ABSTRACT

Title of Dissertation: SPARSE REPRESENTATION, DISCRIMINATIVE DICTIONARIES AND PROJECTIONS FOR VISUAL CLASSIFICATION

Ashish Shrivastava, Doctor of Philosophy, 2015

Dissertation directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

Developments in sensing and communication technologies have led to an explosion in the availability of visual data from multiple sources and modalities. Millions of cameras have been installed in buildings, streets, and airports around the world that are capable of capturing multimodal information such as light, depth, heat etc. These data are potentially a tremendous resource for building robust visual detectors and classifiers. However, the data are often large, mostly unlabeled and increasingly of mixed modality. To extract useful information from these heterogeneous data, one needs to exploit the underlying physical, geometrical or statistical structure across data modalities. For instance, in computer vision, the number of pixels in an image can be rather large, but most inference or representation models use only a few parameters to describe the appearance, geometry, and dynamics of a scene. This has motivated researchers to develop a number of techniques for finding a low-dimensional representation of a high-dimensional dataset. The dominant methodology for modeling and exploiting the low-dimensional structure in high dimensional data is sparse dictionary-based modeling. While discriminative dictionary learning have demonstrated tremendous success in computer vision applications, their performance is often limited by the amount and type of labeled data available for training. In this dissertation, we extend the sparse dictionary learning

framework for weakly supervised learning problems such as semi-supervised learning, ambiguously labeled learning and Multiple Instance Learning (MIL). Furthermore, we present nonlinear extensions of these methods using the kernel trick. We also address the problem of choosing the optimal kernel for sparse representation-based classification using Multiple Kernel Learning (MKL) methods. Finally, in order to deal with heterogeneous multimodal data, we present a feature level fusion method based on quadratic programing. The dissertation has been divided into following four parts:

1) In the first part, we develop a discriminative non-linear dictionary learning technique which utilizes both labeled and unlabeled data for learning dictionaries. We compute a probability distribution over class labels for all the unlabeled samples which is updated together with dictionary and sparse coefficients. The algorithm is also extended for ambiguously labeled data when part of the data contains multiple labels for a training sample.

2) Using non-linear dictionaries, we present a multi-class Multiple Instance Learning (MIL) algorithm where the data is given in the form of bags. Each bag contains multiple samples, called instances, out of which at least one belongs to the class of the bag. We propose a noisy-OR model and a generalized mean-based optimization framework for learning the dictionaries in the feature space. The proposed method can be viewed as a generalized dictionary learning algorithm since it reduces to a novel discriminative dictionary learning framework when there is only one instance in each bag.

3) We propose a Multiple Kernel Learning (MKL) algorithm that is based on the Sparse Representation-based Classification (SRC) method. Taking advantage of the non-linear kernel SRC in efficiently representing the non-linearities in the high-dimensional feature space, we propose an MKL method based on the kernel alignment criteria. Our method uses a two step training method to learn the kernel weights and the sparse codes. At each iteration, the sparse codes are updated first while fixing the kernel mixing coefficients, and then the kernel mixing coefficients are updated while fixing the sparse codes. These two steps are repeated until a

stopping criteria is met.

4) Finally, using a linear classification model, we study the problem of fusing information from multiple modalities. Many current recognition algorithms combine different modalities based on training accuracy but do not consider the possibility of noise at test time. We describe an algorithm that perturbs test features so that all modalities predict the same class. We enforce this perturbation to be as small as possible via a quadratic program (QP) for continuous features, and a mixed integer program (MIP) for binary features. To efficiently solve the MIP, we provide a greedy algorithm and empirically show that its solution is very close to that of a state-of-the-art MIP solver.

SPARSE REPRESENTATION, DISCRIMINATIVE DICTIONARIES AND PROJECTIONS FOR VISUAL CLASSIFICATION

by

Ashish Shrivastava

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Larry S. Davis
Professor Jonathan Simon
Professor Amitabh Varshney
Dr. Vishal M. Patel

Acknowledgments

I owe my gratitude to all the people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Rama Chellappa for giving me an invaluable opportunity and freedom to work on challenging and extremely interesting problems over the past five years. He has always made himself available for help and advice and there has never been an occasion when I've knocked on his door and he hasn't given me time. His outstanding support and encouraging words have always inspired me to work hard and stay focused on my research. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank research associate Dr. Vishal Patel, who helped me with many technical ideas and taught me a great deal on writing papers. Without his fantastic ideas and expertise, this dissertation would have been a distant dream. Thanks are due to Professor Larry Davis, Professor Jonathan Simon and Professor Amitabh Varshney for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript. My special thanks to Professor Larry Davis for his extremely useful comments and suggestions on my multi-modal learning work.

I would like to thank all my colleagues at ECE and UMIACS who have enriched my graduate life in many ways. My co-author and a great friend Jai Pillai deserves special mention for helping me with my first paper and participating in multiple

useful discussions.

I owe my deepest thanks to my parents who have always stood by me and motivated me through my career. Words cannot express the gratitude I owe them. My special gratitude to my wife Prathyusha for her constant encouragement and support. Last but not the least, I am grateful to my friends and roommates who have been a crucial factor in my finishing smoothly.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

In computer vision and machine learning applications, the data is often very high dimensional and usually corrupted by noise. This requires us to develop robust models for data representation to mitigate the effects of *curse of dimensionality*. Recently, researchers have shown that sparse representation-based methods can achieve state-of-the-art performance in many signal and image processing applications. The success of sparse representation and dictionary-based algorithms is essentially due to the fact that the signals or images of interest, though high dimensional, can often be coded using a few representative atoms in some dictionary. This has resulted in rapid development, both in theory and in algorithms, of the field of sparse representation in recent years [1–4].

While discriminative dictionary learning algorithms have demonstrated tremendous success for image classification, their performance is often limited by the amount and type of labeled data available for training. Furthermore, the available labels might be erroneous due to monotonous nature of labeling process. In many cases, the labeling efforts can be significantly reduced by allowing some noise in the labeling process. For example, for an object classifier, instead of drawing a bounding box around an object, it's easier to indicate the presence or absence

of the object in the image. In this dissertation, we present various approaches to extend sparse dictionary learning framework for weakly supervised learning problems such as semi-supervised learning, ambiguously labeled learning and Multiple Instance Learning (MIL). Furthermore, we present non-linear extensions of these methods using the kernel trick. The choice of kernel for non-linear methods is often made using cross-validation. However, joint learning of the kernel and the classification model often improves the performance of the non-linear model. We develop an algorithm for choosing the optimal kernel for sparse representation-based classification using Multiple Kernel Learning (MKL) methods. Finally, in order to deal with heterogeneous multimodal data, we present a feature level fusion method based on quadratic programing.

Next, we give an overview of sparse representation and dictionary learning. Let $\mathbf{D}$ be a redundant dictionary with K atoms in $\mathbb{R}^d$

$$\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K] \in \mathbb{R}^{d \times K}.$$

The atoms have unit Euclidean norm i.e., $\|\mathbf{d}_i\| = 1 \ \ \forall i$. Given a signal $\mathbf{y} \in \mathbb{R}^d$, finding the sparsest representation of $\mathbf{y}$ in $\mathbf{D}$ entails solving the following optimization problem

$$\mathbf{x} = \arg\min_{\mathbf{z}} \|\mathbf{z}\|_0 \quad \text{subject to } \mathbf{y} = \mathbf{D}\mathbf{z}, \tag{1.1}$$

where the $\|\mathbf{z}\|_0 := \#\{j : z_j \neq 0\}$, which is a count for the number of nonzero elements in $\mathbf{z}$. Problem (1.1) is NP-hard and cannot be solved in a polynomial time. Hence, approximate solutions are usually sought [3, 5–7]. For instance, Basis

Pursuit [5] offers the solution via ℓ_1-minimization as

$$\mathbf{x} = \arg\min_{\mathbf{z}} \|\mathbf{z}\|_1 \quad \text{subject to } \mathbf{y} = \mathbf{D}\mathbf{z}, \tag{1.2}$$

where $\|\cdot\|_p$ for $0 < p < \infty$ is the ℓ_p-norm defined as

$$\|\mathbf{z}\|_p = \left(\sum_{j=1}^{d} |z_j|^p \right)^{\frac{1}{p}}.$$

The sparsest recovery is possible provided when certain conditions are met [8], [4]. One can adapt the above framework to a more practical noisy setting, where the measurements are contaminated with an error $\mathbf{n}$ obeying $\|\mathbf{n}\|_2 < \epsilon$, that is

$$\mathbf{y} = \mathbf{D}\mathbf{x} + \mathbf{n} \quad \text{for} \quad \|\mathbf{n}\|_2 < \epsilon. \tag{1.3}$$

A stable solution can be obtained by solving the following optimization problem [4]

$$\mathbf{x} = \arg\min_{\mathbf{z}} \|\mathbf{z}\|_1 \quad \text{subject to } \|\mathbf{y} - \mathbf{D}\mathbf{z}\|_2 < \epsilon. \tag{1.4}$$

One of the major challenges in sparse modeling of the signal is to find an appropriate dictionary $\mathbf{D}$ in which data is well represented with sparse coefficients. This dictionary can be analytic such as overcomplete wavelets, curvelets, contourlets etc. or it can be learned using data. Predetermined dictionaries are appealing due to their simplicity and can lead to fast algorithms for computation of the sparse coefficients. However, it has been observed that learning dictionary directly from data usually leads to better performance. One of the effective methods to learn dictionary using data is called Method of Optimal Directions (MOD) [9] which iteratively updates $\mathbf{D}$ by reducing the mean square error (MSE) at each iteration. Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$ be the data matrix of N samples, and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ be the

the matrix consisting of corresponding N coefficients. The goal in MOD algorithm, at each iterations, is to update $\mathbf{D}$ such that the sum of error norms $\mathbf{r}_i := \mathbf{y}_i - \mathbf{D}\mathbf{x}_i$ is minimized. This can be achieved by minimizing the following cost,

$$\mathcal{E} = \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2, \tag{1.5}$$

where $\|.\|_F^2$ denotes the Frobenius norm. By setting derivative of $\mathcal{E}$ to zero, one obtains the following update rule for the dictionary at $(t+1)^{\text{th}}$ iteration,

$$\mathbf{D}^{(t+1)} = \mathbf{Y}\mathbf{X}^{(t)^T}(\mathbf{X}^{(t)}\mathbf{X}^{(t)^T})^{-1}, \tag{1.6}$$

where, $\mathbf{X}^{(t)}$ is the coefficient matrix at t^{th} iteration. Due to matrix inversion operation in (1.6), this method is impractical for very large number of dictionary columns. Another popular method to learn the dictionary is K-SVD [10] which, similar to MOD, iteratively updates $\mathbf{D}$ and $\mathbf{X}$, however, within an iteration, each atom is sequentially updated using singular value decomposition (SVD). To update k^{th} atom $\mathbf{d}_k$, we seek to minimize

$$\mathcal{E} = \|\mathbf{Y} - \sum_{j \neq k} \mathbf{d}_j \mathbf{x}_T^j - \mathbf{d}_k \mathbf{x}_T^k\|_F^2 = \|\mathbf{E}_k - \mathbf{d}_k \mathbf{x}_T^k\|_F^2 \tag{1.7}$$

with respect to $\mathbf{d}_k$ and $\mathbf{x}_T^k$, simultaneously. Here, $\mathbf{x}_T^i$ is the i^{th} row of the coefficient matrix $\mathbf{X}$. The optimization can be performed by computing the SVD of the matrix $\mathbf{E}_k$. Furthermore, in order to preserve the sparsity of $\mathbf{X}$, only those samples are considered that use the atom $\mathbf{d}_k$, i.e. those columns of $\mathbf{E}_k$ are removed that have corresponding zeros coefficients in $\mathbf{x}_T^k$.

It has been shown that sparse representation and dictionary learning methods work well in many inverse problems where the original signal $\mathbf{y}_t$ needs to be

reconstructed as accurately as possible, such as denoising, deconvolution and image inpainting [11–16]. The focus of this dissertation is classification task where the goal is to learn a model that can predict the class of a novel data sample. Various sparse representation-based methods have been used for classifying an unseen test sample into one of the numerous classes [2, 17–20]. Using labeled data, one can learn dictionary $\mathbf{D}_c$ for each class $c = 1, \ldots C$, where C is the total number of classes. Then, given a novel test sample $\mathbf{y}_t$, its class can be determined by computing its sparse representation in each dictionary separately and computing the residue $\mathbf{r}_{tc} = \mathbf{y}_t - \mathbf{D}\mathbf{x}_t$. Then, the class of $\mathbf{y}_t$ is the one that results in minimum residue norm, i.e.,

$$\text{class of } \mathbf{y}_t = \arg\min_c \|\mathbf{r}_{tc}\|_2. \tag{1.8}$$

There are various approaches for learning dictionary-based classification methods and predicting the class of novel test sample, which have been described or referred to in subsequent chapters. The success of all classification methods rely on the availability of the labeled data. However, collecting labeled data is very expensive and monotonous while unlabeled data can easily be obtained from the Internet or various publicly available datasets. Hence, we develop dictionary learning algorithms with limited labeled data and show that they can out perform existing methods in many applications. Specifically, in the first part of the dissertation, we develop a dictionary learning approach that uses labeled as well as unlabeled data to learn a classification model. Furthermore, linear models are not always the best way to represent the data and this motivates us to extend this to its non-linear version. Next, we note that in many applications, labeling may be provided for the collec-

tion of samples called bags. This falls under the realm of multiple instance learning (MIL) framework [21, 22] and we demonstrate that dictionary-based methods can be adopted to achieve state-of-the-art performance under this setting. The non-linear sparse and dictionary methods, need to choose a kernel function to compute the kernel matrix. This choice is generally made with cross validation. However, it has been shown that using the linear combination of multiple kernels can lead to better performance. Inspired by multiple kernel learning (MKL) approaches, we develop a method for sparse representation-based classification (SRC) using MKL techniques. Finally, we focus on combining information from multiple sources using linear classification models. We present a perturbation-based model that predicts the consistent label from all the available modalities. We introduce the proposed algorithms and their contributions below:

1.1 Proposed Algorithms and their Contributions

We describe the methods introduced in the dissertation and their key contributions below:

1. **Non-Linear Dictionary Learning with Partially Labeled Data:**

   In the first part of the dissertation, we consider the problem of utilizing unlabeled and ambiguously labeled data for visual classification. It has been established in semi-supervised literature [23–25] that the labels from labeled data can be propagated to the unlabeled samples in their proximity. We incorporate this fact by introducing a probability distribution over classes for

each unlabeled sample. Based on these distributions, a dictionary-based classification model is learned with both labeled and unlabeled data, and, using this model, the distributions of unlabeled samples are updated. The process is repeated until a stopping criterion is met.

