BASIC THRESHOLDING CLASSIFICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET ALTAN TOKSÖZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MARCH 2016

Approval of the thesis:

BASIC THRESHOLDING CLASSIFICATION

submitted by **MEHMET ALTAN TOKSÖZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Gönül Turhan Sayan _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. İlkay Ulusoy _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Çağatay Candan
Electrical and Electronics Engineering Department, METU _____

Assoc. Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Department, METU _____

Assoc. Prof. Dr. Alptekin Temizel
Graduate School of Informatics, METU _____

Assoc. Prof. Dr. Selim Aksoy
Computer Engineering Department, Bilkent University _____

Assoc. Prof. Dr. Pınar Duygulu Şahin
Computer Engineering Department, Hacettepe University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MEHMET ALTAN TOKSÖZ

Signature :

ABSTRACT

BASIC THRESHOLDING CLASSIFICATION

Toksöz, Mehmet Altan
Ph.D., Department of Electrical and Electronics Engineering
Supervisor : Assoc. Prof. Dr. İlkay Ulusoy

March 2016, 128 pages

In this thesis, we propose a light-weight sparsity-based algorithm, basic thresholding classifier (BTC), for classification applications (such as face identification, hyper-spectral image classification, etc.) which is capable of identifying test samples extremely rapidly and performing high classification accuracy. Originally BTC is a linear classifier which works based on the assumption that the samples of the classes of a given dataset are linearly separable. However, in practice those samples may not be linearly separable. In this context, we also propose another algorithm namely kernel basic thresholding classifier (KBTC) which is a non-linear kernel version of the BTC algorithm. KBTC can achieve promising results especially when the given samples are linearly non-separable. For both proposals, we introduce sufficient identification conditions (SICs) under which BTC and KBTC can identify any test sample in the range space of a given dictionary. By using SICs, we develop parameter estimation procedures which do not require any cross validation. Both BTC and KBTC algorithms provide efficient classifier fusion schemes in which individual classifier outputs are combined to produce better classification results. For instance, for the application of face identification, this is done by combining the residuals having different random projectors. For spatial applications such as hyper-spectral image classification, the fusion is carried out by incorporating the spatial information, in which the output residual maps are filtered using a smoothing filter. Numerical results on publicly available face and hyper-spectral datasets show that our proposal

outperforms well-known support vector machines (SVM)-based techniques, multinomial logistic regression (MLR)-based methods, and sparsity-based approaches like l_1-minimization and simultaneous orthogonal matching pursuit (SOMP) in terms of both classification accuracy and computational cost.

Keywords: basic thresholding classifier (BTC), kernel basic thresholding classifier (KBTC), sufficient indentification condition (SIC), face identification, hyper-spectral image classification, support vector machines (SVM), multinomial logistic regression (MLR), simultaneous orthogonal matching pursuit (SOMP)

ÖZ

TEMEL EŞİKLEME SINIFLANDIRMA

Toksöz, Mehmet Altan
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. İlkay Ulusoy

Mart 2016 , 128 sayfa

Bu tezde, bazı sınıflandırma uygulamaları (yüz tanıma, hiper-spektral imge sınıflandırma vb.) için yüksek sınıflandırma doğruluğu ile test numunelerini son derece hızlı bir şekilde sınıflandırabilen seyreklik tabanlı temel eşikleme sınıflandırıcı (BTC) önerilmektedir. Orjinalinde BTC doğrusal bir sınıflandırıcı olup verilen bir veri kümesinin sınıflarına ait örneklerin doğrusal olarak ayırt edilebilir varsayımı üzerine çalışmaktadır. Ancak pratikte bu örnekler doğrusal olarak her zaman ayırt edilemeyebilmektedir. Bu kapsamda, BTC'nin doğrusal doğrusal olmayan çekirdek versiyonu KBTC de ayrıca takdim edilmektedir. Özellikle doğrusal bir şekilde ayırt edilemeyen örnekler verildiğinde KBTC gelecek vaat eden sonuçlar elde edebilmektedir. Verilen bir sözlüğün değer kümesi uzayında bulunan herhangi bir test örneği, takdim edilen yeterli tanıma koşulu (SIC) altında sınıflandırılabilmektedir. Bu koşul kullanılarak çapraz doğrulama gerektirmeyen parametre kestirim yöntemleri geliştirilmiştir. BTC ve KBTC algoritmaları, sınıflandırma doğruluğunu arttırmak için bireysel sınıflandırıcı çıkışlarını birleştirerek füzyon tekniklerinin etkili bir şekilde uygulanmasını sağlamaktadır. Örneğin bu işlem, yüz tanıma uygulamaları için farklı rasgele projektörlere sahip sınıflandırıcıların çıkıştaki artık değerlerinin birleştirilmesiyle yapılır. Füzyon işlemi, hiper-spektral imge sınıflandırma gibi uzamsal uygulamalarda ise uzamsal bir filtre kullanılarak çıkıştaki artık değer haritalarının düzleştirilmesiyle yapılmaktadır. Bazı yaygın yüz ve hiper-spektral veri setleri kullanılarak gerçekleştirilen deneylerde, önerdiğimiz BTC ve KBTC algoritmaları, sınıflandırma doğruluğu

ve maliyeti açısından, tanınmış destek vektör makineleri (SVM) tabanlı teknikler, çok terimli lojistik regresyon tabanlı metotlar, l_1-minimizasyonu ve eşzamanlı dik eşleştirme takibi (SOMP) gibi seyreklik tabanlı yaklaşımlardan daha iyi sonuçlar vermektedir.

Anahtar Kelimeler: temel eşikleme sınıflandırıcı (BTC), çekirdek temel eşikleme sınıflandırıcı (KBTC), yeterli tanıma şartı (SIC), yüz tanıma, hiper-spektral imge sınıflandırma, destek vektör makineleri (SVM), çok terimli lojistik regresyon (MLR), eşzamanlı dik eşleştirme takibi (SOMP)

To my family and people who are reading this page

ACKNOWLEDGMENTS

I gratefully thank my supervisor Assoc. Prof. Dr. İlkay Ulusoy, who has shared her expertise with me throughout the development of the thesis, for her support, constructive criticism and supervision. I greatly appreciate her patience and belief on me and also her guidance related to my academic life.

I express my gratitude to thesis monitoring committee members Prof. Dr. Çağatay Candan and Assoc. Prof. Dr. Alptekin Temizel. They have closely monitored the progress of the study and the quality of the thesis has been remarkably improved with their valuable comments and fruitful discussions with them. In particular, I would like to say how lucky I was to meet Dr. Candan who has taught a lot of advanced subjects in signal analysis and processing course.

I would like to thank thesis jury members Assoc. Prof. Dr. Selim Aksoy and Assoc. Prof. Dr. Pınar Duygulu Şahin for sparing their time and effort in the evaluation of the thesis.

I would like to express my gratitude to Tübitak Bilgem İltaren and also my colleagues there for all the support on the study.

I would also like to thank Joel Tropp, Xudong Kang, Jun Li, Julien Mairal, Laurens van der Maaten, Chih-Chung Chang, and Dani Lischinski for kindly providing the source codes and toolboxes used in this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AA	Average Accuracy
ANN	Artificial Neural Networks
ASOMP	Adaptive Simultaneous Orthogonal Matching Pursuit
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
BP	Back Propagation
BPNN	Back Propagation Neural Networks
BT	Basic Thresholding
BTC	Basic Thresholding Classifier
COMP	Cholesky-based Orthogonal Matching Pursuit
FFBPNN	Feed Forward Back Propagation Neural Networks
GF	Guided Filter
HSI	Hyper-spectral Image
KBTC	Kernel Basic Thresholding Classifier
LDA	Linear Discriminant Analysis
LORSAL	Logistic Regression via Splitting and Augmented Lagrangian
MASR	Multi-scale Adaptive Sparse Representation
MLL	Multilevel Logistic
MLR	Multinomial Logistic Regression
NN	Nearest Neighbor
OA	Overall Accuracy
OAA	One-Against-All
OAO	One-Against-One
OMP	Orthogonal Matching Pursuit
PCA	Principal Component Analysis
RBF	Radial Basis Functions
ROC	Receiver Operating Characteristics
ROSIS	Reflective Optics System Imaging Spectrometer
SCI	Sparsity Concentration Index

SIC	Sufficient Identification Condition
SOMP	Simultaneous Orthogonal Matching Pursuit
SRC	Sparse Representation-based Classification
SVM	Support Vector Machines
WLS	Weighted Least Squares
WSOMP	Weighted Simultaneous Orthogonal Matching Pursuit

CHAPTER 1

INTRODUCTION

The term classification is usually referred to as assigning objects into different categories. It can also be interpreted as assigning predefined class labels to each object under testing. Those objects could be human faces, fingerprints, characters, digits, signals, documents, diseases, proteins, genes, speech, emotions, galaxies, objects in a scene, hyper-spectral pixels, etc.

The classification / recognition / identification process can easily be performed by human brains. On the other hand, it is not quite easy for the artificial classifiers or recognizers because the process involves quite complex stages such as sensing, preprocessing, feature extraction, dimension reduction, and decision making. Fig. 1.1 shows a typical classification scheme. Sensors such as camera, microphone or other type of acquisition devices are able to capture high quality raw data in today's technology. Although the sensing and preprocessing stages are performed easily, the feature extraction stage could be problematic. Perhaps, the performance of a classifier is mostly affected by the quality of the features. A good feature is considered to be discriminative, informative, robust, reliable, independent, invariant to scale and transformation.

Figure 1.1: A typical classification scheme

Features in a classification problem could be color, shape, texture, size, sound, intensity of a pixel, gender, height, weight, measured frequency value, etc. They are generally mapped to real line and stored in the vectors. An N-dimensional feature vector can also be interpreted as a point in an N-dimensional feature space. Sometimes using all of the extracted features does not improve the classification process. Instead, it may cause performance degradation in terms of accuracy and speed. To overcome this problem, an information reduction stage namely dimension reduction is integrated to the whole system.

The final stage of a classification process, decision making, is performed by a classifier. if high quality features are passed to the classifier, the objects under test can easily be classified. However, in real world applications, extracting good features from the raw data is not always possible. In this case, in order to increase classification accuracy, we need to design sophisticated classifiers. These kinds of classifiers not only perform high classification accuracy but also act quickly. They also require few processing steps and memory.

In the literature, variety of classifiers have been proposed addressing the classification problem. They can be divided into two main categories namely parametric and non-parametric approaches. One example of parametric techniques is the Bayesian decision theory in which a priori probability densities are known for each class. Those densities are converted to a posteriori probabilities and final decisions are made based on them [1]. Unfortunately, in practice, those densities are generally unknown. Therefore, it is inevitable to use non-parametric approaches. There are various methods in this category such as density or parameter estimation-based techniques in which the underlying densities and parameters are estimated based on the provided training data. Sometimes decision boundaries are formed using the training data, which is referred to as learning. If the class labels of the training data are known, then this kind of learning is called supervised learning. If there is no labeled data, in this case, the learning process becomes unsupervised learning or clustering.

Recently, sparse representation-based classification algorithms in the category of supervised techniques have got significant attention [2, 3, 4, 5, 6, 7, 8]. Over the last two decades, tremendous research activities have been observed in the area of sparse

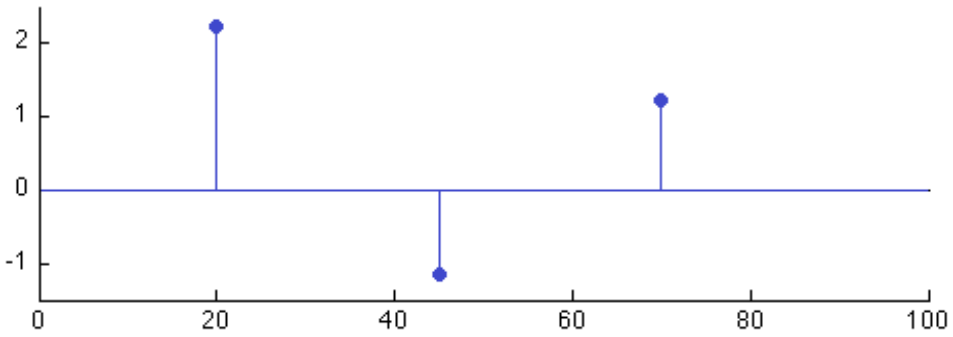


Figure 1.2: A sparse signal

signal representation and compressed sensing [9, 10, 11, 12, 13]. This is mainly because of the fact that significant portion of the signals in the nature are sparse, that is, most of the components of them are zero (Fig. 1.2). Sparsity provides that real world signals can be represented by the combinations of a few basis vectors. For instance, a typical image can be successfully compressed via JPEG technique which works based on the assumption that an image can be represented by a few discrete cosine basis.

While the nice properties of compressible or sparse signals in the area of signal processing are inspiring, the computer vision community is more interested in the semantic information of a signal rather than compressed sensing and compact representation. For instance, state-of-the-art results have been achieved based on sparse representation in image de-noising and in-painting [14, 15, 16], image super-resolution [17, 18, 19, 20], object tracking [4, 21, 22], face recognition [2, 6, 23, 24, 25, 26], image smoothing [27], image classification [28, 7, 29, 30, 31], etc.

Although sparse representation-based techniques achieve promising results, the underlying framework, sparse signal recovery or reconstruction, is not an easy task. Until now, various algorithms have been proposed addressing the problem of sparse reconstruction. Convex relaxation and l_1-minimization-based techniques [10, 32, 33, 34], greedy approaches [35, 36, 37, 38], Bregman iteration-based procedures [39, 40],

and linear programming-based [41] methods have been deeply investigated. Unfortunately, while some of those approaches are extremely costly, the others are highly sensitive to noise and corruption. As we stated previously, most of the classification applications involve noisy and corrupted features such as face recognition under illumination variations, noise, and corruption. In some applications such as hyperspectral image classifications, the problems involve classification of hundred thousands of pixels, which requires extremely cost effective classifiers.

In this thesis, we propose two sparsity-based, light-weight, and easy-to-implement classification algorithms which achieve state-of-the-art results in terms of both accuracy and computational cost. While the first algorithm addresses the applications in which the samples of data are linearly separable, the other one refers the problems involving non-linearly separable data. The following section briefly presents the organization of the thesis.

1.1. Outline

This thesis provides the following contributions to the field of computer vision and classification:

- In Chapter 2, we briefly discuss some of the common existing techniques in classification including non-parametric approaches, neural network-based methods, and sparsity-based classifiers.
- In Chapter 3, we introduce the basic thresholding classification (BTC) algorithm and give the construction of it step by step. We also provide necessary guidance for parameter estimation.
- Chapter 4 introduces the kernel basic thresholding classification (KBTC) algorithm which achieves promising results in the problems especially involving non-linearly separable data. We present full guidance of the parameter estimation steps by utilizing the propositions related to the algorithm.
- In Chapter 5, the performance of the BTC algorithm is compared to those of the state-of-the-art sparsity-based techniques in the application of face identi-

fication. We also provide an effective classification fusion technique in which individual classifiers are combined to achieve better classification performance. At the end of the chapter, an efficient validation scheme is presented in order to reject invalid test samples.

- Chapter 6 compares the performance of the BTC technique with those of the powerful non-linear kernel methods such as SVM in the application of hyper-spectral image classification. This chapter also introduces a spatial-spectral framework in which the residual maps produced by the proposed algorithms are smoothed using edge preserving filtering techniques. This intermediate step extremely improves the classification accuracy.

- In Chapter 7, we compare the performance of the KBTC algorithm with those of the state-of-the-art non-linear kernel approaches as well as the linear BTC in hyper-spectral image classification. This chapter shows how the non-linear similarity-based KBTC achieves significant performance improvements over the linear ones as well as the other techniques. We also provide fixed training sets by which efficient comparison of the algorithms is performed.

- Finally, Chapter 8 concludes this thesis by presenting summary and future directions.

Please note that the Chapter 5, 6, and 7 are based on the following papers:

M. A. Toksoz and I. Ulusoy, "Hyperspectral Image Classification via Kernel Basic Thresholding Classifier", IEEE Transactions on Geoscience and Remote Sensing, Manuscript submitted for publication, 2016.

M. A. Toksoz and I. Ulusoy, "Hyperspectral Image Classification via Basic Thresholding Classifier", IEEE Transactions on Geoscience and Remote Sensing, Manuscript accepted for publication, 2016.

M. A. Toksoz and I. Ulusoy, "Classification via ensembles of basic thresholding classifiers," IET Computer Vision, Manuscript accepted for publication, 2015.

1.2. Notations

Throughout the thesis we will use some notations which are described as follows:

- We will use capital letters for matrices and sets. For instance, a given dictionary consisting of training samples will be shown by the matrix A. Exceptionally, the threshold parameter, the number of features, and the number of classes will be represented by M, B, and C, respectively.
- Small letters will be used to represent vectors. In classification applications, typically the small letter y is used to describe a given test sample which is a vector containing features. Exceptionally, the small letters i, j, k, m, and n will be used for indexes.
- The ith column of a dictionary A will be shown by $A(i)$ which corresponds to a training sample. If we want to extract the sub matrix whose indexes in Λ, we will use the notation $A(\Lambda)$.
- Small Greek letters such as α and γ will denote the constants.
- The notation A_i will represent the sub matrix which contains only the training samples of the ith class. A sample belonging to ith class will be denoted by a_i.
- Finally, the range space of a vector will be shown by $\mathcal{R}(.)$ and the notations $\|.\|_0$, $\|.\|_1$, and $\|.\|_2$ will represent the l_0, l_1, and l_2 norms, respectively.

CHAPTER 2

EXISTING METHODS

In this chapter, we briefly discuss commonly used classification algorithms in the literature.

2.1. Nearest Neighbor Classifier

One of the most intuitive and primitive methods in the class of non-parametric techniques is the nearest neighbor (NN) classifier. The classification is simply performed based on the Euclidean distances between testing and training samples in the feature space. It has been shown in [42] that when the size of the training data goes to infinity, its error rate does not exceed the double Bayes error rate.

NN classifier is commonly used in the classification applications such as face recognition. Usually features are transformed before using it. One of the most popular subspace-based transformation methods is the principal component analysis (PCA) approach [43, 44]. In PCA, high dimensional data is projected to a lower dimensional subspace in which first principal component carries most discriminative information. Another popular approach of the subspace-based algorithms is the linear discriminant analysis (LDA) method, also known as Fisher's LDA [45]. The goal of the LDA is to seek a projection to a lower dimensional subspace such that maximum separability is obtained between samples of different classes. This is achieved by maximizing the ratio $\frac{|V^T S_b V|}{|V^T S_w V|}$, where S_b is the between-class scatter matrix, S_w is the within-class scatter matrix, and V is the projection which can be obtained from the eigenvectors of $S_w^{-1} S_b$. Although LDA is a powerful dimensionality reduction method, it encounters

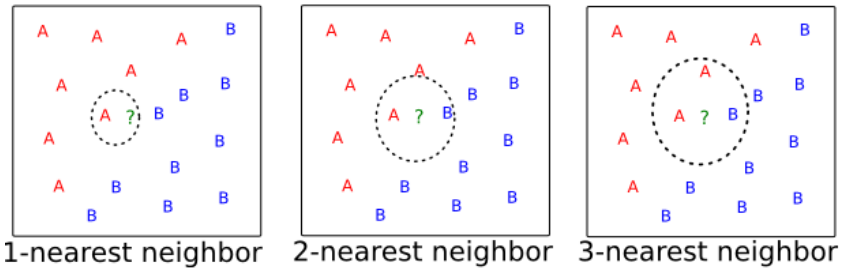


Figure 2.1: Nearest neighbors of a test instance

a common high dimensionality problem in the classification applications. In other words, the size of the feature vectors is generally larger than the number of training samples. This makes the S_w matrix singular, which is usually called small sample size problem (SSS).

Although NN-based approaches work well under normal conditions, they are highly sensitive to corruption and noise in the features. A more sophisticated version of NN classifier is the k-NN technique which executes majority voting among k nearest training samples [46]. As shown in [47], its performance does not exceed those of linear and non-linear SVM classifiers. First, second, and third nearest neighbors of a test instance could be seen in Fig. 2.1.

2.2. Neural Networks

Over the last two decades, artificial neural networks (ANN) has become more important in computer vision and pattern recognition. One of the most popular ANN-based methods is the back-propagation (BP) algorithm which is a gradient based method [48]. At every epoch, the algorithm adjusts the connection weights in the network such that the difference of the desired and actual output vector is minimized. BPNN was successfully applied to hand written digit recognition [49, 50]. Recently, FF-BPNN has been applied to face recognition using PCA [51, 52].

Fig. 2.2 shows a structure of a typical network which has three layers, namely, input,

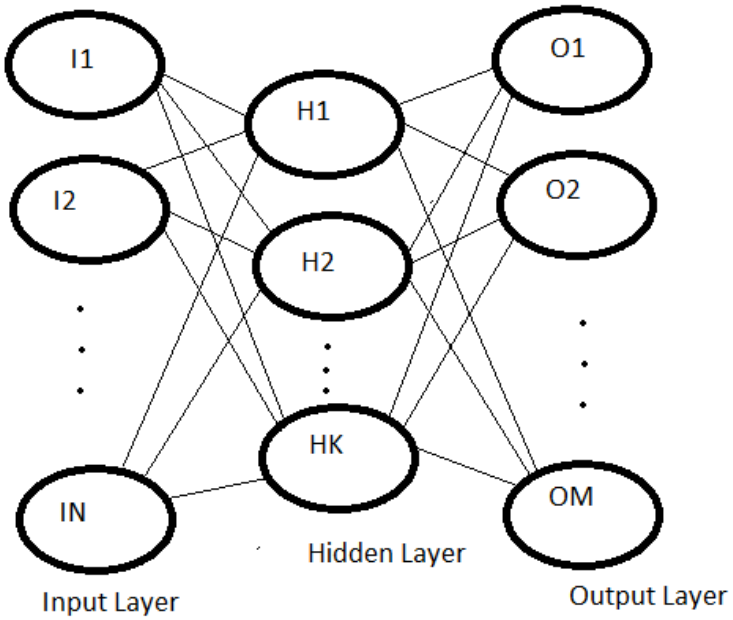


Figure 2.2: A three-layer neural network

hidden, and output ones. In a classification application, generally, the number of input units is equal to the length of feature vectors. The number of hidden units could be determined experimentally, and finally the number of output units is equal to the number of classes in the application. There could be more than one hidden layers in the network. Very recently, deep convolutional neural networks (CNN) has been successfully applied to hyper-spectral image classification [53]. The technique becomes superior to the SVM approach, however, it requires more training samples than a conventional classifier.

2.3. Support Vector Machines (SVM)

SVM is a binary classification technique in which a maximum distance decision surface is found between closest points (support vectors) of two classes [54]. The points are assumed to be linearly separable. In case they are inseparable, a penalty factor C is utilized. Alternatively, a non-linear kernel (RBF, Polynomial, etc.) is used to determine a non-linear decision boundary. The hyper-plane found by the algorithm has maximum distances to the support vectors (Fig. 2.3).

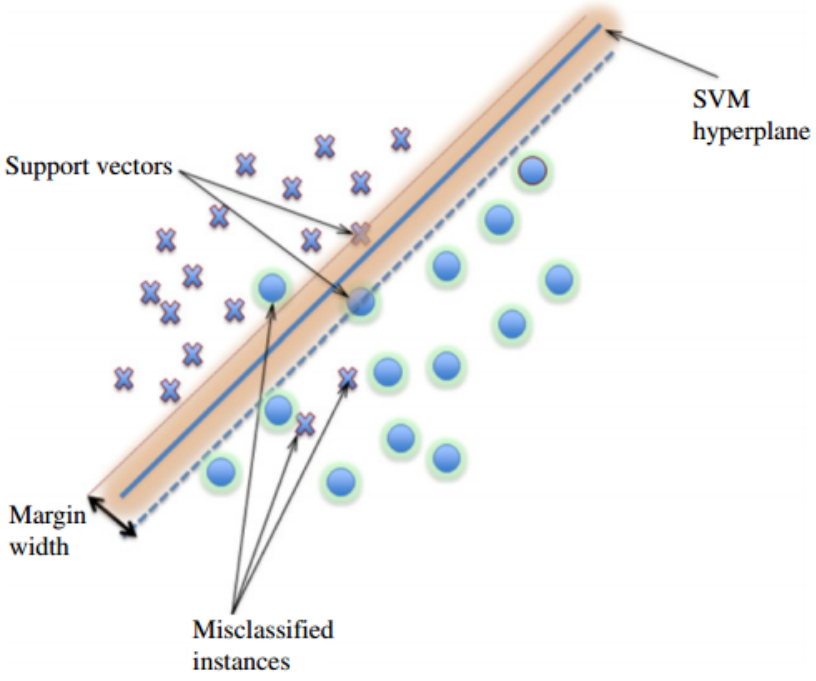


Figure 2.3: Support vector machines

Since the algorithm is originally a binary classification technique, it can not be directly applied to a multi-class classification problem. There are various strategies proposed in the literature to extend the binary SVMs to multi-class SVMs. The two famous of them namely one-against-all (OAA) and one-against-one (OAO) strategies are quite common for multi-class problems. In OAA approach, the number of binary SVMs is equal to the number of classes in the training set. Each SVM finds a separating hyper plane between ith class and the rest of the classes. There are some drawbacks of this strategy. First, the number of train pixels per class is unbalanced since the rest of the classes has more train pixels than the ith class. Second, the size of one SVM classifier is very large and it requires large memories. In OAO approach, for every possible pair of classes there exists a binary SVM. This approach is a symmetric one and every SVM requires less memory than the OAA approach. However, the number of classifiers is larger than the OAA strategy. Therefore, we can say that during the classification procedure, it requires more time. SVM approach has been applied to plenty of classification problems in the literature including text classification [55, 56, 57, 58], face recognition [59, 60, 61], protein classification [62, 63, 64, 65], gene classification [66, 67], hyper-spectral image classification [47, 68, 69, 70, 71],

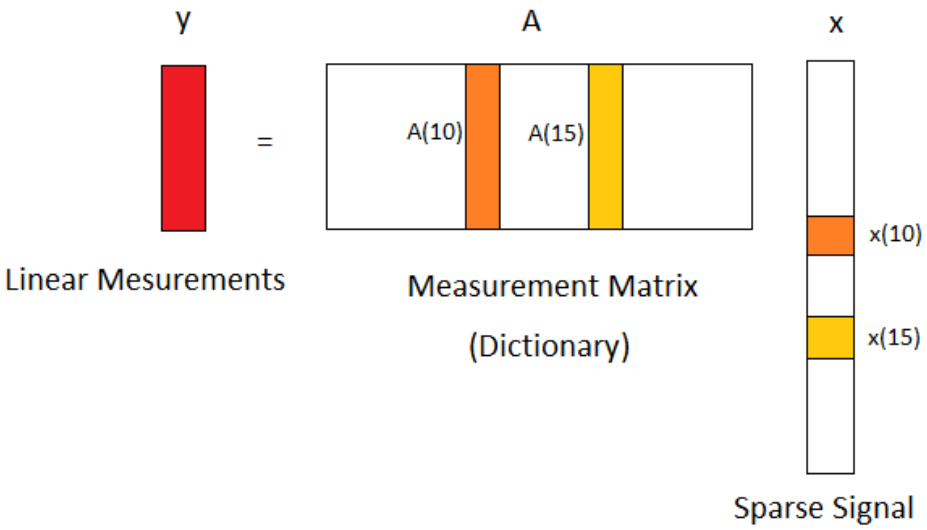


Figure 2.4: Sparse representation

spam categorization [72], etc. Although this technique is quite common, its parameters are determined using cross validation which may result in non-optimal parameter set.

2.4. Sparse Representation-Based Classification (SRC)

As we stated in the previous chapter, most of the signals in the nature are sparse and a few components of them carry information. The major challenge in sparse approximation / reconstruction is to recover the original signal $x \in \mathbb{R}^N$ using a few measurements $y \in \mathbb{R}^B$ and the sensing (measurement) matrix $A \in \mathbb{R}^{B \times N}$ (Fig. 2.4). It is assumed that each column of A has unit Euclidean norm and the system of equations is mostly under-determined ($B \ll N$).

The reconstruction task could be written as the following minimization problem:

$$\hat{x} = \arg\min_{x} \|x\|_0 \ \ subject\ to\ y = Ax \ \ or \ \ subject\ to\ \|y - Ax\|_2 \leq \epsilon \qquad (2.1)$$

where $\|x\|_0$ is l_0 norm of x, which counts the number of non-zero components in x ,and ϵ is small error tolerance. Unfortunately, the problem is not tractable and

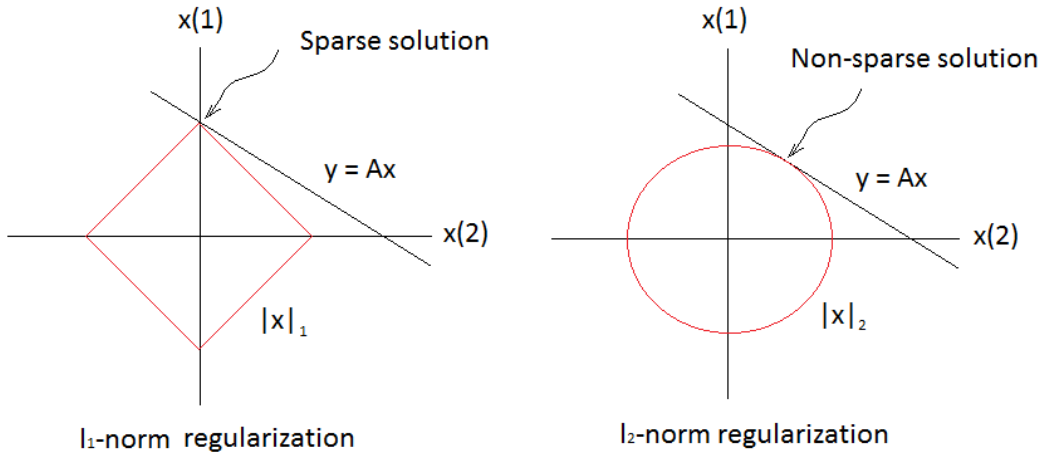


Figure 2.5: l_1 versus l_2 regularized solutions

it is NP-hard. One intuitive solution could be found by replacing l_0 norm with l_2 norm (Euclidean norm) which results in Tikhonov regularization [73]. However, this technique does not produce sparse solution. Alternative approach is to replace l_0 norm with l_1 since it is convex and close to l_0 function. Luckily, this method provides sparse solution. The comparison between these two approaches is demonstrated in Fig. 2.5.

