection method using sparse representation. In Subsection 6.3.1, we reformulate the feature selection as a sparse representation problem, in which discriminative features are selected by finding the sparse representation for the original features. In Subsection 6.3.2, we introduce a compressed sensing-based approach to further reduce the computational cost.

6.3.1 Sparsity-based feature selection

Suppose that we have a data set consisting of Q patterns $(\mathbf{p}_i, \mathbf{z}_i)$ of input vectors and their corresponding desired output patterns. The classifier aims to extract the decision rule subject to the following constraint

$$\mathbf{z}_i = f\left(\mathbf{p}_i\right) + \mathbf{e}_i, \tag{6.14}$$

where $f(\cdot)$ is an unknown decision function to be estimated and $\mathbf{e}_i$ is the corresponding error. In general $f(\mathbf{p}_i)$ can be expressed as

$$f(\mathbf{p}_i) = X^{\mathrm{T}}\Psi(\mathbf{p}_i), \tag{6.15}$$

where X is an $N \times L$ weight matrix and $\Psi(\mathbf{p}_i)$ is an N-dimensional feature vector derived from the input $\mathbf{p}_i$. For instance, in support vector machines (SVMs), $\Psi(\cdot)$ represents the user-defined kernel function, whereas in feed-forward neural networks, $\Psi(\cdot)$ is generated by the hidden layers of the network.

Let $P = \left[\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Q\right]^{\mathrm{T}}$ denote the matrix containing all input patterns arranged into rows and $Z = \left[\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_Q\right]^{\mathrm{T}}$ be the corresponding $Q \times L$ matrix of desired outputs. Combining Eq. (6.14) and (6.15) for all training patterns, we obtain the constraint equation in matrix form:

$$Z = \Psi\left(P\right) X + E, \tag{6.16}$$

where E is the error matrix with $\mathbf{e}_i^{\mathrm{T}}$ as the i-th row. The main problem with Eq. (6.16) is that all N features, i.e., columns of $\Psi\left(P\right)$, contribute to the final output even though some of them may be redundant or irrelevant.

Using feature selection approaches, a classifier can be built with fewer features. When the i-th feature is removed, its corresponding weights (or the i-th row of the weight matrix X) will be assigned zero values. On the other hand, the features associated with nonzero rows from X remain. To select the most representative features, a sparsity-based feature selection algorithm is proposed to find the sparse representation of the original features $\Psi\left(P\right)$. The sparse solution aims to preserve as much of the information from the original features and to minimize the number of selected features simultaneously. Given a multi-dimensional output ($L > 1$), the feature selection is reformulated as a multiple measurement vector (MMV)

problem:

$$\min \; S(X) \quad \text{subject to} \quad \|Z - \Psi(P)\, X\|_2 \leq \epsilon, \tag{6.17}$$

where $S(X)$ is the matrix sparsity and ϵ is the bound on the power of the noise or the error. Here, the matrix $\Psi(P)$ plays the role of a dictionary, and the weight matrix X contains the sparse decomposition coefficients. The problem then is to find a sparse representation of the desired input-output mapping using atoms from the feature dictionary $\Psi(P)$. Since each row of X contains the weights of a particular feature, the problem in (6.17) is to minimize the number of non-zero rows of X. To solve the MMV model, we apply the M-OMP algorithm presented in [60] due to its simplicity and efficiency. Only the features that contribute most to the minimization of the residual error are selected. Furthermore, if we consider the problem with a one-dimensional (1-D) output vector, the problem from Eq. (6.17) is converted to solving a sparse solution for a 1-D weight vector $\mathbf{x}$:

$$\min \; S(\mathbf{x}) \quad \text{subject to} \quad \|\mathbf{z} - \Psi(P)\, \mathbf{x}\|_2 \leq \epsilon, \tag{6.18}$$

where $S(\mathbf{x})$ denotes the measure of the vector sparsity. In this case, we can employ the single measurement vector (SMV) model to solve the feature selection problem.

6.3.2 Feature selection using compressed sensing

So far we have exploited sparse signal representation for selecting discriminative features in pattern classification. However, the minimization problem in Eq. (6.17) contains the entire data set with all Q training samples, which is time-consuming and memory intensive.

Compressed sensing (CS) allows the calculation of the sparse solution from a reduced set of non-adaptive linear measurements, rather than the entire data sets. The stable recovery of the sparse solution can be achieved by simply choosing a

random measurement matrix in most cases [41, 66, 67]. As shown in Chapter 5, the reduced set of measurements, using a linear projection matrix, is sufficient to train the LS-SVM classifier. Similarly, the measurement matrix Φ of size $M \times Q$ ($M << Q$) is introduced here. Then the problem in Eq. (6.17) is rewritten as follows:

$$\min\ S(X) \quad \text{subject to} \quad \|Y - \mathcal{D}X\|_2 \leq \epsilon, \tag{6.19}$$

where $Y = \Phi Z$ is the reduced measurements, and the dictionary $\mathcal{D} = \Phi\Psi(P)$. As a result, the measurement matrix Φ can be used to solve the sparse representation problem using fewer measurements, thereby reducing the computational complexity.

The sparsity-based feature selection algorithm (SFS) is summarized in Algorithm 8. Finally, we should note that the proposed method can also be employed to the input data if the simple linear mapping between input vectors and feature vectors is adopted, $\Psi(P) = P$. In this case, a linear input-output relation, $Z = PX$, is assumed during feature selection; however, after feature selection any nonlinear classifier can still be used to learn the desired input-output mapping.

6.4 Experimental results

This section presents the experimental results and comparison of the proposed SFS approach with other conventional feature reduction methods. First, Subsection 6.4.1 presents experimental results based on several classification problems. Then, Subsection 6.4.2 evaluates the performance of the SFS algorithm on a real word application of pedestrian detection.

6.4.1 Feature Selection for Classification

In this section, we evaluate the performance of the SFS algorithm on several classification problems and compare it to those of existing feature selection methods.

Algorithm 8: Sparse representation-based algorithm for feature selection.

input : The desired output $Z \in \mathbb{R}^{Q\times L}$, and the original feature matrix $\Psi(P) \in \mathbb{R}^{Q\times N}$;

output: Index of selected features I_s;

Randomly initialize the measurement matrix Φ;

Solve the following problem:

$\widehat{X} = \arg\min_X S(X) \quad \text{subject to} \quad \|Y - \mathcal{D}X\|_2 \leq \epsilon,$

where $Y = \Phi Z$ *and* $\mathcal{D} = \Phi\Psi(P)$.

Calculate the relevant residual error for each nonzero row $\mathbf{x}_j^{\mathrm{T}}$ *from* X*:*

$\epsilon_j = \left\| Y - \mathcal{D}_j \mathbf{x}_j^{\mathrm{T}} \right\|_2, \quad j = 1, 2, ..., N,$

where $\mathcal{D}_j$ *is the* j*-th column from* $\mathcal{D}$;

Rank the features according to the error: $I_s = \mathrm{sort}(\epsilon_j)$*, and select the features based on their ranks.*

The data sets and the evaluation criterion are presented in Subsection 6.4.1.1. The effect of the measurement matrix is analyzed in Subsection 6.4.1.2. The comparison results between the proposed algorithm and traditional feature selection algorithms are then presented in Subsection 6.4.1.3.

6.4.1.1 Experimental methods

Eight benchmark problems are selected from UCI data sets[134], see Table 6.1. Each data set is partitioned into two subsets: a training set and a test set. The training patterns are used to select features and train the classifier. The test set is used for measuring the performance of the selected features. The selection of the problems is made considering both continuous and binary patterns and a diverse range of the problem domains.

The proposed SFS algorithm is compared with six existing feature selection methods, i.e., laplacian score (LS) [187], ReliefF [188], Simba-Linear (SL), Simba-Sigmoid (SS) [189], mutual information difference (MID), and mutual information quotient (MIQ) methods [190]. The support vector machine (SVM) with radial

Table 6.1: Data sets used for feature selection.

Dataset	No. Features	No. Training Patterns	No. Test Patterns
Arrhythmia	279	294	146
Cancer (wisconsin)	30	378	189
Dermatology	34	240	120
Soybean	34	453	227
Spectf	44	177	89
Bench	20	135	57
Chess	36	2137	915
Ionosphere	34	175	75

basis function (RBF) kernel is employed at the final classification stage. The performances of the various feature selection algorithms are evaluated using classification accuracy as a function of the number of features. With the same classifier, a better feature selection method should lead to a better classification performance with fewer features. The performance of the proposed SFS algorithm is evaluated in terms of the number of selected measurements and the type of measurement matrix used in the CS formulation. The percentage of selected measurements is measured as

$$m = M/Q \times 100\%, \tag{6.20}$$

where M is the number of selected measurements and Q is the number of all training samples.