Contributions: Researchers have explored dictionary learning methods for supervised and unsupervised methods, however, discriminative dictionary learning for semi-supervised remains largely unexplored. To the best of our knowledge, the proposed method is the first work that develops dictionary-based semi-supervised framework which directly uses the unlabeled samples based on their probability distribution. It significantly improves the classification performance compared to using labeled data alone as well as outperforms competing algorithms using partially labeled data.

2. **Generalized Dictionaries for Multiple Instance Learning:**

   Many object detection and classification algorithms are supervised in nature requiring large amount of training data to learn a good model. However, labels are provided by human annotator and, in many cases, can be slightly inaccurate that can have an adverse impact on the learned model. Also, labeling individual instances requires significantly more effort compared to labeling the sets of them. For example, indicating the presence or absence of an object in an image is much easier than drawing a bounding box around the object. Learning a classification model, when labels are provided for set of instances, is known as multiple instance learning.

Contributions: We develop a novel MIL method based on non-linear dictionary learning algorithm. The proposed method generalizes the discriminative dictionary learning framework using diverse-density criterion. Furthermore, we present the non-linear version of this algorithm that improves over the linear one.

3. **Multiple Kernel Learning for Sparse Representation-based Classification**

   The non-linear sparse representation and dictionary learning based algorithms compute the kernel matrices using a kernel function that is usually chosen with cross validation. We develop an algorithm for choosing an optimal kernel based on linear combination of multiple kernels known as Multiple Kernel Learning (MKL).

   Contributions: We propose a kernel sparse representation-based classification method based on MKL where multiple kernel functions are combined to obtain a better solution. Our method uses a two step training method using the SRC as the base learner. At each iteration, first the combination function parameters are updated while fixing the base learner parameters, and then the base learner parameters are updated while fixing the combination function parameters. These two steps are repeated until convergence.

4. **Class Consistent Multimodal Learning**

   Availability of information from multiple sources enables us to employ effective schemes to combine them in various machine learning tasks. Most of the

existing multi-modal fusion algorithms have been designed for continuous features and are not appropriate for binary features. Binary features help save storage and time, and are more robust to noise. As a result, they have shown remarkable performance in computer vision applications with large datasets. Furthermore, the current recognition algorithms generally combine different modalities based on training accuracy and do not consider the possibility of noise at test time. For the recognition problem, we propose to perturb the test features in a way that all modalities predict the same class.

Contributions: We enforce class consistency across all available modalities in a perturbation model to determine the class of multi-modal data item. Based on this notion of class consistency, we develop an efficient binary feature fusion algorithm.

1.2 Organization

The dissertation is organized as follows. In Chapter 2 we present a semi-supervised dictionary learning algorithm. Next, we develop the dictionary learning algorithm for multiple instance learning problem in Chapter 3. A multiple kernel learning-based sparse representation method is presented in Chapter 4. We develop a perturbation model-based multi-modal fusion algorithm for classification in Chapter 5. Finally, we conclude the dissertation and provide future directions in Chapter 6.

Chapter 2: Non-Linear Dictionary Learning with Partially Labeled Data

2.1 Introduction

While dictionaries are often trained to obtain good reconstruction, training supervised dictionaries with a specific discriminative criterion has also been considered. For instance, linear discriminant analysis (LDA)-based basis selection and feature extraction algorithm for classification using wavelet packets was proposed by Etemand and Chellappa [17] in the late nineties. Recently, similar algorithms for simultaneous sparse signal representation and discrimination have also been proposed [26], [27], [28] [29], [30], [31], [32], [19], [33].

Sparse representation and dictionary learning methods for unsupervised learning have also been proposed. In [34], a method for simultaneously learning a set of dictionaries that optimally represent each cluster is proposed. To improve the accuracy of sparse coding, this approach was later extended by adding a block incoherence term in their optimization problem [35]. Some of the other sparsity motivated clustering and subspace clustering methods include [36], [37].

The performance of a supervised classification algorithm is often dependent

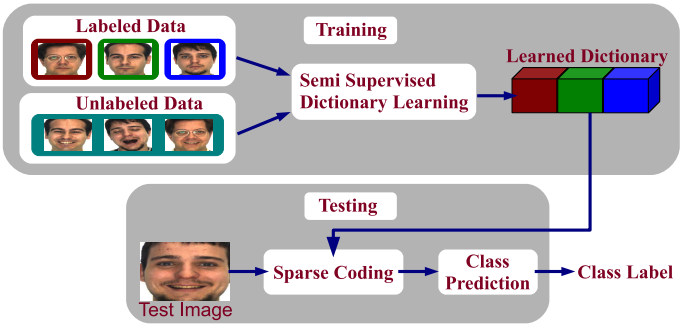


Figure 2.1: Block diagram illustrating semi-supervised dictionary learning.

on the quality and diversity of training images, which are mainly hand-labeled. However, labeling images is expensive and time consuming due to the significant human effort involved. On the other hand, one can easily obtain large amounts of unlabeled images from public image datasets like Flickr or by querying image search engines like Bing. This has motivated researchers to develop semi-supervised algorithms, which utilize both labeled and unlabeled data for learning classifier models. Such methods have demonstrated improved performance when the amount of labeled data is limited. See [25] for an excellent survey of recent efforts on semi-supervised learning.

Two of the most popular methods for semi-supervised learning are Co-Training [38] and Semi-Supervised Support Vector Machines (S3VM) [24]. Co-Training assumes the presence of multiple views for each feature and uses the confident samples in one view to update the other. However, in applications such as image classification, one often has just a single feature vector and hence it is difficult to apply Co-Training. S3VM considers the labels of the unlabeled data as additional unknowns and jointly optimizes over the classifier parameters and the unknown labels in the SVM frame-

work [39].

Using the kernel trick, several methods have been proposed in the literature that exploit sparsity of data in the high dimensional feature space. In these methods, a preselected Mercer kernel is used to map the input data onto a features space where dictionaries are trained. It has been shown that such non-linear dictionaries can provide better discrimination than their linear counterparts [40], [41], [42].

Motivated by the success of non-linear dictionary learning methods [40], [41], we propose a novel method to learn kernel discriminative dictionaries for classification in a semi-supervised manner. Fig. 2.1 shows the block diagram of the proposed approach which uses both labeled and unlabeled data. While learning a dictionary, we maintain a probability distribution over class labels for each unlabeled data. The discriminative part of the cost is made proportional to the confidence over the assigned label of the participating training sample. This makes the proposed method robust to label assignment errors.

This chapter makes the following contributions:

1. We propose a discriminative dictionary learning method that utilizes both labeled and unlabeled data.

2. Using the kernel trick, we extend the formulation for learning linear dictionaries with labeled and unlabeled data to the non-linear case. An efficient optimization procedure is proposed for solving this non-linear dictionary learning problem.

3. We show how the proposed method can be extended to ambiguously labeled

data where each training sample has multiple labels and only one of them is correct.

The methods proposed in [43] is different from the one proposed in this chapter. Specifically, in [43] two linear methods are proposed - one based on soft decision rules and the other based on hard decision rules. In contrast to linear reconstructive dictionary leaning methods in [43] and [29], we propose a general discriminative non-linear kernel dictionary learning method for partially labeled data.

The rest of the chapter is organized as follows. In Section 2.2, we formulate the problem of non-linear dictionary learning with partially labeled data. The optimization of the proposed framework is presented in Section 2.3. Experimental results are presented in Section 2.4, and Section 2.5 concludes the chapter with a brief summary and discussion.

2.2 Problem Formulation

In this section, we formulate the optimization problem for learning discriminative dictionaries with partially labeled data. We first present the linear formulation. We then extend it to the non-linear case.

2.2.1 Linear Dictionary Learning with Partially Labeled Data

Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{d \times N}$ be the data matrix where d is the dimension of each data sample $\mathbf{y}_i$ and N is the total number of training samples. We assume that the data is partially labeled and denote the label of the i^{th} sample by l_i. When the

sample $\mathbf{y}_i$ is not labeled, we set l_i to 0, i.e., $l_i \in \{0, 1, \ldots C\}$, where C is the total number of classes.

Our goal is to learn a dictionary $\mathbf{D} \in \mathbb{R}^{d \times K}$, where K is the number of unit norm atoms. We represent this dictionary as the concatenation of all the classes' dictionary, i.e. $\mathbf{D} \triangleq [\mathbf{D}_1 | \ldots | \mathbf{D}_C]$ such that each $\mathbf{D}_c \in \mathbb{R}^{d \times K_c}$ can represent the c^{th} class data well while not economically representing the other class data. Here, K_c is the number of atoms in dictionary $\mathbf{D}_c$, and hence, $K = \sum_{c=1}^{C} K_c$. Enforcing each $\mathbf{D}_c$ to represent only its own class c improves the discriminative capability of the learned dictionary. We represent each sample $\mathbf{y}_i$ by sparse linear combination of dictionary $\mathbf{D}$'s atoms and represent the sparse coefficient of the i^{th} sample by $\mathbf{x}_i$. Furthermore, we denote the coefficient matrix for all the samples by $\mathbf{X}$, i.e., $\mathbf{X} \triangleq [\mathbf{x}_1, \ldots, \mathbf{x}_N]$.

In order to deal with unlabeled data, we introduce a probability matrix $\mathbf{P} \in \mathbb{R}^{C \times N}$ such that each column of $\mathbf{P}$ represents the class distribution of the corresponding data sample. In other words, $(c, i)^{\text{th}}$ element P_{ci} of $\mathbf{P}$ denotes the probability of the i^{th} sample belonging to class c. Hence, by definition,

$$
\begin{aligned}
&P_{ci} = 1 \quad \text{if } \mathrm{y}_i \text{ is labeled with one class and } l_i = c. \\
&P_{ci} = 0 \quad \text{if } \mathrm{y}_i \text{ is labeled with one class and } l_i \neq c. \\
&0 \leq P_{ci} \leq 1 \quad \text{if } \mathbf{y}_i \text{ is unlabeled or ambiguously labeled.}
\end{aligned} \tag{2.1}
$$

We denote the probability of all the samples belonging to class c by a diagonal matrix $\mathbf{P}_c \in \mathbb{R}^{N \times N}$ such that $\mathbf{P}_c(i, i) = P_{ci}$ and the non-diagonal elements of $\mathbf{P}_c$ are

set equal to zeros. Also, we define a matrix $\mathbf{Q}_c \triangleq 1 - \mathbf{P}_c$ to denote the probability of all the samples not belonging to the c^{th} class. Furthermore, we define $\mathbf{P}_c^{sqrt}$ and $\mathbf{Q}_c^{sqrt}$ the square root of $\mathbf{P}_c$ and $\mathbf{Q}_c$, respectively, i.e., $\mathbf{P}_c = \mathbf{P}_c^{sqrt}\mathbf{P}_c^{sqrt}$ and $\mathbf{Q}_c = \mathbf{Q}_c^{sqrt}\mathbf{Q}_c^{sqrt}$. The Frobenius norm and the sparsity promoting ℓ_1 norm of a matrix $\mathbf{A}$ are denoted as $\|\mathbf{A}\|_F$ and $\|\mathbf{A}\|_1$, respectively.

Equipped with these notations, we formulate the dictionary learning problem as one of optimizing

$$\mathcal{J}_0(\mathbf{D}, \mathbf{X}, \mathbf{P}) = \mathcal{F}_0(\mathbf{Y}, \mathbf{D}, \mathbf{X}, \mathbf{P}) + \mathcal{H}(\mathbf{X}, \mathbf{P}) + \lambda_1 \|\mathbf{X}\|_1, \tag{2.2}$$

where,

$$\begin{aligned} \mathcal{F}_0(\mathbf{Y}, \mathbf{D}, \mathbf{X}, \mathbf{P}) = \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \\ + \tau_1 \sum_{c=1}^{C} \|\big(\mathbf{Y} - \mathbf{D}_c\mathbf{X}^c\big)\mathbf{P}_c^{sqrt}\|_F^2 \\ + \tau_2 \sum_{c=1}^{C} \|\mathbf{D}_c\mathbf{X}^c\mathbf{Q}_c^{sqrt}\|_F^2, \end{aligned} \tag{2.3}$$

$$\mathcal{H}(\mathbf{X}, \mathbf{P}) = \lambda_2\big(tr(S_w(\mathbf{X}, \mathbf{P}) - S_b(\mathbf{X}, \mathbf{P}))\big) + \eta\|\mathbf{X}\|_F^2, \tag{2.4}$$

and $\mathbf{X}^c$ is the coefficient matrix corresponding to the cth class. Here, the first term of $\mathcal{F}_0$ encourages $\mathbf{D}$ to be a good representative of the data matrix $\mathbf{Y}$ without needing any label information. The second term of $\mathcal{F}_0$ enforces that the c^{th} class dictionary $\mathbf{D}_c$ represents well those samples which are likely to belong to class c. Note that $\mathbf{P}_c^{sqrt}$ is a diagonal matrix and hence the contribution of each sample in this part of the cost is proportional to the probability of it having come from the c^{th} class. The third part of $\mathcal{F}_0$ enlarges the reconstruction error of those samples which are

less likely to have come from the c^{th} class. The parameters τ_1 and τ_2 control the discriminative capability of the learned dictionary.

The second term $\mathcal{H}$ of $\mathcal{J}_0$ in (2.2) makes the sparse coefficients of samples discriminative by decreasing the trace of within-class scatter matrix

$$S_w = \sum_{c=1}^{C} \sum_{i:l_i=c} (\mathbf{x}_i - \mathbf{m}_c)(\mathbf{x}_i - \mathbf{m}_c)^T$$

and increasing the trace of between-class scatter matrix

$$S_b = \sum_{c=1}^{C} N_c(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^T,$$

where $\mathbf{m}_c$ is the average of the c^{th} class coefficients, $\mathbf{m}$ is the average of all the coefficients and N_c is the number of samples in class c. However, when the label information is available in the form of probability matrix, these scatter matrices can be defined as follows

$$\begin{aligned} S_w(\mathbf{X}, \mathbf{P}) &= \sum_{c=1}^{C} (\mathbf{X} - \mathbf{M}_c)\mathbf{P}_c(\mathbf{X} - \mathbf{M}_c)^T \\ &= \sum_{c=1}^{C} (\mathbf{X} - \mathbf{X}\mathbf{E}_c)\mathbf{P}_c(\mathbf{X} - \mathbf{X}\mathbf{E}_c)^T, \end{aligned} \tag{2.5}$$

where $\mathbf{E}_c \in \mathbb{R}^{N \times N}$ has N repeated column and each of them, denoted by $\mathbf{e}_c$, has the following form,

$$\mathbf{e}_c(i) = \frac{P_{ci}}{w_c}, \quad \text{where } w_c = \sum_{i=1}^{N} P_{ci}, \tag{2.6}$$

and

$$S_b(\mathbf{X}, \mathbf{P}) = \sum_{c=1}^{C} w_c(\mathbf{X}\mathbf{e}_c - \mathbf{X}\mathbf{b})(\mathbf{X}\mathbf{e}_c - \mathbf{X}\mathbf{b})^T, \tag{2.7}$$

where, $\mathbf{b}(i) = \frac{1}{N}, \forall i = 1, \ldots, N$. Note that $\mathbf{X}\mathbf{e}_c$ is the average of the c^{th} class coefficients and $\mathbf{X}\mathbf{b}$ is the average of all the coefficients.