We can also compare the two methods by performing an experiment. Assume that we have a sparse signal $x \in \mathbb{R}^{512}$ having 15 non-zero entries and a sensing matrix $A \in \mathbb{R}^{170\times512}$ whose entries are filled with numbers drawn from Gaussian distribution. The observation vector y can be formed by decoding x via A, that is, $y = Ax$. We can then recover x given A and y by performing l_1 and l_2 regularizations. The result could be seen in Fig. 2.6. As we can observe, l_1 regularization approach exactly recovers the original x, having very tiny ripples. On the other hand, the result obtained by l_2 regularization is highly dense and only a few peaks can be observed.

l_1 regularized solution could be obtained via some convex relaxation methods such as basis pursuit [13], least absolute shrinkage and selection operator (LASSO) [74], homotopy [75], etc. The common problems of these techniques are heavy computational complexity. Greedy approximations such as orthogonal matching pursuit (OMP) [35], compressive sampling matching pursuit (CoSaMP) [36], stage-wise or-

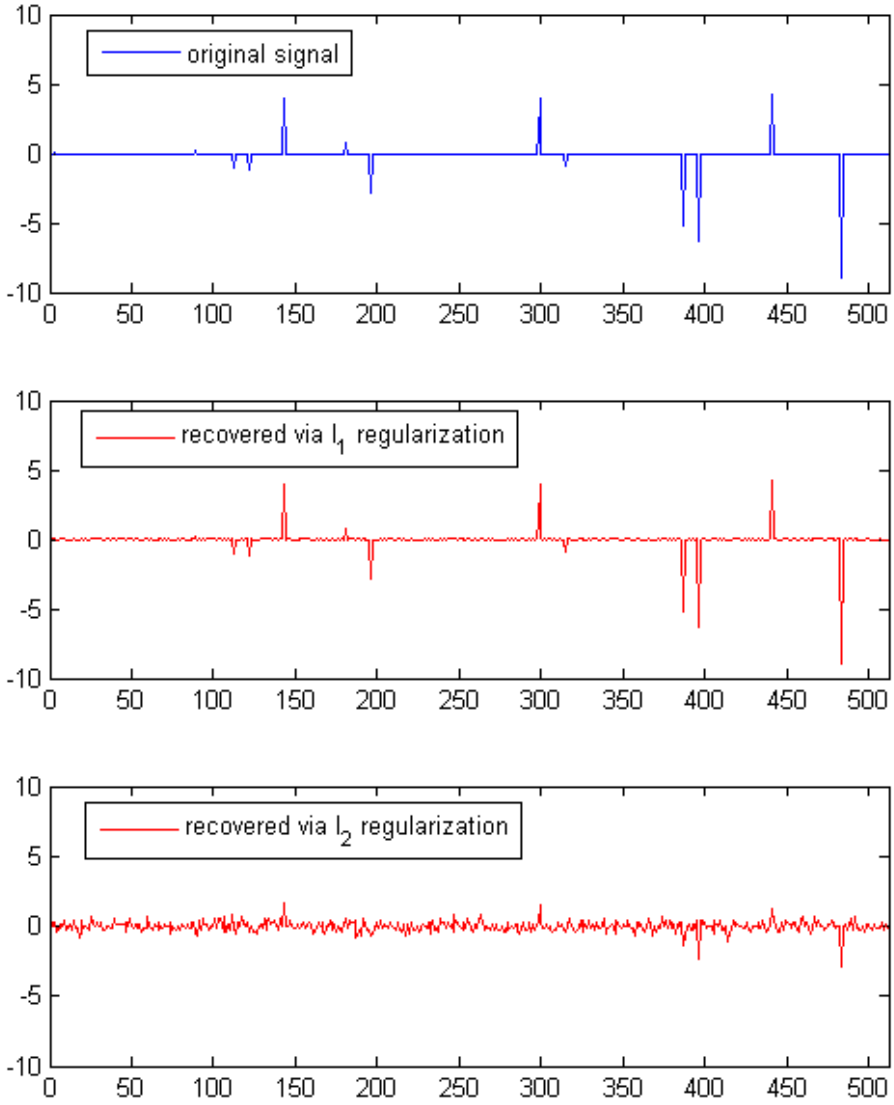


Figure 2.6: An experiment: l_1 versus l_2 regularized solutions

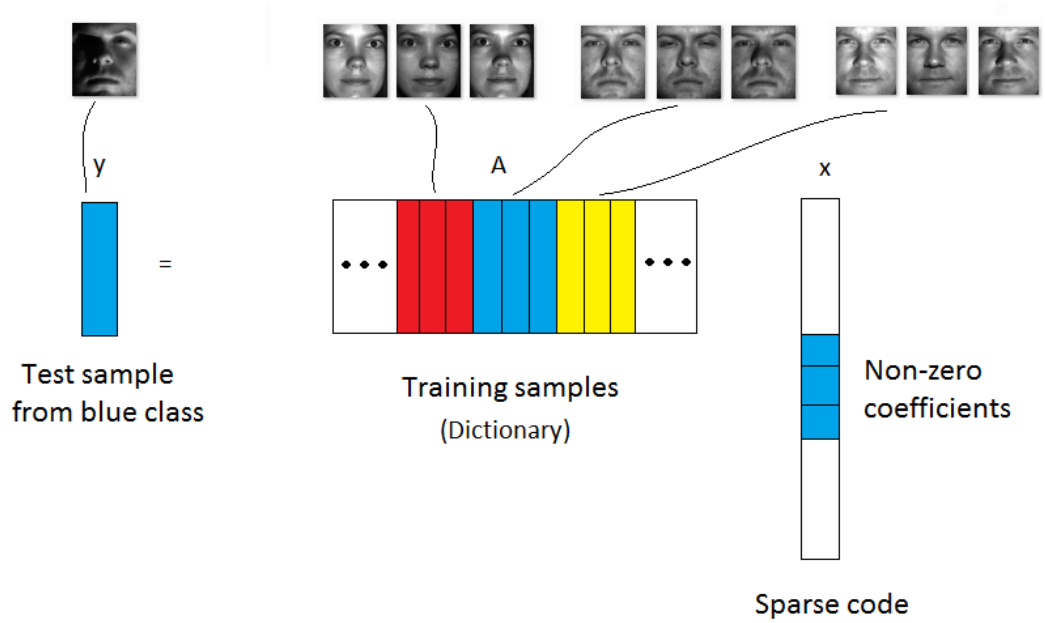


Figure 2.7: Sparse representation in classification

thogonal matching pursuit (StOMP) [38] are available in the literature to reduce the computational burden in sparse approximation.

Recently, sparse representation has been successfully applied to classification problems by exploiting the fact that identity information (class identity) of a given test sample is sparse among the other classes [2, 3]. In this case, the matrix A does not contain the basis elements but the labeled training samples. The measurement vector y becomes the sample to be classified and the sparse signal x is interpreted as the vector consisting of identity information. The new interpretation is shown in Fig. 2.7.

After finding the sparse code using l_1-minimization in the new configuration, class identity of a given test sample is estimated via class-wise regression error. That is,

$$class(y) = \arg\min_i \|y - A\hat{x}_i\|_2 \ \forall i \in \{1, 2, \ldots, C\} \tag{2.2}$$

where $\hat{x}_i$ represents the ith class portion of the estimated sparse code $\hat{x}$ among C many classes. The class estimation stage could also be seen in Fig. 2.8.

The most crucial side of this approach is low sensitivity to corrupted features because of the fact that the errors due to these kinds of features are often sparse with respect to the dictionary elements [2]. Although this approach achieves state-of-the-art results,

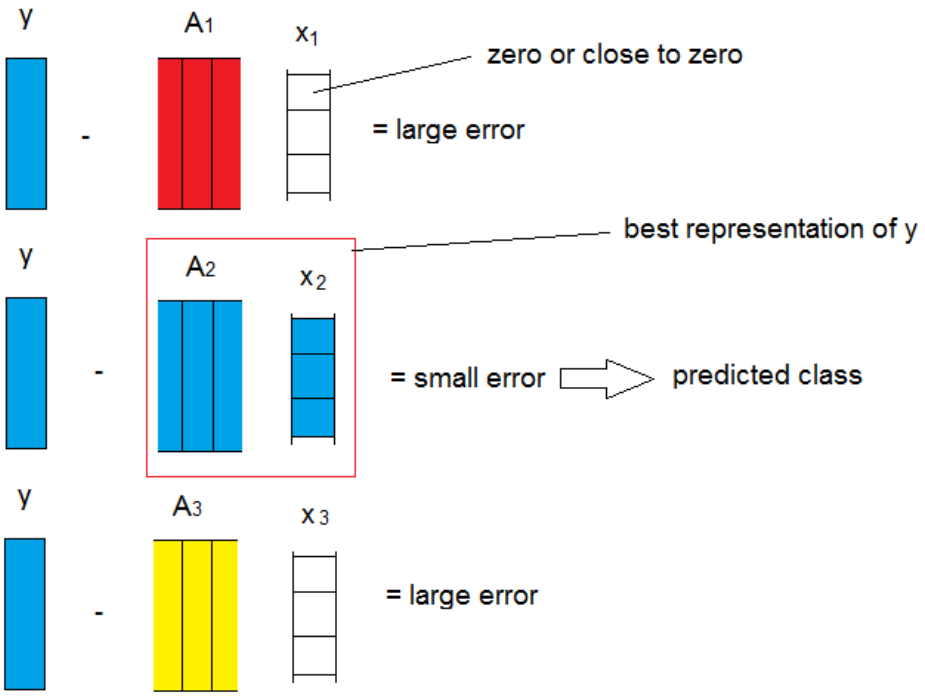


Figure 2.8: Class estimation via class-wise regression error

its sparse recovery part based on the l_1 minimization is extremely costly and infeasible to apply huge size problems such as hyper-spectral image classification. Alternative approaches such as collaborative representation-based classification (CRC) based on l_2 regularization have been proposed to reduce the computational cost. However, as observed in Fig. 2.6, the method uses non-sparse solution which only provides satisfactory results if the problem contains very large number of features. Therefore, this approach fails in most of the conventional classification problems involving a moderate number of features.

CHAPTER 3

BASIC THRESHOLDING CLASSIFICATION

All the limitations with the described algorithms in the previous chapter force us to seek a method for classification problems having the following properties:

- It provides high classification accuracy.
- It is robust, cost effective, and fast.
- It is easy to implement.
- It has no parameter tuning experimentally.

In this context, we propose the BTC algorithm for classification applications which approximately satisfies aforementioned properties. BTC is motivated by *basic thresholding* (BT) algorithm which could be considered one of the simplest techniques in compressed sensing theory [76, 77]. Unlike BT, which is a generic sparse signal recovery algorithm, BTC is a classifier which utilizes Tikhonov's regularization [73] for the overdetermined systems and performs classification using class-wise regression error. In the following section, we will develop the BTC algorithm step by step.

3.1. Basic Thresholding Classifier

In the generic sparse signal recovery problem, BT algorithm applies the following two stages given a dictionary containing orthonormal atoms:

- The first step consists of selecting a subset of the atoms (D) from the whole dictionary A which are close in angle to the signal y.

- The second step estimates the sparse code for y with respect to the pruned dictionary D, which is performed by solving the following minimization problem:

$$\hat{x} = \arg\ \min_{x} \|y - Dx\|_2 \tag{3.1}$$

  The solution is obtained using the ordinary least squares (OLS) technique, $\hat{x}(\Lambda) = (D^T D)^{-1} D^T y$. Note that in the expression, only the entries indexed with Λ are updated and the others are set to zero. The index set Λ consists of the indexes of M (threshold) largest correlations.

Here, the assumption is that y is a linear combination of a subset of orthonormal basis vectors included in the dictionary. In classification problems, the assumption of orthonormal basis vectors fails since the selected subset of the atoms contains extremely correlated columns possibly from the same class. Therefore, the matrix constructed using the selected atoms of the original dictionary becomes singular and the OLS produces meaningless bad sparse approximation. One solution to this problem is to utilize Tikhonov's regularization method for the overdetermined systems. In this case, the minimization problem could be written as follows:

$$\hat{x} = \arg\ \min_{x} \|y - Dx\|_2^2 + \alpha \|x\|_2^2 \tag{3.2}$$

The solution of the minimization problem could be obtained by manipulating the cost function below,

$$J(x) = \|y - Dx\|_2^2 + \alpha \|x\|_2^2 \tag{3.3}$$

The expanded version of it can be written as,

$$\begin{aligned} J(x) &= (y - Dx)^T (y - Dx) + \alpha x^T x \\ &= y^T y - 2D^T yx + x^T D^T Dx + \alpha x^T x \end{aligned} \tag{3.4}$$

In order to find x which minimizes $J(x)$, we take the gradient of $J(x)$ with respect to x and set it to zero, that is,

$$\begin{aligned} \nabla J(x) &= 0 \\ -2D^T y + 2D^T Dx + 2\alpha x &= 0 \\ (D^T D + \alpha)x &= D^T y \\ x &= (D^T D + \alpha I)^{-1} D^T y \end{aligned} \tag{3.5}$$

The final equation which estimates the sparse code becomes as follows,

$$\hat{x}(\Lambda) = (D^T D + \alpha I)^{-1} D^T y \tag{3.6}$$

Here, α is a small regularization constant which is generally problem dependent. Tikhonov's regularization has the following advantages:

- It filters out the small or zero eigenvalues of $D^T D$.
- It is simple to implement and it requires no complex decompositions like singular value decomposition (SVD).
- It preserves the structure of $D^T D$ matrix and the effects of it could easily be analyzed.

We can also repeat the same experiment performed in the previous chapter in order to see how BTC technique produces an efficient sparse solution. The result could be seen in Fig. 3.1. As we can observe, most of the components of sparse x are perfectly recovered by BTC and only a few insignificant components of it are missed.

As we stated previously, BTC not only recovers the sparse code, but also performs classification using a predetermined dictionary containing labeled training samples. It produces the class label of a test sample based on the minimal residual or equivalently class-wise regression error. The implementation is given in **Algorithm** 1.

The BTC algorithm performs the following steps:

- In the first step, BTC finds the correlation vector v containing the linear correlations between the test sample y and the samples of all training set A in the original feature space via inner product.

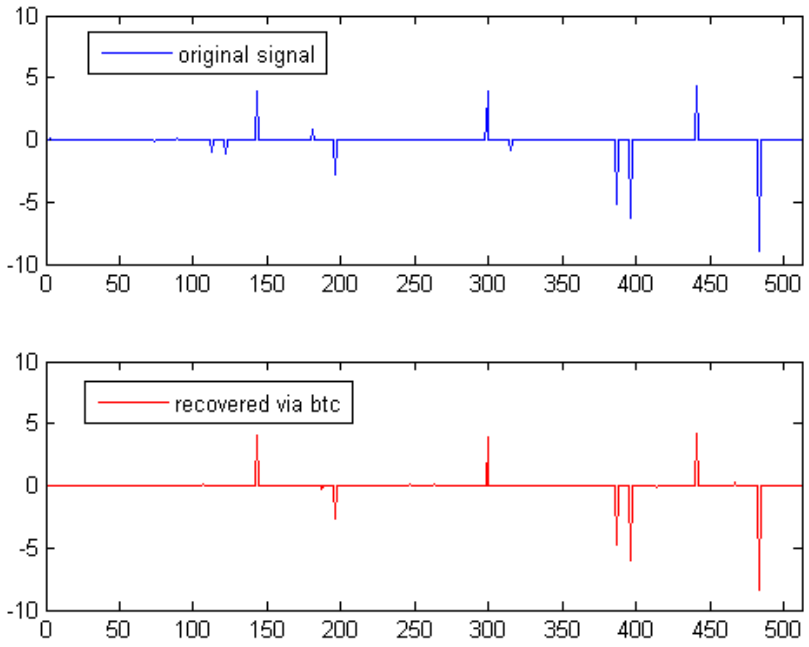


Figure 3.1: An experiment: Sparse solution via BTC

Algorithm 1 BTC

INPUT:

Dictionary $A \in \mathbb{R}^{B \times N}$

Test sample $y \in \mathbb{R}^B$

Threshold $M \in \mathbb{N}$

Regularization constant $\alpha \in (0, 1)$

Initial sparse coefficients with zeros $\hat{x} \in \mathbb{R}^N$

OUTPUT:

Class of y

Residual vector $\epsilon \in \mathbb{R}^C$

PROCEDURE:

1: $v \leftarrow A^T y$

2: $\Lambda \leftarrow L_M(v)$

3: $D \leftarrow A(\Lambda)$

4: $\hat{x}(\Lambda) \leftarrow (D^T D + \alpha I)^{-1} D^T y$

5: $\epsilon(j) \leftarrow \|y - A_j \hat{x}_j\|_2 \quad \forall j \in \{1, 2, \ldots, C\}$

6: $class(y) \leftarrow \arg\min_j \epsilon(j)$

- In the second step, the operator $L_M(.)$ selects the index set Λ consisting of the indexes of M largest correlations.

- Then, D matrix is extracted from the original dictionary A by means of the index set Λ.

- The entires of the sparse code x indexed with Λ are estimated using regularized least squares technique with a small regularization constant α.

- Finally, it calculates the residuals for all classes, and predicts the class of y based on the minimum residual.

3.2. Upper Bound for the Threshold

Tikhonov's regularization not only helps us estimate the sparse code x but also provides the upper bound for the threshold parameter M. Notice that this regularization technique requires the system of equations is overdetermined. Therefore, the number of columns of D, which is equal to the threshold parameter M, can not exceed the number of rows (features (B)) of it. We can then define the following relation,

$$M < B \tag{3.7}$$

This relation highly reduces the cost of the algorithm since $B \ll N$. A nice thing that the boundaries of M provide us is that M does not grow as the number of classes in the dictionary increases. For instance, assume that we have 1000 subjects and each contains 10 samples. Also suppose that we want to reduce the feature vector size to 120. In this example, M will be less than 120. This will highly reduce the cost of BTC algorithm. However, for an l_1-minimization-based classification technique, the dictionary size will be 120×10^4 which will extremely increase the convergence time. In the following part, we will introduce a sufficient identification condition (SIC) for BTC and based on it, we will develop a procedure to determine the best value of the parameter M in the SIC sense.

3.3. Sufficient Identification Condition for BTC

In compressed sensing theory, one of the most fundamental property of a dictionary is the *mutual incoherence* quantity (3.8).

$$\mu \triangleq \max_{i \neq j} | < A(i), A(j) > | \tag{3.8}$$

It simply measures how much any two elements in a dictionary look alike. Using μ one can determine under which conditions an algorithm recovers the correct sparse code. For instance, Tropp [78] showed that if $\mu < \frac{1}{2K-1}$, then the OMP algorithm perfectly recovers any K-sparse vector x from the measurements $y = Ax$. In generic sparse signal recovery, μ is desired to be small. A dictionary having $\mu = 0.05$ will satisfy exact recovery condition for OMP to recover any sparse signal having sparsity of at most 10. Unfortunately, in classification applications, the columns of a dictionary are highly correlated. Therefore, we can not use the quantity μ to determine such conditions [3]. *Cumulative coherence* is another quantity which measures the maximum total similarity between a fixed column and the collection of other columns [78]. Even the conditions based on this measure are not useful for classification problems because of the high correlations.

Luckily, the BTC algorithm enables us to develop such conditions for the dictionaries in classification applications. Let us consider the following proposition:

Proposition 3.3.1. *A sufficient condition for BTC to identify a test sample y belonging to the ith class of the dictionary A is that*

$$\max_{j \neq i} \frac{\|y - A_i \hat{x}_i\|_2}{\|y - A_j \hat{x}_j\|_2} < 1 \tag{3.9}$$

where $\hat{x}_i$ and $\hat{x}_j$ are the ith and jth class portions of $\hat{x}$, respectively.

Proof. A test sample y belonging to the ith class can be successfully identified via BTC if and only if the residual $\|y - A_i \hat{x}_i\|_2$ is minimum. It implies that,

$$\|y - A_i \hat{x}_i\|_2 < \min_{j \neq i} \|y - A_j \hat{x}_j\|_2 \tag{3.10}$$

Dividing both sides of the inequality with the right hand side concludes the proof. □

Based on the proposition, we define a quantity namely the SIC rate as follows: We replace the testing sample y with a_i which is a training sample belonging to A_i. We also replace the A_i matrix with $\overline{A_i}$ which excludes the column a_i. Now the training sample a_i is not belonging to the dictionary A anymore. Then, the quantity could be expressed as follows,

$$\beta_M(a_i) \triangleq \max_{j \neq i} \frac{\left\| a_i - \overline{A_i}\hat{x}_i \right\|_2}{\left\| a_i - A_j\hat{x}_j \right\|_2} \tag{3.11}$$

Notice that we used the notation $\beta_M(a_i)$ because the expression is also a function of the parameter M. If we can find a threshold value which minimizes $\beta_M(a_i)$, we then state that this is the best threshold for the selected a_i in the SIC sense. One could compute the value of $\beta_M(a_i)$ using **Algorithm** 2. In the algorithm, the operator $L_{M-1}(.)$ selects the indexes of M largest correlations excluding the first one which corresponds to the index of a_i. The idea of the best threshold in the SIC sense for

Algorithm 2 $\beta_M(a_i)$

INPUT:

Dictionary $A \in \mathbb{R}^{B \times N}$

Any selected sample from the ith class $a_i \in \mathbb{R}^B$

Threshold $M \in \mathbb{N}$

Regularization constant $\alpha \in (0, 1)$

Initial sparse coefficients with zeros $\hat{x} \in \mathbb{R}^N$

OUTPUT:

$\beta_M(a_i) \in \mathbb{R}$

PROCEDURE:

1: $v \leftarrow A^T a_i$

2: $\Lambda \leftarrow L_{M-1}(v)$

3: $D \leftarrow A(\Lambda)$

4: $\hat{x}(\Lambda) \leftarrow (D^T D + \alpha I)^{-1} D^T a_i$

5: $\epsilon(j) \leftarrow \|a_i - A_j\hat{x}_j\|_2 \quad \forall j \in \{1, 2, \ldots, C\}$

6: $\beta_M(a_i) \leftarrow \max_{j \neq i} \epsilon(i)/\epsilon(j)$

any a_i could be extended to the all samples of A. Let $\overline{\beta}_M$ be the quantity which is computed by averaging the $\beta_M(a_i)$ for all samples of A. This is shown by the

following expression.

$$\overline{\beta}_M \triangleq \frac{1}{N}\sum_{k=1}^{N}\beta_M(A(k)) \tag{3.12}$$

3.4. Parameter Selection

Parameter selection is quite critical for a classifier, which highly effects the classification performance. Some algorithms use cross validation in which some portion of the training data is used for testing purposes. This method may not always provide good parameter estimation. Bad estimation of the parameters highly reduces the classification accuracy. In the following parts, we will provide some procedures to estimate the parameters of BTC.

3.4.1 Estimating the Threshold Parameter

Using the previously defined quantity $\overline{\beta}_M$, one can estimate the threshold parameter. The best estimate of M could be found in the SIC sense using the following minimization expression:

$$\hat{M} = \arg\min_{M} \overline{\beta}_M \tag{3.13}$$

By varying M from 1 to B, we could plot the quantity $\overline{\beta}_M$, and choose the best M which minimizes it. Since $\overline{\beta}_M$ is generally convex, the procedures such as binary search and steepest descent could be utilized and the best value of M could be found in a few steps. However, since the parameter determination is an off-line procedure, exhaustive search could also be used.

3.4.2 Selection of the Regularization Parameter

The best value of α is generally problem dependent. Once it is determined for a classification application, it is fixed for all experiments. For example, for face identification one could set it to 0.01 for all experiments while for pixel-wise hyper-spectral image classification, it can be fixed to 0.0001. Sometimes, the optimal choice of α is not critical and it is set to a quite small number just to prevent the ill-conditioned matrix

operation. For instance, for spatial-spectral hyper-spectral image classification, it is set to very small number such as 10^{-10}.

CHAPTER 4

KERNEL BASIC THRESHOLDING CLASSIFICATION

4.1. Mapping to Hyperspace

As we stated previously, in practice, given test samples belonging to different classes may not be distinguishable or separable in the original B dimensional feature space. One solution to this problem is to map the samples in the original feature space to some arbitrarily large or possibly infinite dimensional hyperspace via a mapping function ϕ [79], that is,

$$\phi : \mathbb{R}^B \longrightarrow \mathcal{F} \;\; by \;\; y \longmapsto \phi(y) \tag{4.1}$$

Fig. 4.1 shows how inseparable data in two dimensional space becomes separable in three dimensional space via a separating hyper-plane. This solution could also be applied to the BTC algorithm. In the following parts, we are going to develop kernel basic thresholding classifier by utilizing this technique.

4.1.1 Mapping the Cost Function

We know that BTC uses the cost function given below,

$$J(x) = \|y - Dx\|_2^2 + \alpha \|x\|_2^2 \tag{4.2}$$

If we map both the test sample y and the selected D matrix to $\mathcal{F}$ via the mapping function ϕ, we then obtain the new cost function as follows,

$$J(x) = \|\phi(y) - \phi(D)x\|_2^2 + \alpha \|x\|_2^2 \tag{4.3}$$

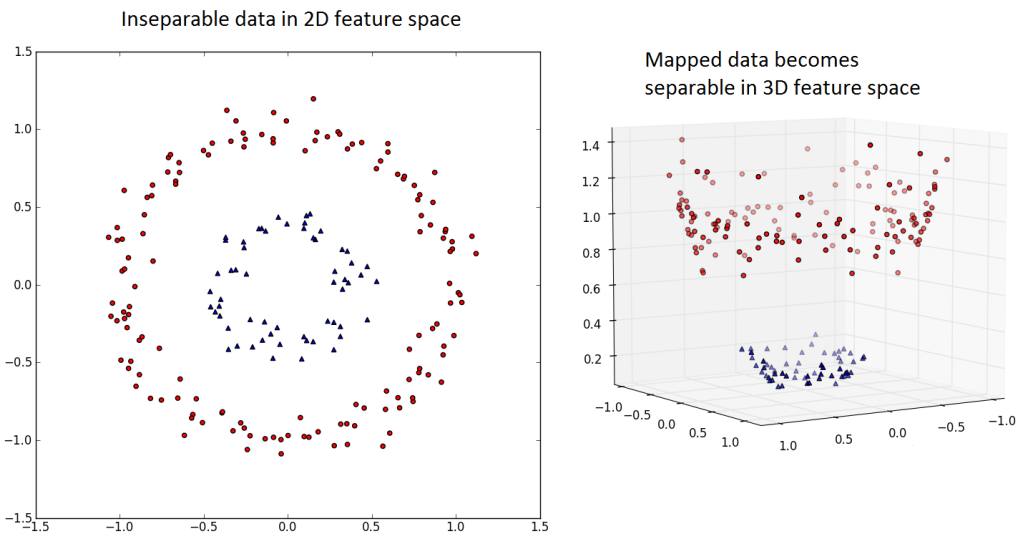


Figure 4.1: Mapping to hyperspace

It could also be expanded as,

$$\begin{aligned} J(x) &= (\phi(y) - \phi(D)x)^T(\phi(y) - \phi(D)x) + \alpha x^T x \\ &= \phi(y)^T\phi(y) - 2\phi(D)^T\phi(y)x + x^T\phi(D)^T\phi(D)x + \alpha x^T x \end{aligned} \tag{4.4}$$

In order to find x which minimizes $J(x)$ in $\mathcal{F}$, we take the gradient of $J(x)$ with respect to x and set it to zero, that is,

$$\begin{aligned} \nabla J(x) &= 0 \\ -2\phi(D)^T\phi(y) + 2\phi(D)^T\phi(D)x + 2\alpha x &= 0 \\ (\phi(D)^T\phi(D) + \alpha)x &= \phi(D)^T\phi(y) \\ x &= (\phi(D)^T\phi(D) + \alpha I)^{-1}\phi(D)^T\phi(y) \end{aligned} \tag{4.5}$$

The final equation which estimates the sparse code in $\mathcal{F}$ in terms of mapped elements becomes as follows,

$$\hat{x}(\Lambda) = (\phi(D)^T\phi(D) + \alpha I)^{-1}\phi(D)^T\phi(y) \tag{4.6}$$

4.1.2 Kernel Trick

Since we do not know the mapping function ϕ, we can not directly calculate $\hat{x}$ using (4.6). However, notice that the expression contains the inner products $\langle\phi(D(i)), \phi(D(j))\rangle$ $\forall i, j \in \{1, 2, \ldots, M\}$ and $\langle\phi(D(i)), y\rangle$ $\forall i \in \{1, 2, \ldots, M\}$. This enables us to use

the kernel trick [80] by which we can calculate the inner products of two vectors in $\mathcal{F}$ implicitly via a kernel function, that is, $K(x, y) = \langle \phi(x), \phi(y) \rangle$. Then, the kernelized version of (4.6) becomes as follows,

$$\hat{x}(\Lambda) = (K(D, D) + \alpha I)^{-1} K(D, y) \tag{4.7}$$

where $K(D, D)$ is an $M \times M$ Gram matrix such that the entry corresponding to the ith row and jth column is equivalent to $K(D(i), D(j)) = \langle \phi(D(i)), \phi(D(j)) \rangle$ and $K(D, y)$ is an $M \times 1$ vector whose entries are $K(D(i), y) = \langle \phi(D(i)), y \rangle \quad \forall i \in \{1, 2, \ldots, M\}$.

4.1.3 Mapping the Residuals

After finding the sparse code estimation in $\mathcal{F}$, we also need to determine the distances or residuals in $\mathcal{F}$ between the test sample and the best representation of it for all classes, that is, $\epsilon(j) = \|\phi(y) - \phi(A_j)\hat{x}_j\|_2 \quad \forall j \in \{1, 2, \ldots, C\}$. Since we can not calculate $\phi(y)$ and $\phi(A_j)$ directly, we are required to expand the expression as follows,

$$\begin{aligned} \epsilon(j) &= \|\phi(y) - \phi(A_j)\hat{x}_j\|_2 \\ &= \sqrt{(\phi(y) - \phi(A_j)\hat{x}_j)^T(\phi(y) - \phi(A_j)\hat{x}_j)} \\ &= \sqrt{\phi(y)^T\phi(y) - 2\hat{x}_j^T\phi(A_j)^T\phi(y) + \hat{x}_j^T\phi(A_j)^T\phi(A_j)\hat{x}_j} \\ &= \sqrt{K(y, y) - 2\hat{x}_j^T K(A_j, y) + \hat{x}_j^T K(A_j, A_j)\hat{x}_j} \end{aligned} \tag{4.8}$$

4.2. Kernel Basic Thresholding Classifier

After obtaining the required kernelized expressions, we can easily construct the KBTC algorithm.

The implementation of the KBTC algorithm is presented in **Algorithm** 3. The KBTC technique performs the following steps:

Algorithm 3 KBTC

INPUT:

Dictionary $A \in \mathbb{R}^{B \times N}$

Test sample $y \in \mathbb{R}^B$

Threshold $M \in \mathbb{N}$

Regularization constant $\alpha \in (0, 1)$

Initial sparse coefficients with zeros $\hat{x} \in \mathbb{R}^N$

OUTPUT:

Class of y

Residual vector $\epsilon \in \mathbb{R}^C$

PROCEDURE:

1: $v \leftarrow K(A, y)$

2: $\Lambda \leftarrow L_M(v)$

3: $D \leftarrow A(\Lambda)$

4: $\hat{x}(\Lambda) \leftarrow (K(D, D) + \alpha I)^{-1} K(D, y)$

5: $\epsilon(j) \leftarrow \sqrt{K(y, y) - 2\hat{x}_j^T K(A_j, y) + \hat{x}_j^T K(A_j, A_j)\hat{x}_j} \;\; \forall j \in \{1, 2, \ldots, C\}$

6: $class(y) \leftarrow \arg\min_j \epsilon(j)$

- In the first step, KBTC finds the non-linear correlations between the test sample y and the samples of all training set A in $\mathcal{F}$ using the kernel function $K(.,.)$.

- In the second step, it chooses the index set of largest M correlations using the operator $L_M(.)$.

- Then, it forms the sub matrix D by using the indexes in the set Λ.