6.4.1.2 Performance evaluation

To reduce the computational complexity, a measurement matrix is employed in the proposed algorithm to reduce the number of selected measurements. In this subsection, we evaluate the performance of the SFS method using two types of measurement matrices: a random measurement matrix and a selection matrix, with each row and each column containing only one non-zero element (equal to 1). We also consider the robustness of the proposed algorithm to various numbers of selected measurements. The percentage of selected measurements is

set to m = 20%, 50% and 100%. Note that using the selection matrix with m = 100% is equivalent to solving the sparse representation for the original features (see Eq. (6.17)). Tables 6.2 and 6.3 show the average classification accuracy of the proposed algorithm with, respectively, the random measurement and selection matrix over 20 runs.

Table 6.2: Average classification accuracy (%) over 20 runs on the test data set as a function of the size of the random measurement matrix. For convenience the number of selected features is enclosed in parentheses next to the classification rate.

Data sets	m =20%	m =50%	m =100%
Arrhythmia	**67.58±4.25** (38)	67.20±3.20 (70)	66.51±2.01 (**27**)
Cancer (wisconsin)	**97.08±1.10** (**26**)	96.68±0.78 (29)	96.68±0.78 (29)
Dermatology	88.73±2.45 (30)	**89.56±3.56** (**29**)	88.72±2.56 (34)
Soybean	**94.89±0.32** (38)	94.86±1.02 (**30**)	93.99±0.62 (35)
Spectf	79.63±1.02 (**8**)	80.99±1.23 (13)	**81.82±1.31** (10)
Bench	65.79±1.07 (20)	65.23±0.77 (**19**)	**65.84±1.38** (20)
Chess	91.99±0.59 (**6**)	92.26±0.45 (10)	**92.68±0.6**5 (10)
Ionosphere	**93.98±1.35** (25)	93.89±0.75 (25)	93.78±0.90 (**18**)
Average	84.96±1.52 (24)	**85.08±1.47** (28)	85.00±1.28 (**23**)

Table 6.3: Summary of classification accuracy (%) over 20 runs on the test data set with different sizes of the selection matrix. For convenience the number of selected features is enclosed in parentheses next to the classification rate.

Data sets	m =20%	m =50%	m =100%
Arrhythmia	**67.77±3.36** (48)	66.00±3.20 (78)	64.11 (**9**)
Cancer (wisconsin)	96.71±0.65 (30)	**96.83±0.63** (15)	96.72 (**11**)
Dermatology	88.23±3.37 (34)	89.23±4.12 (34)	**90.22 (22)**
Soybean	**95.01±0.68** (34)	94.96±0.67 (32)	94.95 (**30**)
Spectf	79.63±0.31 (**6**)	81.12±0.33 (13)	**81.82** (8)
Bench	64.46±1.72 (20)	64.92±0.92 (20)	**67.38 (18)**
Chess	92.21±0.48 (**6**)	92.23±0.41 (**6**)	**92.93** (11)
Ionosphere	92.41±1.20 (29)	**94.32±0.95** (23)	93.25 (**16**)
Average	84.55±1.48 (26)	84.95±1.40 (28)	**85.17** (**15**)

Several observations can be made from these results. First, the proposed method achieves similar classification accuracy regardless of the type of the measurement matrix. For instance, the proposed algorithm generates on average 84.96%, 85.08%, and 85.00% classification accuracy with the random measurement matrix, whereas, the classification accuracy using the selection matrix is

84.55%, 84.95%, and 85.17%, respectively. The maximum difference in classification accuracy from eight data sets is only 0.41%, which is insignificant.

Second, with different sizes of the measurement matrix, the SFS method achieves similar generalization ability. In other words, the classification rate on the test set is not guaranteed to be improved with more measurements. For example, with the selection matrix, the overall average classification accuracy (over the 8 datasets) is 85.17% for m = 100% and 84.55% for m = 20% (see Table 6.3); the difference is only 0.62%, which is not significant at the 95% confidence interval.

Tables 6.4 and 6.5 show the average time for feature selection with different sizes of random measurement and selection matrix, respectively. In all cases, a larger m (more selected measurements) leads to a slower convergence. Overall, using fewer measurements, the proposed algorithm not only achieves the affordable computation, but also maintains the good generalization performance.

Table 6.4: Average time for feature selection (sec) with different sizes of the random measurement matrix.

Data sets	m =20%	m =50%	m =100%
Arrhythmia	0.05±0.05	0.91±0.91	**10.42±1.42**
Cancer (wisconsin)	0.02±0.01	0.02±0.02	**0.02±0.03**
Dermatology	0.02±0.02	0.02±0.02	**0.03±0.03**
Soybean	0.02±0.01	0.02±0.01	**0.03±0.01**
Spectf	0.02±0.01	0.03±0.02	**0.05±0.04**
Bench	**0.02±0.01**	**0.02±0.01**	**0.02±0.01**
Chess	0.05±0.03	0.08±0.05	**0.11±0.07**
Ionosphere	0.01±0.01	0.02±0.01	**0.02±0.02**
Average	0.03±0.02	0.14±0.13	**1.34±0.20**

6.4.1.3 Comparison with conventional feature selection methods

In this subsection, the proposed algorithm is compared with other conventional methods. Note that if the selection matrix is employed in SFS, it is equivalent to selecting a subset of training patterns. To analyze the robustness of various

Table 6.5: Average time for feature selection (sec) as a function of the size of the selection matrix.

Data sets	m =20%	m =50%	m =100%
Arrhythmia	0.06±0.01	0.95±0.13	**11.16±1.16**
Cancer (wisconsin)	0.02±0.01	0.02±0.01	**0.03±0.01**
Dermatology	0.02±0.01	0.02±0.01	**0.03±0.01**
Soybean	0.02±0.01	0.03±0.01	**0.03±0.01**
Spectf	0.01±0.01	0.03±0.01	**0.05±0.02**
Bench	0.01±0.01	0.01±0.01	**0.02±0.02**
Chess	0.06±0.03	0.11±0.05	**0.19±0.05**
Ionosphere	0.02±0.01	0.02±0.01	**0.03±0.01**
Average	0.03±0.01	0.15±0.03	**1.44±0.16**

feature selection methods to the number of training samples, we select a subset from training samples by setting m = 20%, 50%, and 100% with the selection matrix. Then all feature selection methods are evaluated using different numbers of training patterns. The average results over 20 runs are listed in Tables 6.6, 6.7, and 6.8.

In all three cases, the proposed SFS algorithm outperforms other conventional feature selection methods; it always achieves the best classification accuracy. In particular, with all training samples (m =100%), the proposed algorithm achieves the best classification rate from five data sets: namely Cancer (wisconsin), Dermatology, Soybean, Bench, and Chess. The SFS method also selects the fewest number of features (15 features) on average to solve all eight problems.

In conclusion, it is empirically confirmed that the proposed SFS method obtains significant improvement compared to existing feature selection methods. Note that since the SFS algorithm selects features that minimize the output error, the feature selection process has a direct correspondence with the classification accuracy. By contrast, the criteria used by other features selection methods are not directly related to the classification purpose.

Table 6.6: Summary of classification accuracy (%) on the test set using m =20% training patterns. The number of selected features is shown inside the bracket.