In (2.4), $tr(.)$ denotes the matrix trace operator and an elastic term $\|\mathbf{X}\|_F^2$ is added to make the cost with respect to $\mathbf{X}$ convex and stable. Similar formulations have been used in [17, 32]. The last term of $\mathcal{J}_0$ enforces the sparsity of coefficients. Finally, λ_1, λ_2 and η are the parameters controlling sparsity of coefficients, discriminability of sparse codes and elastic term, respectively.

2.2.2 Non-Linear Dictionary Learning

Let $\mathbf{\Phi} : \mathbb{R}^d \rightarrow G$ be a non-linear mapping from d-dimensional space into a dot product space G. Dictionary learning algorithm can be formulated in the feature space by writing $\mathbf{D} = \mathbf{\Phi}(\mathbf{Y})\mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{N \times K}$ is a matrix with K columns [40], [41]. By changing the columns of $\mathbf{A}$, we can learn the dictionary atoms in the feature space. Hence, the columns of $\mathbf{A}$ are referred to as atoms and denoted by $\mathbf{a}_k$, with $k = 1, \ldots, K$. The k^{th} atom in the feature space can be written as $\mathbf{\Phi}(\mathbf{Y})\mathbf{a}_k$. In order to enforce unit norm constraint on the atoms in the feature space, $\mathbf{a}_k \boldsymbol{\mathcal{K}} \mathbf{a}_k$ should be equal to 1 for all k. Also, we define $\mathbf{A}$ as the concatenation of C matrices, one for each class, i.e., $\mathbf{A} = [\mathbf{A}_1 | \ldots | \mathbf{A}_C]$. Next, we can change $\mathcal{F}_0$ and denote it by $\mathcal{F}$ such that,

$$\begin{aligned} \mathcal{F}(\mathbf{Y}, \mathbf{A}, \mathbf{X}, \mathbf{P}) = & \|\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \\ & + \tau_1 \sum_{c=1}^{C} \|\big(\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{X}^c\big)\mathbf{P}_c^{sqrt}\|_F^2 \\ & + \tau_2 \sum_{c=1}^{C} \|\big(\mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{X}^c\big)\mathbf{Q}_c^{sqrt}\|_F^2. \end{aligned} \tag{2.8}$$

As we will see later, each of the terms in $\mathcal{F}$ containing $\mathbf{\Phi}(\mathbf{Y})$ can be written in terms of the dot products $\mathbf{\Phi}(\mathbf{Y})^T\mathbf{\Phi}(\mathbf{Y})$. This allows us to use the kernel trick by writing $\mathbf{\Phi}(\mathbf{Y})^T\mathbf{\Phi}(\mathbf{Y}) = \boldsymbol{\mathcal{K}}(\mathbf{Y},\mathbf{Y}) \in \mathbb{R}^{N\times N}$, where, $\boldsymbol{\mathcal{K}}$ is the kernel matrix whose $(i,j)^{\text{th}}$ element measures the similarity between $\mathbf{y}_i$ and $\mathbf{y}_j$ by means of a mercer kernel function denoted by $\kappa(\mathbf{y}_i,\mathbf{y}_j) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Some commonly used kernels include polynomial kernels

$$\kappa(\mathbf{y}_i,\mathbf{y}_j) = (\mathbf{y}_i^T\mathbf{y}_j + a)^b$$

and Gaussian kernels

$$\kappa(\mathbf{y}_i^T\mathbf{y}_j) = \exp\Big(\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{c}\Big),$$

where a, b and c are the parameters of the kernel functions. The overall cost for the non-linear dictionary learning can be written as follows

$$\mathcal{J}(\mathbf{A},\mathbf{X},\mathbf{P}) = \mathcal{F}(\mathbf{Y},\mathbf{A},\mathbf{X},\mathbf{P}) + \mathcal{H}(\mathbf{X},\mathbf{P}) + \lambda_1\|\mathbf{X}\|_1. \tag{2.9}$$

Having proposed the formulation for learning non-linear dictionaries with partially labeled data, we describe our approach to optimize the cost in (2.9).

2.3 Optimization of the Proposed Formulation

Our optimization problem is to minimize the cost in (2.9) with respect to dictionary $\mathbf{A}$, sparse coefficient matrix $\mathbf{X}$ and probability matrix $\mathbf{P}$,

$$\hat{\mathbf{A}},\hat{\mathbf{X}},\hat{\mathbf{P}} = \arg\min_{\mathbf{A},\mathbf{X},\mathbf{P}} \mathcal{J}(\mathbf{A},\mathbf{X},\mathbf{P})$$
$$\text{subject to} \qquad \mathbf{a}_k^T\boldsymbol{\mathcal{K}}\mathbf{a}_k = 1, \quad \forall k = 1,\ldots,K. \tag{2.10}$$

Equation (2.10) is jointly non-convex in all the three variable. Hence, we resort to optimizing one variable at a time, while keeping the other two fixed.

2.3.1 Optimization of the Dictionary $\mathbf{A}$

When the coefficient matrix $\mathbf{X}$ and the probability matrix $\mathbf{P}$ are fixed, we optimize $\mathbf{A}$ one class at a time. To optimize the c^{th} class dictionary, we write the cost $\mathcal{J}$ with respect to $\mathbf{A}_c$ as

$$\begin{aligned}\mathcal{J}_{\mathbf{A}_c} &= \|\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{X}^c - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_o\mathbf{X}^o\|_F^2 \\ &+ \tau_1\|\big(\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{X}^c\big)\mathbf{P}_c^{sqrt}\|_F^2 \\ &+ \tau_2\|\big(\mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{X}^c\big)\mathbf{Q}_c^{sqrt}\|_F^2, \end{aligned} \tag{2.11}$$

where, $\mathbf{Y}_o$ and $\mathbf{A}_o$ denote the other class (i.e. not c) data matrix and dictionary, respectively. $\mathbf{X}^o$ denotes the coefficient matrix corresponding to $\mathbf{A}_o$. These matrices are defined as,

$$\mathbf{Y}_o \triangleq [\mathbf{Y}_1, \ldots, \mathbf{Y}_{c-1}, \mathbf{Y}_{c+1}, \ldots, \mathbf{Y}_C], \tag{2.12}$$

$$\mathbf{A}_o \triangleq [\mathbf{A}_1, \ldots, \mathbf{A}_{c-1}, \mathbf{A}_{c+1}, \ldots, \mathbf{A}_C], \tag{2.13}$$

$$\mathbf{X}^o \triangleq [\mathbf{X}^{1^T}, \ldots, \mathbf{X}^{c-1^T}, \mathbf{X}^{c+1^T}, \ldots, \mathbf{X}^{C^T}]^T, \tag{2.14}$$

where $\mathbf{Y}_c \in \mathbb{R}^{d \times N_c}$ is part of the data matrix consisting of samples from the c^{th} class. To update $\mathbf{A}$, we solve the following optimization problem for all $c = 1, \ldots, C$,

$$\hat{\mathbf{A}}_c = \arg\min_{\mathbf{A}_c} \mathcal{J}_{\mathbf{A}_c} \tag{2.15}$$

$$\text{subject to} \quad \mathbf{a}_k^T \boldsymbol{\mathcal{K}} \mathbf{a}_k = 1, \quad \forall k = 1, \ldots, K_c, \tag{2.16}$$

where $\mathbf{a}_k$ is the k^{th} columns of $\mathbf{A}_c$.

Next, we optimize one atom at a time while keeping the others fixed. The cost with respect to $\mathbf{a}_k$ can be written as

$$\begin{aligned}\mathcal{J}_{\mathbf{a}_k} = &\|\mathbf{Z}_c^{\mathbf{\Phi}} - \mathbf{\Phi}(\mathbf{Y})\mathbf{a}_k\mathbf{x}^k\|_F^2 + \tau_1\|(\mathbf{U}_c^{\mathbf{\Phi}} - \mathbf{\Phi}(\mathbf{Y})\mathbf{a}_k\mathbf{x}^k)\mathbf{P}_c^{sqrt}\|_F^2 \\ &+ \tau_2\|\mathbf{\Phi}(\mathbf{Y})(\mathbf{a}_k\mathbf{x}^k + \mathbf{W}_c)\mathbf{Q}_c^{sqrt}\|_F^2,\end{aligned} \tag{2.17}$$

where,

$$\begin{aligned}\mathbf{W}_c &:= \sum_{j\neq k}\mathbf{A}_c(:,j)\mathbf{X}^c(j,:), \\ \mathbf{Z}_c^{\mathbf{\Phi}} &:= \mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_o\mathbf{X}^o - \mathbf{\Phi}(\mathbf{Y})\mathbf{W}_c, \quad \text{and} \\ \mathbf{U}_c^{\mathbf{\Phi}} &:= \mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{W}_c.\end{aligned} \tag{2.18}$$

Writing $\mathcal{J}_{\mathbf{a}_k}$ in kernel form (and ignoring the terms independent of $\mathbf{a}_k$), we get

$$\begin{aligned}\mathcal{J}_{\mathbf{a}_k} = &\operatorname{tr}[\mathbf{x}^k\mathbf{x}^{kT}\mathbf{a}_k^T\boldsymbol{\mathcal{K}}\mathbf{a}_k - 2\mathbf{a}_k^T\big(\boldsymbol{\mathcal{K}} - \boldsymbol{\mathcal{K}}\mathbf{A}_o\mathbf{X}^o - \boldsymbol{\mathcal{K}}\mathbf{W}_c\big)\mathbf{x}^{kT}] \\ &+ \tau_1\operatorname{tr}[\mathbf{a}_k^T\boldsymbol{\mathcal{K}}\mathbf{a}_k\mathbf{x}^k\mathbf{P}_c\mathbf{x}^{kT} - 2\mathbf{a}_k^T\big(\boldsymbol{\mathcal{K}} - \boldsymbol{\mathcal{K}}\mathbf{W}_c\big)\mathbf{P}_c\mathbf{x}^{\mathbf{k}T}] \\ &+ \tau_2\operatorname{tr}[\mathbf{a}_k^T\boldsymbol{\mathcal{K}}\mathbf{a}_k\mathbf{x}^{\mathbf{k}}\mathbf{Q}_c\mathbf{x}^{\mathbf{k}T} + 2\mathbf{a}_k^T\boldsymbol{\mathcal{K}}\mathbf{W}_c\mathbf{Q}_c\mathbf{x}^{kT}].\end{aligned} \tag{2.19}$$

To optimize, $\mathcal{J}_{\mathbf{a}_k}$, subject to $\mathbf{a}_k\boldsymbol{\mathcal{K}}\mathbf{a}_k = 1$, we write the Lagrange function as

$$\mathcal{L}(\mathbf{a}_k, \gamma) = \mathcal{J}_{\mathbf{a}_k} + \gamma(\mathbf{a}_k\boldsymbol{\mathcal{K}}\mathbf{a}_k - 1), \tag{2.20}$$

where, γ is a Lagrange multiplier. Next, we take the derivative of $\mathcal{L}(.)$ with respect

to $\mathbf{a}_k$ and set it equal to zero

$$\begin{aligned}\alpha.\boldsymbol{\mathcal{K}}\mathbf{a}_k =&[\boldsymbol{\mathcal{K}} - \boldsymbol{\mathcal{K}}\mathbf{A}_o\mathbf{X}^o - \boldsymbol{\mathcal{K}}\mathbf{W}_c]\mathbf{x}^{k^T} \\ &+ \tau_1[\boldsymbol{\mathcal{K}} - \boldsymbol{\mathcal{K}}\mathbf{W}_c]\mathbf{P}_c\mathbf{x}^{\mathbf{k}^T} - \tau_2\boldsymbol{\mathcal{K}}\mathbf{W}_c\mathbf{Q}_c\mathbf{x}^{k^T},\end{aligned} \tag{2.21}$$

where α is a scalar constant. Denoting the right hand side of the above equation by $\boldsymbol{\mathcal{K}}\mathbf{v}$, we get $\alpha.\mathbf{a}_k = \mathbf{v}$, where,

$$\mathbf{v} \triangleq [\mathbf{I} - \mathbf{A}_o\mathbf{X}^o - \mathbf{W}_c]\mathbf{x}^{k^T} + \tau_1[\mathbf{I} - \mathbf{W}_c]\mathbf{P}_c\mathbf{x}^{k^T} - \tau_2\mathbf{W}_c\mathbf{Q}_c\mathbf{x}^{k^T},$$

and along with the constraint $\mathbf{a}_k^T\boldsymbol{\mathcal{K}}\mathbf{a}_k = 1$, we choose the dual variable γ, and hence α, such that the condition is satisfied. In other words,

$$\mathbf{a}_k = \frac{\mathbf{v}}{\|\mathbf{v}\|_2}. \tag{2.22}$$

2.3.2 Optimization of the Coefficient Matrix $\mathbf{X}$

With the fixed dictionary $\mathbf{A}$, and the probability matrix $\mathbf{P}$, the cost in (2.9) can be re-written with respect to $\mathbf{X}$ as,

$$\begin{aligned}\mathcal{J}_{\mathbf{X}} =&\mathcal{F}_1(\mathbf{X}) + \tau_1\mathcal{F}_2(\mathbf{X}) + \tau_2\mathcal{F}_3(\mathbf{X}) + \\ &\lambda_2\mathcal{H}_1(\mathbf{X}) + \lambda_2\mathcal{H}_2(\mathbf{X}) + \eta\|\mathbf{X}\|_F^2 + \lambda_1\|\mathbf{X}\|_1,\end{aligned} \tag{2.23}$$

where,

$$\mathcal{F}_1 = \|\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2,$$

$$\mathcal{F}_2 = \sum_{c=1}^{C} \|\big(\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{X}^c\big)\mathbf{P}_c^{sqrt}\|_F^2,$$

$$\mathcal{F}_3 = \sum_{c=1}^{C} \|\big(\mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{X}^c\big)\mathbf{Q}_c^{sqrt}\|_F^2,$$

$$\mathcal{H}_1 = tr[\sum_{c=1}^{C}(\mathbf{X} - \mathbf{X}\mathbf{E}_c)\mathbf{P}_c(\mathbf{X} - \mathbf{X}\mathbf{E}_c)^T], \quad \text{and}$$

$$\mathcal{H}_2 = -tr[\sum_{c=1}^{C} w_c(\mathbf{X}\mathbf{e}_c - \mathbf{X}\mathbf{b})(\mathbf{X}\mathbf{e}_c - \mathbf{X}\mathbf{b})^T].$$