- Using the expression $(K(D,D)+\alpha I)^{-1}K(D,y)$, it estimates the sparse code indexed with Λ.

- Finally, it calculates the residuals for all classes, and predicts the class of y based on the minimum residual.

Note that computing $K(D,D)$ whenever a new sample is needed to be classified, is infeasible. Instead, we recommend computing $K(A,A)$ off-line and extract the required matrix as $K(D,D) \leftarrow K(A(\Lambda),A(\Lambda))$. This way of computing $K(D,D)$ highly reduces the computational cost. There are several kernels used in the literature such as the polynomial kernel $K(x,y) = (1+x^T y)^d$ and the radial basis function (RBF) kernel $K(x,y) = exp(-\gamma \left\|x-y\right\|_2^2)$. In this dissertation, we use the RBF kernel which is more commonly preferred. There are three parameters in the KBTC algorithm namely the regularization constant α, the threshold M, and the γ parameter of the kernel function. The choice of α is an easy one because it is used just to prevent the ill-conditioned matrix inversion. Typically, we set it to very small number such as 10^{-9}, 10^{-10}, etc. The most critical parameter is γ which determines the hyperspace $\mathcal{F}$ in which the classes of a dictionary should be separate enough such that a test sample could be classified correctly. SVM technique uses cross validation to determine the γ parameter. However, this method may not provide the best γ for all classifiers. In the next part, we will introduce sufficient identification condition for KBTC and based on it, we will develop some procedures to determine the best values of γ and M parameters.

4.3. Sufficient Identification Condition for KBTC

Proposition 4.3.1. *A sufficient condition for KBTC to identify a test sample y belonging to the ith class of the dictionary A is that*

$$\max_{j \neq i} \frac{\sqrt{K(y,y) - 2\hat{x}_i^T K(A_i, y) + \hat{x}_i^T K(A_i, A_i)\hat{x}_i}}{\sqrt{K(y,y) - 2\hat{x}_j^T K(A_j, y) + \hat{x}_j^T K(A_j, A_j)\hat{x}_j}} < 1 \tag{4.9}$$

where $\hat{x}$ is the sparse code estimated via KBTC, $\hat{x}_i$ and $\hat{x}_j$ are the ith and jth class portions of $\hat{x}$, respectively, and $K(.,.)$ is the kernel function.

Proof. A test sample y belonging to the ith class can be successfully identified if and only if the residual $\|\phi(y) - \phi(A_i)\hat{x}_i\|_2$ is minimum. It implies that,

$$\|\phi(y) - \phi(A_i)\hat{x}_i\|_2 < \min_{j \neq i} \|\phi(y) - \phi(A_j)\hat{x}_j\|_2 \tag{4.10}$$

Expanding both sides of the inequality we obtain that,

$$\begin{aligned} &\sqrt{\phi(y)^T\phi(y) - 2\hat{x}_i^T\phi(A_i)^T\phi(y) + \hat{x}_i^T\phi(A_i)^T\phi(A_i)\hat{x}_i} \\ &< \min_{j \neq i} \sqrt{\phi(y)^T\phi(y) - 2\hat{x}_j^T\phi(A_j)^T\phi(y) + \hat{x}_j^T\phi(A_j)^T\phi(A_j)\hat{x}_j} \end{aligned} \tag{4.11}$$

$$\begin{aligned} &\sqrt{K(y,y) - 2\hat{x}_i^T K(A_i, y) + \hat{x}_i^T K(A_i, A_i)\hat{x}_i} \\ &< \min_{j \neq i} \sqrt{K(y,y) - 2\hat{x}_j^T K(A_j, y) + \hat{x}_j^T K(A_j, A_j)\hat{x}_j} \end{aligned}$$

Finally, dividing both sides of the inequality with the right hand side concludes the proof. □

Now, based on the proposition, we define a quantity by replacing the test sample y with a_i which is a training sample belonging to A_i and also replacing the A_i matrix with $\overline{A_i}$ which excludes the column a_i. It means that a_i is not belonging to the dictionary A anymore. Then, the quantity could be expressed as follows,

$$\beta(\gamma, M, a_i) \triangleq \max_{j \neq i} \frac{\sqrt{K(a_i,a_i) - 2\hat{x}_i^T K(\overline{A_i}, a_i) + \hat{x}_i^T K(\overline{A_i}, \overline{A_i})\hat{x}_i}}{\sqrt{K(a_i,a_i) - 2\hat{x}_j^T K(A_j, a_i) + \hat{x}_j^T K(A_j, A_j)\hat{x}_j}} \tag{4.12}$$

One could easily compute $\beta(\gamma, M, a_i)$ using the **Algorithm** 4. It is similar to the **Algorithm** 3, however, this time the input is any selected training sample belonging to ith class and the operator $L_{M-1}(.)$ chooses the indexes of M largest non-linear correlations excluding the first one which is the index of a_i itself. The final output value is equivalent to the ratio of residual belonging to the ith class and minimum residual whose class is not equal to i.

Algorithm 4 $\beta(\gamma, M, a_i)$

INPUT:

Dictionary $A \in \mathbb{R}^{B \times N}$

Any selected training sample from the ith class $a_i \in \mathbb{R}^B$

Threshold $M \in \mathbb{N}$

Regularization constant $\alpha \in (0, 1)$

Initial sparse coefficients with zeros $\hat{x} \in \mathbb{R}^N$

OUTPUT:

$\beta(\gamma, M, a_i) \in \mathbb{R}$

PROCEDURE:

1: $v \leftarrow K(A, a_i)$

2: $\Lambda \leftarrow L_{M-1}(v)$

3: $D \leftarrow A(\Lambda)$

4: $\hat{x}(\Lambda) \leftarrow (K(D, D) + \alpha I)^{-1} K(D, a_i)$

5: $\epsilon(j) \leftarrow \sqrt{K(a_i, a_i) - 2\hat{x}_j^T K(A_j, a_i) + \hat{x}_j^T K(A_j, A_j)\hat{x}_j} \;\; \forall j \in \{1, 2, \ldots, C\}$

6: $\beta(\gamma, M, a_i) \leftarrow \max_{j \neq i} \epsilon(i)/\epsilon(j)$

As we stated previously, the quantity $\beta(\gamma, M, a_i)$ is quite important because we will utilize it in order to estimate the parameters of the KBTC algorithm.

4.4. Parameter Selection

Notice that we used the notation $\beta(\gamma, M, a_i)$ for our quantity because it is depended on γ and the threshold M. In the following two parts, we develop some methodologies to estimate γ and M using the quantity $\beta(\gamma, M, a_i)$.

4.4.1 Estimating the γ Parameter

Notice that $\beta(\gamma, M, a_i)$ is calculated only for a column of the dictionary A. If we repeat the procedure for all columns of A and then average it, we obtain the following averaged quantity,

$$\overline{\beta}(\gamma, M) \triangleq \frac{1}{N} \sum_{n=1}^{N} \beta(\gamma, M, A(n)) \tag{4.13}$$

Knowing that M could have values from 1 to $B-1$, we could compute $\overline{\beta}(\gamma, M)$ for all Ms and then average it. The final averaged quantity becomes as follows,

$$\overline{\beta}(\gamma) \triangleq \frac{1}{(B-1)} \frac{1}{N} \sum_{m=1}^{B-1} \sum_{n=1}^{N} \beta(\gamma, m, A(n)) \tag{4.14}$$

Using $\overline{\beta}(\gamma)$ one can easily estimate the best γ for KBTC by solving the following minimization problem.

$$\hat{\gamma} = \arg\min_{\gamma} \overline{\beta}(\gamma) \tag{4.15}$$

As we stated previously, $\hat{\gamma}$ determines the hyperspace $\mathcal{F}$ that KBTC works in. After estimating γ, we also need to estimate M. In the following part, we will follow the similar procedures to estimate M.

4.4.2 Estimating the Threshold Parameter

Since we determined the γ, we could set it to the estimated value of it in the following function.

$$\overline{\beta}(\hat{\gamma}, M) \triangleq \frac{1}{N} \sum_{n=1}^{N} \beta(\hat{\gamma}, M, A(n)) \tag{4.16}$$

Now we can easily find the best estimate of M using the following minimization expression.

$$\hat{M} = \arg\min_{M} \overline{\beta}(\hat{\gamma}, M) \tag{4.17}$$

By varying M from 1 to B, we could plot the quantity $\overline{\beta}(\hat{\gamma}, M)$, and choose the best M which minimizes it.

4.5. Alternative 5th Step Calculation

We could also develop some other alternative expressions for the 5th stages of both **Algorithm** 4 and 3. For this purpose, we will use the following proposition:

Proposition 4.5.1. *If the inequality* $\|\phi(y) - \phi(A_i)x_i\|_2 < \min_{j \neq i} \|\phi(y) - \phi(A_j)x_j\|_2$ *holds for a given sample* y*, then the following inequality also holds.*

$$|K(y,y) - x_i^T K(A_i, y)| < \min_{j \neq i} |K(y,y) - x_j^T K(A_j, y)| \tag{4.18}$$

Proof. We know that both $\phi(A_i)x_i$ and $\phi(A_j)x_j$ lie in the range space of $\phi(A)$. The sparse representation assumption $\phi(y) = \phi(A)x$ implies that $\phi(y)$ also lies in $\mathcal{R}(\phi(A))$. Therefore, both the residual vectors $\phi(y) - \phi(A_i)x_i$ and $\phi(y) - \phi(A_j)x_j$ lie in $\mathcal{R}(\phi(A))$. Projecting those vectors on to the vector $\phi(y)$ via dot product with $\phi(y)/\|\phi(y)\|_2$ does not change the direction of the inequality, that is,

$$\left\| \langle \frac{\phi(y)}{\|\phi(y)\|_2}, \phi(y) - \phi(A_i)x_i \rangle \right\|_2 < \min_{j \neq i} \left\| \langle \frac{\phi(y)}{\|\phi(y)\|_2}, \phi(y) - \phi(A_j)x_j \rangle \right\|_2 \tag{4.19}$$

Then the expression becomes as follows,

$$|\phi(y)^T\phi(y) - x_i^T \phi(A_i)^T \phi(y)| < \min_{j \neq i} |\phi(y)^T\phi(y) - x_j^T \phi(A_j)^T \phi(y)| \tag{4.20}$$

Finally, writing the expression in terms of kernel function concludes the proof.

$$|K(y,y) - x_i^T K(A_i, y)| < \min_{j \neq i} |K(y,y) - x_j^T K(A_j, y)| \tag{4.21}$$

□

By using the proposition, we can replace the 5th steps of **Algorithm** 3 and 4 with $\epsilon(j) \leftarrow |K(y,y) - \hat{x}_j^T K(A_j, y)| \ \ \forall j \in \{1, 2, \ldots, C\}$ and with $\epsilon(j) \leftarrow |K(a_i, a_i) - \hat{x}_j^T K(A_j, a_i)| \ \ \forall j \in \{1, 2, \ldots, C\}$, respectively.

CHAPTER 5

FACE RECOGNITION VIA BTC

5.1. Introduction

Face recognition is obviously one of the most widely investigated subjects in computer vision and pattern recognition. Law enforcement, access control systems, and several commercial applications have made face recognition an attractive research area for the last two decades. Researchers have proposed various face recognition methods to provide robust, reliable, low-cost, and high-accuracy automatic identification of frontal-view faces under various difficulties. Among those, appearance based face recognition techniques have been very popular. One of the most popular appearance methods is the Principal Component Analysis (PCA) or Eigenfaces approach [43]. In PCA, high dimensional features of both train and test samples are projected to a lower dimensional subspace. Test images are then classified using Nearest Neighbor (NN) classifier in the new feature space. Another popular subspace method is the Linear Discriminant Analysis (LDA), also known as Fisher's LDA which extracts the lower dimensional features by using both within-class and between-class information [81]. Afterwards, an NN classification is performed in the identification part. All those approaches and variants focus on the feature extraction and dimension reduction stages.

Recently, Wright *et al.* [2] have proposed a robust algorithm, Sparse Representation-based Classification (SRC), for face recognition which focuses on the classification stage rather than the feature extraction. They exploit the fact that the identity information of a person is sparse among all identities in a given face database. They

use a mathematical model in which any test image lying in the span of a class of a given dataset can be represented by a linear combination of all train samples of the same set. This representation is then used for classification. They showed the robustness of the algorithm under both conventional (PCA, LDA, etc.) and unconventional (Down-sampled and Random) features. In the sparse information recovery part, they use l_1-minimization which requires solving a convex optimization problem. Unfortunately, l_1-minimization has a high computational complexity especially for large scale applications.

Recently, Yang *et al.* [82] have investigated the performances of several popular and fast l_1-minimizers. As stated in [82], one of the fastest algorithm is the homotopy method which was originally proposed in [75]. When we consider real time applications with large scale datasets, even homotopy method converges very slowly.

An interesting alternative to l_1-minimizers has been proposed in [83] for face recognition problem. The authors actually propose a hash matrix in the feature extraction part, then they use either l_1-minimization or Orthogonal Matching Pursuit (OMP) for the sparse information recovery. They recommend OMP with hashing which is extremely fast. It is indeed quite fast, however, in case of noisy sparse signal recovery, its performance reduces dramatically. Therefore, it is not recommended for face recognition especially when the train samples have illumination, expression etc. variations. Another alternative has been proposed in [84]. Instead of l_1-minimization, they use l_2-minimization which could be considered fast, however, it is very sensitive to alignment variations, therefore, is not so robust.

In this chapter, we propose the BTC algorithm for face identification problem which is capable of identifying frontal-view test samples extremely rapidly and performing high recognition rates. By exploiting rapid recognition capability, we propose a fusion scheme in which individual BTC classifiers are combined to produce better identification results especially when very small number of features is used. Finally, we propose an efficient validation technique to reject invalid test samples. Numerical results show that BTC is a tempting alternative to greedy and l_1-minimization-based algorithms [85].

5.2. Feature Extraction

In face recognition, the goal is to correctly identify the class label of any given test image using a dictionary containing labeled training samples. In this context, gray scale face samples with size of $w \times h$ are embedded into m dimensional vectors where $m = wh$. Let $a_{i,j} \in \mathbb{R}^m$ with $\|a_{i,j}\|_2 = 1$ be the vector containing the pixels of the jth sample of class i and let $A_i = [a_{i,1}\ a_{i,2}\ \dots\ a_{i,N_i}] \in \mathbb{R}^{m \times N_i}$ represent the matrix containing the training samples of the ith class with N_i many samples and C denote the number of classes. Then, the final face dictionary A could be constructed in such a way that $A = [A_1\ A_2\ \dots\ A_C] \in \mathbb{R}^{m \times N}$ where $N = \sum_{i=1}^{C} N_i$.

When we consider a face image at a typical dimension, 100×100, the final vector storing the pixels of that image will have the size of 10^4. The computational complexity of processing with dictionaries having this number of rows is extremely high. Also most of the data in such vectors are redundant and do not carry discriminative information. Therefore, a projection $R \in \mathbb{R}^{B \times m}$, where $B \ll m$, is required to remedy aforementioned problems. After finding such a transformation, one can calculate the lower dimensional representation of the dictionary, $\Phi \in \mathbb{R}^{B \times N}$, where $\Phi = RA$. In the following part, we will briefly describe the random projections technique.

5.3. Random Projections

According to Johnson-Lindenstrauss Lemma [86], any m dimensional set in Euclidean space could be embedded into $O(\log m/\epsilon^2)$ dimensional Euclidean space preserving the distances between any pair of points with small distortions which are not larger than a factor of $(1 + \epsilon)$, where $0 < \epsilon < 1$. Random projections could be considered one of the such embeddings which project the original data spherically onto a random B-dimensional hyperplane ($B \ll m$). There are various advantages of the use of random projections: 1-) It is a very simple and computationally efficient technique. 2-) It preserves the distances between any pair of points with small distortions. 3-) It is data independent unlike PCA or LDA. 4-) It provides classifier output variability by which we can combine the outputs of several classifiers having different random projectors.

Random projection matrices could be constructed in several ways. For instance, each entry of the matrix is selected from the zero mean and i.i.d. random variables having Normal distribution. Then, each row of the matrix is normalized to unit length. This kind of projection was successfully applied in face recognition in [2]. Another way is to select each entry from i.i.d. random variables from the Bernoulli distribution. Both ways have similar mathematical properties. Storing Bernoulli random variables (integers) obviously requires less memory than storing Normal random variables (double size numbers). Achlioptas [87] proposed sparse random projections in which entries are selected from $\{+\sqrt{3}, 0, -\sqrt{3}\}$ with probabilities $\{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\}$. A more generic version of this technique was proposed in [88]. It is called very sparse random projection in which entries are selected from $\{+\sqrt{S}, 0, -\sqrt{S}\}$ with probabilities $\{\frac{1}{2S}, 1 - \frac{1}{S}, \frac{1}{2S}\}$ where S is a positive integer. Note that the random projection matrices from the Bernoulli distribution and variants are normalized by $\sqrt{m}$ to produce unit length row. In this thesis, we use the last technique in which we can adjust the sparsity of the input features by varying the S parameter. Also, if we are working on a device having limited storage capacity, aggressive choice of S (e.g. 100) will highly reduce the size of the random projection matrix (We can only store the locations of $\{+\sqrt{S}\}$s and $\{-\sqrt{S}\}$s) without losing significant discriminative features.

5.4. Recognition with a Single Classifier

Assume that after random projection we obtained the lower dimensional representation of the dictionary as $\Phi \in \mathbb{R}^{B \times N}$ and the test face image as $y \in \mathbb{R}^{B}$. Then, one can use the following expression in order to identify y:

$$Id(y) \leftarrow BTC(\Phi, y, M, \alpha) \tag{5.1}$$

The details of the algorithm could be found in Chapter 3. Using the described quantity (3.12), the threshold value M can be estimated for any dataset. For instance, we computed $\overline{\beta}_M$ for Extended Yale-B dataset for 504 and 120 features. In Fig. 5.1, we plotted $\overline{\beta}_M$ by simply ranging M over the intervals (1, 120) and (1, 504). By observing the plotted curves, the optimum value of M in SIC sense could be deter-

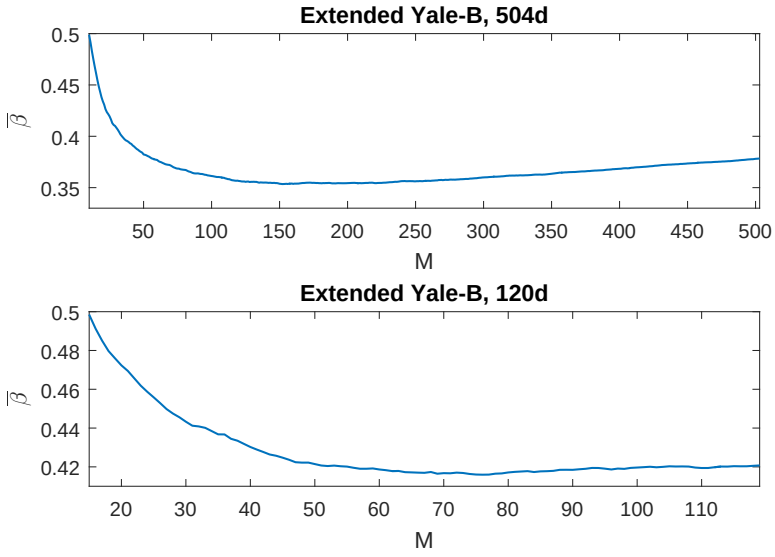


Figure 5.1: $\overline{\beta}_M$ vs threshold M on Extended Yale-B dataset for dimensions 120 and 512

mined approximately by picking the value which makes $\overline{\beta}_M$ minimum. Note that the detailed descriptions of the Extended Yale-B dataset is given in Section 5.6.

In generic classification problems, it is always desired to reduce the dimensionality of the input features. Here, the goal is not only to increase the discriminative properties of the features, but also to reduce the memory usage and computational burden. It is valid for face recognition as well. Consider the portable devices like cell phones and near feature intelligent robots for which energy requirement is always problematic. Therefore, the designers have to keep the processing capability of such intelligent devices in low levels. In this context, we need to construct reliable algorithms which are able to provide highly accurate results using only a small number of features. On the other hand, keeping the number of features small does not always provide good results especially for a single classifier. Immediate remedy of this problem is to combine the outputs of several classifiers. In this sense, we need high accuracy individual classifiers having enough level of output diversity. Those requirements are exactly met by individual BTC classifiers with different random projections. In the following section, we will develop an efficient fusion scheme by combining the outputs of individual BTC classifiers.

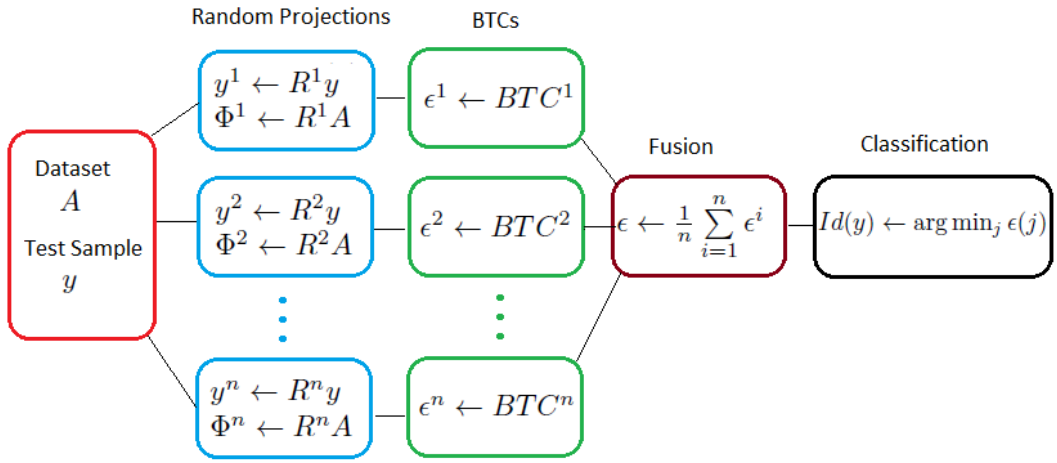


Figure 5.2: Classifier ensembles

5.5. Classifier Ensembles

In this part, we present a classifier fusion scheme to increase the classification accuracy further. In Fig. 5.2, we see how we can combine the outputs of n individual BTCs using parallel topology. This architecture is commonly used in the literature [89], [90], [91]. Assume that the random projector for the ith BTC is $R^i \in \mathbb{R}^{B \times m}$ and the corresponding residual vector containing the errors for each class is $\epsilon^i \in \mathbb{R}^C$. We can then combine the individual residuals to obtain one final residual vector. Here, the sample mean of the residuals, as indicated in [92] and successfully applied in [93], could be considered as a good combiner, that is, $\epsilon = \frac{1}{n} \sum_{i=1}^{n} \epsilon^i$. The technique used in here combining the intermediate function output instead of direct classification results is called support function fusion [89]. After obtaining the overall residual vector, final classification is done by selecting the class which has the minimum residual. **Algorithm** 5 describes the practical implementation of BTC-n. Note that the fusion technique given here is similar to the method used in [93]. Here, we use the outputs of individual BTCs instead of the outputs of l_1 minimizers.

Algorithm 5 BTC-n

INPUT:

Dictionary $A \in \mathbb{R}^{m \times N}$

Test sample $y \in \mathbb{R}^m$

Threshold parameter $M \in \mathbb{N}$

Regularization parameter $\alpha \in (0, 1)$

Random Projectors $R^i \in \mathbb{R}^{B \times m}$ $i \in \{1, 2, ..., n\}$

OUTPUT:

Identity of y

PROCEDURE:

$\epsilon(j) \leftarrow 0 \ \ \forall j \in \{1, 2, ..., C\}$

for $i \leftarrow 1, n$ **do**

    $\Phi^i \leftarrow R^i A$

    $y^i \leftarrow R^i y$

    $\epsilon^i \leftarrow BTC(\Phi^i, y^i, M, \alpha)$

    $\epsilon \leftarrow \epsilon + \epsilon^i$

end for

$\epsilon \leftarrow \epsilon / n$

$Id(y) \leftarrow \arg\min_j \epsilon(j)$

Table 5.1: Computed M values for each dataset and dimension

Dimension(d)	30	56	120	504
Extended Yale-B	29	48	88	172
Faces 94	11	11	33	21
Faces 95	11	26	36	38
Faces 96	11	13	35	43
ORL	6	6	6	6

5.6. Experimental Verification and Performance Comparison

In this section, we will compare the classification performances of SRC, OMP, COMP, and BTC as well as their ensembles, SRC-n, OMP-n, COMP-n, and BTC-n, on publicly available datasets namely Extended Yale-B, Faces 94, 95, 96, and ORL in face identification domain. We tested the performances using 30, 56, 120, and 504 features as used in [2] and [93]. Note that in the beginning of the experiments we computed the threshold values for each dataset and feature vector size according to properties of the datasets. Also note that whenever the dictionary changes, that is, some classes and samples are added or removed from it, the M value for that dictionary must be recalculated. The computed M values are shown in Table 5.1.

5.6.1 Results on Extended Yale-B

The Extended Yale-B face dataset contains 38 classes (subjects) each having about 64 frontal face samples with resolution 192 x 168. The samples have perfect alignment and different illuminations per individual (Fig. 5.3). As in [2] and [93], we randomly selected half of the samples (about 32 per class) for training and remaining half for testing to make sure that the results do not depend on any special configuration of the dictionary. We used the same random instance for all algorithms. For SRC method, we chose Homotopy l_1 minimizer during the experiments with tolerance 1e-12. We set the sparsity level of OMP algorithm to the average number of samples per class ($\overline{N}$) for all experiments. For BTC algorithm, we used the threshold values given in Table 5.1 and for all experiments we set the regularization constant α to 0.01. For face

Figure 5.3: Sample faces from Extended Yale-B dataset

identification, this is the value that we propose based on the experiment we provide on Extended Yale-B. As shown in Fig. 5.5, the classification accuracies are maximized at 0.01 for BTC and BTC-n techniques. We repeated the experiments 50 times with 50 different random projectors (Bernoulli random projections) for each method to make sure that results do not depend on any special choice of the projection matrix. The classification accuracies of each method are shown in Table 5.2. In multi-class problems, considering only classification accuracy may not be enough to show the real performance of a classifier as indicated in [94]. Therefore, we also added kappa (κ) statistic in Table 5.2 which measures pairwise degree of agreement among set of raters [95].

In the single classifier case, we see that the rates of SRC and BTC are very close to each other. In case of 30 and 504 features, BTC slightly outperforms the SRC method. On the other hand, SRC slightly performs better than BTC at 56d and 120d. The performance of OMP algorithm is quite low at dimensions 30d, 56d, and 120d as expected. We also added Cholesky-based OMP (COMP) which performs better than ordinary OMP. In the case of classifier ensembles using 5 individual classifiers, we see that the accuracies highly increase. For this case, BTC-5 outperforms the other

Table 5.2: Recognition rates and κ statistics on Extended Yale-B dataset

	Accuracy				κ statistics			
Dimension(d)	30	56	120	504	30	56	120	504
SRC [%]	83.47	**91.82**	**95.77**	97.40	83.02	**91.60**	**95.65**	97.33
OMP [%]	66.59	76.66	89.91	96.52	65.69	76.03	89.64	96.43
COMP [%]	72.23	86.75	93.24	97.03	71.48	86.40	93.05	96.95
BTC [%]	**83.98**	91.25	95.44	**98.14**	**83.54**	91.01	95.32	**98.09**
SRC-5 [%]	90.87	94.98	96.87	97.45	90.62	94.84	96.79	97.38
OMP-5 [%]	78.20	81.90	93.58	96.87	77.61	81.41	93.41	96.79
COMP-5 [%]	83.96	90.46	94.90	97.37	83.53	90.20	94.76	97.29
BTC-5 [%]	**92.51**	**95.97**	**97.45**	**98.84**	**92.31**	**95.86**	**97.38**	**98.81**

techniques at all dimensions. The ensemble technique namely *E-Random* or SRC-5 proposed in [93] achieves 90.72, 94.12, 96.35, 98.26 percent at the same dimensions, respectively, using 5 classifiers. we see that BTC-5 outperforms *E-Random* which is computationally quite expensive.