Data Sets	LS	Relief	SL	SS	MID	MIQ	SFS
Arrhythmia	65.01±2.23 (40)	67.22±3.10 (57)	60.45±2.10 (**27**)	60.78±2.10 (95)	67.55±4.31 (44)	66.72±3.10 (55)	**67.77±3.36** (48)
Cancer (wisconsin)	96.28±0.51 (**29**)	96.73±0.46 (30)	96.69±0.73 (**29**)	96.56±0.67 (**29**)	**96.82±0.56** (**29**)	96.78±0.56 (**29**)	96.71±0.65 (30)
Dermatology	87.66±4.57 (34)	87.91±4.10 (**28**)	88.12±3.01 (34)	88.21±4.10 (32)	87.55±4.13 (32)	86.78±5.10 (32)	**88.23±3.37** (34)
Soybean	94.57±0.49 (34)	94.63±0.19 (33)	**95.20±0.98** (**25**)	94.82±0.82 (26)	92.89±0.67 (32)	93.29±0.57 (33)	95.01±0.68 (34)
Spectf	79.11±0.22 (13)	79.44±0.23 (14)	79.44±0.33 (10)	79.56±0.53 (7)	79.22±0.42 (5)	79.43±0.42 (**4**)	**79.56±0.31** (6)
Bench	64.67±1.23 (20)	65.31±0.87 (20)	65.55±0.79 (17)	65.21±0.68 (18)	**65.61±0.72** (17)	64.61±1.62 (**15**)	64.46±1.72 (20)
Chess	87.78±5.23 (36)	92.12±0.24 (24)	92.12±0.34 (15)	**92.34±0.44** (22)	91.66±1.32 (29)	92.22±0.62 (12)	92.21±0.48 (**6**)
Ionosphere	89.36±3.01 (28)	92.67±2.20 (29)	93.11±2.20 (**24**)	92.79±2.33 (25)	**93.27±1.61** (25)	92.92±1.81 (**24**)	92.41±1.20 (29)
Average	83.06±2.19 (29)	84.50±1.42 (29)	83.84±1.31 (**23**)	83.78±1.42 (32)	84.32±1.72 (27)	84.09±1.72 (26)	**84.55±1.48** (26)

Table 6.7: Summary of classification accuracy (%) on the test set with m =50% training patterns. The number of selected features is shown inside the bracket.

Data Sets	LS	Relief	SL	SS	MID	MIQ	SFS
Arrhythmia	64.74±2.10 (43)	66.78±3.41 (36)	60.45±2.22 (**27**)	60.78±2.42 (**27**)	**68.23±4.53** (42)	67.56±4.21 (44)	66.00±3.20 (78)
Cancer (wisconsin)	**96.85±0.16** (29)	96.67±0.71 (**15**)	96.61±0.71 (29)	96.57±0.87 (29)	96.78±0.61 (29)	96.85±0.62 (29)	96.83±0.63 (**15**)
Dermatology	88.02±4.10 (34)	87.34±4.22 (33)	88.95±3.01 (29)	88.79±4.71 (**28**)	87.89±4.22 (32)	86.22±5.41 (34)	**89.23±4.12** (34)
Soybean	94.82±0.81 (34)	94.67±0.89 (33)	94.81±0.81 (30)	94.87±0.88 (**28**)	93.89±0.72 (33)	93.92±0.37 (32)	**94.96±0.67** (32)
Spectf	79.23±0.34 (30)	79.21±0.24 (11)	80.67±0.45 (9)	79.78±0.34 (12)	78.56±0.33 (**5**)	79.43±0.45 (12)	**81.12±0.33** (13)
Bench	64.56±1.22 (20)	65.33±0.97 (19)	**66.20±1.31** (18)	65.10±0.88 (18)	65.78±0.39 (19)	64.56±1.12 (**15**)	64.92±0.92 (20)
Chess	88.23±4.23 (27)	92.12±0.24 (10)	92.42±0.45 (17)	92.56±0.55 (19)	91.99±1.20 (10)	**92.78±0.67** (10)	92.23±0.41 (**6**)
Ionosphere	89.78±3.10 (30)	92.67±1.20 (27)	92.89±1.23 (23)	93.11±1.20 (**18**)	93.31±0.89 (20)	92.78±0.89 (26)	**94.32±0.95** (23)
Average	83.28±2.00 (31)	84.35±1.48 (23)	84.13±1.27 (23)	83.94±1.48 (**22**)	84.55±1.61 (24)	84.26±1.72 (25)	**84.95±1.40** (28)

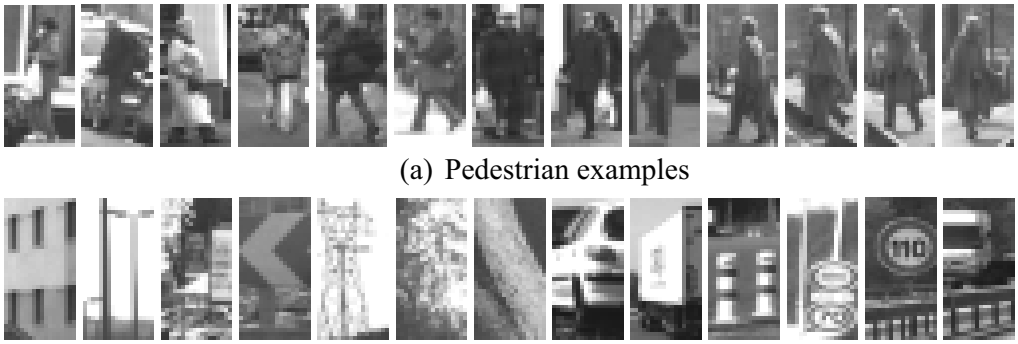
Table 6.8: Summary of classification accuracy (%) on the test data set based on the entire data set (m =100%). The number of selected features is shown inside the bracket.

Data Sets	LS	Relief	SL	SS	MID	MIQ	SFS
Arrhythmia	68.45±2.21 (15)	**70.22±3.12** (28)	60.45±2.51 (27)	60.55±2.21 (27)	68.22±4.34 (19)	69.21±4.21 (46)	64.11 (**9**)
Cancer (wisconsin)	96.72±0.85 (13)	96.67±0.71 (14)	96.56±0.83 (29)	96.55±0.82 (28)	96.67±0.62 (29)	96.67±0.61 (29)	**96.72 (11)**
Dermatology	88.33±3.01 (32)	88.32±5.12 (29)	88.92±4.71 (27)	88.78±4.11 (26)	88.33±5.10 (27)	85.89±6.11 (34)	**90.22 (22)**
Soybean	94.67±0.78 (31)	94.67±0.94 (27)	94.78±0.88 (32)	94.89±0.82 (**27**)	93.80±0.58 (32)	93.71±0.86 (31)	**94.95** (30)
Spectf	80.23±0.92 (31)	81.92±1.52 (8)	80.67±0.95 (6)	81.45±0.38 (8)	81.75±1.40 (**2**)	**81.83±1.82** (**2**)	81.82 (8)
Bench	64.44±1.22 (20)	67.11±1.82 (2)	65.44±0.68 (16)	65.82±0.91 (17)	65.11±0.57 (**1**)	64.73±1.12 (**1**)	**67.38** (18)
Chess	88.56±4.41 (31)	92.33±0.24 (17)	92.23±0.51 (24)	92.67±0.52 (**8**)	91.89±2.12 (10)	92.79±0.81 (11)	**92.93** (11)
Ionosphere	90.00±2.10 (**15**)	92.69±1.56 (18)	93.23±1.20 (23)	93.08±1.22 (20)	**93.89±1.32** (20)	93.68±1.80 (25)	93.25 (16)
Average	83.93±1.89 (24)	85.49±1.88 (18)	84.04±1.53 (23)	84.22±1.38 (20)	84.96±2.00 (18)	84.81±2.17 (22)	**85.17** (**15**)

6.4.2 Pedestrian detection

In this subsection, the proposed feature selection method is combined with a hierarchical visual pattern recognition architecture for pedestrian detection. The proposed method is also compared with existing dimensionality reduction and feature selection methods on the pedestrian detection task.

Pedestrian detection has many applications in traffic safety, law enforcement and car industry. Its objective is to determine the presence and the location of people or pedestrians in images and video. A comprehensive study on pedestrian detection, using the Daimler-Chrysler database, is given in [202]. The Daimler-Chrysler database consists of three training sets and two test sets. Each set has 4800 pedestrian patterns and 5000 non-pedestrian patterns; each image pattern is of size 36×18 pixels. Some examples of pedestrian and non-pedestrian patterns are shown in Fig. 6.1.



(a) Pedestrian examples

(b) Non-pedestrian examples

Figure 6.1: Image patterns from the Daimler-Chrysler pedestrian detection database.

A feature extraction method is proposed in [19] to detect the features from the input image, which is based on directional and adaptive filters, inspired by the human visual system. Compared to other existing methods presented in [202], the architecture presented in [19] achieves better detection accuracy. However, the drawback is that the number of generated features is too large, over 2016. Therefore, the proposed feature selection method is applied to reduce the number of features. In SFS, the percentage of selected measurements is set to m =80%

with the selection matrix.

We first conduct the comparison between the SFS-based system and the full architecture [19]. The SFS algorithm is applied after all the features have been detected to select the important features. Table 6.9 presents the average results from the proposed algorithms over 20 independent runs, using a linear classifier. Note that the original architecture achieves 99.70% and 91.72% on training and testing data, respectively.