The problem of updating $\mathbf{X}$ can be written as

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \mathcal{J}_{\mathbf{X}}. \tag{2.24}$$

In order to minimize $\mathcal{J}_{\mathbf{X}}$ with respect to $\mathbf{X}$, we use the Iterative Projection Method (IPM) that minimizes a cost consisting of a convex term with an additional ℓ_1 regularizer [32, 44]. IPM is an iterative algorithm that computes the derivative of all the terms except the ℓ_1 part of the cost and takes a gradient descent step at each iteration. Followed by this gradient descent at each iteration, the values of $\mathbf{X}$ are soft thresholded [44]. The required derivative of all the terms in (2.23) can be

computed as follows

$$\frac{\partial \mathcal{F}_1}{\partial \mathbf{X}} = 2\mathbf{A}^T\boldsymbol{\mathcal{K}}\mathbf{A}\mathbf{X} - 2\mathbf{A}^T\boldsymbol{\mathcal{K}}, \tag{2.25}$$

$$\frac{\partial \mathcal{F}_2}{\partial \mathbf{X}^c} = \frac{\partial}{\partial \mathbf{X}^c} tr[\mathbf{A}_c^T\boldsymbol{\mathcal{K}}\mathbf{A}_c\mathbf{X}^c\mathbf{P}_c\mathbf{X}^{\mathbf{c}T} - \mathbf{A}_c^T\boldsymbol{\mathcal{K}}\mathbf{P}_c\mathbf{X}^{cT}] \tag{2.26}$$

$$= 2\mathbf{A}_c^T\boldsymbol{\mathcal{K}}\mathbf{A}_c\mathbf{X}^c\mathbf{P}_c - 2\mathbf{A}_c^T\boldsymbol{\mathcal{K}}\mathbf{P}_c, \tag{2.27}$$

$$\frac{\partial \mathcal{F}_3}{\partial \mathbf{X}^c} = 2\mathbf{A}_c^T\boldsymbol{\mathcal{K}}\mathbf{A}_c\mathbf{X}^c\mathbf{Q}_c. \tag{2.28}$$

Note that $\mathcal{H}_1(\mathbf{X}) = \sum_{c=1}^{C} tr[\mathbf{X}^T\mathbf{S}_c\mathbf{X}]$, where $\mathbf{S}_c := (\mathbf{I} - \mathbf{E}_c)\mathbf{P}_c(\mathbf{I} - \mathbf{E}_c)^T$. Hence,

$$\frac{\partial \mathcal{H}_1}{\partial \mathbf{X}} = \sum_{c=1}^{C} 2\mathbf{X}\mathbf{S}_c. \tag{2.29}$$

Similarly,

$$\frac{\partial \mathcal{H}_2}{\partial \mathbf{X}} = -\frac{\partial}{\partial \mathbf{X}} \sum_{c=1}^{C} tr[\mathbf{X}\mathbf{T}_c\mathbf{X}^T] \tag{2.30}$$

$$= -\sum_{c=1}^{C} 2\mathbf{X}\mathbf{T}_c, \tag{2.31}$$

where, $\mathbf{T}_c := w_c(\mathbf{e}_c - \mathbf{b})(\mathbf{e}_c - \mathbf{b})^T$.

2.3.3 Optimization of the Probability Matrix $\mathbf{P}$

With the fixed dictionary $\mathbf{A}$, and the coefficient matrix $\mathbf{X}$, the cost in (2.9) can be re-written with respect to $\mathbf{P}$ as,

$$\begin{aligned}\mathcal{J}_{\mathbf{P}} =&\tau_1 \sum_{c=1}^{C}\sum_{i=1}^{N} P_{ci}\|\mathbf{\Phi}(\mathbf{y}_i) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{x}_i^c\|_2^2 + \tau_2 \sum_{c=1}^{C}\sum_{i=1}^{N}(1-P_{ci})\|\mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{x}_i^c\|_2^2 \\ &+ \lambda_2 \sum_{c=1}^{C}\sum_{i=1}^{N} P_{ci}\|\mathbf{x}_i - \mathbf{m}_c\|_2^2 - \lambda_2 \sum_{c=1}^{C} N_c\|\mathbf{m}_c - \mathbf{m}\|_2^2. \end{aligned} \tag{2.32}$$

We can solve the above problem by optimizing for the class probabilities for the i^{th} sample $\mathbf{p}_i$ independently, where $\mathbf{p}_i = [P_{1i}, \ldots, P_{Ci}]^T$, provided that $\mathbf{m}_c$ does not change much with each update. Hence, the cost with respect to $\mathbf{p}_i$ is given by

$$\mathcal{J}_{\mathbf{p}_i} = \mathbf{p}_i^T\mathbf{v}_i, \tag{2.33}$$

where the cth element of $\mathbf{v}_i$ is given by,

$$\begin{aligned}\mathbf{v}_i(c) =&\tau_1\|\mathbf{\Phi}(\mathbf{y}_i) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{x}_i^c\|_2^2 \\ &- \tau_2\|\mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{x}_i^c\|_2^2 + \lambda_2\|\mathbf{x}_i - \mathbf{m}_c\|_2^2. \end{aligned} \tag{2.34}$$

The goal is to, minimize $\mathcal{J}_{\mathbf{p}_i}$ subject to $\mathbf{p}_i^T\mathbf{1} = 1, \mathbf{p}_i \geq 0$. To minimize a linear cost subject to linear constraints is a linear programming (LP) optimization problem whose solution is on one of the vertices. In other words, the element of $\mathbf{p}_i$ corresponding to minimum value in $\mathbf{v}_i$ would be 1 and other elements would be zeros. This is to say that each sample will be assigned to a fixed class rather than a class distribution. Hence, instead of solving this LP, we compute the probability of each sample based on the reconstruction error e_{ci} of the i^{th} sample on the c^{th} class

dictionary, defined as

$$\begin{aligned} e_{ci} &= \|\mathbf{\Phi}(\mathbf{y}_i) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{x}_i^c\|_2^2 \\ &= \boldsymbol{\mathcal{K}}(\mathbf{y}_i, \mathbf{y}_i) + (\mathbf{x}_i^c)^T\mathbf{A}\boldsymbol{\mathcal{K}}\mathbf{A}\mathbf{x}_i^c - \boldsymbol{\mathcal{K}}(\mathbf{y}_i, \mathbf{Y})\mathbf{A}_c\mathbf{x}_i^c, \end{aligned} \tag{2.35}$$

where $\mathbf{x}_i^c$ is the sparse coefficient of the i^{th} sample corresponding to dictionary $\mathbf{A}_c$. Now, the probability of the i^{th} sample belonging to the c^{th} class can be defined as

$$P_{ci} = \begin{cases} \frac{\exp\{-\frac{e_{ci}}{\sigma}\}}{\sum_{c=1}^{C}\exp\{-\frac{e_{ci}}{\sigma}\}} & \text{if} \quad \frac{\exp\{-\frac{e_{ci}}{\sigma}\}}{\sum_{c=1}^{C}\exp\{-\frac{e_{ci}}{\sigma}\}} > \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{2.36}$$

Here, σ is a parameter that controls how sharp the probability distributions are. Furthermore, we want to add only those samples which are quite confident about its class and remove the ones that have similar probability of having come from multiple classes. This is achieved by setting the probability of those samples to zero which are less than a certain parameter θ. Furthermore, instead of updating $\mathbf{P}$ at each iteration, we skip a few iteration(s) (typically $1-5$) before updating the probability matrix. This gives some time for the learned dictionary to converge before adding more samples. The proposed method for learning dictionary is summarized in Algorithm 1.

2.3.4 Dictionary Learning with Ambiguously Labeled Data

In many practical situations there might be multiple labels available for each training sample. For example, given a picture with multiple faces and a caption specifying who are in the picture, the reader may not know which face goes with

Algorithm 1: Algorithm for learning non-linear dictionary $\mathbf{A}$ by solving (2.9).

Input: Training Data $\mathbf{Y}$, Partial Labels $l_i, \forall i = 1, \ldots N$, Kernel Function κ.

Output: Dictionary $\mathbf{A}$.

Initialize Dictionary $\mathbf{A}$, sparse Coefficient matrix $\mathbf{X}$ and Probability matrix $\mathbf{P}$.

$itr = 0$

repeat

  $itr = itr + 1$

  Update sparse coefficient matrix $\mathbf{X}$ by solving (2.24).

  if *mod(itr, skipItr)*=0 **then**

    Update Probability matrix $\mathbf{P}$ using (2.36)

  end

  for $c = 1, \ldots, C$ **do**

    for $k = 1, \ldots, K_c$ **do**

      Update atom $\mathbf{a}_k$ using (2.22).

    end

  end

until *convergence or maximum iterations*;

return $\mathbf{A}$.

the names in the caption. The problem of learning identities where each example is associated with multiple labels, when only one of which is correct is often known as ambiguously labeled learning [45].

This ambiguously labeled data can be easily handled using the proposed formulation by giving equal probabilities to each of the given class for that sample. For example, if a sample $\mathbf{y}_i$ has labels $1, 4, 5, 7$, we can set $\mathbf{P}(c, i) = 0.25$, for $c = 1, 4, 5, 7$. However, a major challenge in handling such ambiguously labeled data is to learn an initial dictionary [43]. For the cases where data is either unambiguously labeled or completely unlabeled, we can use the unambiguously labeled data to learn an initial dictionary for each class. However, when each sample has multiple labels, we first need to cluster the data into different classes to make sure that the learned dictionary for each class is not influenced by the samples of the other classes.

Let $\mathbf{y}_i$ have multiple labels denoted by the set L_i and the number of ambiguous labels be denoted by $C_i \triangleq |L_i|$. In order to assign one cluster label to $\mathbf{y}_i$, we learn C_i dictionaries, one for each ambiguous class label, using all the samples excluding $\mathbf{y}_i$. While learning the c^{th} class dictionary $\mathbf{D}_{ci}$, where $c \in L_i$, for the i^{th} sample, we use all the samples excluding $\mathbf{y}_i$ and with at least one class label as c. Let the set of these samples be denoted by $\mathbf{Y}_{ci}$. We learn a dictionary $\mathbf{D}_{ci}$ with the data matrix $\mathbf{Y}_{ci}$ using the KSVD algorithm [10] for each $c \in L_i$. The reconstruction error of $\mathbf{y}_i$ is computed on $\mathbf{D}_{ci}$ as follows,

$$r_{ci} = \|\mathbf{y}_i - \mathbf{D}_{ci}\mathbf{x}\|_2, \tag{2.37}$$

where, $\mathbf{x} = (\mathbf{D}_{ci}^T\mathbf{D}_{ci})^{-1}\mathbf{D}_{ci}^T\mathbf{y}_i$. Next, $\mathbf{y}_i$ is assigned to the cluster c with the minimum

reconstruction error r_{ci}. These steps are summarized in Algorithm 2.

Algorithm 2: Algorithm for clustering ambiguously labeled data into C clusters.

Input: Training Data $\mathbf{Y}$, Partial Labels L_i, $\forall i = 1, \ldots N$.

Output: Cluster labels $h_i \in \{1, \ldots, C\}$ for each sample $\mathbf{y}_i$, for all

$$i = 1, \ldots, N.$$

for *i = 1, . . . , N* **do**

  for $j = 1, \ldots, C_i$ **do**

    $c = L_i(j)$

    Collect all the samples except $\mathbf{y}_i$ with at least one class label as c into data matrix $\mathbf{Y}_{ci}$.

    Learn dictionary $\mathbf{D}_{ci}$ with $\mathbf{Y}_{ci}$ using KSVD algorithm.

    $\mathbf{x} = (\mathbf{D}_{ci}^T \mathbf{D}_{ci})^{-1} \mathbf{D}_{ci}^T \mathbf{y}_i$.

    $r_{ci} = \|\mathbf{y}_i - \mathbf{D}_{ci}\mathbf{x}\|_2$.

  end

  Cluster label $h_i = \arg\min_{c \in L_i} r_{ci}$

end

return $h_i, \forall i = 1, \ldots, N$.

For each class, an initial dictionary $\mathbf{D}_c^{(0)}$ is learned with samples in the c^{th} cluster using the KSVD algorithm. Finally, initial non-linear dictionary $\mathbf{A}_c^{(0)}$ is computed using $\mathbf{D}_c^{(0)}$ as

$$\mathbf{A}_c^{(0)} = \text{pinv}(\mathbf{Y})\mathbf{D}_c^{(0)}, \tag{2.38}$$

where $\text{pinv}(\mathbf{Y})$ is the pseudo-inverse of the data matrix $\mathbf{Y}$.