We also compared the computation times per sample for each method in order to further investigate the feasibility of each proposal. Fig. 5.4 shows the classification times per individual for each method at each dimension. Note that all experiments were performed in MATLAB on a workstation with dual quad-core 2.67 GHz Intel processors and 4GB of memory. As expected SRC is slow and OMP and COMP are fast. On the other hand, we see that single BTC is extremely fast as compared to the others. The speed of the proposal enables us efficiently use the ensemble technique which is superior to the single classifiers. The SRC-n approach is highly inefficient in terms of computational cost although its classification accuracy is high. We also compared the performances of the ensemble techniques with respect to the number of classifiers using only 30 features in Fig. 5.6. We observe that for all cases BTC-n is superior to the other methods. Up to 5 classifiers, the classification accuracies immediately increase. Based on this observation, combining 5 or 10 classifiers could be good choice depending on the speed of the algorithm. The corresponding computation times for the ensemble techniques are given in Fig. 5.7. The results show that the computation performance differences between BTC-n and the other approaches

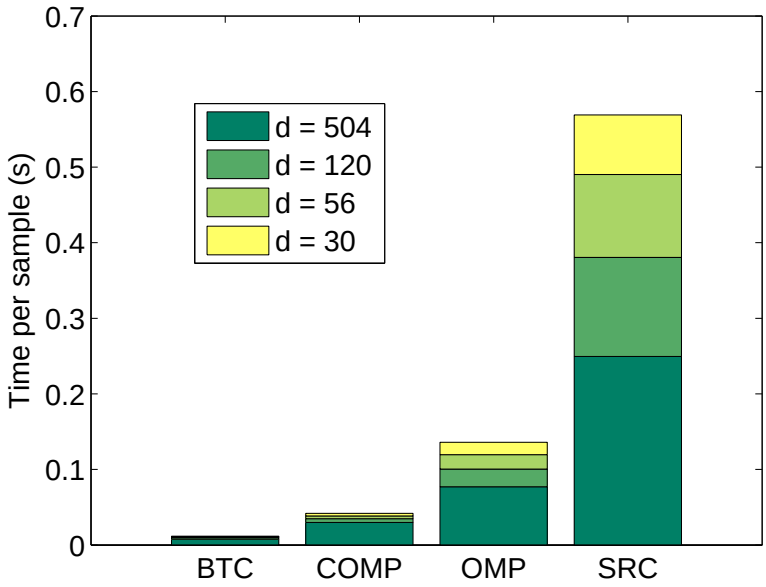


Figure 5.4: Classification times on Extended Yale-B

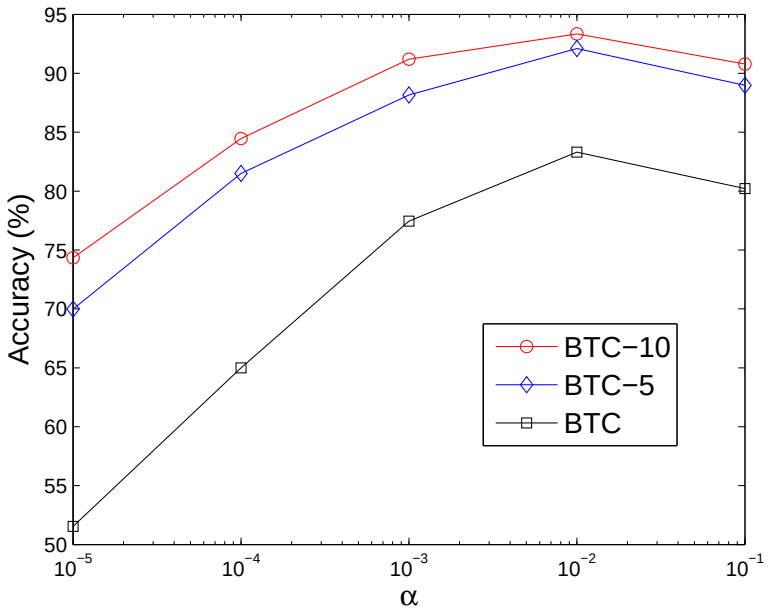


Figure 5.5: Classification accuracies with respect to regularization parameter (α)

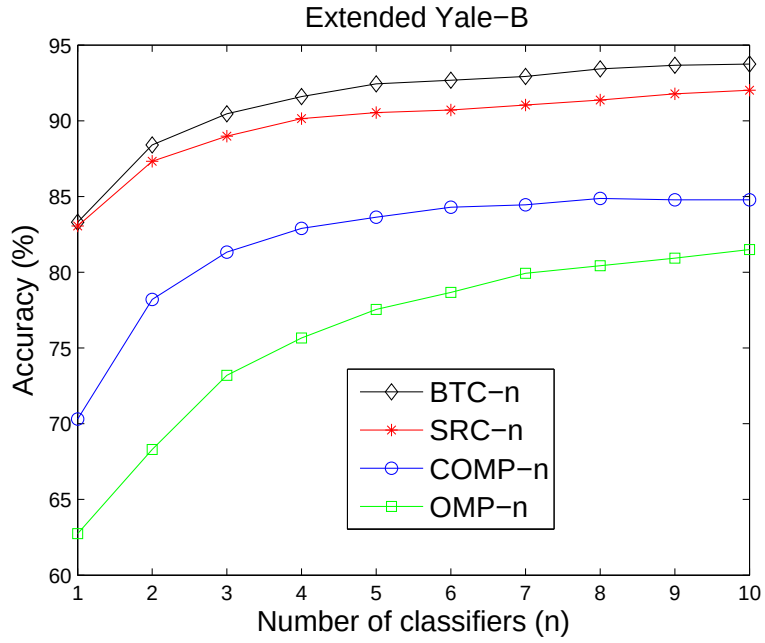


Figure 5.6: Classification accuracies on Extended Yale-B using ensemble techniques

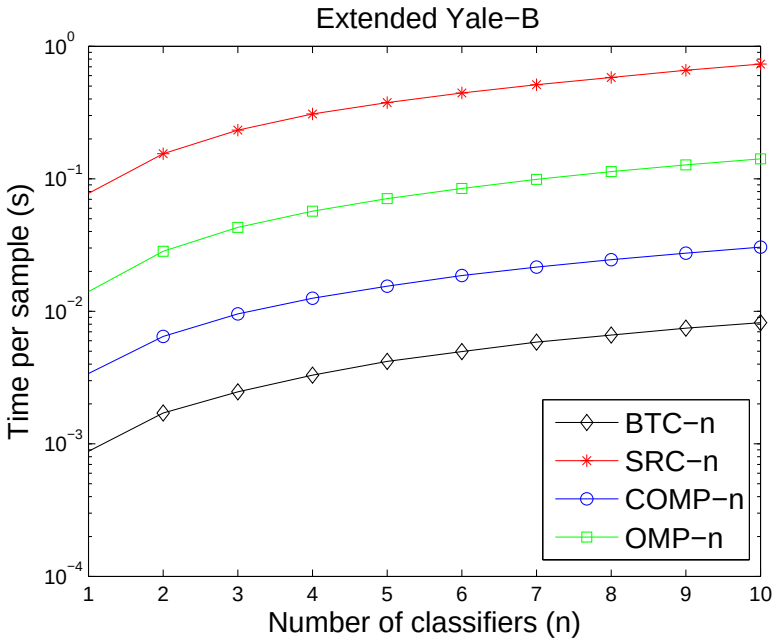


Figure 5.7: Classification times on Extended Yale-B using ensemble techniques

Table 5.3: Description of each dataset

Dataset	Difficulty	Resolution	Captured resolution	Num. of classes	Tr. smp. per class	Te. smp. per class
Faces 94	Easy	180×200	123×123	152	10	10
Faces 95	Medium	180×200	75×75	72	10	10
Faces 96	Hard	196×196	98×98	151	10	10
ORL	Medium	9×112	92×112	40	5	5

are significant.

5.6.2 Results on Faces 94, 95, 96, and ORL

The goal of the experiments on Faces 94, 95, and 96 is to compare the classification performances under automatic object detection framework. Note that we used *Viola-Jones* detector to capture the faces [96]. Since the detector that we use is not able to perfectly capture and align the test samples, the maximum identification rates of the classifiers are decreased. Because of this reason, we focus on the performance differences rather than the maximum rates achieved. Brief description of the datasets could be seen in Table 5.3. The detailed description of each dataset (Faces 94, 95, 96) could be found in [97]. Notice that this time we have more realistic scenarios which contain misalignments, head scale, expression, and illumination variations in the cropped faces. We have also chance to compare the performances under ORL which has a few (5) training samples per subject. Configurations for these experiments were similar to the previous ones. The only differences were the number of classes and samples for each dataset.

Fig. 5.8 shows the recognition rates on Faces 94 dataset for SRC-n, OMP-n, COMP-n, and BTC-n methods with respect to the number of classifiers using 30 features. As we see in the figure, BTC-n outperforms the other techniques as expected. Notice that this time, OMP-n's and COMP-n's performances are acceptable as compared to the results on Extended Yale-B. This is because the illumination variations are not significant which corresponds to less-noisy sparse recovery. They even outperform the SRC-n method. However, previous results on Extended Yale-B show that those approaches are not as robust as the SRC-n and BTC-n techniques.

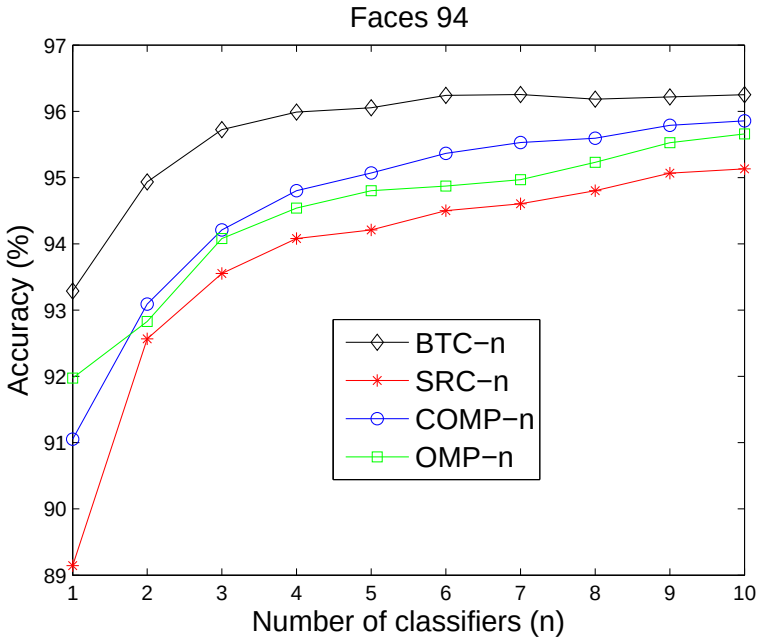


Figure 5.8: Recognition rates on Faces 94

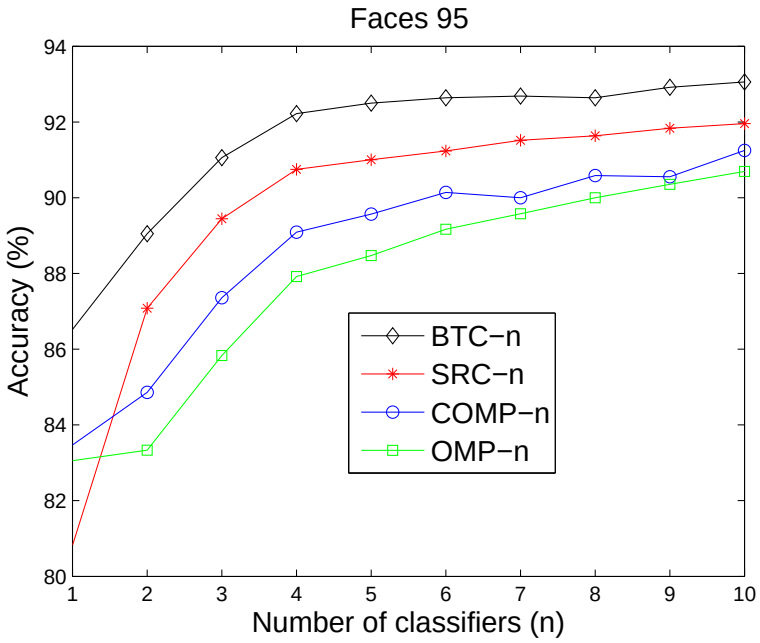


Figure 5.9: Recognition rates on Faces 95

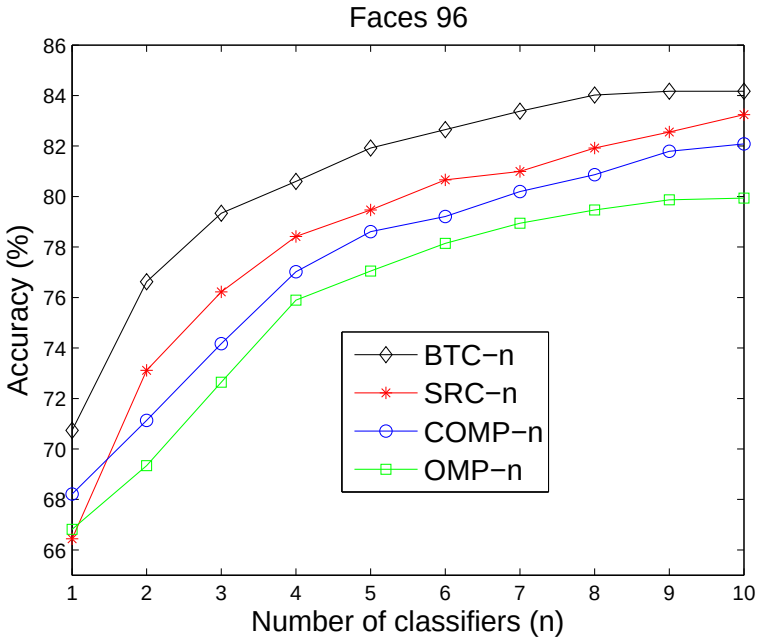


Figure 5.10: Recognition rates on Faces 96

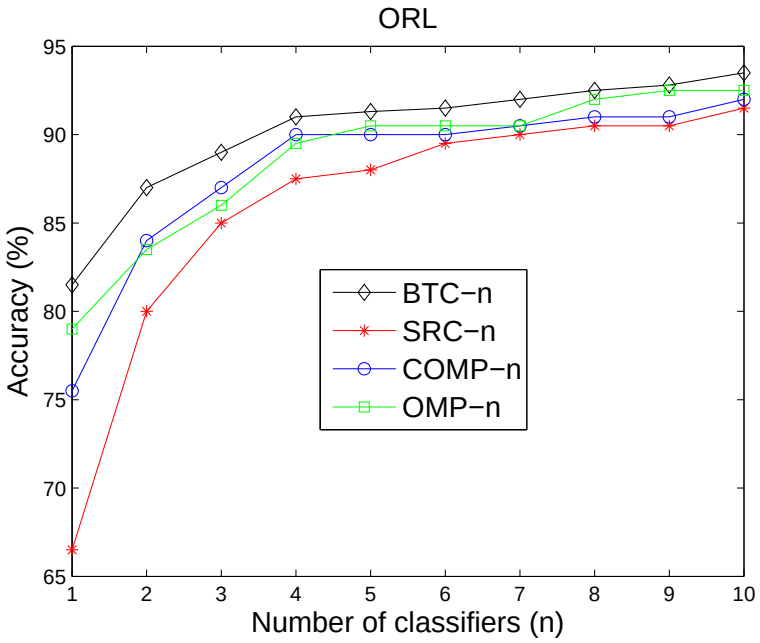


Figure 5.11: Recognition rates on ORL

In Fig. 5.9 we see the classification performances on Faces 95 dataset. This dataset is more difficult than the previous one. Therefore, the performances of all methods slightly decreased as compared to the previous case. This time also BTC-n technique outperforms the others. We observe that SRC-n is superior to the OMP-n and COMP-n techniques because of the difficulty of the dataset.

In Fig. 5.10 we see the classification accuracies on Faces 96 dataset which could be considered the most difficult one. This time the performances of the all methods decreased. The performance differences are similar to those of the previous experiment. The best results also were obtained by BTC-n.

Fig. 5.11 shows the classification performances on ORL dataset. As expected BTC-n achieves best results as in the previous cases. This time the performances of BTC-n, OMP-n, and COMP-n are very close. However, BTC-n slightly outperforms the OMP-n and COMP-n methods. SRC-n achieves the lowest rates. This shows that SRC-n is vulnerable to the number of samples per subject especially when the dictionary has very small number of samples per class.

5.6.3 Comparison with the Correlation Classifier

One could wonder what happens if we directly use simple correlations instead of sparse representation. For this purpose, we designed an algorithm namely simple correlation classifier (CORR) which performs the following steps:

- Find the correlation vector v containing the M largest linear correlations between the test sample y and the samples of all training set $A \in \mathbb{R}^{B \times N}$.
- Set the remaining $N - M$ entries in v to zero.
- Perform classification using the sum of the linear correlations within each class, that is,

$$class(y) = \arg\max_j \sum_i v_j(i) \;\; \forall j \in \{1, 2, \ldots, C\} \tag{5.2}$$

  where i represents the correlation index within a class and j shows the class index.

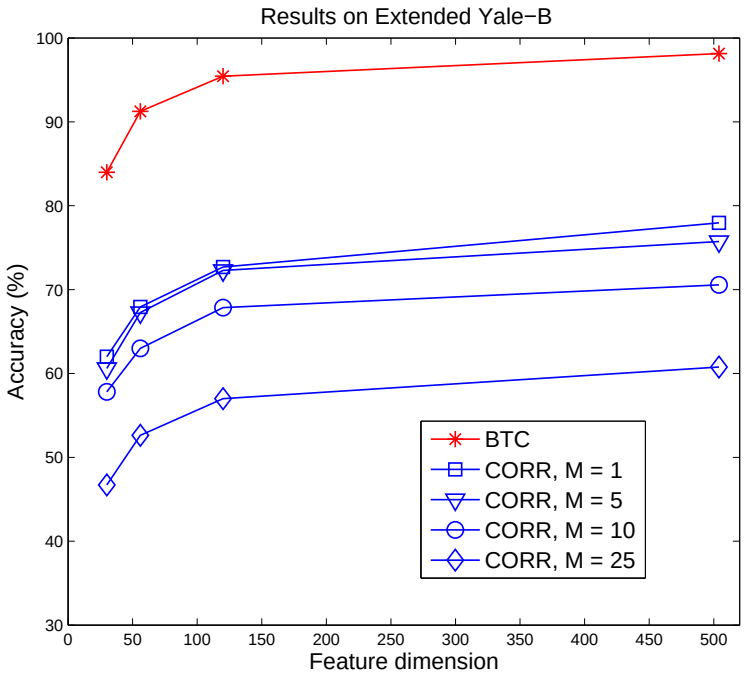


Figure 5.12: Comparison with correlation classifier (Extended Yale-B)

Using this method, we repeated the same experiment previously performed on Extended Yale-B dataset. The recognition rates for different feature vector sizes and threshold values (M) could be seen in Fig. 5.12. We can observe that the best results are achieved when M is set to 1. This means that the best policy is to find the class label of the dictionary element having the highest correlation with the test sample. Note that we could have used the majority voting (MV) technique instead of the sum operation. However, the sum operation is superior to the MV technique.

By observing the results, we can also compare the performance of this intuitive method with that of the proposed algorithm. We see that the accuracies obtained by BTC are far beyond the CORR approach. This experiment not only shows the superiority of BTC but also the power of the sparse representation.

5.6.4 Rejecting Invalid Test Samples

Evaluating a classifier considering only classification performance and computational cost is not enough in real world applications. A classifier must also be able to reject

invalid test samples and correctly classify the valid ones at the same time. Up to now we only considered the cases where the test sample belongs to one of the classes in the dictionary. This time we consider a test sample which does not belong to any of the classes. Rejection could be performed by using a predefined threshold value. In this thesis, we propose a validation mechanism based on the residual vector which is produced when the test sample y is applied to the BTC algorithm. As we stated previously, the residual vector contains entries for all classes in the dictionary. In this context, we define the following measure for any test sample y.

$$\gamma(y) \triangleq 1 - \frac{\epsilon(i)}{\epsilon(j)} \tag{5.3}$$

where $i = \arg\min_k \epsilon(k)$ and $j = \arg\min_{k \neq i} \epsilon(k)$. Here, i and j are simply the class indexes which give the smallest and the second smallest residuals, respectively. Notice that the ratio in (5.3) is the natural result of $\overline{\beta}_M$ that we mentioned previously. Assume that we apply y to the BTC algorithm and obtain $\gamma(y)$ being close to 1. Then, we say that with high probability it belongs to one of the classes. If the result is close to 0, then we say that it probably does not belong to any of the classes. Let us define $\tau \in (0, 1)$ as the rejection threshold. If the following condition is not satisfied, then the test sample is rejected.

$$\gamma(y) \geq \tau \tag{5.4}$$

Unlike the rejection rule defined here, SRC algorithm uses the *Sparsity Concentration Index* (SCI) to reject invalid test samples [2]. It is based on the estimated sparse code x. The details of the rule could be found in [2]. In order to compare the rejection performances of BTC, SRC, OMP, and COMP we designed a new experimental configuration. This time in the experiments, we included Extended Yale-B, Faces 94, 95, and 96 at the dimension 120d. We also included only half of the classes in the training sets. However, the test sets remained the same. Therefore, half of the classes and their samples in the test sets were invalid for the new dictionaries. Since the dictionaries were redesigned, we recalculated the M values which are 72, 11, 33, 24 for Extended Yale-B, Faces 94, 95, and 96, respectively. We then generated the normalized Receiver Operating Characteristics (ROC) curves for each dataset and algorithm by simply sweeping τ over the range (0,1). Note that for SRC, OMP, and COMP we used the SCI rule.

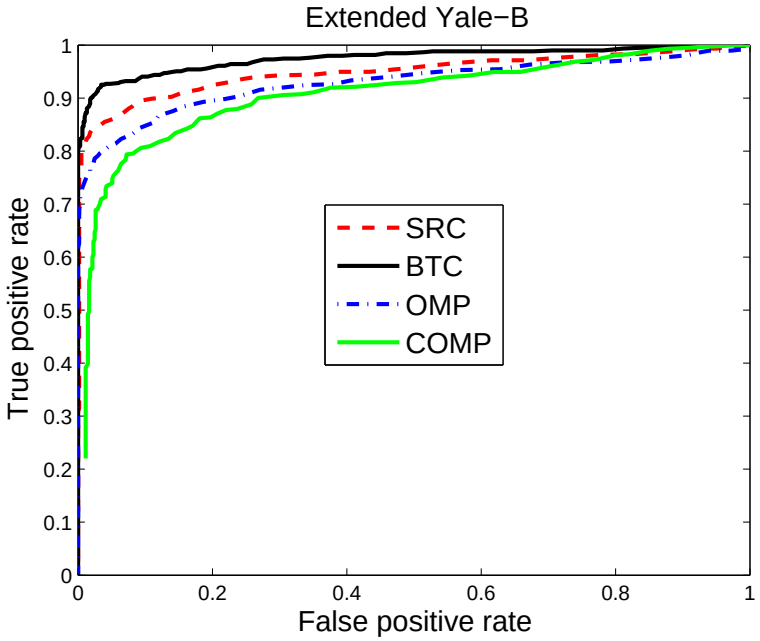


Figure 5.13: ROC curves on Extended Yale-B

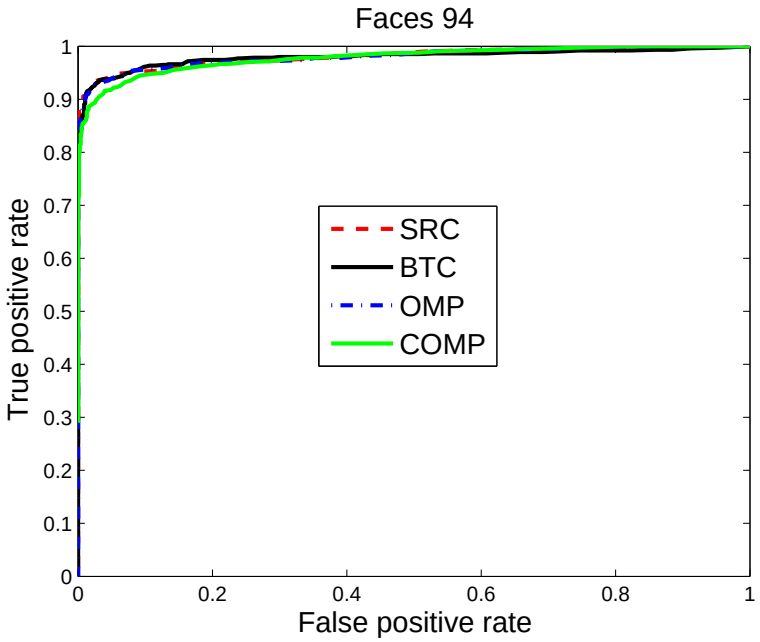


Figure 5.14: ROC curves on Faces 94

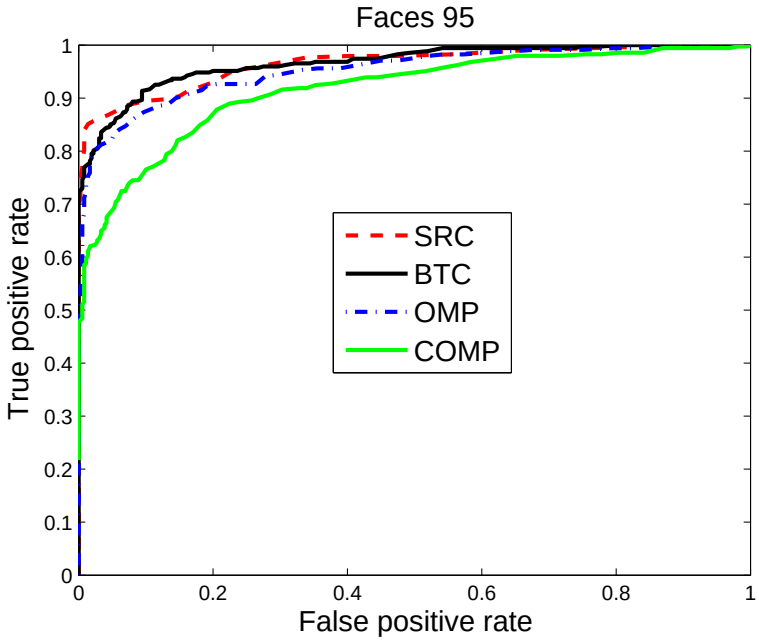


Figure 5.15: ROC curves on Faces 95

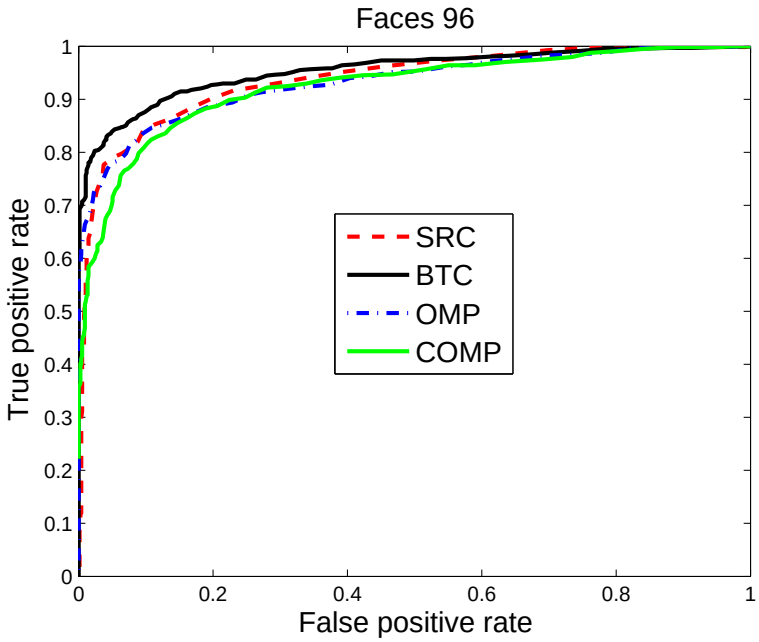


Figure 5.16: ROC curves on Faces 96

Fig. 5.13 shows the ROC curves for Extended Yale-B dataset. In this set, BTC significantly outperforms the other algorithms. Since the dataset Faces 94 is an easy one, all algorithms generate nearly the same curves in Fig. 5.14. In Fig. 5.15, we see the results for Faces 95. In this case, up to 0.1 false positive rate, SRC slightly outperforms BTC. After that point, BTC performs better than SRC up to 0.3. After this rate, the performances are nearly the same. Finally, Fig. 5.16 shows the ROC curves for Faces 96, which was the most difficult one. This time again BTC algorithm achieves the best results.

5.7. Conclusions

In this chapter, we introduced basic thresholding classification for face recognition which has significant speed, accuracy, and rejection performance improvements over l_1-minimization-based and greedy approaches. One of the main contributions is that unlike most of the classification algorithms, the computation performance of the proposed algorithm does not depend on the number of classes in the dictionary. It depends only on the feature vector size which is already intended to be small. By exploiting the output diversity property of the random projections and speed of the proposed algorithm, we developed a classifier ensemble mechanism which efficiently combines the outputs of the individual classifiers to further increase the classification accuracy especially in the case of small feature vector size. The ensemble technique also enables us to run each individual classifier in parallel manner. Finally, in order to reject invalid samples, we presented a rejection methodology which was actually developed by using the natural results of the sufficient identification condition rate. We demonstrated the performance and robustness of the algorithm under various well-known publicly available datasets. Simulation results showed that basic thresholding classification is a quite reasonable alternative to some state-of-the-art l_1-minimization-based and greedy classifiers.

CHAPTER 6

HYPER-SPECTRAL IMAGE CLASSIFICATION VIA BTC

6.1. Introduction

Different materials on the surface of earth have different electromagnetic spectral signatures. In a given scene, those signatures could be captured by remote sensors with a small spatial and spectral resolution. Each pixel of a captured hyper-spectral image (HSI) or data cube contains very useful spectral measurements or features which could be used to distinguish different materials and objects. With the advancement of the sensor technology, current sensors are able to capture hundreds of spectral measurements. However, increasing the number of spectral features of a HSI pixel does not always help to increase the correct classification rate of the pixel. For instance, in the supervised classification techniques, at some point increasing the input feature vector size further may reduce the classification accuracy. This is known as high dimensionality problem or Hughes phenomenon [98].

Several dimension reduction techniques have been proposed to eliminate the effects of Hughes phenomenon [99], [100], [101]. Besides, to increase the classification accuracy, various approaches have been proposed. Among those, support vector machines (SVM) technique outperformed classical methods such as K-nearest neighbor (K-NN) and radial basis function (RBF) networks [47]. It has been shown that SVM distinguishes in terms of classification accuracy, computational cost, low vulnerability to Hughes phenomenon, and it requires a few training samples [47], [71]. On the other hand, SVM approach has some limitations. First, it has parameter tuning (C, γ, error tolerance) and kernel (linear, RBF, etc.) selection steps which are done using

k-fold cross validation in which some portion of the training data is used for testing purposes. Those procedures are cumbersome and the resulting parameter set may not be optimum for the test sets [102]. Second, since SVM is a binary classifier, a conversion strategy to multi-class case is required. An easy one is the one-against-all (OAA) strategy in which a test sample may result as unclassified which causes low classification accuracies [102]. Another strategy is the one-against-one approach in which number of binary classifiers ($K(K-1)/2$) increases dramatically as the number of classes (K) increases. The final limitation is that the probability outputs of the SVM classifier can not be directly provided and an estimation procedure is required such as logistic sigmoid [103]. Therefore, the SVM-based methods using probability outputs must rely on those estimates.

The SVM approach described above is in the class of pixel-wise algorithms since it uses only the spectral features. It is well known that the performance of a pixel-wise classifier could be improved by incorporating spatial information based on the fact that neighboring pixels in the homogeneous regions of a HSI have similar spectral signatures. Therefore, various approaches have been proposed to combine the spectral and spatial information. For instance, a composite kernels approach has been proposed in [104] which successfully enhances the classification accuracy of the SVM. A segmentation-based technique proposed in [105] combines the segmentation maps obtained via clustering and pixel-wise classification results of the SVM technique. Final decisions are made by majority voting in the adaptively defined windows. A similar framework has been proposed in [106] which utilizes segmentation maps using watershed transformation. All these methods share common limitations since they are based on the SVM approach.

One of the recent spatial-spectral frameworks, which utilizes an edge-preserving filter, has been proposed in [107]. The method uses the SVM classifier as the pixel-wise classification step. For each class, the probability maps, which are the posterior probability outputs of the SVM classifier, are smoothed by an edge-preserving filter with a gray scale or rgb guidance image. Final decision for each pixel is then made based on the maximum probabilities. As an edge-preserving filter, they use one of the recent state-of-the-art techniques namely guided image filtering [108]. Since the proposed framework is based on SVM, it has also common problems with the

SVM-based classifiers. One alternative to SVM classifier is multinomial logistic regression (MLR)[109] method in which class posterior probability distributions are learned using Bayesian framework. MLR has been successfully applied to HSI classification in [110], [111], and [112]. One of the recent techniques based on MLR has been proposed in [113]. The method uses logistic regression via splitting and augmented Lagrangian (LORSAL) [114] algorithm with active learning in order to estimate the posterior distributions. In the segmentation stage, it utilizes a multilevel logistic (MLL) prior to encode the spatial information. LORSAL-MLL (L-MLL) technique achieves promising results as compared to classical segmentation methods.

Recently, sparsity-based methods, sparse representation-based classification (SRC) and joint SRC (J-SRC), alternative to SVM-based frameworks have been successfully applied to HSI classification [115, 115, 116, 117, 118, 119]. SRC originally was proposed for face identification in [2]. Since SRC is based on l_1 minimization which includes solving costly convex optimization problem, greedy algorithms like orthogonal matching pursuit (OMP) [78] and simultaneous OMP (SOMP) have been preferred for HSI classification in [115]. SOMP was originally proposed in [37] for generic simultaneous sparse information recovery. HSI version of SOMP is based on a joint sparsity model assuming that the pixels in the small neighborhood of a test pixel share a common sparsity pattern. Those pixels are simultaneously represented by the linear combinations of the training samples of a predetermined dictionary. Kernelized versions of SOMP have been developed in [116], [120], and [121] to exploit the non-linear nature of the kernels for a better class separability. Another version of SOMP namely weighted joint sparsity (W-JSM) or WSOMP has been proposed in [122]. The method calculates a non-local weight matrix for neighboring pixels of each test pixel. It then executes the standard SOMP with the calculated weight matrix. Multi-scale adaptive sparse representation (MASR), which utilizes spatial information at multiple scales, has been proposed in [123]. A final one is the adaptive SOMP (ASOMP) which adaptively selects the neighborhood of the test pixel according to a predetermined segmentation map [124]. All SOMP-based approaches have several common drawbacks. The most important one is the extensive computational cost due to simultaneous sparse recovery of the surrounding pixels of the test sample. Another limitation is the parameter tuning step in which the sparsity level K_0, error

tolerance ϵ, maximum iterations, and the weight thresholds are needed to be tuned experimentally. Once the parameters are determined for a dataset, they may not be optimum for some other datasets. Therefore, there is no guidance for the parameter selection.