Table 6.9: Summary of the classification accuracy using the SFS-based algorithm for the pedestrian detection.

Number of selected features	Training Accuracy (%)	Test Accuracy (%)
100	93.21±2.72	87.55±1.22
300	96.63±2.53	90.95±0.41
500	97.53±1.37	91.72±0.61
1000	**98.21±1.33**	**91.81±0.63**

Several observations can be made from Table 6.9. Firstly, when more features are employed, the performance of the SFS algorithm is improved in terms of the training and test accuracy. Although the training error is larger than that of the initial structure, the proposed algorithm is built on fewer features. Secondly, the average test accuracy obtained from the SFS-based algorithm is comparable to that of the full architecture, especially when 500 or 1000 features are selected. For instance, in SFS 91.72% classification accuracy is achieved when only 500 features are provided. In other words, when SFS is employed, the same classification accuracy is obtained with 75% reduction in number of features. Figure 6.2 gives the typical results obtained by the SFS-based pedestrian detection system.

The proposed approach was also compared on the pedestrian detection problem to four feature reduction methods: PCA, LDA, MID and MIQ. Table 6.10 shows the average detection accuracy with respect to the number of features for 20 independent runs. Clearly, the proposed SFS algorithm outperforms the other 4 feature reduction methods, especially when the number of selected feature is

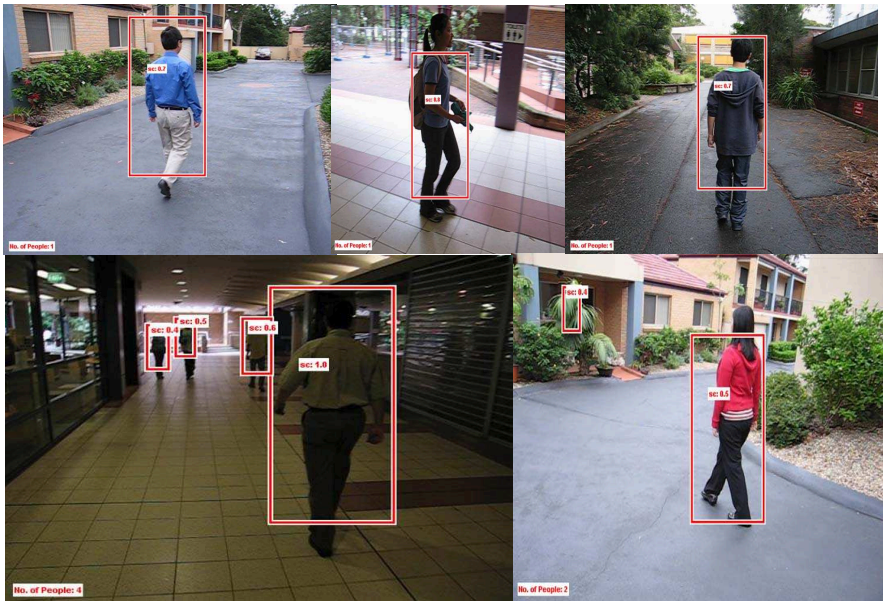


Figure 6.2: The pedestrian detection outputs for some test images. The detection score has been highlighted.

small. For example, with only 100 features, the SFS achieves 87.55% classification accuracy, whereas, the PCA, LDA, MID and MIQ methods give 83.35%, 83.33%, 80.22%, and 83.13% accuracy, respectively. In terms of memory requirements, the proposed SFS approach is much better since SFS only needs to record the indices of the discriminative features ($O(K)$) while PCA and LDA need additional memory storage for the transformation matrix ($O(K \times N)$), where K is the number of selected features and N is the number of all available features.

Table 6.10: Performance of different dimensionality reduction algorithms: PCA, LDA, MID, MIQ, and SFS via the pedestrian detection problems.

Algorithm	Number of selected features			
	100	300	500	1000
PCA	83.35	90.26	91.75	91.80
LDA	83.33	86.25	89.62	91.51
MID	80.22	84.62	87.92	88.63
MIQ	83.13	86.17	89.51	90.27
SFS	**87.55±1.22**	**90.95±0.41**	**91.72±0.61**	**91.81±0.63**

Figure 6.3 presents some examples of the features selected by the SFS and MID algorithms in six images chosen from the training data. The light regions

indicate the location of the selected features by each algorithm. Both methods select features from the centre of the input pattern and ignore the surrounding region, which does not contain useful features for the classification task. However, the region from which the features are extracted in the original image is bigger for SFS. This indicates that SFS considers more information from the original input pattern. Compared to SFS, the MID features tend to cluster in certain parts in the input pattern, which may not be as informative for the classification. For example, as we can see from the input image B, SFS almost covers the entire body of the pedestrian, while the MID method ignores the head region.

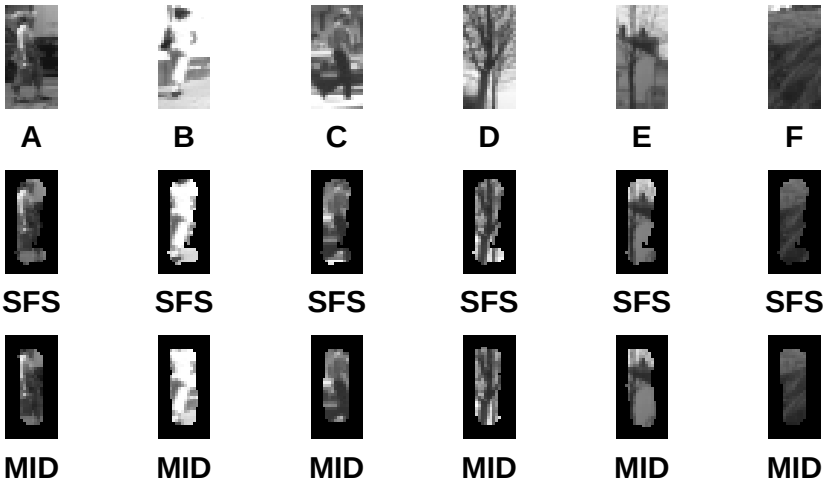


Figure 6.3: An example of different features (regions) selected by SFS and MID algorithms when the same number of features (500) is employed. The first row displays six input images (A-F) from the training data. The second and third rows show the corresponding highlighted regions from SFS and MID algorithm, respectively.

6.5 Conclusions

In this chapter, we have presented a novel algorithm for feature selection, termed sparse feature selection (SFS). In the proposed SFS algorithm, original features are used as the dictionary in the sparse representation. That is, each column from the dictionary is associated with a specific feature from the original data. Then the SFS

algorithm calculates the sparse solution, which is equivalent to selecting features that minimize the residual output error. Moreover, a measurement matrix is also introduced to reduce the computational cost.

Experimental results on various classification problems show that the proposed SFS method is superior to existing feature selection methods; it achieves better classification accuracy with fewer number of selected features. Furthermore, the proposed algorithm is tested with a hierarchical visual pattern recognition architecture for pedestrian detection. The results show that the proposed method achieves the same test accuracy as the original full architecture using only 25% of features.

Chapter 7

Conclusion and Future Work

Chapter contents

In this dissertation, a variety of algorithms are proposed based on sparse signal representation to solve different machine learning problems, such as structural optimization, supervised learning, and feature selection. The motivation is that many natural signals have a sparse structure, where only few nonzero elements are capable of representing the majority information conveyed by the target signal. Inspired by the sparse signal representation paradigm, we developed several robust machine learning algorithms for neural network pruning, supervised neural network learning, least squares support vector machines-based training, and feature selection. The remainder of the chapter presents a summary of major contributions of this thesis and proposes directions for future research.

7.1 Summary of research outcomes

The main contributions of the thesis can be summarized as follows:

- A comprehensive review of conventional sparse representation techniques

is conducted in Chapter 2. Based on the number of observed measurements, three sparse models are presented, namely single measurement vector (SMV), multiple measurement vector (MMV), and infinite measurement vector (IMV). The chapter also presents the common optimization algorithms used to solve the various sparse representation models. Dictionary learning methods are also discussed.

- A novel method for structural optimization is proposed for feed-forward neural networks in Chapter 3. The important network parameters are selected by finding the sparse representation for the network structure. The proposed network pruning algorithm was tested on eight datasets and its performance was compared with that of state-of-the-art methods. The proposed algorithm leads to faster convergence, smaller number of hidden units and the better generalization accuracy.