2.3.5 Classification

Having learned the non-linear dictionary $\mathbf{A}$, we classify a given test sample $\mathbf{y}_t$ by first computing its sparse code $\mathbf{x}_t$ by solving the following optimization problem,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 \tag{2.39}$$

$$= \arg\min_{\mathbf{x}} \Big(\kappa(\mathbf{y}_t, \mathbf{y}_t) + \mathbf{x}^T\mathbf{A}^T\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})\mathbf{A}\mathbf{x} - 2\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{Y})\mathbf{A}\mathbf{x} + \lambda\|\mathbf{x}\|_1\Big). \tag{2.40}$$

The above problem in (2.40) is solved using the IPM. Next, to determine the class of the test sample, we compute the reconstruction error for each class as

$$r_c = \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}_c\mathbf{x}^c\|_2^2 \tag{2.41}$$

$$= \kappa(\mathbf{y}_t, \mathbf{y}_t) + (\mathbf{x}^c)^T\mathbf{A}_c^T\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})\mathbf{A}_c\mathbf{x}^c - 2\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{Y})\mathbf{A}_c\mathbf{x}^c. \tag{2.42}$$

Finally, the test sample is assigned the class corresponding to the minimum reconstruction error as

$$\text{class of } \mathbf{y}_t = \arg\min_c r_c. \tag{2.43}$$

2.4 Experimental Results

To illustrate the effectiveness of our method, we present experimental results on some of the publicly available databases such as the USPS digit dataset [46], the Kimia's object dataset [47] and TV LOST dataset [48, 49] that consists of cropped

face images from TV series 'LOST'. A comparison with other existing object recognition methods in [32] suggests that the discriminative dictionary learning algorithm known as Fisher Discriminant Dictionary Learning (FDDL) is among the best dictionary-based method for classification. Hence, we use FDDL and a semi-supervised dictionary learning algorithm S2D2 [50] to compare the performance on semi-supervised experiments. We also compare our method with that of Support Vector Machines (SVM) as well as a semi-supervised extension of SVM known as (S3VM) [24]. Also, we compare our method with recently proposed Pseudo Multi-view Automatic Feature Decomposition for Co-training (PMC) method [51]. In all of our experiments, λ is set equal to 0.05 and η is set equal to 0.001. The number of iterations are set to a maximum value of 30. All the other parameters are set using cross-validation separately for each experiment. For big training datasets, they can be optimized on a small validation dataset to reduce training time. In our experiments, we optimized the sparsity parameter over the set $\{0.01, 0.05, 0.1, 0.5\}$. The discriminative parameters τ_1 and τ_2 were optimized over the set $\{0.1, 1, 5, 10\}$. We skipped a few iterations when updating $\mathbf{P}$ to ensure the convergence of the cost function. This allows dictionary atoms to converge before using them to compute the probability matrix. Furthermore, the parameter σ controls the sharpness of probability distribution. Although, this can be computed in each iteration as the average reconstruction error as was done in [43], we set this equal to 1 for simplicity. If the probability distributions appear very flat, we reduce it to a smaller value.

2.4.1 Digit Recognition

The USPS digit dataset [46] consists of gray images of hand written digits from 0 to 9. This dataset contains 7291 training samples and 2007 test samples. From the training data, four samples from each class are randomly chosen as the labeled samples and the rest of the training data is used as the unlabeled data. The original images are of size 16×16 which forms the feature vector of dimension 256. We added a maximum of 10 unlabeled samples per class at each iterations. For this experiment we used polynomial kernel of degree 4, and set sparsity parameter $\lambda_1 = 0.01$. Furthermore, to avoid low confidence samples we set $\theta = 0.5$.

We compare the recognition accuracies of the proposed method with other methods in Table 2.1. The parameters τ_1 and τ_2 were set equal to 10 and 0.1, respectively, for this dataset. Observe that the proposed method outperforms the other methods by more than 5%. The major difference between S2D2 and the proposed method is the use of non-linear kernel. This confirms the importance of non-linear kernels in dictionary learning methods. The improvement in performance compared to SVM and FDDL is due to the fact that we utilize the unlabeled data for updating dictionaries in the training stage. Being supervised techniques, the performance of SVM and FDDL reduces when the available labeled samples are small. Unlike S3VM which assigns hard labels to the unlabeled data points at each iteration, the proposed method assigns only a soft probability of class for each unlabeled data.The reason why the proposed method performs better than S3VM is because the soft assignment approach is more robust to labeling errors when

Algorithms	Accuracy(%)
SVM	74.47
S3VM [24]	75.61
FDDL [32]	79.24
PMC [51]	79.78
S2D2 [50]	85.61
Proposed Method	**90.60**

Table 2.1: Recognition accuracy for the proposed method on USPS Digit dataset.

compared to the hard assignment.

Pre-Images of the learned dictionary atoms: Recall that the k^{th} atom of the learned non-linear dictionary is represented as $\mathbf{\Phi}(\mathbf{Y})\mathbf{a}_k$ with respect to the base $\mathbf{\Phi}(\mathbf{Y})$ in the feature space G. Since G is large, and possibly of infinite dimension, we visualize the pre-image [52] of dictionary atoms. The pre-image of a dictionary atom $\mathbf{\Phi}(\mathbf{Y})\mathbf{a}_k$ is obtained by seeking a vector $\mathbf{d}_k$ in input space $\mathbb{R}^d$ that minimizes the cost function $\|\mathbf{\Phi}(\mathbf{d}_k) - \mathbf{\Phi}(\mathbf{Y})\mathbf{a}_k\|_2$. Due to various noise effects and the generally non-invertible mapping $\mathbf{\Phi}$, pre-image does not always exist. However, an approximated pre-image can be reconstructed without venturing into feature space using techniques described in [52]. In Fig. 2.2, we show the pre-images of some of the learned dictionary atoms from each class.

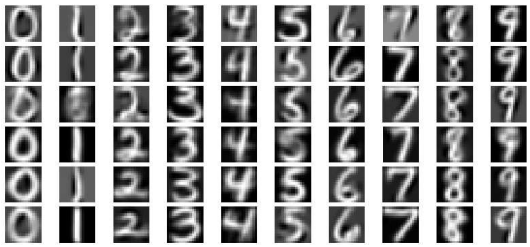
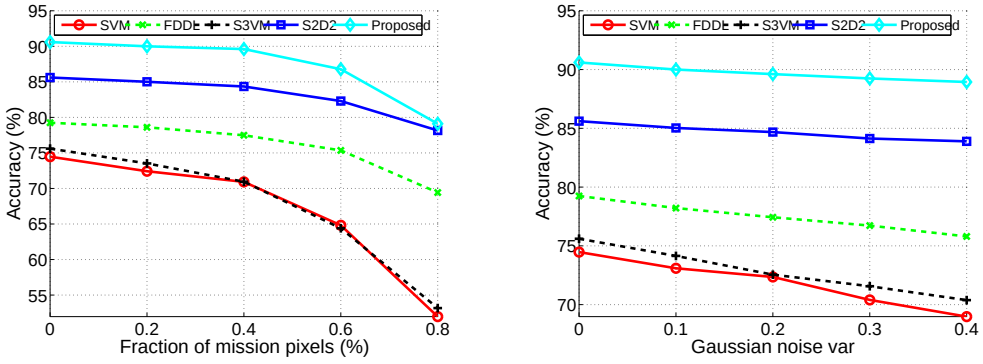


Figure 2.2: Pre-images of the learned atoms of USPS digits. Columns show the learned dictionary atoms for each class.

Performance in the presence of missing and noisy pixels: To further evaluate

the robustness of the proposed method, we computed the recognition performance of the proposed method when pixels in the image are either missing or corrupted by noise. In the missing data experiment, we set pixels at random locations to zero for test images in the digit recognition application. The number of corrupted pixels was varied and we plot the corresponding accuracy in Fig. 2.3(a). Note that the recognition accuracy falls as expected when the amount of missing pixels is increased. But the fall in accuracy is much lower for the proposed technique when compared to the other methods. This clearly demonstrates the improved robustness of the proposed method compared to the competing methods. Similarly to study the robustness of our method in the presence of noise, we added independent and identically distributed Gaussian noise to the pixels. We varied the variance of the added noise and compute the recognition accuracy for all the methods. The results are shown in Fig. 2.3(b). We observed a similar improvement in robustness of the proposed technique.



(a) (b)

Figure 2.3: Accuracy for two kinds of corruption for digit recognition. (a) accuracy vs missing data. (b) Accuracy vs noise variance.

Algorithms	Accuracy(%)
SVM	84.26
S3VM [24]	84.26
FDDL [32]	86.11
PMC [51]	88.89
S2D2 [50]	87.96
Proposed Method	**92.59**

Table 2.2: Recognition accuracy for the proposed method, compared to competing ones for shape recognition.

2.4.2 Object Recognition

In the next set of experiments, we use Kimia's object dataset [47] which has 18 object categories each with 12 binary shapes. We randomly chose six images per class for training and the remaining six for testing. Furthermore, we randomly picked four images per class as the labeled data and the remaining two as the unlabeled data. Each image was resized to 16×16 and intensity values were used as features. The classification rates for all the algorithms are compared in Table 2.2. We see that the proposed method performs better than the other methods. In this experiment we used polynomial kernel of degree 2. We set sparsity parameter $\lambda_1 = 0.5$, $\tau_1 = 0.1$ and $\tau_2 = 1$. Furthermore, to avoid low confidence samples we set $\theta = 0.5$. These results clearly demonstrate that the performance of discriminative dictionary learning methods can be improved significantly by using unlabeled data, when the available labeled data is limited. Furthermore, the use of non-linear kernel can improve the performance of dictionary learning methods for classification.

Caltech101 object recognition: The Caltech101 dataset contains 102 object categories and each category has about 40 to 80 images downloaded from Internet. We

Algorithms	Accuracy(%)
SVM	60.8
FDDL [32]	61.1
PMC [51]	58.4
S3VM [24]	65.6
Proposed Method	**66.4**

Table 2.3: Recognition accuracy for the proposed method on Caltech101 dataset.

randomly selected 10 labeled and 10 unlabeled training images from each category to evaluate the proposed algorithm. To evaluate our method on this dataset, we used spatial pyramid features [30]. For each image, dense SIFT descriptors were extracted from 16×16 patches, separated by 6 pixels. To train the codebook for spatial pyramid, standard k-means clustering with k = 1024 was used. Finally, the dimension of spatial pyramid features were reduced to 3000 dimensions by PCA. The results of our comparison are provided in Table 2.3. As can be seen from this table, the proposed method compares favorably even on the large dataset.

2.4.3 Ambiguously Labeled Data

In order to test our algorithm on ambiguously labeled data we chose the TV LOST dataset as used by [43]. This dataset consists of face images from TV series 'LOST'. In original dataset, there are 1122 registered face images corresponding to a total of 14 subjects, each containing from 18 to 204 images. In our experiment, we followed the same setting as [43] and chose 12 subjects with at least 25 face images per subject. For each subject, first 25 images were selected to evaluate our method. Each image was resized to 30×30 pixels, and histogram-equalized intensities were used as features. This experiment was conducted under transductive setting, mean-

ing all the data was available at training time. We ambiguously labeled 85% of the data and remaining 15% of the data was correctly labeled. For each ambiguously labeled sample, we assigned one correct label and 3 randomly chosen incorrect class labels. We compare our method with the Convex Learning from Partial Labels (CLPL) presented in [49], and various dictionary learning-based methods proposed in [43]. DLHD [43] clusters training data into various clusters based on the reconstruction error, and then learn dictionary for each cluster. DLSD [43] assigns a soft label to each sample based on the the reconstruction error and learns a dictionary for each class based on the assigned soft labels. Equally-weighted K-SVD [43] learns a dictionary using K-SVD for each class by giving equal weight to each ambiguous class. We compare our method with the other methods in Table 2.4. We use a polynomial kernel of degree 4 and set sparsity parameter $\lambda_1 = 0.05$. Furthermore, discriminative parameters τ_1, and τ_2 are set equal to 1 and 0.1, respectively. In order to visualize the dictionary atoms, we plot pre-images of the dictionary atoms for each class in Figure 2.4. As we can see the learned dictionary atoms capture the variations present in each class. Furthermore, we analyze the convergence of our algorithm. In Figure 2.5, we display the probability matrices at the start, end and intermediate iterations. We can clearly visualize how the label accuracy improves over iterations. We also plot the total cost over iterations in Figure 2.6. As can be seen from this figure, our cost decreases with increase in iterations.

Algorithms	Accuracy(%)
CLPL [49]	78.53
Equally-Weighted K-SVD [43]	81.67
DLHD [43]	86.17
DLSD [43]	86.63
Proposed Method	**88.33**

Table 2.4: Recognition accuracy for the proposed method, compared to competing ones for TV LOST dataset.

Figure 2.4: Pre-images of dictionary atoms for TV LOST dataset.

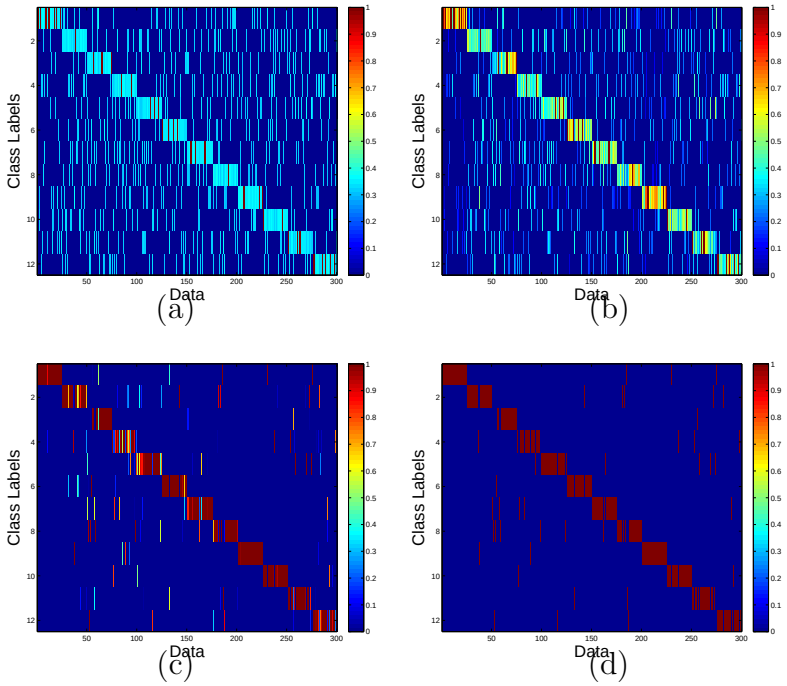


Figure 2.5: Convergence of probability matrices for TV LOST dataset. Figures (a), (b), (c), (d) show the probability matrix $\mathbf{P}$ at intermediate iterations.

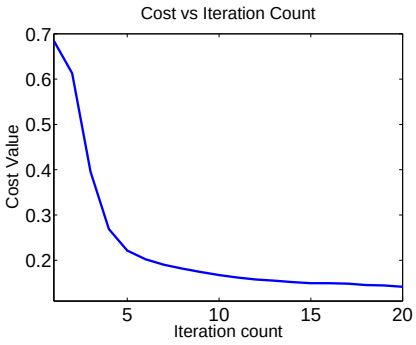


Figure 2.6: Convergence of cost over iterations for TV LOST dataset

2.5 Conclusion

We proposed a method that utilizes unlabeled and ambiguously labeled training data for learning non-linear discriminative dictionaries. The proposed method iteratively estimates the confidence of unlabeled samples belonging to each of the classes and uses it to refine the learned dictionaries. Experiments using various publicly available datasets demonstrate the improved accuracy and robustness to noise and missing information of the proposed method compared to state-of-the-art dictionary learning techniques.