In this chapter, we propose the basic thresholding classifier (BTC) for HSI classification. It is a pixel-wise light-weight method which classifies every pixel of an HSI image using only spectral features. During the classification it uses a predetermined dictionary containing labeled training pixels. For each pixel, it produces two outputs which are the error vector consisting of the residuals and the class label selected based on the minimal residual. To improve the classification accuracy of BTC, we extend our framework to a three-step spatial-spectral procedure. First, we run pixel-wise classification step using BTC for each pixel of a given HSI. The output residual vectors form a cube which is also interpreted as a stack of images. Every image is also called as residual map. Secondly, we smooth every residual map using an averaging filter. In the final step, we determine the class label of each pixel based on the minimal residual. The contribution of this chapter is threefold:

- We introduce a new sparsity-based algorithm for HSI classification which is light-weight, cost effective, easy to implement, and provides high classification accuracy. Unlike classical approaches such as minimum distance, K-NN, and SVM techniques, our method has low vulnerability to the corrupted, noisy, and partial features in the test samples since it is based on sparse representation which exploits the fact that the errors due to these kinds of features are often sparse with respect to the standard basis [2]. It also distinguishes from previous sparsity-based techniques in its ability to classify test pixels extremely rapidly.

- The proposed approach eliminates the limitations of well-known SVM technique. First, unlike SVM, it does not have training and cross-validation stages. We give the full guidance of threshold and regularization parameter selection of the BTC method. On the other hand, the parameters of SVM (C, γ) have to be determined using cross-validation which may result in a non-optimal set. Second, the computational cost of BTC does not significantly increase as the number of classes (K) increases. However, since the number of binary classifiers

is dependent on the square of K in one of the common conversion strategies (OAO) of SVM to multi-class case, the cost of SVM dramatically increases as K increases. Finally, SVM does not provide residuals which might be used for intermediate processing such as smoothing.

- Our proposal can easily be extended to spatial-spectral case by smoothing the residual maps. This procedure eliminates high computational cost of joint sparsity model or SOMP-based techniques in which simultaneous sparse code recovery is essential. This low cost intermediate process extremely increases the classification accuracy of the proposed method. It is even able to outperform non-linear SVM-based techniques which use direct classification output maps of the spectral-only SVM.

6.2. HSI Classification

In the context of HSI classification, spectral measurements are embedded into B dimensional feature vectors. Let $a_{i,j} \in \mathbb{R}^B$ with $\|a_{i,j}\|_2 = 1$ be the vector consisting of the spectral features of the jth sample of the class i and let $A_i = [a_{i,1}\ a_{i,2}\ \dots\ a_{i,N_i}] \in \mathbb{R}^{B \times N_i}$ denote the matrix containing the training pixels of the ith class with N_i many pixels. Then, one can construct the dictionary A with C many classes in a way that $A = [A_1\ A_2\ \dots\ A_C] \in \mathbb{R}^{B \times N}$ where $N = \sum_{i=1}^{C} N_i$.

Suppose that we constructed an HSI dictionary $A \in \mathbb{R}^{B \times N}$ and we are given a test pixel $y \in \mathbb{R}^B$ to be classified. In the sparse representation model, the assumption is that there exists a minimum l_1-norm sparse code $x \in \mathbb{R}^N$ such that $y = Ax$ or $y = Ax + \sigma$ where σ is a small error [2]. The problem is equivalent to

$$\hat{x} = \arg\min_x \|x\|_1 \ \ subject\ to\ y = Ax \tag{6.1}$$

or alternatively subject to $\|y - Ax\|_2 \leq \sigma$. The class of y is then found using the following expression.

$$class(y) = \arg\min_i \|y - A\hat{x_i}\|_2 \ \ \forall i \in \{1, 2, \dots, C\} \tag{6.2}$$

where $\hat{x_i}$ represents the ith class portion of the estimated sparse code $\hat{x}$. As we

stated previously, solving (6.1) is not an easy problem. l_1-minimization or convex relaxation-based techniques such as homotopy method [82] could be used. However, those techniques are quite expensive for the applications such as HSI classification. A faster way of solving (6.1) is the OMP technique which has been successfully applied in HSI classification [115]. When we use OMP, the expression in (6.1) is replaced with (6.3).

$$\hat{x} = OMP(A, y) \tag{6.3}$$

The details of the OMP algorithm could be found in [78]. In order to incorporate spatial information, Chen *et al.* proposed the joint sparsity model (JSM) in [116]. In this case, not only the test pixel y is used but also the surrounding $n-1$ pixels $y^1, y^2, \ldots, y^{n-1}$ in a given window T are used in the minimization problem where the corresponding sparse codes are $x^1, x^2, \ldots, x^{n-1}$. This time expression (6.1) is replaced with the following:

$$\hat{X} = \arg\min_X \|X\|_{row,0} \; subject\, to\, Y = AX \tag{6.4}$$

or alternatively subject to $\|Y - AX\|_F \leq \sigma$ where $X = [x\ x^1\ x^2\ \ldots\ x^{n-1}]$, $Y = [y\ y^1\ y^2\ \ldots\ y^{n-1}]$, and $\|X\|_{row,0}$ is the number of nonzero rows. The class of y is then found using the following expression:

$$class(y) = \arg\min_i \left\| Y - A\hat{X}_i \right\|_F \; \forall i \in \{1, 2, \ldots, C\} \tag{6.5}$$

where $\hat{X}_i$ represents the ith class portion of the estimated sparse code matrix $\hat{X}$. For this case, greedy SOMP algorithm is preferred to solve (6.4) in [115]. The details of SOMP could be found both in [115] and [37]. As we stated previously, although SOMP technique is a greedy approach, it is computationally expensive since it simultaneously recovers the sparse codes of test and surrounding pixels.

All the limitations with the SVM and SOMP algorithms force us to use BTC for HSI classification [125]. One can use the following expression in order to classify the given pixel y:

$$Class(y) \leftarrow BTC(A, y, M, \alpha) \tag{6.6}$$

Note that the details of proposed technique could be found in Chapter 3. Before using the BTC algorithm, first, we need to determine the parameters M and α. In the

following section, we will show how the parameters are selected using the quantity $\overline{\beta}_M$.

6.3. Parameter Selection

We plotted $\overline{\beta}_M$s for the dictionaries constructed using some publicly available well-known hyper-spectral images. The $\overline{\beta}_M$s for the dictionaries of Indian Pines, Salinas and Pavia University images are given in Fig. 6.1. The detailed description of each dataset and the corresponding dictionaries are given in Section 6.5. We see that in all plots of Fig. 6.1, $\overline{\beta}_M$ decays and reaches approximately to some minimum value. As we stated previously, any M at which $\overline{\beta}_M$ is close to the minimum value is an acceptable choice. However, we need to consider that increasing M will also increase the computational cost. For the given plots, we selected the regularization constant α as 10^{-4} which is a good choice for pixel-wise HSI classification. The effects of α could also be seen in the figures. Without α, the decaying $\overline{\beta}_M$ would start to increase at some point due to the noisy eigenvalues which reduce the classification performance. One could think that the optimum choice of α is quite critical for BTC. It is important in spectral-only classification, however, in the following part, we will show that the effects of it will be compensated by the post processing smoothing in the spatial extension case. Therefore, using the optimal choice of α will not be critical anymore. Instead of optimal choice, we will prefer a quite small α in order to avoid an ill-conditioned matrix operation.

6.4. Extension to Spatial-Spectral BTC

In this section, we extend our pixel-wise proposal to a three-step spatial-spectral framework in order to incorporate spatial information. In the first step, each pixel $y \in \mathbb{R}^B$ in a given HSI, $H \in \mathbb{R}^{n_1 \times n_2 \times B}$ containing $n_1 \times n_2$ pixels, is classified using BTC. The produced outputs are not only the class labels but also the residual vectors, $\epsilon \in \mathbb{R}^C$, for all pixels. The resulting residual vectors form a residual cube $R \in \mathbb{R}^{n_1 \times n_2 \times C}$ which could also be interpreted as a stack of images representing residual maps ($map_i \in \mathbb{R}^{n_1 \times n_2}$ for all $i \in \{1, 2, \ldots, C\}$). In the second step,

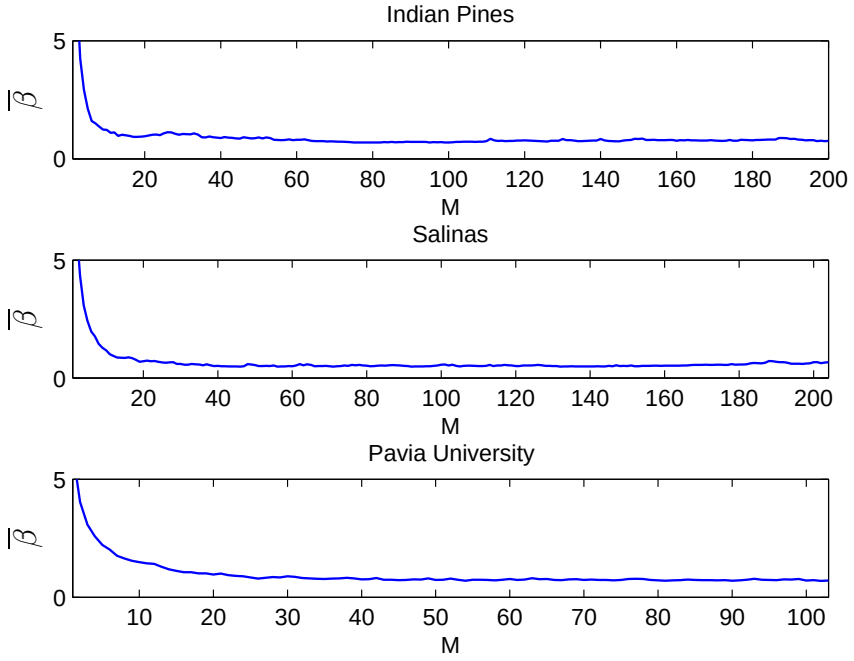


Figure 6.1: $\overline{\beta}_M$ vs threshold for Indian Pines, Salinas, and Pavia University ($\alpha = 10^{-4}$)

each map_i for all $i \in \{1, 2, \ldots, C\}$ is smoothed via an averaging filter producing $\overline{map}_i$ for all $i \in \{1, 2, \ldots, C\}$. Before smoothing operation, the values in the residual cube R are normalized between 0 and 1. Also note that by using the intermediate classification map of spectral-only BTC, we simply set the residual values to the maximum value 1 in map_i for the entries whose labels are not equal to i. This improves the classification performance. The smoothed maps form a smoothed residual cube $\overline{R} \in \mathbb{R}^{n_1 \times n_2 \times C}$. In the final step, class label of each pixel is determined based on minimal smoothed residuals, that is, $class(y) = \arg\min_i \overline{\epsilon}(i)$ where $i \in \{1, 2, \ldots, C\}$. The overall framework is shown in Fig. 6.2. For the spatial-spectral extension case, in which α is set to very small number (10^{-10}), we also plotted $\overline{\beta}_M$s in Fig. 6.3 for the dictionaries of the images used in this work. As we see in the figure, the decaying $\overline{\beta}_M$s start to increase at some point because of the noisy eigenvalues. As we stated in the previous part, the optimal choice of α for BTC in the spatial extension is not critical and a quite small value ($10^{-8}, 10^{-9}, 10^{-10}$, etc.) could be used only to prevent the singular matrix inverse. Also note that using a bit larger value such as 10^{-4} will reduce the classification accuracy of the spatial-spectral classifier since it eliminates

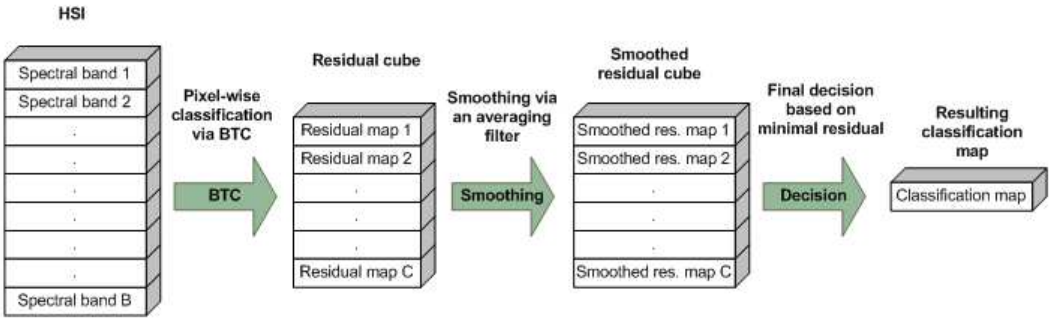


Figure 6.2: Spatial-Spectral BTC

Table 6.1: Description of each dataset

Dataset	Size	Spatial resolution	Spectral coverage	Num. of classes	Sensor	Num. of bands
Indian Pines	$145 \times 145 \times 220$	20 m	$0.4 - 2.5\ \mu$m	16	AVIRIS	200
Salinas	$512 \times 217 \times 224$	3.7 m	$0.4 - 2.5\ \mu$m	16	AVIRIS	204
Pavia University	$610 \times 340 \times 115$	1.5 m	$0.43 - 0.86\ \mu$m	9	ROSIS	103

some discriminative small eigenvalues. Therefore, in the spatial extension, we always prefer to use a very small regularization constant.

6.5. Experimental Results

6.5.1 Datasets

As in [107], we also used three well-known publicly available datasets namely Indian Pines, Salinas, and Pavia University in this work. We present the brief description of each dataset in Table 6.1. Both the Indian Pines and Salinas images were captured by Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor. The Pavia University image was captured by Reflective Optics System Imaging Spectrometer (ROSIS) sensor. Before the experiments, some noisy water absorption bands (20 bands for Indian Pines and Salinas, 12 bands for Pavia University) were discarded. The 3-Band color image, ground truth, each class and the corresponding number of training and test pixels are given for Indian Pines, Salinas, and Pavia University in Fig. 6.4, Fig. 6.5, and Fig. 6.6, respectively.

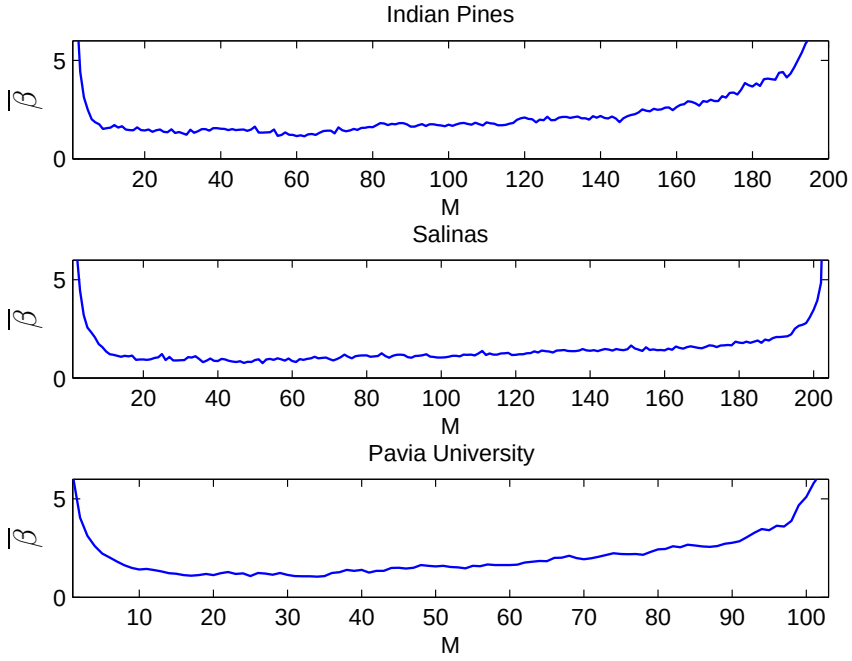


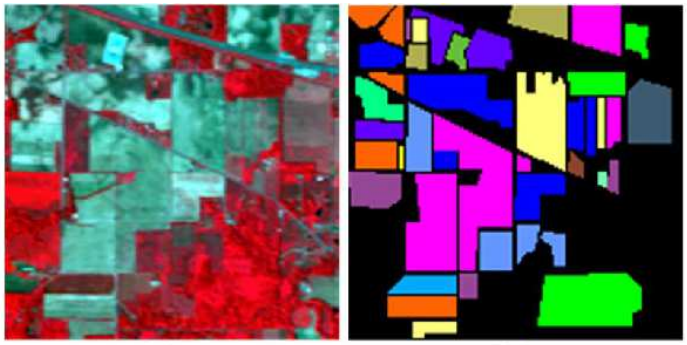
Figure 6.3: $\overline{\beta}_M$ vs threshold for Indian Pines, Salinas, and Pavia University ($\alpha = 10^{-10}$)

6.5.2 Performance Indexes

During the experiments, we used three commonly preferred performance indexes namely overall accuracy (OA), average accuracy (AA), and the kappa coefficient (κ). In addition to them, we also included the computation time. OA shows the percentage of the correctly classified samples and AA gives the average of the percentages of the correctly classified samples in each class. The κ coefficient is used to measure the degree of consistency [126]. The computation time is also an important measure which determines whether the classifier is feasible for the real time applications or not.

6.5.3 Experimental Setup

In the experiments, we included various spectral-only algorithms such as SVM [47], OMP [115], and BTC as well as some state-of-the-art spatial-spectral methods such as EPF-G-g [107], L-MLL [113], SOMP [115], and spatial-spectral BTC. EPF-G-g



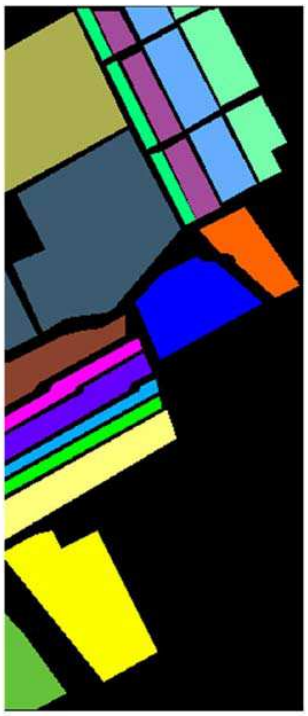
(a) (b)

Color	Class No	Class	Train	Test
	1	Alfalfa	10	36
	2	Corn-no till	143	1285
	3	Corn-min till	83	747
	4	Corn	24	213
	5	Grass/pasture	49	434
	6	Grass/trees	73	657
	7	Grass/pasture-mowed	10	18
	8	Hay-windrowed	48	430
	9	Oats	10	10
	10	Soybeans-no till	98	874
	11	Soybeans-min till	246	2209
	12	Soybeans-clean till	60	533
	13	Wheat	21	184
	14	Woods	127	1138
	15	Buildings-grass-tree-drives	39	347
	16	Stone-steel towers	10	83

(c)

Figure 6.4: a-) 3-Band color image b-) ground truth image, and c-) each class and the corresponding number of training and test pixels of Indian Pines dataset

(a)



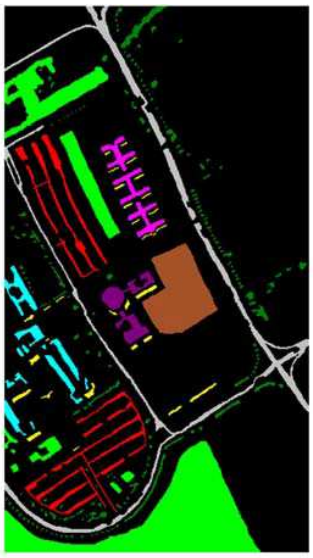
(b)

Color	Class No	Class	Train	Test
	1	Weeds 1	101	1908
	2	Weeds 2	187	3539
	3	Fallow plow	99	1877
	4	Fallow smooth	70	1324
	5	Stubble	134	2544
	6	Celery	198	3761
	7	Grapes untrained	179	3400
	8	Soil	564	10707
	9	Corn	311	5892
	10	Soybean-notill	164	3114
	11	Lettuce 4 wk	54	1014
	12	Lettuce 5 wk	97	1830
	13	Lettuce 6 wk	46	870
	14	Lettuce 7 wk	54	1016
	15	Vinyard untrained	364	6904
	16	Vinyard trellis	91	1716

(c)

Figure 6.5: a-) 3-Band color image b-) ground truth image, and c-) each class and the corresponding number of training and test pixels of Salinas dataset

(a)



(b)

Color	Class No	Class	Train	Test
	1	Asphalt	332	6299
	2	Meadows	933	17716
	3	Gravel	105	1994
	4	Trees	154	2910
	5	Metal sheets	68	1277
	6	Bare soil	252	4777
	7	Bitumen	67	1263
	8	Bricks	185	3497
	9	Shadows	48	899

(c)

Figure 6.6: a-) 3-Band color image b-) ground truth image, and c-) each class and the corresponding number of training and test pixels of Pavia University dataset

is based on SVM and guided image filtering. We used two versions of this method namely SVM-GF (based on guided filter [108]) and SVM-WLS (based on weighted least squares filtering [127]). For our spatial-spectral proposal, we also used the same filtering techniques in order to smooth the resulting residual maps and called the two versions of it as BTC-GF and BTC-WLS. For SVM-based classifiers, we used well-known and fast LIBSVM library which was written in C++ [128]. The parameters (C, γ) of SVM were chosen by 5-fold cross validation by varying C from 10^{-2} to 10^{4} and γ from 2^{-3} to 2^{4}. Since the non-linear RBF kernel is superior to the linear kernel (dot product), we used the RBF kernel for the SVM-based methods. In all experiments, for SVM-GF and BTC-GF, we set the filtering parameters namely filtering size (r) and blur degree (ϵ) to 3 and 0.01, respectively. Those values are proposed in [107] for HSI classification. Since both GF and WLS filters require a gray-scale guidance image, we obtained it by extracting the first principal component of the given HSI using the principal component analysis (PCA) [129] procedure. Since the L-MLL approach requires an active learning stage, we set the initial training set to the half of the all training set for all experiments. We then incremented the samples by 50 using random selection (RS) method up to the whole training set. The classification and segmentation stages were performed after the learning stage. For OMP and SOMP classifiers, we used the SPArsa toolbox provided by Julien Mairal [130]. There is no guidance for the selection of sparsity parameter (L) of the OMP and SOMP techniques. However, based on the experiments in [115] and [122], we set L to 25 for OMP and 30 for SOMP. Similarly, we set the spatial window size (T) to 25 for SOMP technique. Note that we used the same values for all experiments.

The reason that we include the WLS filter is because it does not cause halo artifacts at the edges of an image as the degree of smoothing increases [127]. However, the guided filter and bilateral filter [131] techniques tend to blur over the edges. Similar to the filtering techniques mentioned here, WLS filtering also has two input parameters namely the degree of smoothing (λ) and the degree of sharpening (α) over the preserved edges. Note that the reader should not confuse the parameter α here with the regularization parameter of BTC method. Since there is no guidance for the selection of WLS filtering parameters in HSI classification, in this thesis, our proposal is 0.4 for λ and 0.9 for α based on the experiment we performed on the Indian Pines dataset.

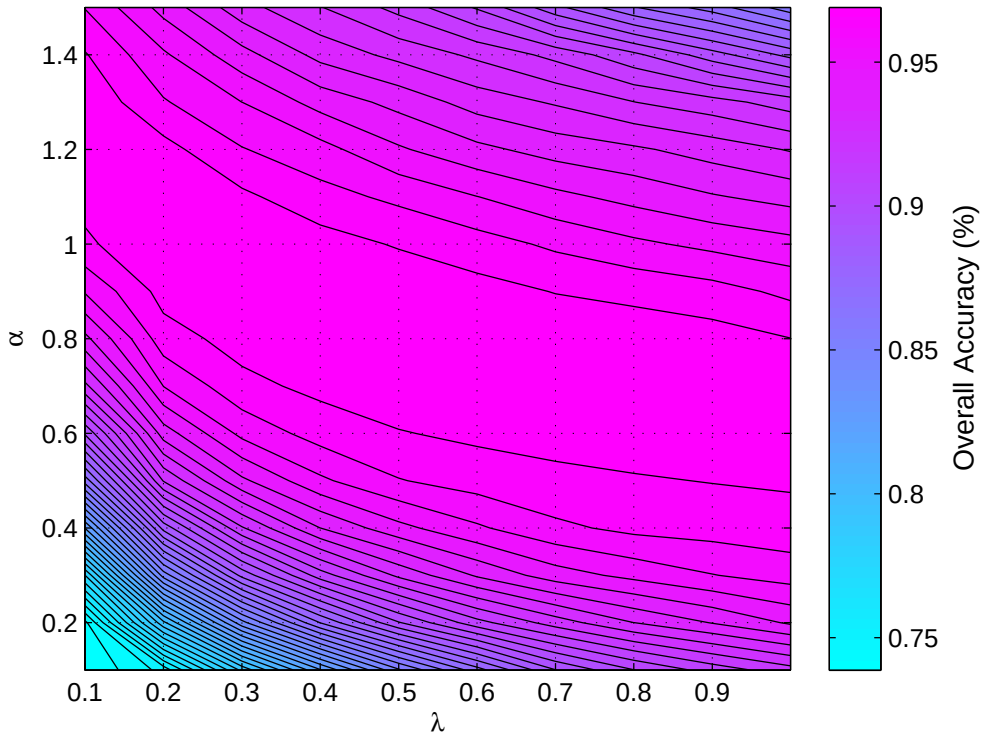


Figure 6.7: Overall accuracy (%) grid by varying the λ and α pair of WLS filter on Indian Pines dataset using BTC-WLS method

Note that the details of the experiment are presented in the following subsection. We obtained the metric OA by varying λ from 0.1 to 1.0 and α from 0.1 to 1.5 using the BTC-WLS method. The resulting OA grid is shown in Fig. 6.7. The wide contour in the middle of the figure shows the highest OA region. The coordinates of the points in this region are also acceptable choices for the WLS filtering in our case. Note that the WLS filter parameters are fixed to the proposed values ($\lambda = 0.4, \alpha = 0.9$) for all experiments.

Regarding the regularization parameter (α) of our proposal, as we pointed out in the previous section, we set it to 10^{-4} for the spectral-only BTC and 10^{-10} for BTC-GF and BTC-WLS techniques. Note that like spectral-only SVM and OMP methods, spectral-only BTC is also not a practical HSI classifier. We used those methods as the reference for our experiments. Therefore, the choice of proposed regularization parameter in the spectral-only case is not critical for real-world applications. However, for BTC-GF and BTC-WLS approaches, we should select a quite small number in order to only prevent ill-conditioned matrix operation. Since any quite small number is acceptable, there is no guidance required for the selection of this parameter.

For the choice of threshold parameter (M) of the proposed BTC-based techniques, once the dictionary is determined, one can immediately plot the $\overline{\beta}_M$ by varying M from 1 to the number of bands (B) available. Then, the best value of it in the described sense could be easily determined by looking at the resulting plot. Notice that the procedure does not require any cross validation or experiment and it is totally based on the predetermined dictionary. In the following subsections, we will provide the estimated M values for each experiment based on the $\overline{\beta}_M$ plots given in the previous section. A final note about the setup is that we performed all experiments on a PC with a quad-core 2.67 GHz processor and 4GB of memory.

6.5.4 Results on Indian Pines Dataset

We performed the first experiment on the Indian Pines dataset. We randomly selected 10% of the ground truth pixels for training set (dictionary) and the remaining 90% for testing. We limited the minimum number of training pixels to 10 for each class. Each class and the corresponding number of test and training pixels are given in Fig. 6.4c.