- In Chapter 4, a novel network training algorithm is introduced as an extension of the sparse structural optimization technique presented in Chapter 3. This method is capable of training the neural network and optimizing the architecture simultaneously. The dictionary learning-based algorithm is also employed to balance the trade-off between the training performance and architecture complexity. In general, the proposed training method achieves lower training accuracy compared to existing methods with much higher generalization accuracy for both regression and classification problems.

- In Chapter 5, a fast and efficient training algorithm is proposed for least squares support vector machines (LS-SVM). Significant support vectors from LS-SVM are selected during training. Experimental comparison with existing algorithms for function approximation and classification tasks was conducted. The proposed LS-SVM training method achieves a significant improvement in terms of classification accuracy using the same number of support vectors. On average, a classification accuracy of 84.17% is obtained

on the test sets from the six problems, which is much better than other sparse training approaches. Furthermore, the proposed algorithm with an average of 27.87% support vectors outperforms the standard LS-SVM training.

- Chapter 6 presents a sparsity-based algorithm to address the feature reduction problem. The method regards the approximation of the desired output as the sum of independent feature vectors. Therefore, the feature selection problem is converted to a sparse representation model. The experiment consists of two parts. First, the proposed sparse algorithm outperforms other conventional feature selection methods using the synthetic datasets; it always achieves the best classification accuracy. Meanwhile, the proposed algorithm also requires the fewest number of features (15 features) on average to solve all eight problems. Second, the proposed feature selection method was evaluated on a pedestrian detection problem. The results show that the proposed method achieves the same test accuracy as the original full architecture using only 25% of features.

In summary, the new paradigm of sparse signal representation is exploited to develop robust machine learning algorithms. The proposed sparsity-based algorithms are capable of representing salient information only using few parameters. This enables the design of high performance algorithms for structural (model) optimization, supervised training and feature selection. Experimental results show that the proposed sparse algorithms are superior to existing methods in terms of generalization ability, computational cost, and numerical stability.

7.2 Future work

Research on sparse representation is in its early stages, and progress in this areas is still on-going. The research work presented in this thesis has just investigated few machine learning areas using sparse representation models. There are many

possibilities for future research directions, a small sample of which is listed below:

1. **Online sparse learning:** We have applied the sparse model to optimize supervised learning algorithms to achieve a better generalization ability. All proposed methods discussed so far are off-line learning methods, in which the performance depends on the volume of available data. It would be very interesting to extend the proposed methods to online learning.

2. **Other learning tasks:** This work considers few supervised learning methods, i.e., neural network and least squares support vector machines. An obvious extension is to apply the sparse representation-based algorithms to other machine learning problems, such as classifier ensembles. Several classifiers can be modeled together and the final prediction is formed by combining predictions of those classifiers. Implementing sparsity-based ensemble methods can further improve the generalization ability of the machine learning algorithm. Furthermore, investigation of how the proposed learning framework can be used in conjunction with meta-learning algorithms such as Adaboost would also constitute an interesting extension of this work.

3. **Advanced optimization algorithms:** The simulation results indicate the efficiency of the sparsity-based technique. Clearly the performance of the sparse representation relies heavily on optimization algorithms employed to compute sparse solutions. Thus, the performance is expected to be improved when sophisticated recovery methods are developed. Furthermore, heuristic methods for dictionary learning can also be considered to improve the speed and solution accuracy of the proposed sparsity-based supervised learning algorithms.

4. **Real-world applications:** Finally, we believe that this thesis has opened an area of research for sparse representation and its wide applications. Potential

examples will be pattern recognition, parallel computing and information retrieval. For instance, in information retrieval, the target keyword can be represented using an overcomplete dictionary which consists of training samples. If sufficient training samples are available, the target keyword can be obtained using a linear combination of relatively fewer training samples. Seeking the sparse representation of the target keyword therefore automatically segregates the information of interest from irrelevant data.

References

[1] A. Bouzerdoum and R. Pinter, "Shunting inhibitory cellular neural networks: derivation and stability analysis," vol. 40, pp. 215–221, 1993.

[2] F. H. C. Tivive and A. Bouzerdoum, "Efficient training algorithms for a class of shunting inhibitory convolutional neural networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 541 –556, 2005.

[3] K. Methaprayoon, L. Lei, S. Rasmiddatta, J. R. Liao, and R. J. Ross, "Multistage artificial neural network short-term load forecasting engine with front-end weather forecast," *IEEE Transactions on Industry Applications*, vol. 43, no. 6, pp. 1410–1416, 2007.

[4] J. S. Lim, "Finding features for real-time premature ventricular contraction detection using a fuzzy neural network system," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 522–527, 2009.

[5] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[6] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in kernel methods: support vector learning*, pp. 169–184, 1998.

[7] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[8] J. A. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," vol. 2, pp. 757 –760, 2000.

[9] J. A. K. Suykens, *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[10] L. P. Chang and T. Pavlidis, "Fuzzy decision tree algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, no. 1, pp. 28 –35, 1977.

[11] L. Siegel and A. Bessey, "A decision tree procedure for voiced/unvoiced/mixed excitation classification of speech," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 53 – 56, 1980.

[12] M. Qing, H. Qiang, L. Ning, D. Xiang, and S. Li, "Crisp decision tree induction based on fuzzy decision tree algorithm," in *International Conference on Information Science and Engineering (ICISE)*, pp. 4811 –4814, 2009.

[13] A. Abdelhalim and I. Traore, "A new method for learning decision trees from rules," in *International Conference on Machine Learning and Applications*, pp. 693 –698, 2009.

[14] M. Yu, "The practice on using machine learning for network anomaly intrusion detection," in *International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 2, pp. 576 –581, 2011.

[15] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE Symposium on Security and Privacy (SP)*, pp. 305 –316, 2010.

[16] L. Dong, H. Chun, Y. Sheng, and C. Wei, "Initiated language learning machine with multi-media and speech-recognition techniques," in *International*

Conference on Machine Learning and Cybernetics (ICMLC), vol. 6, pp. 2985 – 2989, 2010.

[17] H. Xiao and D. Li, "Speech recognition, machine translation, and speech translation: A unified discriminative learning paradigm [lecture notes]," *IEEE Signal Processing Magazine*, vol. 28, no. 5, pp. 126 –133, 2011.

[18] J. Yang, A. Bouzerdoum, and L. P. Son, "Dimensionality reduction using compressed sensing and its application to a large-scale visual recognition task," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 1 –8, 2010.

[19] F. H. C. Tivive, A. Bouzerdoum, L. Son, and K. M. Iftekharuddin, "Adaptive hierarchical architecture for visual recognition," *Applied Optics*, vol. 49, no. 10, pp. 131–138, 2010.

[20] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[21] W. L. Chan, D. Charan, K. F. Kelly, and D. M. Mittleman, "A single-pixel terahertz imaging system based on compressive sensing," *Applied Physics Letters*, vol. 93, p. 121105, 2008.

[22] M. Mishali and Y. Eldar, "Blind multiband signal reconstruction: Compressed sensing for analog signals," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 993 –1009, 2009.

[23] ——, "From theory to practice: Sub-nyquist sampling of sparse wideband analog signals," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 375 –391, 2010.

[24] F. Ahmad, M. G. Amin, and S. A. Kassam, "A beamforming approach to stepped-frequency synthetic aperture through-the-wall radar imaging,"

in *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 24–27, 2005.

[25] C. Debes, M. G. Amin, and A. M. Zoubir, "Target detection in single- and multiple-view through-the-wall radar imaging," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 5, pp. 1349–1361, 2009.

[26] T. Dogaru and L. Calvin, "Recent investigations in sensing through the wall radar modeling," in *IEEE Antennas and Propagation Society International Symposium*, pp. 1–4, 2008.

[27] Y. S. Yoon and M. G. Amin, "Compressed sensing technique for high-resolution radar imaging," in *Proceeding of SPIE Signal Processing, Sensor Fusion, and Target Recognition XVII, Orlando, FL*, pp. 6958, 2008.

[28] M. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. Davies, "Sparse representations in audio and music: From coding to source separation," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995 –1005, 2010.

[29] L. Xin, "Parallel and distributed audio concealment using nonlocal sparse representations," in *IEEE International Conference on Multimedia and Expo*, pp. 775–778, 2007.

[30] P. Boufounos, G. Kutyniok, and H. Rauhut, "Sparse recovery from combined fusion frame measurements," *IEEE Transactions on Information Theory*, vol. 57, no. 6, pp. 3864–3876, 2011.