Chapter 3: Generalized Dictionaries for Multiple Instance Learning

3.1 Introduction

Machine learning has played a significant role in developing robust computer vision algorithms for object detection and classification. Most of these algorithms are supervised learning methods, which assume the availability of labeled training data. Label information often includes the type and location of the object in the image, which are typically provided by a human annotator. The human annotation is expensive and time consuming for large datasets. Furthermore, multiple human annotators can often provide inconsistent labels which could affect the performance of the subsequent learning algorithm [53]. However, it is relatively easy to obtain weak labeling information either from search queries on Internet or from amateur annotators providing the category but not the location of the object in the image. This necessitates the development of learning algorithms from weakly labeled data.

A popular approach to incorporate partial label information during training is through Multiple Instance Learning (MIL). Unlike supervised learning algorithms, MIL framework does not require label information for each training instance, but just for collection of instances called bags. A bag is positive if at least one of its instances is a positive example otherwise the bag is negative. One of the first al-

gorithms for MIL, named the Axis-Parallel Rectangle (APR), was proposed by [21] which attempts to find an APR by manipulating a hyper rectangle in the instance feature space to maximize the number of instances from different positive bags enclosed by the rectangle while minimizing the number of instances from the negative bags within the rectangle. The basic idea of APR led to several interesting MIL algorithms. A general framework, called Diverse Density (DD), was proposed by [22] which measures the co-occurrence of similar instances from different positive bags. The idea is to learn the desired concept by maximizing the DD function. An approach based on Expectation - Maximization and DD, called EM-DD, for MIL was proposed by [54]. EM-DD was later extended by [55], called DD-SVM, that essentially trains an SVM in a feature space constructed from a mapping defined by the maximizers and minimizers of the DD function. More recently, an MIL algorithm for randomized trees, named MIForest, was proposed by [56]. An interesting approach, called Multiple Instance Learning via Embedded instance Selection (MILES), was proposed by [57]. This method converts the MIL problem to a standard supervised learning problem that does not impose the assumption relating instance labels to the bag labels.

In this chapter, we develop a general DD-based dictionary learning framework for MIL where labels are available only for the bags, and not for the individual samples. In recent years, sparse coding and dictionary learning-based methods have gained a lot of traction in computer vision and image understanding fields [2], [1], [19], [58], [59]. Dictionary-based algorithms have produced state-of-the-art results in many practical problems such as object recognition, object detection and tracking [2,

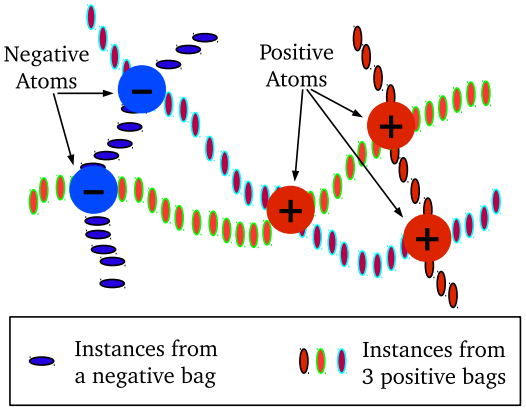


Figure 3.1: Motivation behind the proposed DD-based MIL dictionary learning framework.

19]. In particular, non-linear dictionaries have been shown to produce better results than the linear dictionaries in object recognition tasks [60–62]. While the MIL algorithms exist for popular classification methods like Support Vector Machines (SVM) [63] and decision trees [56], such algorithms have been studied only recently in the literature using the dictionary learning framework.

A dictionary-based MIL algorithm was recently proposed for event detection by [64] that iteratively prunes negative samples from positive bags based on the dictionary learned from the negative bags. One of the limitations of this approach is that, it may not generalize well for multi-class classification where computing a negative dictionary might be difficult. Another max-margin dictionary learning algorithm was proposed for computing spatial pyramid features from gray images for object and scene recognition by [65] . This dictionary consists of rows containing SVM weight vectors computed using the approach similar to MI-SVM. This dictionary is pre-multiplied to the dense features and the resulting coefficients are max-pooled. This algorithm takes dense features as its input and does not address

general image features.

Figure 3.1 provides the motivation behind the proposed method. Instances in a bag are points in feature space. Our goal in learning a positive concept is to find a point in the feature space that can represent at least one instance in each positive bag and does not represent any of the negative instances. In practical applications, with high dimensional feature space, it is difficult to represent each bag with just one such point. We seek to represent multiple such points as dictionary atoms. In this figure, we show instances from one negative bag and 3 positive bags. They can be imagined intersecting at different locations. From the problem definition, the negative bag contains only negative class samples, hence the region around the negative instances is very likely to be a negative concept, even if it intersects with the positive bags. However, the intersection of positive bags, is likely to belong to the positive concept. Traditional diverse density based approaches [22] can find only one positive concept that is close to the intersection of positive bags and away from the negative bags. Since one point in the feature space can not describe the positive class distribution, these approaches tend to compute different positive concepts with multiple initializations. In this work, we show that the multiple concepts are naturally captured by dictionary atoms and lead to a better performance. Figure 3.2 shows an overview of the proposed MIL dictionary learning method.

Key contributions of this chapter are as follows:

1. We propose a general dictionary learning and sparse coding based framework for MIL by learning a representation for the components common in the in-

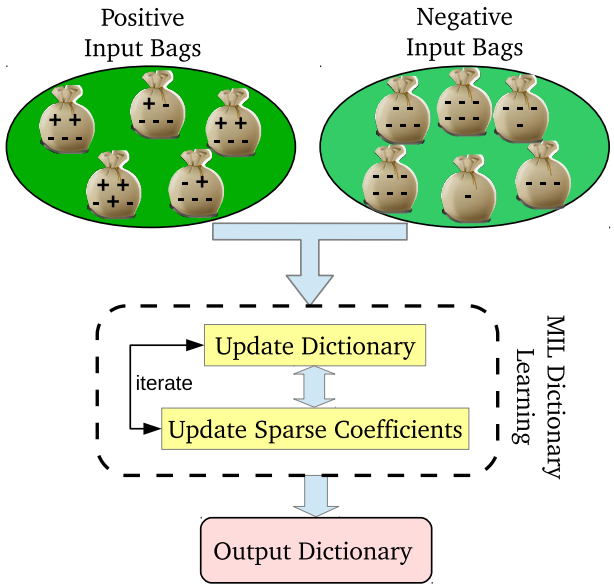


Figure 3.2: An overview of the proposed MIL dictionary learning framework.

stances of the same class bags and different for different class bags.

2. Under the MIL setting, we propose to exploit the non-linearity of data by learning a dictionary in the high dimensional feature space using a predetermined kernel function.

3. We propose two models for learning the sparse features of positive bags under the MIL setting; one is based on the noisy-OR model and the other is based on the Generalized Mean (GM) model.

4. We evaluate our method on various standard MIL datasets and advance the state-of-the-art on pain detection.

This chapter is organized as follows. Background discussion on sparse coding and dictionary learning are given in Section 3.2. Section 3.3 gives an overview of the proposed method and formulates the proposed MIL dictionary learning problem.

The details of the optimization steps are given in Section 3.4. The classification procedure using the learned dictionaries is described in Section 3.5. Experimental results are presented in Section 3.6 and Section 3.7 concludes the chapter with a brief summary and discussion.

3.2 Background

In this section, we give a brief background on sparse coding and dictionary learning.

3.2.1 Sparse Coding

Let $\mathbf{D}$ be a redundant (overcomplete) dictionary with K elements in $\mathbb{R}^d$

$$\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_K] \in \mathbb{R}^{d \times K}. \tag{3.1}$$

The elements of $\mathbf{D}$ (also known as atoms) are normalized to unit Euclidean norm i.e., $\|\mathbf{d}_i\| = 1 \ \ \forall i$. Given a signal $\mathbf{y}_t \in \mathbb{R}^d$, finding the sparsest representation of $\mathbf{y}_t$ in $\mathbf{D}$ entails solving the following optimization problem

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject } \text{ to } \mathbf{y}_t = \mathbf{D}\mathbf{x}, \tag{3.2}$$

where $\|\mathbf{x}\|_0 := \#\{j : x_j \neq 0\}$, is a count of the number of nonzero elements in $\mathbf{x}$. Problem (3.2) is NP-hard and cannot be solved in a polynomial time. Hence, approximate solutions are usually sought. For instance, a stable solution can be obtained by solving the following optimization problem, provided certain conditions

are met [4]

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{y}_t - \mathbf{D}\mathbf{x}\|_2 + \lambda\|\mathbf{x}\|_1, \tag{3.3}$$

where λ is a regularization parameter and $\|\cdot\|_p$ for $0 < p < \infty$ is the ℓ_p-norm defined as

$$\|\mathbf{x}\|_p = \left(\sum_{j=1}^{d} |x_j|^p\right)^{\frac{1}{p}}. \tag{3.4}$$

3.2.2 Dictionary Learning

Traditionally, the dictionary $\mathbf{D}$ in (3.1), is predetermined; e.g., wavelets. It has been observed [66] that learning a dictionary directly from the training data rather than using a predetermined dictionary usually leads to a more compact representation and hence can provide improved results in many practical computer vision applications [2, 4, 58].

Several algorithms have been developed for the task of learning a dictionary from data samples [4, 10]. One of the most well-known algorithms is the KSVD algorithm proposed by [10]. Given a data matrix $\mathbf{Y} \in \mathbb{R}^{d\times N}$ with its columns as data samples $\mathbf{y}_i, i = 1, \ldots, N$, the goal of the KSVD algorithm is to find a dictionary $\mathbf{D}$ and a sparse matrix $\mathbf{X}$ that minimize the following representation error

$$(\hat{\mathbf{D}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \text{ such that } \|\mathbf{x}_i\|_0 \leq T_0 \ \forall i, \tag{3.5}$$

where $\mathbf{x}_i$'s denote the columns of $\mathbf{X}$, $\|.|_F$ denotes the Frobenius norm and T_0 denotes the sparsity level. The KSVD algorithm is an iterative method and alternates between sparse-coding and dictionary update steps. First, a dictionary $\mathbf{D}$ with

ℓ_2 normalized columns is initialized. Then, the main iteration is composed of the following two stages:

- *Sparse coding*: In this step, $\mathbf{D}$ is fixed and the following optimization problem is solved to compute the representation vector $\mathbf{x}_i$ for each sample $\mathbf{y}_i, i = 1, \cdots, N$,

$$\min_{\mathbf{x}_i} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 \text{ such that } \|\mathbf{x}_i\|_0 \leq T_0. \tag{3.6}$$

  Any standard technique can be used to solve this problem. In fact, approximate solutions can be obtained by solving problems similar to (3.3).

- *Dictionary update*: In KSVD, the dictionary update is performed atom-by-atom in a computationally efficient way rather than using a matrix inversion. For a given atom l, the quadratic term in (3.5) can be rewritten as

$$\|\mathbf{Y} - \sum_{i \neq l} \mathbf{d}_i \mathbf{x}_i^T - \mathbf{d}_l \mathbf{x}_l^T\|_F^2 = \|\mathbf{E}_l - \mathbf{d}_l \mathbf{x}_l^T\|_F^2, \tag{3.7}$$

  where $\mathbf{E}_l$ is the residual matrix, $\mathbf{d}_l$ is the l^{th} atom of the dictionary $\mathbf{D}$ and $\mathbf{x}_i^T$ are the rows of $\mathbf{X}$. The atom update is obtained by minimizing (3.7) for $\mathbf{d}_l$ and $\mathbf{x}_l^T$ through a simple rank-1 approximation of $\mathbf{E}_l$ [10].

3.2.3 Discriminative Dictionary Learning

Given a data matrix $\mathbf{Y}$, the general cost function for learning a dictionary takes the following form

$$\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda \Psi(\mathbf{X}), \tag{3.8}$$

where λ is a parameter and columns of $\mathbf{Y}$, $\mathbf{D}$, and $\mathbf{X}$ contain the training signals, the dictionary atoms, and their coefficients, respectively. While these approaches are purely generative, the design of supervised discriminative dictionaries has also gained a lot of traction in recent years [2], [19]. The design of such dictionaries entails modification of the function $\Psi(\mathbf{X})$ in (3.8) so that not only sparsity is enforced but discrimination is also maintained. This is often done by introducing linear discriminant analysis type of discrimination on the sparse coefficients which essentially enforces separability among dictionary atoms of different classes [67], [31], [30], [29], [32], [68]. Manipulation of $\Psi(\mathbf{X})$ so that it enforces group sparsity can also lead to the design of hierarchical dictionaries.

3.2.4 Non-Linear Dictionary Learning

Kernel-based non-linear sparse coding and dictionary learning methods have also been proposed in the literature [40, 41, 69]. These methods essentially map the input data onto a high dimensional feature space using a predetermined kernel function. Sparse codes and dictionaries are then trained on the feature space for better representation and discrimination. Let $\mathbf{\Phi}(.) : \mathbb{R}^d \rightarrow G$ be a mapping from a d-dimensional space into a dot product space G. A non-linear dictionary can be trained in the feature space G by solving the following optimization problem

$$(\hat{\mathbf{A}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{A},\mathbf{X}} \|\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \text{ subject to}$$

$$\|\mathbf{x}_i\|_0 \leq T_0 \quad \forall i \tag{3.9}$$

where

$$\mathbf{\Phi}(\mathbf{Y}) = [\mathbf{\Phi}(\mathbf{y}_1), \cdots, \mathbf{\Phi}(\mathbf{y}_N)].$$

Since the dictionary lies in the linear span of the samples $\mathbf{\Phi}(\mathbf{Y})$, in (3.9) we have used the following model for the dictionary in the feature space,

$$\mathbf{\Phi}(\mathbf{D}) = \mathbf{\Phi}(\mathbf{Y})\mathbf{A},$$

where $\mathbf{A} \in \mathbb{R}^{N \times K}$ is a matrix with K atoms [40, 60],

$$\mathbf{\Phi}(\mathbf{D}) = [\mathbf{\Phi}(\mathbf{d}_1), \ldots, \mathbf{\Phi}(\mathbf{d}_K)].$$

This model provides adaptivity via modification of the matrix $\mathbf{A}$. Through some algebraic manipulations, the cost function in (3.9) can be rewritten as,

$$\begin{aligned} &\|\mathbf{\Phi}(\mathbf{Y}) - \mathbf{\Phi}(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \\ &\qquad = \mathbf{tr}((\mathbf{I} - \mathbf{A}\mathbf{X})^T \boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})(\mathbf{I} - \mathbf{A}\mathbf{X})), \end{aligned} \tag{3.10}$$

where $\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})$ is a kernel matrix whose elements are computed from

$$\kappa(i, j) = \mathbf{\Phi}(\mathbf{y}_i)^T \mathbf{\Phi}(\mathbf{y}_j).$$

It is apparent that the objective function is feasible since it only involves a matrix of finite dimension $\boldsymbol{\mathcal{K}} \in \mathbb{R}^{N \times N}$, instead of dealing with a possibly infinite dimensional dictionary.