Table 6.2: The results (accuracy per class (%), OA (%), AA (%), κ (%), Time (s) of twenty Monte Carlo runs) for spectral-only and spatial-spectral methods on Indian Pines dataset

	Spectral-Only			Spatial-Spectral					
Class	SVM	OMP	BTC	SOMP	L-MLL	SVM-GF	SVM-WLS	BTC-GF	BTC-WLS
1	60.28	57.50	**76.11**	81.11	87.22	98.33	96.67	**100.00**	98.89
2	**75.86**	61.63	72.82	89.06	92.61	91.59	90.88	**95.67**	94.27
3	**65.19**	53.00	62.24	84.75	89.18	85.74	86.79	95.88	**97.51**
4	**47.70**	31.60	43.71	69.81	79.90	87.65	90.94	**95.40**	94.41
5	87.12	85.51	**88.23**	93.02	90.69	**95.46**	95.23	95.18	94.63
6	95.40	93.15	**96.26**	99.21	99.54	**100.00**	99.74	**100.00**	**100.00**
7	**88.89**	81.67	85.56	89.44	93.33	**97.22**	**97.22**	**97.22**	96.11
8	95.19	92.47	**97.33**	99.86	99.69	**100.00**	**100.00**	**100.00**	**100.00**
9	**91.00**	63.00	84.00	78.00	96.00	98.00	86.00	97.00	**100.00**
10	**74.45**	57.25	70.35	85.27	89.01	**93.83**	93.36	93.27	93.59
11	**83.73**	72.87	81.98	90.38	95.10	98.07	98.58	98.81	**99.41**
12	65.35	50.99	**65.63**	87.90	93.65	95.03	96.81	98.59	**99.25**
13	94.62	95.71	**98.91**	98.15	99.18	99.67	**100.00**	**100.00**	**100.00**
14	95.02	92.55	**96.26**	98.91	97.78	**100.00**	**100.00**	99.93	99.83
15	**52.31**	44.84	51.84	81.67	83.97	84.06	84.15	91.87	**93.29**
16	78.19	81.93	**85.42**	**99.40**	75.42	95.06	91.93	98.43	96.99
OA	**80.18**	70.80	79.17	90.74	93.36	95.16	95.34	97.38	**97.51**
AA	**78.14**	69.73	78.04	89.12	91.39	94.98	94.27	97.33	**97.39**
κ	**77.28**	66.53	76.11	89.43	92.42	94.46	94.66	97.01	**97.15**
Time	11.44	7.33	**4.62**	86.98	**3.23**	26.48	27.47	7.89	8.77

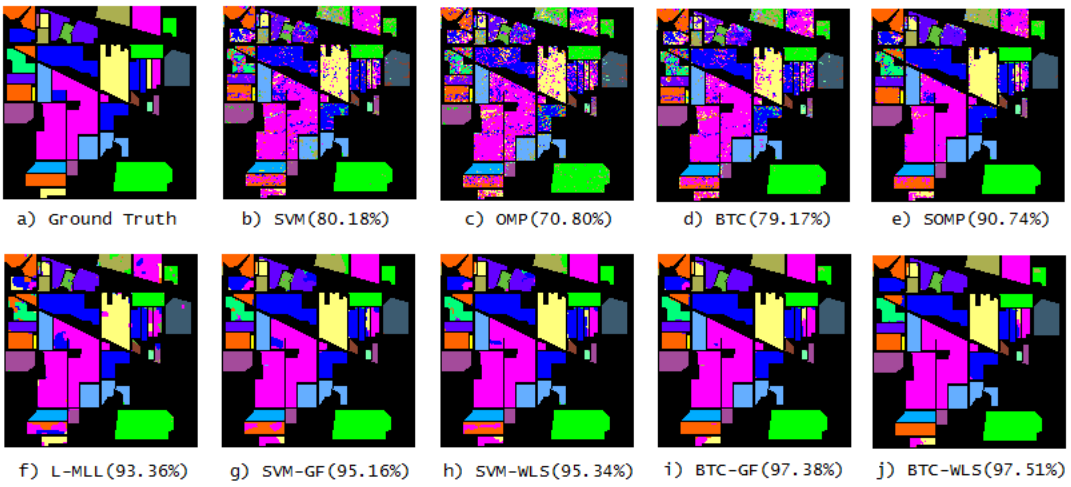


Figure 6.8: Classification Maps on Indian Pines Dataset with Overall Accuracies

For sparsity-based algorithms, each training pixel (column) of the constructed dictionary was l_2 normalized. For SVM-based approaches, both the training and testing pixels were normalized between -1 and 1. For BTC method, based on the $\overline{\beta}_M$ plot given in Fig. 6.1, we set the threshold M to 80 at which $\overline{\beta}_M$ approaches approximately to the minimum. Similarly, for BTC-GF and BTC-WLS methods, by observing the plot given in Fig. 6.3, we set the M parameter to 60. Based-on this configurations, we repeated the experiments for twenty Monte Carlo runs both for spectral-only methods (SVM, OMP, BTC) and for spatial-spectral approaches (SOMP, L-MLL, SVM-GF, SVM-WLS, BTC-GF, BTC-WLS). The classification results are shown in Table 6.2. We have also provided the classification maps with overall accuracies (%) in Fig. 6.8. In spectral-only case, as expected the SVM method having non-linear kernel (RBF) achieves best results in terms of OA, AA, and κ. This is because unlike SVM approach both OMP and BTC methods use linear kernel (dot product). On the other hand, classification results of the BTC method are very close to those of SVM. In terms of computation time, the best result is achieved by the BTC method. In spatial-spectral case, both BTC-GF and BTC-WLS approaches achieve best results in terms of all metrics except the computation time. The OA and AA differences between BTC-WLS and SVM-WLS are about 2% and 3%, respectively. When we compare the BTC-GF and BTC-WLS methods with the SOMP method, the performance differences are significant. This results show that smoothing residual maps is quite effective way of improving the classification accuracy. The L-MLL method achieves better than the SOMP technique and it is the fastest algorithm in this case. However, the performance differences in terms of classification accuracies between the proposed BTC-based methods and L-MLL are significant. The SVM-based approaches perform quite slower than the proposed algorithms and the L-MLL method. The results also show that the SOMP method is computationally very expensive. Note that the time metric in the table include the classification time as well as the smoothing time. When implementing the WLS filter, we applied the preconditioned conjugate gradient method with incomplete Cholesky decomposition which highly reduces the computational cost [132]. A final note about this experiment is that WLS-based approaches are generally superior to the GF-based methods.

Table 6.3: The results (accuracy per class (%), OA (%), AA (%), κ (%), Time (s) of twenty Monte Carlo runs) for spectral-only and spatial-spectral methods on Salinas dataset

	Spectral-Only			Spatial-Spectral					
Class	SVM	OMP	BTC	SOMP	L-MLL	SVM-GF	SVM-WLS	BTC-GF	BTC-WLS
1	99.15	99.28	**99.44**	90.68	99.88	**100.00**	**100.00**	**100.00**	**100.00**
2	99.76	**99.85**	99.22	92.11	99.97	**100.00**	**100.00**	99.99	**100.00**
3	**99.03**	97.19	97.38	91.10	99.87	**100.00**	**100.00**	**100.00**	**100.00**
4	**99.34**	98.56	99.33	86.71	99.05	**100.00**	**100.00**	**100.00**	99.95
5	98.26	98.16	**98.74**	88.77	99.04	99.58	99.45	**99.83**	**99.83**
6	99.77	**99.88**	99.77	88.52	99.87	**100.00**	99.97	**100.00**	**100.00**
7	99.66	**99.84**	99.64	93.53	99.80	**100.00**	**100.00**	99.94	**100.00**
8	87.30	78.07	**88.83**	88.54	93.79	96.03	98.35	98.04	**99.43**
9	99.56	**99.77**	99.42	91.09	99.98	**100.00**	**100.00**	**100.00**	**100.00**
10	94.65	**96.18**	94.38	84.43	96.48	99.51	**100.00**	99.86	**100.00**
11	96.98	**98.08**	97.93	86.16	97.82	**100.00**	**100.00**	**100.00**	**100.00**
12	99.49	99.32	**99.98**	86.49	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
13	97.79	**97.86**	97.51	85.94	97.94	**100.00**	99.49	99.90	99.34
14	95.70	95.30	**96.85**	90.20	97.76	99.84	98.93	**99.85**	99.64
15	**71.51**	64.93	65.89	72.43	72.91	82.15	86.21	85.61	**89.36**
16	98.42	**98.50**	98.38	89.11	98.87	99.99	**100.00**	99.92	99.99
OA	**92.68**	89.94	92.20	87.02	94.59	96.72	97.74	97.63	**98.42**
AA	**96.02**	95.05	95.79	87.86	97.06	98.57	98.90	98.93	**99.22**
κ	**91.84**	88.80	91.30	85.61	93.96	96.34	97.48	97.36	**98.24**
Time	61.81	113.51	**24.37**	988.99	**30.16**	135.85	138.94	42.80	45.55

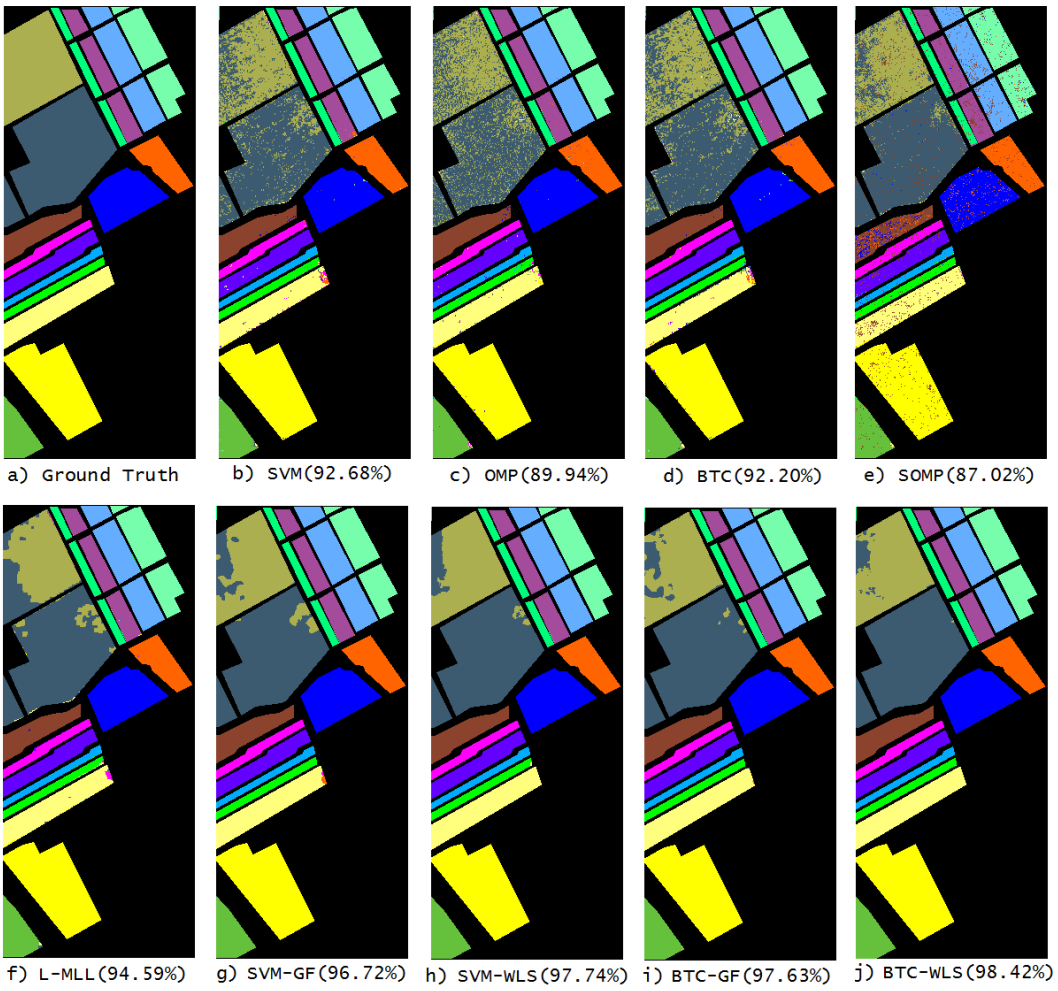


Figure 6.9: Classification Maps on Salinas Dataset with Overall Accuracies

6.5.5 Results on Salinas Dataset

The second experiment was performed on the Salinas dataset. Since the number of ground truth pixels is large as compared to the first dataset, this time we selected 5% of the ground truth pixels for training set (dictionary) and the remaining 95% for testing. Similarly, each class, the number of training, and test pixels are given in Fig. 6.5c. The normalization process for the SVM and sparsity-based approaches was performed as described in the first experiment. For spectral-only and spatial-spectral BTC methods, we set M to 50, and to 20, respectively based on the $\overline{\beta}_M$ plots given in Fig. 6.1 and in Fig. 6.3. We repeated the experiments for twenty Monte Carlo runs for all methods. The classification results are shown in Table 6.3. We have also included the classification maps with overall accuracies (%) in Fig. 6.9. Since the HSI contains large homogeneous areas as compared to the previous case, all spectral-only approaches perform quite similarly. The SVM method again achieves best results except the classification time due to the reason we explained in the previous subsection. In terms of classification time, again the fastest one is the BTC method. Since the dictionary size is larger in this case, the OMP method performs quite slowly as compared to the others. In the spatial-spectral case, similar to the previous experiment, BTC-WLS technique again achieves the best results in terms of all metrics except the computation time. The performance differences between the filtering-based methods and the SOMP method are significant. In terms of OA, BTC-WLS performs approximately 4% better than the L-MLL method. In terms of computational cost, although L-MLL approach slightly outperforms the BTC-based techniques, the computational performance differences between these methods and the SVM-based approaches are significant.

6.5.6 Results on Pavia University Dataset

The third experiment was performed on the Pavia University dataset. Similar to the previous experiment, we chose the 5% of the ground truth pixels for training and 95% for testing. Each class of this dataset, the number of training and testing pixels for each class are shown in Fig. 6.6c. The threshold parameter (M) was set to 40 for BTC and 35 for BTC-GF and BTC-WLS techniques based on the plots given in

Table 6.4: The results (accuracy per class (%), OA (%), AA (%), κ (%), Time (s) of twenty Monte Carlo runs) for spectral-only and spatial-spectral methods on Pavia University dataset

	Spectral-Only			Spatial-Spectral					
Class	SVM	OMP	BTC	SOMP	L-MLL	SVM-GF	SVM-WLS	BTC-GF	BTC-WLS
1	**93.07**	66.58	87.74	87.30	98.66	98.74	98.69	**98.88**	98.79
2	**97.93**	88.51	93.47	99.69	99.70	99.92	**100.00**	99.83	**100.00**
3	**75.02**	56.61	72.84	79.80	80.98	85.40	85.88	90.71	**93.79**
4	**93.44**	84.41	91.87	92.58	95.94	96.41	90.89	**96.53**	90.71
5	99.21	**99.80**	99.50	**100.00**	99.53	**100.00**	**100.00**	**100.00**	**100.00**
6	**86.50**	58.13	73.35	65.62	98.97	98.55	**99.97**	94.55	99.27
7	**84.74**	55.31	73.29	80.59	90.25	99.81	**100.00**	97.27	99.46
8	**90.09**	60.33	73.32	67.78	94.64	98.78	**99.45**	96.79	98.87
9	**99.86**	79.78	88.55	63.87	**99.87**	99.86	98.77	99.35	97.58
OA	**93.39**	76.38	86.81	88.16	97.54	98.51	98.37	98.03	**98.59**
AA	**91.10**	72.16	83.77	81.92	95.39	97.50	97.07	97.10	**97.61**
κ	**91.19**	68.67	82.46	83.98	96.73	98.02	97.84	97.38	**98.12**
Time	17.58	47.19	**13.48**	503.57	**37.37**	92.21	96.65	63.80	67.94

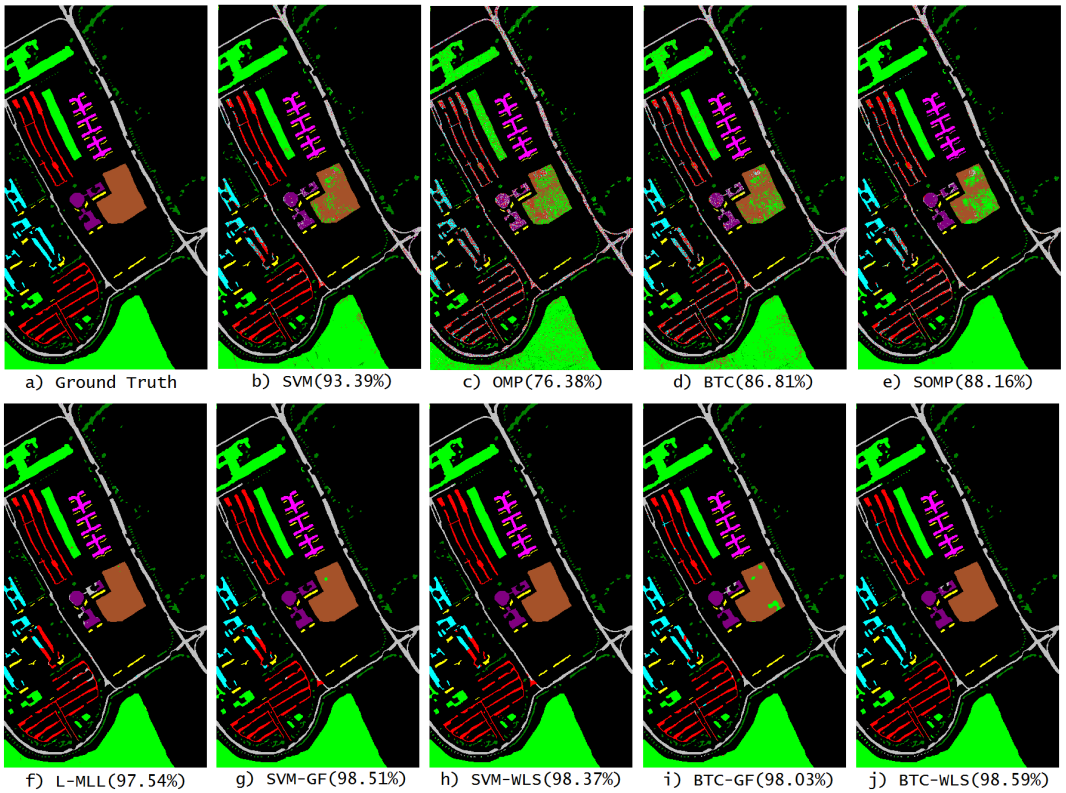


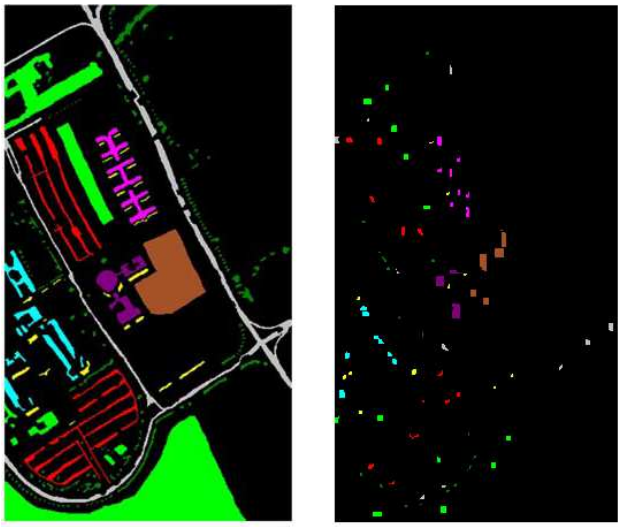
Figure 6.10: Classification Maps on Pavia University Dataset with Overall Accuracies

Fig. 6.1 and in Fig. 6.3, respectively. The experiments were repeated for twenty Monte Carlo runs for all methods using the same configurations given in the previous experiments. The classification results are given in Table 6.4. Similarly, we have also provided the classification maps with overall accuracies (%) for each method in Fig. 6.10. This time in spectral-only case the performance differences between the SVM method and the sparsity-based methods are significant in terms of OA, AA, and κ metrics. It even outperforms the SOMP method. This is because the classes of Pavia University dataset are highly non-linearly separable. On the other hand, in spatial-spectral case, BTC and SVM-based methods perform quite similarly. The best results are achieved by the BTC-WLS method. This time L-MLL technique achieves 1% worse than BTC-WLS in terms of OA. For this case, the fastest method is the L-MLL approach as well. Classification accuracy difference on the third class is significant between the BTC and SVM-based approaches. SVM-GF method outperforms the SVM-WLS technique. We believe this is the reason that the SVM-GF method has been proposed since it seems more robust for HSI classification.

6.5.7 Results using Fixed Training Set

We performed the last experiment on the Pavia University dataset using fixed training set which is available on Dr. Li's web page[1]. The original set contains 3921 training samples, however, we noticed that only 2777 of them are included in the ground truth (Fig. 6.11a). Therefore, for this experiment, we used 2777 of the training samples, which are shown in Fig. 6.11b. The number of training and testing pixels for each class are shown in Table 6.5. Fig. 6.11b also shows that most of the pixels are grouped together which means that the samples belonging to the same class are highly similar to each other. This reduces the diversity of the input features causing lower classification accuracies as compared to the previous experiments. For this experiment, we evaluated the performances of the BTC-WLS technique and the final spatial-spectral techniques (SOMP, L-MLL, SVM-GF) proposed in [115], [113], and [107], respectively. The threshold parameter (M) was set to 5 for the BTC-WLS technique based on the $\overline{\beta}_M$ plot using fixed training set. The classification results are given in Table 6.5. Again BTC-WLS achieves best results in terms of OA, AA, and κ. This time the

[1] http://www.lx.it.pt/ jun/



(a) (b)

Color	Class No	Class	Train	Test
	1	Asphalt	327	6304
	2	Meadows	503	18146
	3	Gravel	284	1815
	4	Trees	152	2912
	5	Metal sheets	232	1113
	6	Bare soil	457	4572
	7	Bitumen	349	981
	8	Bricks	318	3364
	9	Shadows	152	795

(c)

Figure 6.11: a-) Ground truth image b) fixed training samples, and c-) class numbers and the corresponding classes of Pavia University dataset

performance differences between the proposed method and the other techniques are significant. Although L-MLL outperforms the other approaches in terms of classification time, the performance difference is not significant between our proposal and the L-MLL technique in terms of this metric. On the other hand, SOMP and SVM-GF performs quite slowly as compared to L-MLL and BTC-WLS.

Table 6.5: The results (accuracy per class (%), OA (%), AA (%), κ (%), Time (s) using fixed training set) for spatial-spectral methods on Pavia University dataset

Class	Train	Test	SOMP [115]	L-MLL [113]	SVM-GF [107]	BTC-WLS
1	327	6304	62.72	92.61	**96.19**	91.89
2	503	18146	71.49	70.80	79.00	**86.75**
3	284	1815	81.60	76.80	62.98	**82.26**
4	152	2912	89.66	91.79	**96.81**	92.41
5	232	1113	**100.00**	99.82	**100.00**	**100.00**
6	457	4572	94.03	**99.02**	98.60	98.43
7	349	981	95.51	98.37	**100.00**	**100.00**
8	318	3364	65.67	98.69	99.05	**99.35**
9	152	795	57.99	**99.87**	99.75	97.74
OA	-	-	75.09	83.67	87.71	**91.07**
AA	-	-	79.85	91.98	92.48	**94.31**
κ	-	-	68.29	79.21	84.12	**88.27**
Time	-	-	660.18	**51.18**	100.14	63.36

6.6. Conclusions

In this chapter, we proposed a light-weight sparsity-based classifier (BTC) for HSI classification alternative to the well-known SVM-based approaches and other greedy methods such as SOMP. The proposed method is easy to implement and performs quite fast. One of the most important advantages of the proposal is that it does not require any cross validation or classification experiment for parameter selection. Based on the guidance we have presented in chapter 3, one could easily select the threshold parameter once the dictionary is constructed. To improve the classification accuracy, we have proposed quite efficient framework in which the output residual maps of the pixel-wise classification procedure are smoothed using an edge preserving filtering method. Simulation results on the publicly available datasets showed that this intermediate procedure highly improves the classification accuracy. This approach could also be applied to any other sparsity-based classifier.

CHAPTER 7

HYPER-SPECTRAL IMAGE CLASSIFICATION VIA KBTC

7.1. Introduction

The spectral signatures obtained by remote sensors could be used to distinguish the materials and objects on the surface of the earth. Those signatures are stored in the three dimensional data cubes called hyper-spectral images (HSI). While the first two dimensions of an image are used to describe the spatial coordinates, the last dimension is used to represent the spectral coordinate. The pixels of an image can be interpreted as the feature vectors containing spectral measurements. Various methods have been proposed in the literature to classify those pixels using the spectral measurements. Among those, support vector machines (SVM) [54] approach with radial basis function (RBF) kernel is commonly preferred. It is well known that the SVM classifier can achieve satisfactory results using only a few training samples [47], [71]. It also has low sensitivity to Hughes phenomenon [98]. On the other hand, it has some limitations such as conversion from binary classification to multi-class one and parameter determination via cross validation. When one-against-all approach is used as a conversion technique, some of the samples may be evaluated as unclassified [102]. If one-against-one approach is preferred, this time an increment in the number of classes significantly increases the binary classifiers used. Based on SVM, spatial-spectral techniques such as composite kernels [104], segmentation maps [105], [106], edge-preserving filtering [107] have been developed to increase the classification accuracies by incorporating the spatial information. Although those approaches achieve promising results, they share common limitations with SVM.

Multinomial logistic regression (MLR)[109], which is based on Bayesian framework, has been used as an alternative to SVM in HSI classification [110], [111], and [112]. One of the MLR-based techniques which uses active learning namely logistic regression via splitting and augmented Lagrangian (LORSAL) has been proposed in [114] and achieved promising results with the segmentation framework namely L-MLL (LORSAL multi-level logistic prior). It has the advantage of using active learning and RBF kernel.

A powerful alternative to SVM is the sparse representation-based classification scheme which is based on the assumption that the test samples can be modeled as the sparse linear combinations of the basis dictionary elements. It was introduced in [2] and successfully applied to many classification problems. Unlike classical methods, this technique has low sensitivity to the corrupted or partial features exploiting the fact that the errors are sparse with respect to dictionary elements. Recently, joint sparsity-based methods (JSM) have been successfully applied to HSI classification. A greedy approach namely the simultaneous orthogonal matching pursuit (SOMP) [37] was used for simultaneous classification of neighboring pixels in a given region based on the assumption that the pixels in a predetermined window share a common sparsity model [115]. Non-linear kernel versions of this approach have been proposed [116], [120], and [121] to achieve better performance when the samples of different classes are linearly non-separable. To improve the classification accuracy, a weighting matrix based version of SOMP namely WSOMP has been proposed in [122]. An adaptive version of SOMP namely ASOMP has been developed in [124] to adaptively define the neighborhood pixels in the supervision of a segmentation map. Finally, a multi-scale adaptive sparse representation, which incorporates spatial information at different scales, has been successfully applied to HSI classification. However, JSM-based approaches suffer from the extensive computational cost because of simultaneous sparse recovery of the neighboring pixels. Another drawback is to select the sparsity level and the neighborhood window experimentally which may vary and cause non-optimal results.

To eliminate the problems with the aforementioned approaches, basic thresholding classifier (BTC) was introduced in 3. It is a sparsity-based light-weight linear classifier which is able to achieve promising results and classify test samples extremely

rapidly as compared to the other sparsity-based algorithms. In this chapter, we propose the kernelized version of the BTC algorithm namely kernel basic thresholding classifier (KBTC) which is capable of classifying the samples of classes of a given dataset when they are linearly non-separable [133]. Three major contributions of this chapter are follows:

- We propose a non-linear kernel version of the basic thresholding classification algorithm namely KBTC for HSI classification. The proposed non-linear sparsity-based method can achieve higher classification results as compared to the linear sparsity based ones especially when the classes of a given dataset are linearly non-separable.
- Unlike SVM, in which the parameters are obtained via cross validation, we provide a full parameter selection guidance by which the kernel and threshold parameters can easily be estimated without using any experiment and cross validation.
- We extend the proposal to a spatial-spectral framework in which the final classification is performed based on the smoothed residual maps. We also provide more realistic fixed train sets alternative to those selected randomly from a given dataset.

7.2. HSI Classification

In this section, we will briefly describe the HSI classification problem in the sparse representation model. Let $A_i \in \mathbb{R}^{B\times N_i}$ be the matrix containing B dimensional N_i many training pixels which are belonging to the ith class. Then, with C many classes one could construct the dictionary A in such a way that $A = [A_1\ A_2\ \dots\ A_C] \in \mathbb{R}^{B\times N}$ where $N = \sum_{i=1}^{C} N_i$. In this context, a test pixel $y \in \mathbb{R}^B$ can be classified as follows: Sparse representation model states that there exists a sparse code $x \in \mathbb{R}^N$ having minimum l_1 norm such that $y = Ax$ or $y = Ax + \epsilon$ where ϵ is a small error [2]. The minimization problem is equivalent to

$$\hat{x} = \arg\min_{x} \|x\|_1 \ \ subject\ to\ y = Ax \tag{7.1}$$

or subject to $\|y - Ax\|_2 \leq \epsilon$. Then, we could find the class of y using the following expression

$$class(y) = \arg\min_i \|y - A\hat{x}_i\|_2 \ \forall i \in \{1, 2, \ldots, C\} \tag{7.2}$$

where $\hat{x}_i$ denotes the ith class portion of the estimated sparse code vector $\hat{x}$. The minimization problem (7.1) could be solved using convex relaxation-based techniques (homotopy [82]) or alternatively greedy approaches such as OMP [37]. However, those techniques and their variants [115] suffer from high computational cost. To eliminate the problems with those techniques, a simple and light-weight sparsity-based method, basic thresholding classifier (BTC) was introduced in chapter 3. Unfortunately, this approach works based on the fact that the classes of a given HSI are linearly separable. In practice, the situation may not be like this. Therefore, in order to eliminate the limitations with the aforementioned algorithms, we propose KBTC algorithm for HSI classification. In this context, one can use the following expression in order to classify a given pixel:

$$Class(y) \leftarrow KBTC(A, y, \gamma, M, \alpha) \tag{7.3}$$

The details of the algorithm could be found in Chapter 4.

7.3. Extension to Spatial-Spectral KBTC

Since incorporating spatial information highly improves the classification accuracy, we extend our pixel-wise proposal (KBTC) to a spatial-spectral framework namely KBTC-WLS. Notice that when a test pixel y is classified, KBTC produces not only the class identity of y but also a residual vector $\epsilon \in \mathbb{R}^C$ which contains distances to each class. Suppose that we classified every pixel of a given HSI, $H \in \mathbb{R}^{n_1 \times n_2 \times B}$ consisting of $n_1 \times n_2$ pixels, using the KBTC method. We could interpret the resulting residual vectors as a residual cube $R \in \mathbb{R}^{n_1 \times n_2 \times C}$ in which each layer represents a residual map. The idea is to smooth those maps using an averaging filter and then determine the class of each pixel based on minimal smoothed residuals. This intermediate step highly improves the classification results. We use weighted least squares method (WLS) [127] as the smoothing filter which successfully preserves the edges

via a gray scale guidance image. We prefer this filter because it does not cause halo artifacts at the edges. The gray scale guidance image could be obtained via principal component analysis (PCA) technique by reducing the dimensions of the original HSI from B to 1. We give the overall framework step-by-step as follows:

- The parameters γ and M are estimated for a given dictionary A using the procedures provided in chapter 4.
- Every pixel of a given HSI is classified via KBTC. Note that the entries of the resulting residual cube are normalized between 0 and 1.
- We obtain the gray scale guidance image by reducing the dimension of the original HSI from B to 1 using the PCA technique.
- We apply WLS filtering to each residual map by means of the guidance image obtained in the previous step. In order to reduce the computation cost of the WLS filtering, preconditioned conjugate gradient (PCG) method with incomplete Cholesky decomposition [132] could be used when taking the inverses of the large sparse matrices.
- The class of each pixel is determined based on minimal smoothed residuals.

The overall framework could also be seen in Fig. 7.1. Please note that since we obtain the pixel-wise class map before the smoothing procedure, we could set the error values to the maximum value 1 in the ith residual map for the entries whose labels are not equal to i using the pixel-wise class map. This improves the classification performance further.

7.4. Experimental Results

7.4.1 Scaling

Scaling both the testing and training data is quite important before applying the KBTC algorithm. We recommend scaling the entries in the feature vectors to the range $[-1, +1]$ or $[0, 1]$. Suppose that we scaled a training pixel from $[100, 20, 50, ...]^T$ to

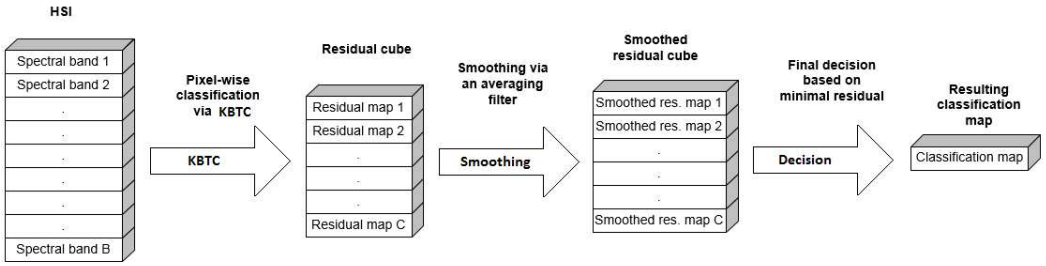


Figure 7.1: Spatial-Spectral KBTC (KBTC-WLS)

$[1, 0.2, 0.5, ...]^T$. If a test pixel having the feature vector $[110, 25, 45, ...]^T$ is needed to be classified, then it should be scaled to $[1.1, 0.25, 0.45, ...]^T$. It is similar to the scaling in RBF kernel SVM [134]. Please note that linear BTC uses different approach in which only the training pixel vectors are l_2 normalized.