[31] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[32] D. L. Donoho, M. Elad, and V. N. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 6–18, 2006.

[33] Y. Li, S. I. Amari, A. Cichocki, and C. Guan, "Probability estimation for recoverability analysis of blind source separation based on sparse representation," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 3139 –3152, 2006.

[34] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1475–1490, 2004.

[35] M. Elad and A. M. Bruckstein, "A generalized uncertainty principle and sparse representation in pairs of bases," *IEEE Transactions on Information Theory*, vol. 48, no. 9, pp. 2558–2567, 2002.

[36] A. Feuer and A. Nemirovski, "On sparse representation in pairs of bases," *IEEE Transactions on Information Theory*, vol. 49, no. 6, pp. 1579–1581, 2003.

[37] S. O. Aase, J. H. Husoy, K. Skretting, and K. Engan, "Optimized signal expansions for sparse representation," *IEEE Transactions on Signal Processing*, vol. 49, no. 5, pp. 1087–1096, 2001.

[38] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2845–2862, 2001.

[39] J. Xi, Y. Pi, and R. Ming, "Sar image compression based on sparse representation," in *11th International Radar Symposium (IRS)*, pp. 1 –4, 2010.

[40] B. Huang and H. Song, "Spatiotemporal reflectance fusion via sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, no. 99, pp. 1 –10, 2012.

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

[42] H. Rauhut, K. Schnass, and P. Vandergheynst, "Compressed sensing and redundant dictionaries," *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2210–2219, 2008.

[43] M. Elad, "Optimized projections for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 55, no. 12, pp. 5695 –5702, 2007.

[44] H. Lou, L. Wei, and J. Wang, "Data compression based on compressed sensing and wavelet transform," in *IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, vol. 8, pp. 537 –542, 2010.

[45] M. Akcandakaya, N. Seunghoon, H. Peng, and R. Nezafat, "Compressed sensing with wavelet domain dependencies for coronary mri: A retrospective study," *IEEE Transactions on Medical Imaging*, vol. 30, no. 5, pp. 1090 –1099, 2011.

[46] M. Yaghoobi and M. Davies, "Compressible dictionary learning for fast sparse approximations," in *IEEE/SP 15th Workshop on Statistical Signal Processing*, pp. 662 –665, 2009.

[47] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[48] C. E. Shannon, "Communication in the presence of noise," *Proc. Institute of Radio Engineers*, vol. 37, p. 1021, 1949.

[49] B. V. Gowreesunker, A. H. Tewfik, V. A. Tadipatri, J. Ashe, G. Pellize, and R. Gupta, "A subspace approach to learning recurrent features from brain activity," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 3, pp. 240–248, 2011.

[50] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements

via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

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

[52] D. L. Donoho, "For most large undetermined systems of linear equations the minimal '1-norm near-solution is also the sparsest near-solution," *Communications on Pure and Applied Mathematics*, vol. 59, pp. 797 – 829, 2006.

[53] J. Haupt and R. Nowak, "Signal reconstruction from noisy random projections," *IEEE Transactions on Information Theory*, vol. 52, no. 9, pp. 4036–4048, 2006.

[54] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1," *Vision Res*, vol. 37, no. 23, pp. 3311–3325, 1997.

[55] S. Gleichman and Y. C. Eldar, "Blind compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6958–6975, 2011.

[56] D. P. Wipf, B. D. Rao, and S. Nagarajan, "Latent variable bayesian models for promoting sparsity," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6236–6255, 2011.

[57] S. G. Mallat and Z. Zheng, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[58] J. Chen and X. Huo, "Sparse representations for multiple measurement vectors (mmv) in an over-complete dictionary," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP '05)*, vol. 4, pp. 257–260, 2005.

[59] ——, "Theoretical results on sparse representations of multiple-measurement vectors," *IEEE Transactions on Signal Processing*, vol. 54, no. 12, pp. 4634–4643, 2006.

[60] S. F. Cotter, B. D. Rao, E. Kjersti, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2477–2488, 2005.

[61] M. Mishali and Y. C. Eldar, "Reduce and boost: Recovering arbitrary sets of jointly sparse vectors," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4692 –4702, 2008.

[62] R. Gribonval and P. Vandergheynst, "On the exponential convergence of matching pursuits in quasi-incoherent dictionaries," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 255–261, 2006.

[63] B. D. Rao, K. Engan, S. Cotter, J. Palmer, and K. Kreutz-Delgado, "Subset selection in noise based on diversity measure minimization," *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 760–770, 2003.

[64] J. B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions with application to arithmetic complexity and statistics," *Linear Algebra Its Appl*, vol. 18, no. 2, pp. 95–138, 1977.

[65] D. L. Donoho and M. Elad, "Optimally sparse representation in general (non-) dictionaries via l1 minimization," *Proceedings of the National Academy of Sciences, USA*, vol. 100, pp. 2197–2202, 2002.

[66] R. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118 –121, 2007.

[67] E. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21 –30, 2008.

[68] R. Justin, "Imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14 – 20, 2008.

[69] T. Blumensath and M. Davies, "Stagewise weak gradient pursuits," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4333 –4346, 2009.

[70] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600 –616, 1997.

[71] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 5302 –5316, 2009.

[72] M. Herman and T. Strohmer, "High-resolution radar via compressed sensing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2275–2284, 2009.

[73] B. D. Jeffs, "Sparse inverse solution methods for signal and image processing applications," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 1885 –1888, 1998.

[74] W. Lin and J. C. Preisig, "Estimation and equalization of rapidly varying sparse acoustic communication channels," in *OCEANS 2006*, pp. 1 –6, 2006.

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

[76] K. Engan, S. O. Aase, and J. Hakon, "Method of optimal directions for frame design," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2443–2446, 1999.

[77] M. Jafari and M. Plumbley, "Fast dictionary learning for sparse representations of speech signals," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1025 –1031, 2011.

[78] M. Yaghoobi, T. Blumensath, and M. E. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2178 –2191, 2009.

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

[80] I. Tosic and P. Frossard, "Dictionary learning: What is the right representation for my signal?" *IEEE Signal Processing Magazine*, vol. 28, pp. 27–38, 2011.

[81] P. Schmid-Saugeon and A. Zakhor, "Dictionary design for matching pursuit and application to motion-compensated video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 880 – 886, 2004.

[82] B. Mailhe, D. Barchiesi, and M. D. Plumbley, "INK-SVD: Learning incoherent dictionaries for sparse representations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3573 –3576, 2012.

[83] Q. Zhang and B. Lin, "Discriminative K-SVD for dictionary learning in face recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2691 –2698, 2010.

[84] Z. Jin, L. Zhe, and L. Davis, "Learning a discriminative dictionary for sparse coding via label consistent K-SVD," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1697 –1704, 2011.

[85] C. Ze and L. Jian, "IBP-SVD: A practical method for learning adaptive

dictionaries for image de-noising," in *International Conference on Wavelet Analysis and Pattern Recognition*, vol. 2, pp. 641 –646, 2007.

[86] Z. Jian, S. Li, Y. Xiao, and Z. Wen, "Sub clustering K-SVD: Size variable dictionary learning for sparse representations," in *IEEE International Conference on Image Processing (ICIP)*, pp. 2149 –2152, 2009.

[87] R. Mazhar and P. Gader, "EK-SVD: Optimized dictionary design for sparse representations," in *International Conference on Pattern Recognition*, pp. 1 –4, 2008.

[88] M. G. Jafari and M. D. Plumbley, "Fast dictionary learning for sparse representations of speech signals," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1025–1031, 2011.

[89] V. Cevher and A. Krause, "Greedy dictionary selection for sparse representation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 979–988, 2011.

[90] S. Ravishankar and Y. Bresler, "Mr image reconstruction from highly undersampled k-space data by dictionary learning," *IEEE Transactions on Medical Imaging*, vol. 30, no. 5, pp. 1028–1041, 2011.

[91] C. Rusu and B. Dumitrescu, "Stagewise k-svd to design efficient dictionaries for sparse representations," *IEEE Signal Processing Letters*, vol. 19, no. 10, pp. 631–634, 2012.

[92] R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, 2013.

[93] A. A. Liu, T. Hao, Z. Gao, Y. T. Su, and Z. X. Yang, "Sequential sparse representation for mitotic event recognition," *Electronics Letters*, vol. 49, no. 14, 2013.

[94] T. Aritake, H. Hino, and N. Murata, "Learning ancestral atom via sparse coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 4, pp. 586–594, 2013.