An important property of this formulation is that the computation of $\boldsymbol{\mathcal{K}}$ only requires dot products. Therefore, one can employ Mercer kernel functions to compute these dot products without carrying out the mapping $\mathbf{\Phi}$. Some commonly used

kernels include polynomial kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle(\mathbf{x}, \mathbf{y}\rangle + a)^b$$

and Gaussian kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{c}\right),$$

where a, b and c are the parameters.

Similar to the optimization of (3.5) using the linear KSVD algorithm, the optimization of (3.9) involves sparse coding and dictionary update steps in the feature space which results in the kernel KSVD algorithm. Details of the optimization can be found in the paper by [40].

Supervised dictionary learning methods (both linear and nonlinear) have shown to produce sate-of-the-art results in many classification tasks. However, in the presence of label ambiguity such is the case in MIL, supervised dictionary learning methods are no longer applicable and don't work well in practice. As a result, a new dictionary learning framework for MIL is necessary.

3.3 Overview and Problem Formualtion

In this section, we give an overview of the proposed MIL dictionary learning framework. We then formulate the proposed multi-class MIL dictionary learning problem.

3.3.1 Overview of the Proposed Approach

Assume that we are given N labeled bags $\mathbf{Y}_i$ and their corresponding labels l_i for all $i = 1, \dots N$. Each label can be from one of the C classes, i.e. $l_i \in \{1, \dots, C\}$. A bag $\mathbf{Y}_i$ can have one or more samples, called instances, denoted by $\mathbf{y}_{ij}, j = 1, \dots M_i$ where M_i is the number of instances in the i^{th} bag. In multi-class MIL setting, if the label of a bag is l_i, at least one of its instances should belong to class l_i. In many computer vision applications a bag corresponds to an image and its instances can be created by varying the scale, position or region of interest. For example, in tracking by detection application [70] multiple overlapping patches are used as instances and in object recognition application multiple regions of an image are treated as instances [55, 56, 71].

The main focus of this work is to obtain a good representation by learning a dictionary for each class with the given labeled training bags. We represent each instance as a sparse linear combination of the dictionary atoms that are representative of the true class. However, when learning the underlying structure in each class, it is important to consider only those instances which belong to the bag's class and disregard the instances from other classes. Existing algorithms for dictionary learning need samples as input and can not work with bags. Hence, in this work we propose a general DD-based dictionary learning algorithm that can learn the representation of each class from bags under the MIL setting.

Our approach relies on learning a dictionary

$$\mathbf{\Phi}(\mathbf{D}_c) = \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c$$

for each class c in high-dimensional feature space, where the matrix $\mathbf{Y}^c$ contains all the instances of the c^{th} class bags, and $\mathbf{A}_c$ is a matrix that we want to learn as a part of the non-linear dictionary learning process. We learn $\mathbf{\Phi}(\mathbf{D}_c)$ by adapting columns of $\mathbf{A}_c$. The instances in bag $\mathbf{Y}^c$ that truly have the bag label c, should be well represented by this dictionary. Towards achieving this goal, we define the probability of an instance $\mathbf{y}_{ij}$ belonging to the c^{th} class as,

$$p_{ij} := \exp\left(-\|\mathbf{\Phi}(\mathbf{y}_{ij}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2^2/\sigma\right), \tag{3.11}$$

where $\mathbf{x}_{ij}$ is the sparse coefficient corresponding to $\mathbf{y}_{ij}$ and $\|\mathbf{\Phi}(\mathbf{y}_{ij}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2$ is the reconstruction error in the feature space. The hyperparameter σ is usually set to 1 for learning dictionaries.

Our goal is to learn $\mathbf{\Phi}(\mathbf{D}_c)$ via $\mathbf{A}_c$ for which at least one instance in each bag of class c is well represented (i.e., the probability is high) and the bags of all the other classes (i.e., not c) are poorly represented. This objective can be captured by computing positive bag probabilities as the maximum probability of its instances

$$\tilde{\mathcal{J}} = \prod_{i:l_i=c}\Big(\max_j \; p_{ij}\Big)\prod_{i:l_i\neq c}\Big(\prod_{j=1}^{M_i}(1-p_{ij})\Big). \tag{3.12}$$

Note that, for $\tilde{\mathcal{J}}$ to be high, at least one instance from each bag of class c should have high p_{ij}, while all the instances in the bags of the other classes should have low probability. If we maximize the above cost with respect to the matrix $\mathbf{A}_c$, we can learn the structure common to all the c^{th} class bags and absent from the bags of other classes. Since max operation is highly non-smooth, we need to approximate it with a smooth function to be able to optimize the cost. A popular choice explored

in many MIL works [22, 54, 72] is to approximate the max function with a smooth noisy-OR (NOR) model defined as

$$\mathcal{S}_{NOR}(\mathbf{p}_i) = 1 - \prod_{j=1}^{M_i}(1 - p_{ij}), \tag{3.13}$$

where $\mathbf{p}_i := [p_{i1}, \ldots, p_{iM_i}]^T$. Note that if one instance in the i^{th} bag is positive with a very high probability, the product term is going to be close to zero and the bag probability will be close one. One limitation of this model is that the probability is biased to bag size and for a large bag the product term diminishes very fast even if each instance has very low probability. For example a bag of 100 instances each with probability 0.05 will result in $\mathcal{S}_{NOR}(\mathbf{p}_i) = 0.9941$ which is very high considering that true $\max_j p_{ij} = 0.05$. Another approximation of the max function can be formulated in the form of generalized mean as explored by [72] and [73]. This model is not sensitive to bag size but averages out the instance probabilities after raising them to a high power as defined by

$$\mathcal{S}_{GM}(\mathbf{p}_i) = \Big(\frac{1}{M_i}\sum_{j=1}^{M_i} p_{ij}^r\Big)^{1/r}, \tag{3.14}$$

where r is a parameter that controls the approximation of S_{GM} to the true max function. A higher value of r results into a better approximation. However, a very high value can result in numerical instability. In our experiments, we set it equal to 10. The GM approximation under-estimates the true max value while NOR model over-estimates it. For a smaller bag size where a few instances have much higher probability compared to the rest of them, NOR model is a better approximation. For example, consider a case where a positive bag has two instances with their respective probabilities of belonging to positive class as 0.9 and 0.1. A GM approximation in

this case is 0.84 while NOR results in a better approximation of 0.91. Let us denote this general soft-max function by $\mathcal{S}$, where $\mathcal{S}$ can be replaced by either $\mathcal{S}_{NOR}$ or $\mathcal{S}_{GM}$, i.e.,

$$\max_j \; p_{ij} \approx \mathcal{S}(\mathbf{p}_i).$$

Hence, the objective (3.12) can be approximated as

$$\tilde{\mathcal{J}} = \prod_{i:l_i=c} \mathcal{S}(\mathbf{p}_i) \prod_{i:l_i \neq c} \Big(\prod_{j=1}^{M_i} (1 - p_{ij}) \Big). \tag{3.15}$$

Once the dictionaries are learned for each class by minimizing the above objective with the sparsity constraint, one can concatenate them to form a global dictionary and compute the representation of the instances using this dictionary. Features can be computed for each bag from this representation and classified using the popular classification algorithms such as Support Vector Machine (SVM). Figure 3.3 presents an overview of our method. We refer to this method as Generalized Dictionaries for MIL (GD-MIL).

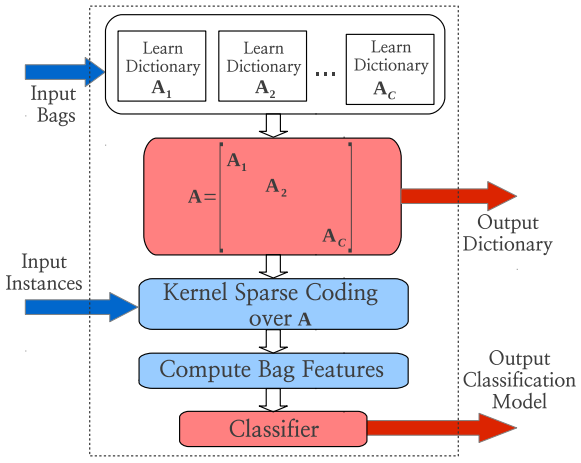


Figure 3.3: Block diagram of the proposed GD-MIL method.

Table 3.1 summarizes the notations used in this chapter. We would like to

draw the reader's attention to subtle but important difference between subscript and superscript of $\mathbf{Y}$ and M, where subscript refers to the bag index while superscript refers to the class index.

Notation	Description
N	Total number of bags
C	Number of classes
$l_i \in \{1, \ldots, C\}$	Label of the i^{th} bag
M_i	Number of instances in the i^{th} bag
d	Dimension of each instance
M	Total number of instances in all the bags
M^c	Total number of instances in all the c^{th} class bags
$\mathbf{Y} \in \mathbb{R}^{d \times M}$	Data matrix with columns as instances from all the bags
$\mathbf{Y}^c \in \mathbb{R}^{d \times M^c}$	Data matrix with columns as instances from all the c^{th} class bags
$\mathbf{Y}_i \in \mathbb{R}^{d \times M_i}$	Matrix with columns as instances from the i^{th} bag
$\mathbf{y}_{ij} \in \mathbb{R}^d$	j^{th} instance of the i^{th} bag
$\mathbf{A}_c \in \mathbb{R}^{M^c \times K_c}$	Matrix whose columns control the dictionary atoms in feature space. It is also referred to as c^{th} class dictionary
K_c	Number of atoms (or columns) in the c^{th} class dictionary $\mathbf{A}_c$
$\mathbf{X} \in \mathbb{R}^{K_c \times M}$	Sparse coefficient matrix of all instances corresponding to dictionary $\mathbf{A}_c$
$\mathbf{X}_i \in \mathbb{R}^{K_c \times M_i}$	Sparse coefficient matrix i^{th} bag instances corresponding to dictionary $\mathbf{A}_c$
$\mathbf{x}_{ij}$	j^{th} coefficient vector of the i^{th} bag. Vector length depends on implicit dictionary size it is computed with
x_{ijk}	k^{th} element of $\mathbf{x}_{ij}$
p_{ij}	Probability that the j^{th} instance of the i^{th} bag belongs to a positive (c^{th}) class
$\mathbf{p}_i \in \mathbb{R}^{M_i}$	Vector containing the probabilities of all the instances in the i^{th} bag, i.e., $\mathbf{p}_i := [p_{i1}, \ldots, p_{iM_i}]$
$\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c) \in \mathbb{R}^{M^c \times M^c}$	Kernel matrix computed from the c^{th} class instances
κ	Kernel function used to compute the elements of the kernel matrix

Table 3.1: Summary of key notations.

3.3.2 Problem Formulation

We denote the data matrix by $\mathbf{Y} = [\mathbf{Y}_1, \ldots, \mathbf{Y}_N] \in \mathbb{R}^{d \times M}$. Here, $M = M_1 + \cdots + M_N$ is the total number of instances in all the bags, M_i is the number of instances

in the i^{th} bag and d is the dimension of the features for each instance. Let $\mathbf{Y}^c$ be the concatenation of all the c^{th} class bags, i.e, $\mathbf{Y}^c = [\mathbf{Y}_i : l_i = c] \in \mathbb{R}^{d \times M^c}$. Note that the subscript i in $\mathbf{Y}_i$ denotes the bag index and superscript c in $\mathbf{Y}^c$ denotes the matrix of all the bags that belong to class c. Similarly, M^c is the total number of instances in all the c^{th} class bags, i.e. $M^c = \sum_{i:l_i=c} M_i$. For the simplicity of notation, we re-index instances of all the c^{th} class bags and write $\mathbf{Y}^c = [\mathbf{y}_1^c, \ldots, \mathbf{y}_{M^c}^c]$, where $\mathbf{y}_i^c$ is the i^{th} instance of the c^{th} class after re-indexing.