7.4.2 Datasets

We performed the experiments using three publicly available HSI datasets namely Indian Pines, Salinas, and Pavia University. Detailed description of each dataset is given Table 7.1. The original Indian Pines image contains 16 different classes. However, we discarded 7 of them because of insufficient number of training and testing samples [135]. We carried out the experiments using fixed and grouped training pixels instead of random selection. This is because in case of random selection, it is highly likely to have a training sample which may be closely related to a testing sample. Therefore, this type of experiment may not present realistic results. On the other hand, in our case, we exclude training pixels from the testing areas which is quite similar to real world scenarios. The experiments using fixed and grouped training pixels could also be seen in [136],[115], and [116] for Pavia University dataset. In our experiments, we used 27 training pixels for each class of Indian Pines and Pavia University datasets. Those samples were taken from 3 distinct 3×3 blocks. For Salinas dataset we used 32 training pixels for each class. This time the samples were taken from 2 distinct 4×4 blocks. The ground truth image, each class with corresponding number of training and testing pixels, and the starting coordinates of the distinct blocks from which the

Table 7.1: Description of each dataset

Dataset	Size	Spatial resolution	Spectral coverage	Num. of classes	Sensor	Num. of bands
Indian Pines	$145 \times 145 \times 220$	20 m	$0.4 - 2.5$ μm	9	AVIRIS	200
Salinas	$512 \times 217 \times 224$	3.7 m	$0.4 - 2.5$ μm	16	AVIRIS	204
Pavia University	$610 \times 340 \times 115$	1.5 m	$0.43 - 0.86$ μm	9	ROSIS	103

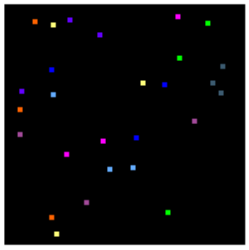
fixed training pixels were taken are given for each dataset in Fig. 7.2, Fig. 7.3, and Fig. 7.4, respectively. We provided the coordinates of the blocks because anyone can easily repeat the experiments and compare the results.

7.4.3 Experimental Setup

Before the experiments, we estimated the γ and M parameters of KBTC using the training sets and the procedures provided in the previous section. For each dataset, we computed the values of $\overline{\beta}(\gamma)$ by varying γ from 2^{-10} to 2^{1}. The results could be seen in Table 7.2. If we keep track of the computed values of $\overline{\beta}(\gamma)$ for any dataset, we observe that the resulting function is strictly convex. This implies that $\overline{\beta}(\gamma)$ has unique minimum. Therefore, the best value of γ in the described sense could easily be estimated. The estimated γ values which minimize $\overline{\beta}(\gamma)$ are 2^{-6}, 2^{-6}, 2^{-1} for the dictionaries constructed using Indian Pines and Salinas and Pavia University datasets, respectively. If we insert the estimated γ values to $\overline{\beta}(\hat{\gamma}, M)$ function, then we could estimate the best value of the threshold by varying M from 1 to $B - 1$. The M value that minimizes $\overline{\beta}(\hat{\gamma}, M)$ is considered to be the best estimate of it in the described sense. We performed this procedure and obtained $\overline{\beta}(\hat{\gamma}, M)$ plots in Fig. 7.5 for each dataset used in this chapter. The estimated threshold values are 95, 92, and 35 for Indian Pines, Salinas, and Pavia University datasets, respectively.

We included the spectral-only classifiers as well as the spatial-spectral ones in the experiments. RBF kernel SVM [47] could be considered one of the strongest spectral-only classifiers in the literature. Another approach in this category is the LORSAL [113] technique which also uses the RBF kernel. It has the advantage of using active learning. Although it is not fair, we included the linear similarity-based BTC in order

a)

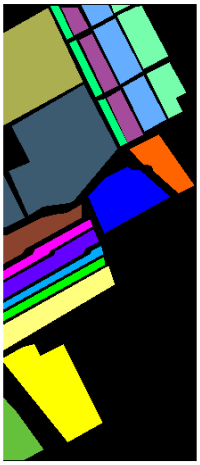


b)

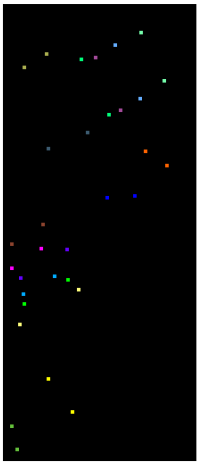
Color	Class No	Class	Train	Test	T. Pixels 1 (3x3)	T. Pixels 2 (3x3)	T. Pixels 3 (3x3)
	1	Corn no-till	27	1401	28,39	96, 48	79, 80
	2	Corn min-till	27	803	18, 10	9, 63	28, 128
	3	Grass / Pasture	27	456	9, 78	49, 119	114, 70
	4	Grass / Trees	27	703	29, 54	63, 99	77, 98
	5	Hay-windrowed	27	451	131, 37	125, 47	130, 53
	6	Soybeans no-till	27	945	83, 47	29, 12	31, 138
	7	Soybeans min-till	27	2428	37, 90	59, 82	104, 7
	8	Soybeans clean-till	27	566	10, 52	39, 9	57, 18
	9	Woods	27	1238	98, 125	105, 32	122, 11

c)

Figure 7.2: a-) Ground truth image, b-) fixed training pixels, c-) each class with corresponding number of training and testing pixels, and the coordinates of the grouped training pixels for Indian Pines dataset



a)

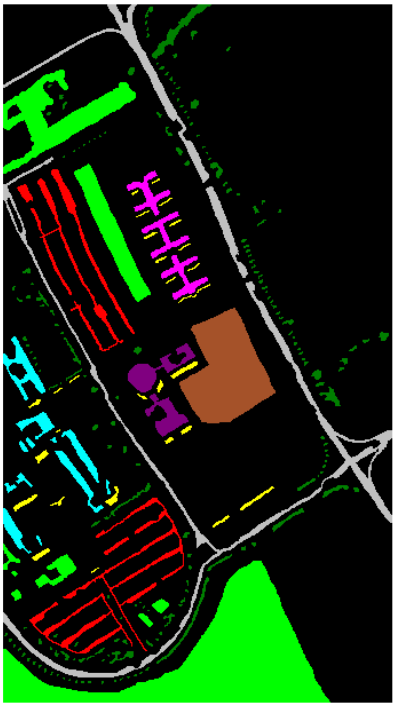


b)

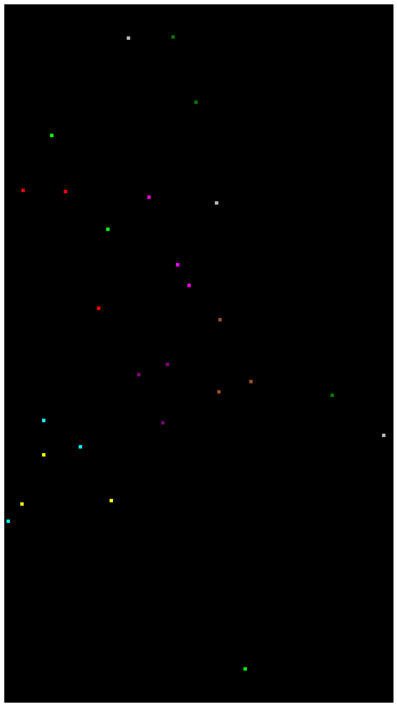
Color	Class No	Class	Train	Test	T. Pixels 1 (4x4)	T. Pixels 2 (4x4)
	1	Weeds 1	32	1977	9, 268	44, 246
	2	Weeds 2	32	3694	116, 216	147, 214
	3	Fallow plow	32	1944	159, 164	183, 180
	4	Fallow smooth	32	1362	87, 61	118, 123
	5	Stubble	32	2646	103, 59	131, 118
	6	Celery	32	3927	125, 45	153, 105
	7	Grapes untrained	32	3547	154, 31	180, 85
	8	Soil	32	11239	50, 161	94, 143
	9	Corn	32	6171	50, 419	77, 456
	10	Soybean-notill	32	3246	18, 358	84, 319
	11	Lettuce 4 wk	32	1036	9, 295	42, 273
	12	Lettuce 5 wk	32	1895	19, 306	71, 274
	13	Lettuce 6 wk	32	884	22, 324	57, 304
	14	Lettuce 7 wk	32	1038	23, 335	72, 308
	15	Vinyard untrained	32	7236	23, 70	48, 55
	16	Vinyard trellis	32	1775	9, 472	15, 498

c)

Figure 7.3: a-) Ground truth image, b-) fixed training pixels, c-) each class with corresponding number of training and test pixels, and the coordinates of the grouped training pixels for Salinas dataset



a)



b)

Color	Class No	Class	Train	Test	T. Pixels 1 (3x3)	T. Pixels 2 (3x3)	T. Pixels 3 (3x3)
	1	Asphalt	27	6604	108, 29	185, 173	331, 376
	2	Meadows	27	18622	41, 114	90, 196	210, 580
	3	Gravel	27	2072	34, 363	3, 451	66, 386
	4	Trees	27	3037	147, 28	286, 341	167, 85
	5	Metal sheets	27	1318	126, 168	161, 245	151, 227
	6	Bare soil	27	5002	188, 275	187, 338	215, 329
	7	Bitumen	27	1303	117, 323	138, 365	142, 314
	8	Bricks	27	3655	53, 163	82, 265	16, 162
	9	Shadows	27	920	93, 433	34, 393	15, 436

c)

Figure 7.4: a-) Ground truth image, b-) fixed training pixels, c-) each class with corresponding number of training and test pixels, and the coordinates of the grouped training pixels for Pavia University dataset

Table 7.2: $\overline{\beta}(\gamma)$ values for each γ and dataset

	$\overline{\beta}(\gamma)$		
γ	Indian Pines	Salinas	Pavia University
2^{1}	0.9314	0.3322	0.4612
2^{0}	0.8250	0.2410	0.4263
2^{-1}	0.6950	0.1836	**0.4176**
2^{-2}	0.5719	0.1499	0.4265
2^{-3}	0.4751	0.1302	0.4549
2^{-4}	0.4093	0.1183	0.5025
2^{-5}	0.3743	0.1123	0.5712
2^{-6}	**0.3644**	**0.1106**	0.6630
2^{-7}	0.3737	0.1127	0.7669
2^{-8}	0.3949	0.1176	0.8925
2^{-9}	0.4263	0.1251	1.0429
2^{-10}	0.4681	0.1355	1.2670

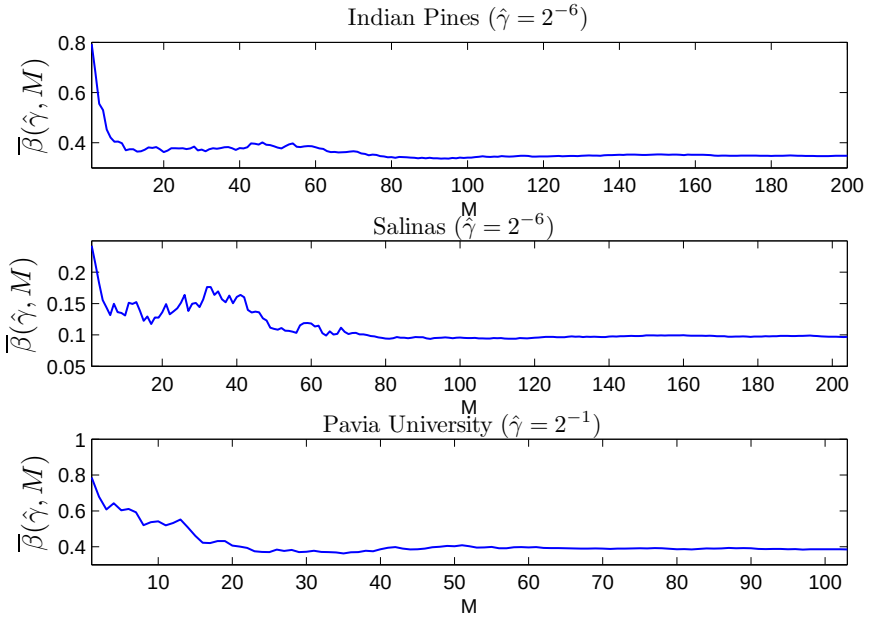


Figure 7.5: $\overline{\beta}(\hat{\gamma}, M)$ vs threshold values for Indian Pines, Salinas, and Pavia University

to show how the non-linear KBTC algorithm is superior to it. For SVM method, we used LIBSVM library [128] and we selected the parameters (C, γ) of it via 5-fold cross validation by varying C from 10^{-5} to 10^{5} and γ from 2^{-5} to 2^{5}. The spatial extensions of those algorithms highly improve the results obtained in the pixel-wise classification stage. Those techniques used in this chapter are SVM-GF [107] (based on guided filter [108]), L-MLL (based on LORSAL), BTC-WLS, and KBTC-WLS. In order for fair comparison we used SVM-WLS instead of SVM-GF since the WLS filter-based techniques achieve better results. WLS filter has two parameters namely the smoothing (λ) and the sharpening (α) degrees. We set those parameters to 0.4 and 0.9 for all experiments, respectively. For LORSAL and L-MLL techniques, we set the initial training samples to the half of the all available training pixels, and during the learning stage we incremented the samples by 20 using random selection (RS). Under this configuration we repeated the experiments for all datasets in a PC having a quad-core 3.60 GHz processor and 16GB of memory.

7.4.4 Performance Indexes

The performance indexes used in this work are as follows:

- Overall Accuracy (OA): It is the percentage of correctly classified pixels among the whole test samples.
- Average Accuracy (AA): It shows the mean of individual class accuracies.
- The κ coefficient: It measures the degree of consistency [126].
- Computation time: It is used to measure the computational complexity of an algorithm. It also determines if an algorithm is suitable for real time applications.

7.4.5 Classification Results

Using the training and testing samples given in Fig. 7.2, we carried out the first experiment on Indian Pines dataset both for spectral-only and spatial-spectral approaches.

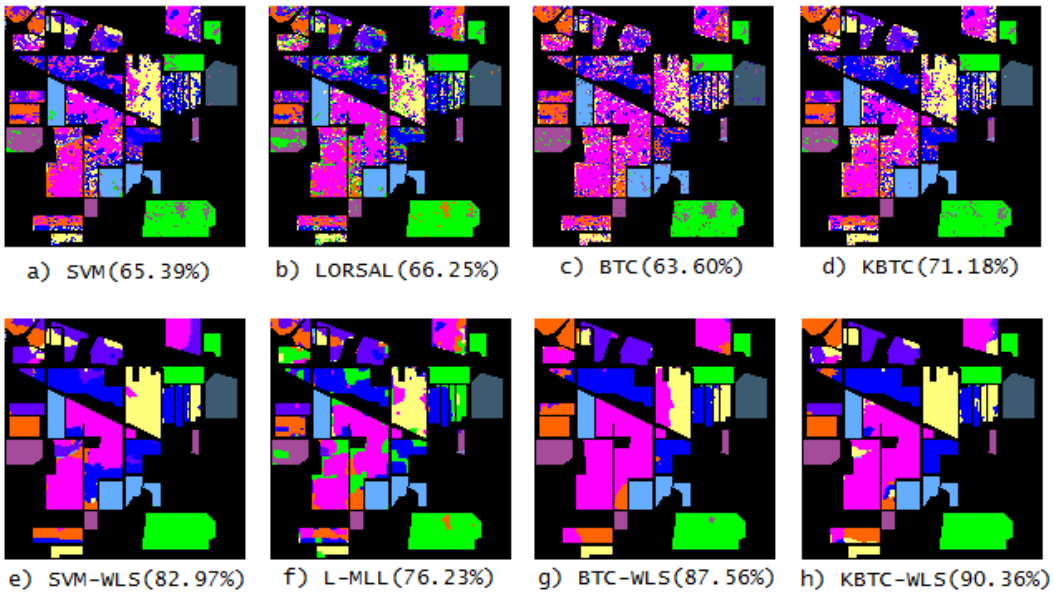


Figure 7.6: Classification Maps on Indian Pines Dataset with Overall Accuracies

The classification results are shown in Table 7.3. We also provided the corresponding classification maps with OAs(%) in Fig. 7.6. Both in the spectral-only and spatial-spectral cases, KBTC achieves best results in terms of all metrics except the computation time. The performance differences between the KBTC and the other algorithms are significant. In the first case, KBTC performs about 5% better than LORSAL technique in terms of overall accuracy. It also improves the result of linear BTC approximately 7.5%. LORSAL technique performs about 1% better than the SVM method using the advantage of active learning. In the latter case, BTC-WLS achieves promising results by means of smoothing the residual maps. KBTC-WLS exploits the same technique and outperforms the BTC-WLS approach by achieving about 3% better in terms of OA. Although the LORSAL approach performs well in the spectral-only case, L-MLL technique, which is based on LORSAL, performs worse than the other approaches in the latter case. This experiment shows that KBTC significantly improves the performance of linear similarity-based BTC both in spectral-only and spatial-spectral cases. In terms of computation time, LORSAL and its spatial extension are the fastest ones. Although KBTC and KBTC-WLS are the slowest ones, there is no significant difference between the computation times of these methods and those of the other techniques except LORSAL and L-MLL.

Table 7.3: The results (accuracy per class (%), OA (%), AA (%), κ (%), Time (s) using fixed training set) for spectral-only and spatial-spectral methods on Indian Pines dataset

	Spectral-Only				Spatial-Spectral			
Class No	SVM	LORSAL	BTC	KBTC	SVM-WLS	L-MLL	BTC-WLS	KBTC-WLS
1	52.53	51.46	47.47	**64.03**	77.02	67.81	80.66	**81.87**
2	49.32	45.08	**56.04**	**56.04**	68.99	51.93	**82.81**	81.32
3	**89.47**	73.68	88.82	88.82	**96.05**	84.65	**96.05**	**96.05**
4	92.75	95.73	90.75	**98.58**	99.86	99.57	**100.00**	**100.00**
5	98.45	98.45	95.79	**99.56**	**100.00**	**100.00**	**100.00**	**100.00**
6	65.50	56.40	44.66	**69.95**	89.63	73.54	70.05	**99.26**
7	49.63	57.50	**58.81**	54.57	68.57	65.53	**89.17**	83.61
8	56.36	61.13	37.81	**65.90**	85.87	81.27	76.15	**93.11**
9	88.93	92.57	86.03	**92.65**	**100.00**	97.25	99.19	**100.00**
OA	65.39	66.25	63.60	**71.18**	82.97	76.23	87.56	**90.36**
AA	71.43	70.22	67.35	**76.67**	87.33	80.17	88.23	**92.80**
κ	60.15	60.76	57.53	**66.68**	80.27	72.32	85.26	**88.73**
Time	1.89	**0.18**	1.78	2.90	2.44	**0.41**	2.33	3.47

We performed the second experiment on the Salinas dataset using the training and testing samples given in Fig. 7.3. The classification results could be seen in Table 7.4. We also included the corresponding classification maps with OAs(%) in Fig. 7.6. As we can see in Fig. 7.3, this HSI consists of large homogeneous areas as compared to the previous HSI. Therefore, the classification results achieved by the spectral-only techniques are closer to each other. The OA achieved by KBTC is about 1% better than that of BTC approach. This result also shows that the dictionary constructed for Salinas image is more linearly separable than the previous one. KBTC also outperforms the SVM and LORSAL techniques having approximately 3% and 1.5% better accuracies, respectively. In the spatial-spectral case, the accuracies are also closer to each other. In this case, again KBTC-WLS achieves the best results in terms of all metrics except the computation time.

Finally, the last experiment was performed on the Pavia University dataset using the training and testing samples given in Fig. 7.4. The classification results are given in Table 7.4. We also included the corresponding classification maps with OAs(%) in Fig. 7.8. The results obtained both in the spectral-only and spatial-spectral cases

Table 7.4: The results (accuracy per class (%), OA (%), AA (%), κ (%), Time (s) using fixed training set) for spectral-only and spatial-spectral methods on Salinas dataset

	Spectral-Only				Spatial-Spectral			
Class No	SVM	LORSAL	BTC	KBTC	SVM-WLS	L-MLL	BTC-WLS	KBTC-WLS
1	96.56	95.45	97.02	**97.42**	**100.00**	96.61	**100.00**	**100.00**
2	**99.76**	99.38	98.51	99.46	**100.00**	99.73	**100.00**	**100.00**
3	73.15	86.37	**87.65**	82.46	95.27	97.43	**100.00**	**100.00**
4	98.53	91.48	**98.83**	97.21	**100.00**	91.48	**100.00**	**100.00**
5	97.73	**98.11**	95.39	97.73	99.36	98.37	**99.77**	99.51
6	96.79	95.80	**99.41**	98.55	99.92	96.87	**100.00**	99.92
7	97.86	96.48	97.60	**99.10**	**100.00**	97.07	**100.00**	**100.00**
8	73.28	73.25	75.18	**75.35**	93.28	86.24	88.81	**93.44**
9	97.18	97.21	96.29	**98.31**	**100.00**	97.50	**100.00**	**100.00**
10	85.77	77.85	85.37	**90.48**	**99.14**	80.04	96.03	**99.14**
11	96.81	96.43	**99.52**	98.75	**100.00**	96.72	**100.00**	**100.00**
12	**99.63**	94.67	99.47	94.72	**100.00**	95.62	**100.00**	**100.00**
13	97.96	**98.98**	97.85	98.30	99.66	98.98	99.21	**99.89**
14	90.46	89.40	**93.26**	90.37	98.55	92.20	99.42	**99.52**
15	59.72	**74.47**	69.79	69.00	81.45	**88.82**	78.15	83.32
16	77.07	82.48	78.87	**94.20**	92.90	91.15	92.28	**100.00**
OA	85.08	86.67	87.39	**88.14**	95.55	92.47	94.17	**96.28**
AA	89.89	90.48	91.87	**92.58**	97.47	94.05	97.10	**98.42**
κ	83.38	85.18	85.98	**86.81**	95.04	91.62	93.50	**95.85**
Time	8.66	**1.94**	6.69	17.44	12.98	**3.84**	10.93	21.71

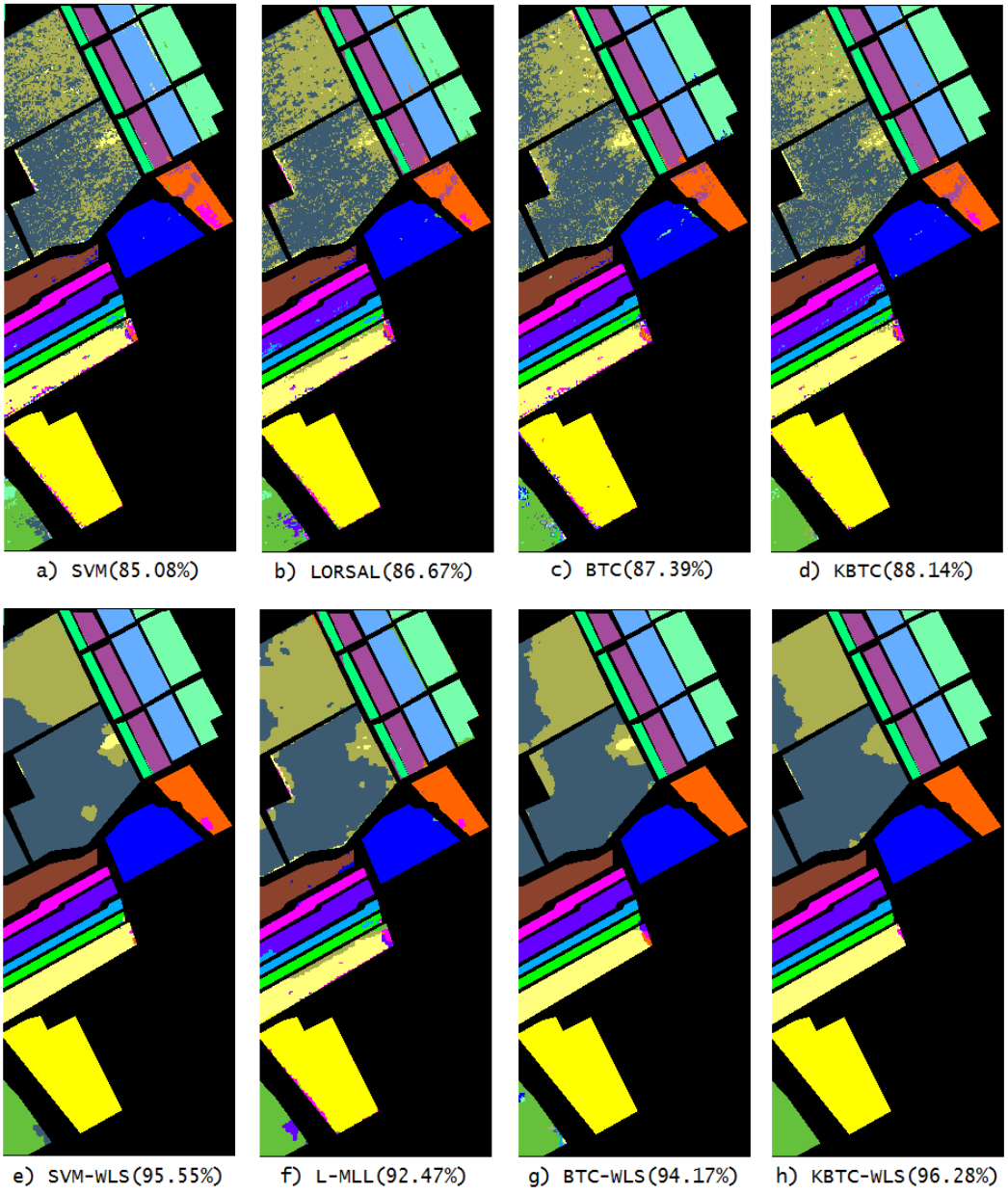


Figure 7.7: Classification Maps on Salinas Dataset with Overall Accuracies

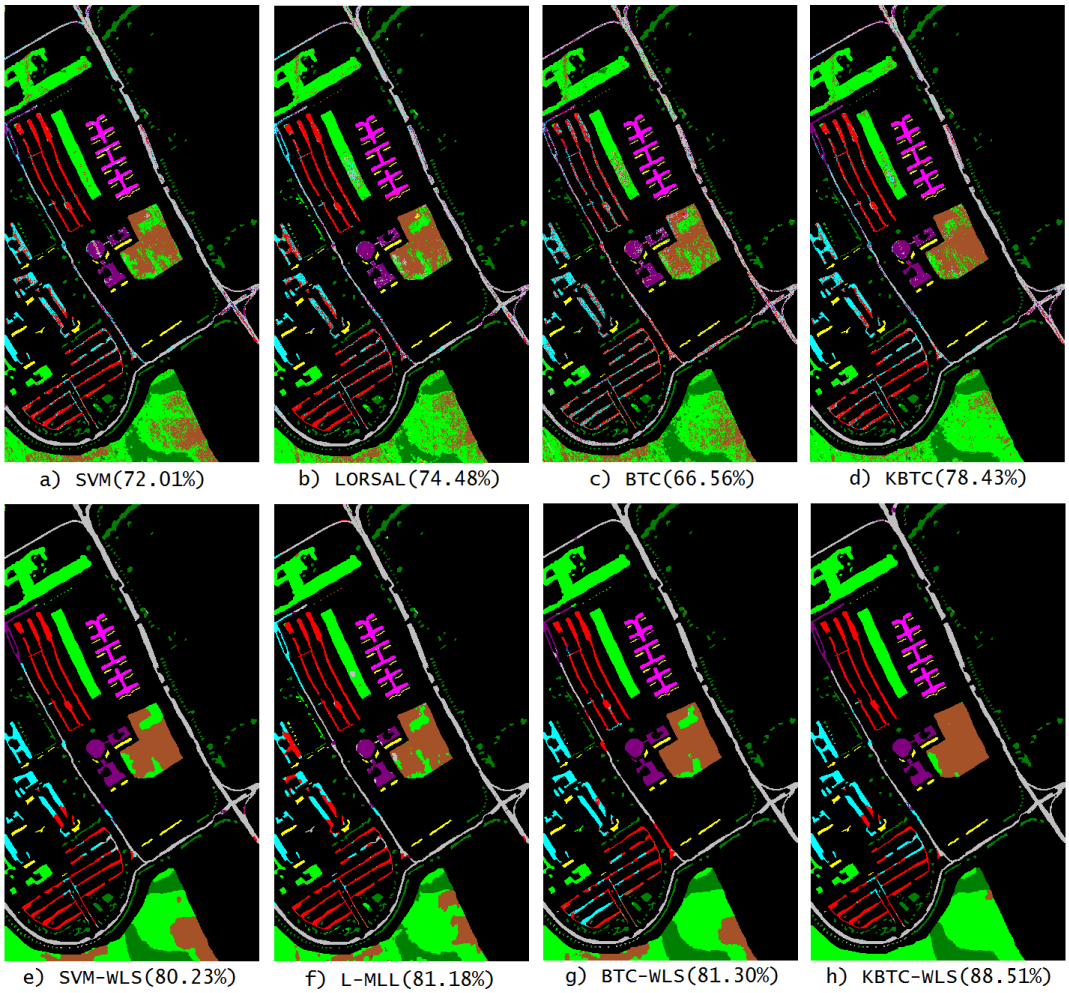


Figure 7.8: Classification Maps on Pavia University Dataset with Overall Accuracies

Table 7.5: The results (accuracy per class (%), OA (%), AA (%), κ (%), Time (s) using fixed training set) for spectral-only and spatial-spectral methods on Pavia University dataset

	Spectral-Only				Spatial-Spectral			
Class No	SVM	LORSAL	BTC	KBTC	SVM-WLS	L-MLL	BTC-WLS	KBTC-WLS
1	74.11	75.44	66.81	**75.79**	89.58	89.52	**90.48**	90.40
2	63.44	67.82	62.03	**73.87**	68.59	74.21	70.98	**82.92**
3	69.35	63.08	66.17	**75.63**	89.48	63.75	**97.06**	90.64
4	97.07	90.12	94.80	**97.14**	93.97	89.96	**94.01**	93.71
5	86.12	93.17	**99.09**	94.01	99.24	96.97	**100.00**	**100.00**
6	69.07	74.29	65.23	**78.83**	77.75	80.69	85.67	**94.86**
7	85.96	89.10	82.27	**93.17**	**100.00**	94.32	**100.00**	**100.00**
8	**79.48**	83.31	46.62	75.73	90.78	**91.57**	76.17	88.59
9	99.78	95.00	81.85	**100.00**	**99.13**	93.15	90.54	98.70
OA	72.01	74.48	66.56	**78.43**	80.23	81.18	81.30	**88.51**
AA	80.48	81.25	73.87	**84.90**	89.83	86.01	89.43	**93.31**
κ	64.92	67.83	58.22	**72.67**	75.06	75.98	76.40	**85.26**
Time	6.56	**1.51**	6.49	7.85	11.36	**3.80**	11.19	12.58

show that this dataset is the most difficult one. The pixels of this HSI are quite mixed and those belonging to different classes are highly non-linearly separable. The performance differences between the linear similarity-based BTC and the kernelized approaches are significant. KBTC achieves about 12% overall accuracy improvement over the BTC technique by means of RBF kernel. This time the LORSAL method performs about 2.5% better than the SVM technique. However, its performance is about 4% less than that of KBTC. Although BTC achieves quite low results in the spectral-only case, BTC-WLS closes the gap between the other approaches using the smoothed residual maps. It even outperforms the SVM-GF and L-MLL methods. On the other hand, KBTC-WLS performs about 7% better than BTC-WLS in terms of OA.