[95] T. Peleg, Y. C. Eldar, and M. Elad, "Exploiting statistical dependencies in sparse representations for signal recovery," *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2286–2303, 2012.

[96] K. Charalampous, I. Kostavelis, A. Amanatiadis, and A. Gasteratos, "Sparse deep-learning algorithm for recognition and categorisation," *Electronics Letters*, vol. 48, no. 20, pp. 1265–1266, 2012.

[97] J. F. Murray and K. Kreutz-Delgado, "An improved focuss-based learning algorithm for solving sparse linear inverse problems," in *Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 347–351, 2001.

[98] K. Engan, S. O. Aase, and J. H. Husoy, "Multi-frame compression: Theory and design," in *Signal Processing*, vol. 80, no. 10, pp. 2121–2140, 2000.

[99] M. Julien, B. Francis, P. Jean, and S. Guillermo, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[100] J. Chen and S. Lin, "A neural network approach-decision neural network (dnn) for preference assessment," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 34, no. 2, pp. 219–225, 2004.

[101] A. Zaknich, "Introduction to the modified probabilistic neural network for general signal processing applications," *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 1980 –1990, 1998.

[102] J. Ma, Y. Fan, and Y. Sheng, "A neural networks-based clustering collaborative filtering algorithm in e-commerce recommendation system," in *In-*

ternational Conference on Web Information Systems and Mining, pp. 616 –619, 2009.

[103] M. Mozer and P. Smolensky, "Skeletonization: a technique for trimming the fat from network via relevance assessment," in *Advances in Neura Information Processing Systems*, vol. 1, pp. 107–115, 1991.

[104] J. Sietsma and R. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67–79, 1991.

[105] E. J. Teoh, K. C. Tan, and X. Chen, "Estimating the number of hidden neurons in a feedforward network using the singular value decomposition," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1623–1629, 2006.

[106] P. Lauret, E. Fock, and T. Mara, "A node pruning algorithm based on a fourier amplitude sensitivity test method," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 273–293, 2006.

[107] H. Xing and B. Hu, "Two-phase construction of multilayer perceptrons using information theory," *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 715–721, 2009.

[108] T. J. Cholewo and J. M. Zurada, "Sequential network construction for time series prediction," in *International Conference on Neural Networks*, vol. 4, pp. 2034–2038, 1997.

[109] A. L. P. Tay and J. M. Zurada, "The hierarchical fast learning artificial neural network: An autonomous platform for hierarchical neural network construction," *IEEE Transactions on Neural Networks,*, vol. 18, no. 6, pp. 1645 –1657, 2007.

[110] A. Krzyzak and T. Linder, "Radial basis function networks and complexity regularization in function learning," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 247 –256, 1998.

[111] S. Razavi and B. A. Tolson, "A new formulation for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1588 –1598, 2011.

[112] S. P. Adhikari, Y. Changju, K. Hyongsuk, and L. O. Chua, "Memristor bridge synapse-based neural network and its learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1426–1435, 2012.

[113] G. Wei, "New evolutionary neural networks," in *International Conference on Neural Interface and Control*, pp. 167 – 171, 2005.

[114] A. A. Salah and Y. Al-Salqan, "Meta-learning evolutionary artificial neural networks: by means of cellular automata," in *International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 1, pp. 186 –192, 2005.

[115] S. Haykin, "Neural networks: A comprehensive foundation," in *New Jersey: Prentice-Hall, Inc.*, 1999.

[116] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 5, pp. 533–536, 1986.

[117] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the rprop algorithm," in *IEEE International Conference on Neural Networks*, pp. 586–591, 1993.

[118] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing System*, vol. 2, pp. 524–532, 1990.

[119] Y. F. Hu and C. Storey, "Global convergence result for conjugate gradients methods," *Journal of Optimization Theory Application*, vol. 71, pp. 399–405, 1991.

[120] P. Van Der Smagt P, "Minimisation methods for training feedforward neural networks," *Neural Networks*, vol. 7, no. 1, pp. 1–11, 1994.

[121] N. Ampazis and S. J. Perantonis, "Two highly efficient second-order algorithms for training feedforward networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1064–1074, 2002.

[122] I. Jordanov and A. Georgieva, "Neural network learning with global heuristic search," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 937 –942, 2007.

[123] J. Peng, L. Kang, and G. W. Irwin, "A new jacobian matrix for optimal learning of single-layer neural networks," *IEEE Transactions on Neural Networks*, vol. 19, no. 1, pp. 119 –129, 2008.

[124] G. Wei, "Study on new evolutionary neural network," in *International Conference on Machine Learning and Cybernetics*, vol. 2, pp. 1287 – 1292 , 2003.

[125] ——, "Evolutionary neural network based on new ant colony algorithm," in *International Symposium on Computational Intelligence and Design*, vol. 1, pp. 318 –321, 2010.

[126] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, vol. 2, pp. 598–605, 1990.

[127] B. Hassibi and D. G. Stork, "second-order derivatives for network pruning: optimal brain surgeon," in *Advances in Neural Information Processing Systems*, vol. 5, pp. 164–171, 1993.

[128] M. Hagiwara, "Removal of hidden units and weights for back propagation networks," in *Proceedings of 1993 International Joint Conference on Neural Networks*, vol. 1, pp. 351–354, 1993.

[129] R. Reed, "Pruning algorithms-a survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[130] R. Parekh, J. Yang, and V. Honavar, "Constructive neural-network learning algorithms for pattern classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 436–451, 2000.

[131] J. O. Moody and P. J. Antsaklis, "The dependence identification neural network construction algorithm," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 3–15, 1996.

[132] D. White and P. Ligomenides, "Gannet: A genetic algorithm for optimizing topology and weights in neural network design," in *International Workshop on Artificial Neural Networks, in New Trends in Neural Computation*, pp. 332–327, 1993.

[133] L. Prechelt, "Proben1-a set of neural networks benchmark problems and benchmarking rules." 1994.

[134] A. Asuncion and D. J. Newman, "Uci machine learning repository," 2007.

[135] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Network*, vol. 2, no. 5, pp. 359–366, 1989.

[136] P. Fuangkhon and T. Tanprasert, "An incremental learning algorithm for supervised neural network with contour preserving classification," in *International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009*, vol. 02, pp. 740 –743, 2009.

[137] T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Supervised neural network modeling: An empirical investigation into learning from imbalanced data with labeling errors," *IEEE Transactions on Neural Networks*, vol. 21, no. 5, pp. 813 –830, 2010.

[138] K. Ristovski, S. Vucetic, and Z. Obradovic, "Uncertainty analysis of neural-network-based aerosol retrieval," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 2, pp. 409–414, 2012.

[139] S. Ansari, I. Shafi, J. Ahmad, and S. Ismail Shah, "Neural network-based approach for the non-invasive diagnosis and classification of hepatotropic viral disease," *IET Communications*, vol. 6, no. 18, pp. 3265–3273, 2012.

[140] M. Rizwan, M. Jamil, and D. P. Kothari, "Generalized neural network approach for global solar energy estimation in india," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 3, pp. 576–584, 2012.

[141] S. A. Sadrossadat, C. Yazi, and Z. Qi-Jun, "Parametric modeling of microwave passive components using sensitivity-analysis-based adjoint neural-network technique," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 5, pp. 1733–1747, 2013.

[142] S. van den Dries and M. A. Wiering, "Neural-fitted td-leaf learning for playing othello with structured neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 11, pp. 1701–1713, 2012.

[143] S. Sahami and M. G. Shayesteh, "Bi-level image compression technique using neural networks," *IET Image Processing*, vol. 6, no. 5, pp. 496–506, 2012.

[144] S. Becker, "Unsupervised learning procedures for neural networks," *International Journal of Neural Systems*, vol. 2, pp. 17–33, 1991.

[145] A. Meyer and V. Thummler, "Local and global stability analysis of an unsupervised competitive neural network," *IEEE Transactions on Neural Networks*, vol. 19, no. 2, pp. 346 –351, 2008.

[146] C. Dong, Z. Li, and W. Ju, "Spatio temporal adaptation in the unsupervised

development of networked visual neurons," *IEEE Transactions on Neural Networks*, vol. 20, no. 6, pp. 992 –1008, 2009.

[147] U. Angadi and M. Venkatesulu, "Structural scop superfamily level classification using unsupervised machine learning," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 2, pp. 601 –608, 2012.