7.5. Conclusions

In this chapter, we proposed a non-linear kernel version of the previously introduced basic thresholding classification algorithm for HSI classification. The pro-

posed method achieves significant performance improvement over the linear version of it especially in the experiments in which the samples of the classes are linearly non-separable. The classification results on the publicly available datasets showed that the proposed algorithm also outperforms the well-known RBF kernel SVM and recently introduced logistic regression-based LORSAL technique. Based on the weighted least squares filter, we also presented the spatial-spectral version of the proposal, which achieves better performances as compared to the recently introduced state-of-the-art spatial-spectral approaches such as SVM-WLS (GF) and L-MLL. Another significance of the proposed framework is that the threshold and the kernel parameter could be easily estimated via the procedures we provided in Chapter 4 without any cross validation or experiment.

CHAPTER 8

CONCLUDING REMARKS

8.1. Summary

In this thesis, we have addressed the problem of classification in computer vision and pattern recognition by introducing two sparsity-based methods. While the first algorithm (BTC) refers the applications involving linearly separable data, the other one (KBTC) addresses the problems consisting of non-linearly separable classes. The techniques are easy to understand and require a few steps to implement. In some challenging applications such as face recognition and hyper-spectral image classification, we have shown that the proposed approaches achieve state-of-the-art classification accuracies as compared to the strongest classifiers in the literature. They also outperform those methods by classifying the given testing samples extremely rapidly. The proposals require a few parameters which could be determined via efficient off-line procedures. These procedures do not involve experiments such as cross validation in which the parameters are determined experimentally. Moreover, we have proposed some problem-specific fusion techniques which significantly improve the classification performances of our individual classifiers. For instance, in face recognition, the fusion is performed by means of taking the average of the output residuals provided by individual classifiers having different random projections. In case of HSI classification, this is achieved by smoothing the output residual maps using recently introduced edge preserving filtering techniques. The proposed fusion mechanisms could also be applied to other sparsity-based classification algorithms. We believe that BTC and KBTC algorithms together constitute a complete classification framework.

8.2. Discussion

Although the proposed algorithms are based on sparse representation, they significantly differ from the other sparsity-based techniques when performing sparse recovery. It is known that conventional methods (l_1 minimization, greedy pursuits) use iterative expressions at this stage. However, our proposals consist of non-iterative structures such as thresholding and Tikhonov regularized sparse code estimation which result in fast classification operation. Other than speed issues, it is unclear if an iteration-based approach can perform satisfactory results or it can converge in most cases. Further studies are required on the robustness of the iteration-based methods for classification applications.

At the dictionary pruning stages of the proposed techniques, we apply a fixed thresholding policy which has been shown to be robust. On the other hand, it is doubtful whether an adaptive pruning stage will improve the classification accuracies or not. Even so, it is difficult to adapt a correlation based adaptive stage and the performance improvement is not guaranteed. It is also worth mentioning that the kernelized version of the proposal uses RBF kernel which is quite common and popular kernel function in classification applications. It is suspicious if the other type of kernel functions such as polynomial kernel will improve the performance. We also note that KBTC is superior to BTC especially in non-linearly separable cases. However, it does not mean that it is always superior. In some applications, linear algorithms outperform the non-linear ones.

8.3. Future Directions

There are many future directions related to the proposed algorithms to consider. Let us mention them one by one:

- As we can observe, both proposals involve inverse matrix operation. Using the properties of symmetric positive definite matrices, the inverse operation could be performed more efficiently in order to reduce the computational cost. Also for this purpose, the properties of Gram matrices could be further investigated.

- Currently, selection of the pruned dictionary is performed based on linear and non-linear correlations. More sophisticated approaches could be utilized in order to improve this step.

- We have measured the performances of the proposed techniques using the applications involving elementary features or those containing simple feature projections. It is required to investigate them under more advanced transform techniques such as scale-invariant feature transform (SIFT) [137, 138], histogram of oriented gradients (HOG) [139], Hough transform [140], etc.

- In HSI classification, spatial-spectral extensions of the proposals currently utilize the gray-scale guidance image obtained via PCA of the given HSI in the edge preserving smoothing stages. We believe that filtering the guidance image using an aggressive edge-aware filter such as L_0 smoothing technique [27] further improves the classification performance.

- It would be interesting to investigate the performance of the KBTC algorithm under multiple kernel learning framework [141, 142] based on the fact that the real world data in the feature space is highly heterogeneous. Another future direction could be adapting a dictionary learning [143] stage which may improve the classification accuracy as well as the computational efficiency.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[2] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[3] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.

[4] X. Mei and H. Ling, "Robust visual tracking and vehicle classification via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2259–2272, 2011.

[5] X.-T. Yuan, X. Liu, and S. Yan, "Visual classification with multitask joint sparse representation," *IEEE Transactions on Image Processing*, vol. 21, no. 10, pp. 4349–4360, 2012.

[6] L. Zhang, M. Yang, and X. Feng, "Sparse representation or collaborative representation: Which helps face recognition?" in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 471–478.

[7] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Kernel sparse representation for image classification and face recognition," in *ECCV Computer Vision*. Springer, 2010, pp. 1–14.

[8] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 543–550.

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

[10] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via l_1 minimization," *Proceedings of the National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.

[11] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[12] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution," *Communications on pure and applied mathematics*, vol. 59, no. 6, pp. 797–829, 2006.

[13] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[14] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[15] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 2272–2279.

[16] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, 2008.

[17] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[18] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[19] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 6, pp. 1127–1133, 2010.

[20] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3467–3478, 2012.

[21] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *2012 IEEE Conference on Computer vision and pattern recognition (CVPR)*. IEEE, 2012, pp. 1822–1829.

[22] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, "Online discriminative object tracking with local sparse representation," in *IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 2012, pp. 425–432.

[23] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma, "Toward a practical face recognition system: Robust alignment and illumination by sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 372–386, 2012.

[24] M. Yang and L. Zhang, "Gabor feature based sparse representation for face recognition with gabor occlusion dictionary," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 448–461.

[25] H. Zhang, F. Wang, Y. Chen, W. Zhang, K. Wang, and J. Liu, "Sample pair based sparse representation classification for face recognition," *Expert Systems with Applications*, vol. 45, pp. 352–358, 2016.

[26] Y. Xu, Z. Zhang, G. Lu, and J. Yang, "Approximately symmetrical face images for image preprocessing in face recognition and sparse representation based classification," *Pattern Recognition*, 2016.

[27] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via l0 gradient minimization," in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6. ACM, 2011, p. 174.

[28] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1794–1801.

[29] R. Rigamonti, M. A. Brown, and V. Lepetit, "Are sparse representations really relevant for image classification?" in *2011 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 1545–1552.

[30] Y. Liu, X. Li, C. Liu, and H. Liu, "Structure-constrained low-rank and partial sparse representation with sample selection for image classification," *Pattern Recognition*, 2016.

[31] M. Huang, Z. Mu, and H. Zeng, "Efficient image classification via sparse coding spatial pyramid matching representation of sift-wcs-ltp feature," *Image Processing, IET*, vol. 10, no. 1, pp. 61–67, 2016.

[32] E. Candes and J. Romberg, "l1-magic: Recovery of sparse signals via convex programming," 2005, last accessed on: 2016-03-24. [Online]. Available: www.acm.caltech.edu/l1magic/downloads/l1magic.pdf

[33] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted l_1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[34] S. Becker, J. Bobin, and E. J. Candès, "Nesta: a fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[35] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[36] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[37] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. part i: Greedy pursuit," *Signal Processing*, vol. 86, no. 3, pp. 572–588, 2006.

[38] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.

[39] T. Goldstein and S. Osher, "The split bregman method for l1-regularized problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009.

[40] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, "Bregman iterative algorithms for l_1-minimization with applications to compressed sensing," *SIAM Journal on Imaging Sciences*, vol. 1, no. 1, pp. 143–168, 2008.

[41] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[42] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[43] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[44] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition*. IEEE, 1991, pp. 586–591.

[45] A. M. Martínez and A. C. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[46] Y. Xu, Q. Zhu, Z. Fan, M. Qiu, Y. Chen, and H. Liu, "Coarse to fine k nearest neighbor classifier," *Pattern recognition letters*, vol. 34, no. 9, pp. 980–986, 2013.

[47] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 8, pp. 1778–1790, 2004.

[48] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[49] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[50] B. B. Le Cun, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*. Citeseer, 1990.

[51] M. A. Kashem, M. N. Akhter, S. Ahmed, and M. M. Alam, "Face recognition system based on principal component analysis (pca) with back propagation neural networks (bpnn)," *Canadian Journal on Image Processing and Computer Vision*, vol. 2, no. 4, pp. 36–45, 2011.

[52] P. Latha, L. Ganesan, and S. Annadurai, "Face recognition using neural networks," *Signal Processing: An International Journal (SPIJ)*, vol. 3, no. 5, pp. 153–160, 2009.

[53] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 2015, 2015.

[54] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[55] T. Joachims, *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

[56] ——, "Transductive inference for text classification using support vector machines," in *ICML*, vol. 99, 1999, pp. 200–209.

[57] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[58] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[59] G. Guo, S. Z. Li, and K. Chan, "Face recognition by support vector machines," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. IEEE, 2000, pp. 196–201.

[60] B. Heisele, P. Ho, and T. Poggio, "Face recognition with support vector machines: Global versus component-based approach," in *IEEE International Conference on Computer Vision*, vol. 2. IEEE, 2001, pp. 688–694.

[61] P. J. Phillips *et al.*, *Support vector machines applied to face recognition*. Citeseer, 1998, vol. 285.

[62] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, "Mismatch string kernels for discriminative protein classification," *Bioinformatics*, vol. 20, no. 4, pp. 467–476, 2004.

[63] C. S. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: A string kernel for svm protein classification." in *Pacific symposium on biocomputing*, vol. 7, 2002, pp. 566–575.

[64] E. Eskin, J. Weston, W. S. Noble, and C. S. Leslie, "Mismatch string kernels for svm protein classification," in *Advances in neural information processing systems*, 2002, pp. 1417–1424.

[65] K. Tsuda, H. Shin, and B. Schölkopf, "Fast protein classification with multiple networks," *Bioinformatics*, vol. 21, no. suppl 2, pp. ii59–ii65, 2005.

[66] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[67] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[68] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using svms and morphological profiles," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 11, pp. 3804–3814, 2008.

[69] Y. Bazi and F. Melgani, "Toward an optimal svm classification system for hyperspectral remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 11, pp. 3374–3385, 2006.

[70] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "Svm-and mrf-based method for accurate classification of hyperspectral images," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 4, pp. 736–740, 2010.

[71] G. Camps-Valls and L. Bruzzone, "Kernel-based methods for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 6, pp. 1351–1362, 2005.

[72] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[73] G. H. Golub, P. C. Hansen, and D. P. O'Leary, "Tikhonov regularization and total least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 1, pp. 185–194, 1999.

[74] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[75] M. R. Osborne, B. Presnell, and B. A. Turlach, "A new approach to variable selection in least squares problems," *IMA journal of numerical analysis*, vol. 20, no. 3, pp. 389–403, 2000.

[76] 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.

[77] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*. Springer, 2013.

[78] 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.

[79] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[80] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[81] P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[82] A. Y. Yang, S. S. Sastry, A. Ganesh, and Y. Ma, "Fast l_1-minimization algorithms and an application in robust face recognition: A review," in *17th IEEE International Conference on Image Processing*, Hong Kong, 2010, pp. 1849–1852.

[83] Q. Shi, H. Li, and C. Shen, "Rapid face recognition using hashing," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2753–2760.

[84] Q. Shi, A. Eriksson, A. Van Den Hengel, and C. Shen, "Is face recognition really a compressive sensing problem?" in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 553–560.

[85] M. A. Toksoz and I. Ulusoy, "Classification via ensembles of basic thresholding classifiers," *IET Computer Vision*, 2015, accepted for publication.

[86] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[87] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2001, pp. 274–281.

[88] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 287–296.

[89] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[90] B. Krawczyk, M. Woźniak, and B. Cyganek, "Clustering-based ensembles for one-class classification," *Information Sciences*, vol. 264, pp. 182–195, 2014.

[91] B. Krawczyk, M. Woźniak, and F. Herrera, "On the usefulness of one-class classifier ensembles for decomposition of multi-class problems," *Pattern Recognition*, 2015.

[92] B. Cyganek, "One-class support vector ensembles for image segmentation and classification," *Journal of Mathematical Imaging and Vision*, vol. 42, no. 2-3, pp. 103–117, 2012.

[93] A. Y. Yang, J. Wright, Y. Ma, and S. S. Sastry, "Feature selection in face recognition: A sparse representation perspective," *IEEE Transactions Pattern Analysis and Machine Intelligence*, 2007.

[94] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Drcw-ovo: Distance-based relative competence weighting combination for one-vs-one strategy in multi-class problems," *Pattern Recognition*, vol. 48, no. 1, pp. 28–42, 2015.

[95] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, 1960.

[96] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2001, pp. I–511.

[97] L. Spacek, "Collection of facial images," 2007, last accessed on: 2016-03-24. [Online]. Available: http://cswww.essex.ac.uk/mv/allfaces/index.html

[98] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55–63, 1968.

[99] J. C. Harsanyi and C.-I. Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 4, pp. 779–785, 1994.

[100] J. Wang and C.-I. Chang, "Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 6, pp. 1586–1600, 2006.

[101] W. Li, S. Prasad, J. E. Fowler, and L. M. Bruce, "Locality-preserving dimensionality reduction and classification for hyperspectral image analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 4, pp. 1185–1198, 2012.

[102] G. Mountrakis, J. Im, and C. Ogole, "Support vector machines in remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, no. 3, pp. 247–259, 2011.

[103] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[104] G. Camps-Valls, L. Gomez-Chova, J. Muñoz-Marí, J. Vila-Francés, and J. Calpe-Maravilla, "Composite kernels for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 1, pp. 93–97, 2006.

[105] Y. Tarabalka, J. A. Benediktsson, and J. Chanussot, "Spectral–spatial classification of hyperspectral imagery based on partitional clustering techniques," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 8, pp. 2973–2987, 2009.

[106] Y. Tarabalka, J. Chanussot, and J. A. Benediktsson, "Segmentation and classification of hyperspectral images using watershed transformation," *Pattern Recognition*, vol. 43, no. 7, pp. 2367–2379, 2010.

[107] X. Kang, S. Li, and J. A. Benediktsson, "Spectral-spatial hyperspectral image classification with edge-preserving filtering," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 5, pp. 2666–2677, 2014.

[108] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proceedings of European Conference on Computer Vision*, Heraklion, Crete, Greece, 2010, pp. 1–14.

[109] D. Böhning, "Multinomial logistic regression algorithm," *Annals of the Institute of Statistical Mathematics*, vol. 44, no. 1, pp. 197–200, 1992.

[110] J. S. Borges, A. R. Marçal, and J. M. Bioucas-Dias, "Evaluation of bayesian hyperspectral image segmentation with a discriminative class learning," in *IEEE International Geoscience and Remote Sensing Symposium*, Barcelona, Spain, 2007, pp. 3810–3813.

[111] J. Li, J. Bioucas-Dias, and A. Plaza, "Spectral–spatial hyperspectral image segmentation using subspace multinomial logistic regression and markov random fields," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 3, pp. 809–823, 2012.

[112] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Semi–supervised hyperspectral image segmentation using multinomial logistic regression with active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4085–4098, 2010.

[113] J. Li, J. Bioucas-Dias, and A. Plaza, "Hyperspectral image segmentation using a new bayesian approach with active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3947–3960, 2011.

[114] J. Bioucas-Dias and M. Figueiredo, "Logistic regression via variable splitting and augmented lagrangian tools," *Instituto Superior Técnico, TULisbon, Tech. Rep*, 2009.

[115] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3973–3985, 2011.

[116] ——, "Hyperspectral image classification via kernel sparse representation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 1, pp. 217–231, 2013.

[117] E. Zhang, X. Zhang, L. Jiao, L. Li, and B. Hou, "Spectral-spatial hyperspectral image ensemble classification via joint sparse representation," *Pattern Recognition*, 2016.

[118] E. Zhang, X. Zhang, L. Jiao, H. Liu, S. Wang, and B. Hou, "Weighted multifeature hyperspectral image classification via kernel joint sparse representation," *Neurocomputing*, vol. 178, pp. 71–86, 2016.

[119] C. Bo, H. Lu, and D. Wang, "Hyperspectral image classification via jcr and svm models with decision fusion," 2016.

[120] J. Liu, Z. Wu, Z. Wei, L. Xiao, and L. Sun, "Spatial-spectral kernel sparse representation for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 6, pp. 2462–2471, 2013.

[121] J. Li, H. Zhang, and L. Zhang, "Column-generation kernel nonlocal joint collaborative representation for hyperspectral image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 94, no. 1, pp. 25–36, 2014.

[122] H. Zhang, J. Li, Y. Huang, and L. Zhang, "A nonlocal weighted joint sparse representation classification method for hyperspectral imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2056–2065, 2014.

[123] L. Fang, S. Li, X. Kang, and J. A. Benediktsson, "Spectral-spatial hyperspectral image classification via multiscale adaptive sparse representation," *IEEE*

Transactions on Geoscience and Remote Sensing, vol. 52, no. 12, pp. 7738–7749, 2014.

[124] J. Zou, W. Li, X. Huang, and Q. Du, "Classification of hyperspectral urban data using adaptive simultaneous orthogonal matching pursuit," *Journal of Applied Remote Sensing*, vol. 8, no. 1, pp. 085 099:1–085 099:14, 2014.

[125] M. A. Toksoz and I. Ulusoy, "Hyperspectral image classification via basic thresholding classifier," *IEEE Transactions on Geoscience and Remote Sensing*, 2016, accepted for publication.

[126] J. Cohen *et al.*, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[127] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," in *ACM Transactions on Graphics*, vol. 27, no. 3. ACM, 2008, pp. 67:1–67:10.

[128] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[129] L. J. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," *Journal of Machine Learning Research*, vol. 10, no. 1-41, pp. 66–71, 2009.

[130] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[131] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the Sixth International Conference on Computer Vision*, Bombay, India, 1998, pp. 839–846.

[132] Y. Saad, *Iterative methods for sparse linear systems*. Siam, 2003.

[133] M. A. Toksoz and I. Ulusoy, "Hyperspectral image classification via kernel basic thresholding classifier," *IEEE Transactions on Geoscience and Remote Sensing*, 2016, manuscript submitted for publication.

[134] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.

[135] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote sensing of environment*, vol. 113, pp. S110–S122, 2009.

[136] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral–spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 844–856, 2013.

[137] D. G. Lowe, "Object recognition from local scale-invariant features," in *The proceedings of the seventh IEEE international conference on Computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[138] ——, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[139] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2005, pp. 886–893.

[140] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[141] F. R. Bach, G. R. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 6.

[142] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *The Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.

[143] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 689–696.

APPENDIX A

MATLAB CODES

The following function implements the BTC algorithm.

```
function [btcResult, errMatrix] = btc(labels, A, Y, M, alpha)
  % This function implements the basic thresholding classifier for any
  % classification applications such as face recognition, hyperspectral
  % image classification, etc.
  %
  % INPUTS:
  % labels: 1 X N vector of integers consisting of class labels
  % of the input training samples where N denotes the number of
  % training samples.
  % A: B X N dictionary whose columns represent L2 normalized training
  % samples where B represents the number of features.
  % Y: B X L matrix consisting of testing samples where L represents
  % the number of testing samples. For fast computation, it is
  % recommended that L is less than 10000. If it is larger, then Y can
  % be partitioned.
  % M: Threshold parameter (integer) which is less than B.
  % alpha: Regularization parameter between 0 and 1.
  %
  % OUTPUTS:
  % errMatrix: C X L error matrix containing residual for each sample
  % btcResult: 1 X L decision vector containing predicted labels

 [~, N] = size(A);
 [~, L] = size(Y);
 C = max(labels);  % C shows the number of classes
 X = zeros(N, L);  % initialize the sparse codes for each test sample
```

```
 % calculate gram matrix, this can be performed outside the func
mappedTrains = A' * A;
 % linear correlation vector is calculated for each test sample
mappedAy = A' * Y;
 % correlation vectors are sorted in descending order
[~, sortedLabels] = sort(abs(mappedAy), 'descend');
I = alpha * eye(M);   % regularization matrix
for k = 1:L  % sparse code vector is calculated for each sample
    support = sortedLabels(1:M, k);
    X(support, k) = (mappedTrains(support,support) + I) \ ...
    mappedAy(support,k);
end
errMatrix = zeros(C, L);   % initialize error (residual) matrix
 % calculate class-wise regression error vector for each sample
for classIndex = 1:C
    ind = (labels == classIndex);
    Yp = A(:,ind)*X(ind, :);
    errMatrix(classIndex,:) = sqrt(sum((Y-Yp).^2, 1));
end
 % make decision for each sample based on minimum residual
[~, btcResult] = min(errMatrix, [], 1);
end
```

The function below is used to calculate $\overline{\beta}$ quantity for BTC. It also estimates the threshold parameter M.

```
function [avgBeta, bestM] = averageBetaBtc(labels, A, alpha)
 % This function implements the quantity average beta for btc.
 %
 % INPUTS:
 % labels: 1 X N vector of integers consisting of class labels
 % of the input training samples where N denotes the number of
 % training samples.
 % A: B X N dictionary whose columns represent L2 normalized training
 % samples where B represents the number of features.
 % alpha: Regularization parameter between 0 and 1.
 %
 % OUTPUTS:
 % avgBeta: 1 X B-1 vector containing beta values. Plot it and see at
```

```
  % which index it becomes minimum. This index gives the best M.
  % bestM: The best value of the threshold M (estimated)

[B, N] = size(A);
C = max(labels);
avgBeta = zeros(1, B-1);
mappedTrains = A' * A;
[~, sortedLabels] = sort(abs(mappedTrains), 'descend');
 % for each M, determine the beta value
for m=1:B-1
    X = zeros(N, N);
    I = alpha * eye(m);
    for k = 1:N
        support = sortedLabels(2:m+1, k);
        X(support, k) = (mappedTrains(support,support) + I) \...
            mappedTrains(support,k);
    end
     % for each sample determine the residual
    errMatrix = zeros(C, N);
    for classIndex = 1:C
        ind = (labels == classIndex);
        Yp = A(:,ind)*X(ind, :);
        errMatrix(classIndex,:) = sqrt(sum(abs(A-Yp).^2,1));
    end
     % calculate the average beta
    [res, sorted] = sort(errMatrix);
    res1 = res(1,:);
    res2 = res(2,:);
    ind = (labels ~= sorted(1,:));
    res2(ind) = res1(ind);
    trueResiduals = errMatrix(labels + (0:N-1)*C);
    avgBeta(m) = mean(trueResiduals./res2);

end
[~, bestM] = min(avgBeta);  % estimate the threshold M.

end
```

The following function implements the KBTC algorithm. Please make sure that the columns of the training and testing matrices are normalized as described in Chapter

4.

```
function [kbtcResult, errMatrix] = kbtc(labels, A, Y, M, alpha, gamma)
  % This function implements the kernel basic thresholding classifier
  % for any classification applications such as face recognition,
  % hyperspectral image classification, etc.
  %
  % INPUTS:
  % labels: 1 X N vector of integers consisting of class labels
  % of the input training samples where N denotes the number of
  % training samples.
  % A: B X N dictionary whose columns represent normalized training
  % samples where B represents the number of features.
  % Y: B X L matrix consisting of testing samples where L represents
  % the number of testing samples. For fast computation, it is
  % recommended that L is less than 10000. If it is larger, then Y can
  % be partitioned.
  % M: Threshold parameter (integer) which is less than B.
  % alpha: Regularization parameter between 0 and 1.
  % gamma: Kernel parameter.
  %
  % OUTPUTS:
  % errMatrix: C X L error matrix containing residual for each sample
  % kbtcResult: 1 X L decision vector containing predicted labels

[~, N] = size(A);
[~, L] = size(Y);
C = max(labels);  % C shows the number of classes
X = zeros(N, L);  % initialize sparse codes for each sample
 % calculate gram matrix, this can be performed outside the func
mappedTrains = kernelFunction(gamma, A, A);
 % calculate nonlinear correlations
mappedAy = kernelFunction(gamma, A, Y);
[~, sortedLabels] = sort(abs(mappedAy), 'descend');
I = alpha * eye(M);  % regularization matrix

for k = 1:L
    support = sortedLabels(1:M, k);
    X(support, k)  = (mappedTrains(support,support) + I) \...
        mappedAy(support,k);
```

```
end
errMatrix = zeros(C, L); % initialize error (residual) matrix
 % calculate class-wise regression error vector for each sample
c1 = kernelFunction(gamma,Y);
for classIndex = 1:C
    ind = (labels == classIndex);
    KA = mappedTrains(ind,ind);
    c2 = -2*dot(X(ind,:),mappedAy(ind,:));
    c3 = X(ind,:)'*KA;
    c3 = dot(c3', X(ind,:));
    errMatrix(classIndex,:) = sqrt(abs(c1 + c2 + c3));
end
  % make decision for each sample based on minimum residual
[~, kbtcResult] = min(errMatrix, [], 1);
end
```

The function below is used to determine the parameters of the KBTC algorithm.

```
function [gamma, M] = determineKbtcParams(labels, A, alpha)
  % This function determines the parameters of kbtc
  %
  % INPUTS:
  % labels: 1 X N vector of integers consisting of class labels
  % of the input training samples where N denotes the number of
  % training samples.
  % A: B X N dictionary whose columns represent normalized training
  % samples where B represents the number of features.
  % alpha: Regularization parameter between 0 and 1.
  %
  % OUTPUTS:
  % gamma: Estimated kernel parameter
  % M: Estimated threshold

[B, ~] = size(A);
minBeta = inf;
bestGamma = 0;
bestGammaIndex = 1;
k = 1;
 % call average beta function for each gamma and determine the best
```

```
 % gamma at which beta becomes minimum.
for gammaIndex = 5:-1:-10
    gamma = 2^(gammaIndex);
    avgBetaGamma(k,:) = averageBetaKbtc(labels, A, alpha, gamma);
    avgBeta = mean(avgBetaGamma(k,:));
    if avgBeta < minBeta
        minBeta = avgBeta;
        bestGamma = gamma;
        bestGammaIndex = k;
    end
    disp(['gamma = ', num2str(gamma), ', avgbeta = ', num2str(avgBeta)])
    k = k + 1;
end
gamma = bestGamma;
disp(['best gamma = ', num2str(bestGamma)]);
 % estimate the threshold parameter
figure;
plot(1:B-1, avgBetaGamma(bestGammaIndex, :));
xlabel('threshold (M)');
ylabel('average beta');
[~, M] = min(avgBetaGamma(bestGammaIndex, :));
disp(['best threshold = ', num2str(M)]);
end
```

The following function implements the $\overline{\beta}$ quantity for KBTC.

```
function [avgBeta] = averageBetaKbtc(labels, A, alpha, gamma)
  % This function implements the quantity average beta for kbtc.
  %
  % INPUTS:
  % labels: 1 X N vector of integers consisting of class labels
  % of the input training samples where N denotes the number of
  % training samples.
  % A: B X N dictionary whose columns represent normalized training
  % samples where B represents the number of features.
  % alpha: Regularization parameter between 0 and 1.
  % gamma: RBF Kernel parameter.
  %
  % OUTPUTS
```

```
  % avgBeta: 1 X B-1 vector containing beta values. Plot it and see at
  % which index it becomes minimum. This index gives the best M.


[B, N] = size(A);
C = max(labels);
avgBeta = zeros(1, B-1);
mappedTrains = kernelFunction(gamma, A, A);
[~, sortedLabels] = sort(abs(mappedTrains), 'descend');

for m =  1:B-1  % for each M, determine the beta value
    X = zeros(N, N);
    I = alpha * eye(m);
    for k = 1:N
        support = sortedLabels(2:m+1, k);
        X(support, k)  = (mappedTrains(support,support) + I) \...
            mappedTrains(support,k);
    end
    errMatrix = zeros(C, N);
     % for each sample determine the residual
    c1 = kernelFunction(gamma,A);
    for classIndex = 1:C
        ind = (labels == classIndex);
        KA = mappedTrains(ind,ind);
        c2 = -2*dot(X(ind,:), mappedTrains(ind,:));
        c3 = X(ind,:)'*KA;
        c3 = dot(c3',X(ind,:));
        errMatrix(classIndex,:) = sqrt(abs(c1 + c2 + c3));
    end
     % calculate the average beta
    [res, sorted] = sort(errMatrix);
    res1 = res(1,:);
    res2 = res(2,:);
    ind = (labels ~= sorted(1,:));
    res2(ind) = res1(ind);
    trueResiduals = errMatrix(labels + (0:N-1)*C);
    avgBeta(m) = mean(trueResiduals./res2);

end
```

```
end
```

The function below is used to determine the kernel matrix for KBTC algorithm.

```
function [corr] = kernelFunction(gamma, D1, D2)
 % This function implements the RBF kernel
 %
 % INPUTS:
 % gamma: Kernel parameter
 % D1: First matrix
 % D2: Second matrix
 % OUTPUT:
 % corr: Correlation matrix or vector

if nargin > 2
    n1sq = sum(D1.^2,1);
    n1 = size(D1,2);
    n2sq = sum(D2.^2,1);
    n2 = size(D2,2);
    c = (ones(n2,1)*n1sq)' + ones(n1,1)*n2sq -2*(D1'*D2);
else
    c = sum((D1-D1).^2,1);
end
corr = exp(-gamma*c);
end
```

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Toksöz, Mehmet Altan
Nationality: Turkish (TC)
Date and Place of Birth: 01.05.1982, Ankara
Marital Status: Single
Phone: +90 506 959 16 09
Fax: +90 312 291 6004

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Bilkent University Electrical and Electronics Engineering	2009
B.S.	University of Applied Sciences Upper Austria Electrical Engineering and Computer Science	2006
B.S.	Anadolu University Electrical and Electronics Engineering	2006

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2010-2016	Tübitak Bilgem İltaren	Senior Researcher
2007-2009	Bilkent University	Research and Teaching Assistant
2006-2007	Akad Elektronik	Software Engineer

HONORS, SCHOLARSHIPS AND AWARDS

- Ranked 79th in the quantitative area among 1.5 million students in the National University Entrance Exam of 2001.
- Scholarship for undergraduate study in Austria for one year, 2005-2006.
- Full scholarship for Master of Science Study awarded by Bilkent University.
- Master of Science scholarship awarded by the Scientific and Technological Research Council of Turkey.

PUBLICATIONS

International Journal Papers

- M. A. Toksoz and I. Ulusoy, "Hyperspectral Image Classification via Kernel Basic Thresholding Classifier", IEEE Transactions on Geoscience and Remote Sensing, Manuscript submitted for publication, 2016.
- M. A. Toksoz and I. Ulusoy, "Hyperspectral Image Classification via Basic Thresholding Classifier", IEEE Transactions on Geoscience and Remote Sensing, Manuscript accepted for publication, 2016.
- M. A. Toksoz and I. Ulusoy, "Classification via ensembles of basic thresholding classifiers", IET Computer Vision, Manuscript accepted for publication, 2015.
- N. Akar and M. A. Toksoz, "MPLS Automatic Bandwidth Allocation via Adaptive Hysteresis", Computer Networks, vol. 55, no. 5, pp. 1181-1196, April 2011.
- M. A. Toksoz and N. Akar, "Dynamic Threshold-based Assembly Algorithms for Optical Burst Switching Networks Subject to Burst Rate Constraints", Photonic Network Communications, vol. 20, no. 2, pp. 120-130, Oct. 2010.