[148] A. Rahman and B. Verma, "Novel layered clustering-based approach for generating ensemble of classifiers," *IEEE Transactions on Neural Networks*, vol. 22, no. 5, pp. 781 –792, 2011.

[149] B. Verma and A. Rahman, "Cluster-oriented ensemble classifier: Impact of multicluster characterization on ensemble classifier learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 605 –618, 2012.

[150] Y. Qi, J. Vance, and S. Jagannathan, "Control of nonaffine nonlinear discrete-time systems using reinforcement-learning-based linearly parameterized neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 4, pp. 994 –1001, 2008.

[151] H. Pingan and S. Jagannathan, "Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 425 –436, 2007.

[152] X. Ma and K. Likharev, "Global reinforcement learning in neural networks," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 573 –577, 2007.

[153] P. Shih, B. Kaul, S. Jagannathan, and J. Drallmeier, "Reinforcement-learning-based output-feedback control of nonstrict nonlinear discrete-time systems with application to engine emission control," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 5, pp. 1162 –1179, 2009.

[154] R. Battiti, "Accelerated backpropagation learning: Two optimization methods," *Complex Systems*, vol. 3, pp. 331–342, 1989.

[155] H. Hsi-Chin, C. C. Li, S. Mingui, and R. J. Sclabassi, "An adaptive training algorithm for back-propagation neural networks," in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1049–1052, 1992.

[156] B. Verma, "Fast training of multilayer perceptions," *IEEE Transactions on Neural Networks*, vol. 8, pp. 1314–1321, 1997.

[157] S. L. Hung and H. Adeli, "A parallel genetic/neural network learning algorithm for mimd shared memory machines," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 900 –909, 1994.

[158] M. Elbashir and J. Wang, "Protein secondary structure prediction: Speeding up conjugate gradient neural network," in *International Conference on Biomedical Engineering and Informatics (BMEI)*, vol. 6, pp. 2347 –2351, 2010.

[159] B. Wilamowski and Y. Hao, "Neural network learning without backpropagation," *IEEE Transactions on Neural Networks*, vol. 21, no. 11, pp. 1793 –1803, 2010.

[160] D. J. C. MacKay, "A practical bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, pp. 448–472, 1992.

[161] B. D. Ripley, "Pattern recognition and neural networks." Cambridge, MA: Cambridge Univ. Press, 1996.

[162] K. Tin and Y. Dit, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 630–645, 1997.

[163] E. Cant-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems,"

IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 4, pp. 915–927, 2005.

[164] M. Zhang and A. A. Sawchuk, "Human daily activity recognition with sparse representation using wearable sensors," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 553–560, 2013.

[165] P. Gurram and K. Heesung, "Sparse kernel-based ensemble learning with fully optimized kernel parameters for hyperspectral classification problems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 787–802, 2013.

[166] D. Geebelen, J. A. K. Suykens, and J. Vandewalle, "Reducing the number of support vectors of svm classifiers using the smoothed separable case approximation," *IEEE Transactions on Neural Networks*, vol. 23, no. 4, pp. 682–688, 2012.

[167] Q. Sami, T. Linmi, S. Fu, and Y. Shi, "A fast and robust sparse approach for hyperspectral data classification using a few labeled samples," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 6, pp. 2287–2302, 2012.

[168] P. Gurram, K. Heesung, and T. Han, "Sparse kernel-based hyperspectral anomaly detection," *IEEE Geoscience and Remote Sensing Letters,*, vol. 9, no. 5, pp. 943–947, 2012.

[169] L. Yuan, C. Lin, and W. Zhang, "Improved sparse least-squares support vector machine classifiers," vol. 69, no. 13, pp. 1655–1658, 2006.

[170] B. J. de Kruif and T. J. A. De Vries, "Pruning error minimization in least squares support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 696–702, 2003.

[171] M. Espinoza, J. Suykens, and B. Moor, "Fixed-size least squares support vector machines: a large scale application in electrical load forecasting," *Computer Management Science*, vol. 3, pp. 113–129, 2006.

[172] K. Suykens, "Nonlinear modelling and support vector machines," *IEEE IMTC Instrumentation and Measurement Technology Conference, Hungary*, pp. 287–294, 2001.

[173] C. Wei, J. Chong, and S. Keerthi, "An improved conjugate gradient scheme to the solution of least squares svm," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 498 –501, 2005.

[174] L. Yanhua, H. Chunhua, and L. Bingjun, "An efficient computational model for ls-svm and its applications in time series prediction," in *International Conference on Computer Science and Education (ICCSE)*, pp. 467 –470, 2010.

[175] A. Kuh and P. De Wilde, "Comments on "pruning error minimization in least squares support vector machines," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 606 –609, 2007.

[176] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. Moor, "A comparison of pruning algorithms for sparse least squares support vector machines," in *Neural Information Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, vol. 3316, pp. 1247–1253, 2004.

[177] J. Yang, D. Zhang, and A. F. Frangi, "Two-dimensional PCA: a new approach to appearance-based face representation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[178] H. Abdi and L. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, pp. 433–459, 2010.

[179] I. T. Jolliffe, "Principal Component Analysis," 2002.

[180] M. Ahdesmaki and K. Strimmer, "Feature selection in omics prediction problems using cat scores and false nondiscovery rate control," pp. 503–519, 2010.

[181] A. M. Martinez and A. C. Kak, "PCA versus LDA," pp. 228–233, 2001.

[182] B. Bin, Z. Guang, S. Jia, and Y. Sha, "Robust image analysis with sparse representation on quantized visual features," *IEEE Transactions on Image Processing*, vol. 22, no. 3, pp. 860–871, 2013.

[183] L. W. Zhong and J. T. Kwok, "Efficient sparse modeling with automatic feature grouping," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1436–1447, 2012.

[184] G. S. V. S. Sivaram and H. Hermansky, "Sparse multilayer perceptron for phoneme recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 23–29, 2012.

[185] H. Xiong, Z. Pan, and Z. Ye, "Sparse spatio-temporal representation with adaptive regularized dictionary learning for low bit-rate video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 4, pp. 710–728, 2013.

[186] L. Luo and W. Wei, "Feature extraction and representation for distributed multi-view human action recognition," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 2, pp. 145–154, 2013.

[187] X. F. Huang, D. Chen, and N. Partha, *Laplacian Score for Feature Selection*. MIT Press, 2005.

[188] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning*, pp. 249–256, 1992.

[189] R. Gilad, A. Navot, and N. Tishby, "Margin based feature selection-theory and algorithms," in *Advances in Neural Information Processing Systems*, pp. 43–50, 2004.

[190] H. Peng, L. Fulmi, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[191] W. Zuo and D. Zhang, "BDPCA plus LDA: a novel fast feature extraction technique for face recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, pp. 946–953, 2006.

[192] B. ZMojaradi, H. Abrishami-Moghaddam, M. J. V. Zoej, and R. P. W. Duin, "Dimensionality reduction of hyperspectral data via spectral feature extraction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, pp. 2091–2105, 2009.

[193] S. Gao and I. W. H. Tsang, "Laplacian sparse coding, hypergraph laplacian sparse coding, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 92–104, 2013.

[194] X. Dong, H. Yi, Z. Zeng, and X. Xin, "Human gait recognition using patch distribution feature and locality-constrained group sparse representation," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 316–326, 2012.

[195] Z. Lai, L. Wong, J. Zhong, Y. Jian, and X. Yong, "Sparse approximation to the eigensubspace for discrimination," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 12, pp. 1948–1960, 2012.

[196] Z. Zheng, W. Lei, L. Huan, and Y. Jie, "On similarity preserving feature selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 3, pp. 619–632, 2013.

[197] S. Gao, I. W. Tsang, and L. Chia, "Sparse representation with kernels," *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 423–434, 2013.

[198] G. Gokhan, C. Zehra, and Y. Lei, "Stable and accurate feature selection," *Machine Learning and Knowledge Discovery in Databases*, vol. 5781, pp. 455–468, 2009.

[199] L. Chen, "Gabor-based kernel PCA with fractional power polynomial models for face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 572 –581, 2004.

[200] D. Shi, H. Rui, and T. Wei, "Enhanced principal component using polar coordinate PCA for stereo audio coding," in *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 628 –633, 2012.

[201] F. R. K. Chung, *Spectral Graph Theory*. Regional Conference Series in Mathematics. Providence, RI: American Mathematical Society (AMS)., 1997.

[202] S. Munder and D. Gavrila, "An experimental study on pedestrian classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1863–1868, 2006.