Our objective is to learn a dictionary $\mathbf{\Phi}(\mathbf{D}_c)$ defined as $\mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c$ for each class in the feature space, where columns of $\mathbf{A}_c \in \mathbb{R}^{M^c \times K_c}$ are optimized to learn the non-linear dictionary. For simplicity, we refer to $\mathbf{A}_c$ as the dictionary for the c^{th} class. Given $\mathbf{A}_c$, we can represent an instance $\mathbf{y}$ as a sparse linear combination of the columns of $\mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c$ in the feature space as follows

$$\mathbf{\Phi}(\mathbf{y}) = \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x} + \epsilon, \tag{3.16}$$

where $\mathbf{\Phi}(\mathbf{Y}^c) = [\mathbf{\Phi}(\mathbf{y}_1^c), \ldots, \mathbf{\Phi}(\mathbf{y}_{M^c}^c)]$ and ϵ is the error term. The sparse coefficient $\mathbf{x}$ can be obtained by solving the following optimization problem [67]

$$\mathbf{x} = \arg\min_{\mathbf{z}} \|\mathbf{\Phi}(\mathbf{y}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1. \tag{3.17}$$

Next, we represent the j^{th} instance of the i^{th} bag using the dictionary $\mathbf{A}_c$ and write its probability p_{ij} in terms of the representation error as follows,

$$\begin{aligned} p_{ij}(\mathbf{A}_c, \mathbf{x}_{ij}) &= \exp\big(-\|\mathbf{\Phi}(\mathbf{y}_{ij}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2^2\big) \\ &= \exp\big(-\boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{y}_{ij}) - \mathbf{x}_{ij}^T\mathbf{A}_c^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij} \\ &\quad + 2\boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\big), \end{aligned} \tag{3.18}$$

where $\mathbf{x}_{ij}$ is the sparse coefficient of $\mathbf{y}_{ij}$,

$$
\begin{aligned}
[\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)]_{i,j} &= [\langle \boldsymbol{\Phi}(\mathbf{Y}^c), \boldsymbol{\Phi}(\mathbf{Y}^c) \rangle]_{i,j} \\
&= \boldsymbol{\Phi}(\mathbf{y}_i^c)^T \boldsymbol{\Phi}(\mathbf{y}_j^c) = \kappa(\mathbf{y}_i^c, \mathbf{y}_j^c), \\
\boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{y}_{ij}) &= \kappa(\mathbf{y}_{ij}, \mathbf{y}_{ij}), \text{and} \\
\boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{Y}^c) &= [\kappa(\mathbf{y}_{ij}, \mathbf{y}_1^c), \ldots, \kappa(\mathbf{y}_{ij}, \mathbf{y}_{M^c}^c)] \in \mathbb{R}^{1 \times M^c}.
\end{aligned}
$$

In order to learn the dictionary $\mathbf{A}_c = [\mathbf{a}_1, \cdots, \mathbf{a}_{K_c}]$ for class c, we need to optimize the cost in (3.15) with respect to $\mathbf{A}_c$ and all the sparse coefficients $\mathbf{x}_{ij}$. We denote all the sparse coefficients for the c^{th} class dictionary by the matrix $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_N] \in \mathbb{R}^{K_c \times M}$ where $\mathbf{X}_i = [\mathbf{x}_{i1}, \ldots, \mathbf{x}_{iM_i}] \in \mathbb{R}^{K_c \times M_i}$. In other words, $\mathbf{X}_i$ contains the sparse coefficients for all the instances of the i^{th} bag and $\mathbf{X}$ contains all the sparse coefficients from all the bags. Note that, for notational simplicity, we have not used any subscript/superscript c with $\mathbf{X}$, $\mathbf{X}_i$ and $\mathbf{x}_{ij}$ to indicate that these sparse coefficients are computed using the c^{th} class dictionary. Next, we take the negative log of the cost $\tilde{\mathcal{J}}$ in (3.15), and introduce a parameter α that controls the influence of the non-c^{th} class bags,

$$
\begin{aligned}
\mathcal{J}(\mathbf{A}_c, \mathbf{X}) = &- \sum_{i:l_i=c} \log \mathcal{S}(\mathbf{p}_i) \\
&- \alpha \sum_{i:l_i \neq c} \sum_{j=1}^{M_i} \log(1 - p_{ij}).
\end{aligned} \tag{3.19}
$$

The resulting problem of learning the non-linear dictionaries can be captured in following optimization problem,

$$
\hat{\mathbf{A}}_c, \hat{\mathbf{X}} = \arg\min_{\mathbf{A}_c, \mathbf{X}} \mathcal{J}(\mathbf{A}_c, \mathbf{X}) + \lambda \|\mathbf{X}\|_1, \tag{3.20}
$$

where $\|\mathbf{X}\|_1 = \sum_n \|\mathbf{x}_n\|_1$. Note that $\mathcal{J}(\mathbf{A}_c, \mathbf{X})$ is a function of p_{ij}. The atoms of a dictionary are normalized to unit norm. This can be enforced by adding the following constraint in the optimization problem (3.20),

$$(\mathbf{\Phi}(\mathbf{Y}^c)\mathbf{a}_m)^T(\mathbf{\Phi}(\mathbf{Y}^c)\mathbf{a}_m) = \mathbf{a}_m^T \boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_m = 1.$$

Hence, the overall optimization problem (3.20) can be re-written as

$$\hat{\mathbf{A}}_c, \hat{\mathbf{X}} = \arg\min_{\mathbf{A}_c, \mathbf{X}} \mathcal{J}(\mathbf{A}_c, \mathbf{X}) + \lambda\|\mathbf{X}\|_1,$$

subject to

$$\mathbf{a}_m^T \boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_m = 1, m = 1, \ldots, K_c. \tag{3.21}$$

3.4 Optimization Approach

In this section, we develop an approach to solve (3.21) by alternatively optimizing the dictionary $\mathbf{A}_c$ and coefficient matrix $\mathbf{X}$. Similar to the KSVD approach [10], for updating the dictionary, we optimize one atom at a time while keeping the others fixed. To satisfy the unit norm constraint on the atoms, we re-normalize the atom at each step of the proposed gradient descent algorithm. We first write instance probabilities p_{ij} as a function of $\mathbf{a}_k$, and then utilize it to update $\mathbf{a}_k$.

3.4.1 Instance Probabilities p_{ij} in terms of $\mathbf{a}_k$

Using (3.18), we can re-write p_{ij} as a function of the k^{th} atom $\mathbf{a}_k$ as

$$\begin{aligned} p_{ij}(\mathbf{a}_k, x_{ijk}) &= \exp\big(-\|\mathbf{\Phi}(\mathbf{y}_{ij}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{ij}\|_2^2\big) \\ &= \exp\big(-\|\mathbf{\Phi}(\mathbf{y}_{ij}) - \mathbf{\Phi}(\mathbf{Y}^c)\sum_{\substack{m=1\\m\neq k}}^{K_c}\mathbf{a}_m x_{ijm} \\ &\qquad - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{a}_k x_{ijk}\|_2^2\big) \\ &= \exp\left(-\|\mathbf{\Phi}(\mathbf{r}_{ij}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{a}_k x_{ijk}\|_2^2\right). \end{aligned} \tag{3.22}$$

Here, x_{ijk} is the k^{th} element of the sparse vector $\mathbf{x}_{ij}$ and

$$\mathbf{\Phi}(\mathbf{r}_{ij}) = \mathbf{\Phi}(\mathbf{y}_{ij}) - \mathbf{\Phi}(\mathbf{Y}^c)\sum_{\substack{m=1\\m\neq k}}^{K_c}\mathbf{a}_m x_{ijm}.$$

One can clearly see the similarity between this expression and the one in (3.7).

After a few algebraic manipulations, p_{ij} in (3.22) can be rewritten in terms of the kernel matrices as follows

$$\begin{aligned} p_{ij}(\mathbf{a}_k, x_{ijk}) = \exp\big(&-\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{r}_{ij}) - x_{ijk}^2\mathbf{a}_k^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_k \\ &+ 2x_{ijk}\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c)\mathbf{a}_k\big), \end{aligned} \tag{3.23}$$

where,

$$\begin{aligned} \boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{r}_{ij}) &= \boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{y}_{ij}) + \sum_{\substack{m=1\\m\neq k}}^{K_c} x_{ijm}^2\mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_m \\ &\quad - 2\sum_{\substack{m=1\\m\neq k}}^{K_c} x_{ijm}\mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{y}_{ij}), \quad \text{and} \end{aligned} \tag{3.24}$$

$$\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c) = \boldsymbol{\mathcal{K}}(\mathbf{y}_{ij}, \mathbf{Y}^c) - \sum_{\substack{m=1\\m\neq k}}^{K_c} x_{ijm}\mathbf{a}_m^T\boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c). \tag{3.25}$$

3.4.2 Atom Update

We propose a gradient descent method to optimize the k^{th} atom $\mathbf{a}_k$. Recall that we denote the coefficient of the j^{th} instance of i^{th} bag corresponding to the k^{th} atom by x_{ijk}. Now, we collect the coefficients of all the instances in i^{th} bag into a vector $\mathbf{x}_i^k := [x_{i1k}, \ldots, x_{iM_ik}]$. Denote the cost for optimizing $\mathbf{a}_k$ by $\mathcal{J}_{\mathbf{a}_k}$. Note that $\mathcal{J}_{\mathbf{a}_k}$, from (3.19), is a function of p_{ij} and, together with the definition of p_{ij} in (3.23), can be written as,

$$\begin{aligned}\mathcal{J}_{\mathbf{a}_k}(\mathbf{a}_k) = &- \sum_{i:l_i=c} \log \mathcal{S}\big(\mathbf{p}_i(\mathbf{a}_k, \mathbf{x}_i^k)\big) \\ &- \alpha \sum_{i:l_i\neq c} \sum_{j=1}^{M_i} \log(1 - p_{ij}\big(\mathbf{a}_k, x_{ijk})\big).\end{aligned} \tag{3.26}$$

Hence, the optimization problem (3.21) can be reformulated for the k^{th} atom as,

$$\hat{\mathbf{a}}_k = \arg\min_{\mathbf{a}_k} \mathcal{J}_{\mathbf{a}_k}(\mathbf{a}_k), \tag{3.27}$$

$$\text{subject to} \quad \mathbf{a}_k^T \boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a}_k = 1. \tag{3.28}$$

Optimization of $\mathbf{a}_k$ in (3.27) can be viewed as minimizing the negative log likelihood and it can be solved using the gradient descent method. To perform gradient descent on $\mathcal{J}_{\mathbf{a}_k}$, we need to compute the derivatives of the softmax functions with respect to $\mathbf{a}_k$. For the NOR model, we get

$$\frac{\partial \log \mathcal{S}_{NOR}}{\partial \mathbf{a}_k} = \frac{1 - b_i}{b_i} \sum_{j=1}^{M_i} \Big(\frac{1}{1 - p_{ij}} \frac{\partial p_{ij}}{\partial \mathbf{a}_k}\Big), \tag{3.29}$$

where

$$b_i := 1 - \prod_{j=1}^{M_i} (1 - p_{ij}),$$

and $\frac{\partial p_{ij}}{\partial \mathbf{a}_k}$ is the partial derivative of the instance probability with respect to the atom. Similarly, for the GM model, we get

$$\frac{\partial \log \mathcal{S}_{GM}}{\partial \mathbf{a}_k} = \frac{1}{\sum_{j=1}^{M_i} p_{ij}} \sum_{j=1}^{M_i} \Big(p_{ij}^{r-1} \frac{\partial p_{ij}}{\partial \mathbf{a}_k} \Big). \tag{3.30}$$

The derivative of p_{ij} with respect to $\mathbf{a}_k$ is calculated as follows

$$\frac{\partial p_{ij}}{\partial \mathbf{a}_k} = 2p_{ij}[\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c)\mathbf{a}_k - \boldsymbol{\mathcal{K}}(\mathbf{Y}^c, \mathbf{Y}^c)\mathbf{a_k} x_{ijk}]. \tag{3.31}$$

The derivative of the part that involves the negative instances in (3.26) with respect to $\mathbf{a}_k$ is computed in a straight forward manner as,

$$\frac{\partial}{\partial \mathbf{a}_k} \log(1 - p_{ij}) = -\frac{1}{1 - p_{ij}} \frac{\partial p_{ij}}{\partial \mathbf{a}_k}. \tag{3.32}$$

Finally, from (3.26) the derivative of $\mathcal{J}_{\mathbf{a}_k}$ is computed as

$$\begin{aligned} \frac{\partial \mathcal{J}_{\mathbf{a}_k}}{\partial \mathbf{a}_k} = &- \sum_{i:l_i=c} \frac{\partial \log \mathcal{S}(\mathbf{p}_i)}{\partial \mathbf{a}_k} \\ &- \alpha \sum_{i:l_i \neq c} \sum_{j=1}^{M_i} \frac{\partial}{\partial \mathbf{a}_k} \log(1 - p_{ij}), \end{aligned} \tag{3.33}$$

where, $\mathcal{S}$ can be replaced with either $\mathcal{S}_{NOR}$ or $\mathcal{S}_{GM}$ depending on the choice of the soft-max function.

3.4.3 Coefficient Update

In this sub-section, we describe how to update the sparse coefficients for different instances. Note that in (3.19) the probabilities of the instances from negative bags are separable while that of the instances from positive bags are not. Hence, we update the coefficients of the negative bags instances and the positive bags instances differently. From (3.19), for each negative instance coefficient, the cost can

be written as,

$$\mathcal{J}^{-}_{\mathbf{x}_{ij}}(\mathbf{x}_{ij}) = -\log(1 - p_{ij}(\mathbf{x}_{ij})) + \lambda\|\mathbf{x}_{ij}\|_1. \tag{3.34}$$

Since the positive instances are not separable, we update i^{th} bag coefficient matrix $\mathbf{X}_i$, if $l_i = c$, by minimizing (3.19) w.r.t. $\mathbf{X}_i$. Lets denote this cost for c^{th} class bags by $\mathcal{J}^{+}_{\mathbf{X}_i}$ which can be defined as,

$$\mathcal{J}^{+}_{\mathbf{X}_i}(\mathbf{X}_i) = -\log \mathcal{S}(\mathbf{p}_i(\mathbf{X}_i)) + \lambda\|\mathbf{X}_i\|_1. \tag{3.35}$$

Note that the cost in (3.34) and (3.35) are non-differentiable due to the ℓ_1 regularization term. Multiple approaches have been developed to minimize such functions [74,75] when the cost without ℓ_1 regularization is smooth. In particular, we use the active set method described by [75]. This method requires the computation of the derivative of the smooth part of the cost. For the positive bags, it can be computed similar to (3.29) or (3.30) depending on the choice of the softmax function. The only difference is that we need to compute $\frac{\partial p_{ij}}{\partial \mathbf{X}_i}$ instead of $\frac{\partial p_{ij}}{\partial \mathbf{a}_k}$ which is done as follows

$$\begin{aligned}\frac{\partial p_{ij}}{\partial \mathbf{X}_i} = 2\big[&\mathbf{A}_c^T \boldsymbol{\mathcal{K}}(\mathbf{Y}_c, \mathbf{Y}_i) \\ &- \mathbf{A}_c^T \boldsymbol{\mathcal{K}}(\mathbf{Y}_c, \mathbf{Y}_c)\mathbf{A}_c\mathbf{X}_i\big]\mathbf{P}_i,\end{aligned} \tag{3.36}$$

where $\mathbf{P}_i$ is a diagonal matrix with instance probabilities of the i^{th} bag in its diagonal

$$\mathbf{P}_i := \begin{bmatrix} p_{i1} & & \\ & \ddots & \\ & & p_{iM_i} \end{bmatrix}. \tag{3.37}$$

The derivative of $\mathcal{J}^-_{\mathbf{x}_{ij}}$ w.r.t. $\mathbf{x}_{ij}$ is computed similar to (3.32). For faster implementation, we collect the derivative of all the instances from positive as well as negative bags to compute $\frac{\partial \mathcal{J}}{\partial \mathbf{X}}$, and optimize $\mathcal{J}$ to update $\mathbf{X}$.

Different steps of the optimization for $\mathbf{A}_c$ are summarized in Algorithm 3.

Algorithm 3: Algorithm for Learning c^{th} Class Dictionary $\mathbf{A}_c$

Input: Bags $\mathbf{Y}_i$, Labels $l_i, \forall i = 1, \dots N$, Kernel Function κ, Parameters

$\alpha, \lambda, K_c, maxItr$.

Output: $\mathbf{A}_c$.

for $itr = 1, \dots, maxItr$ **do**

  for $k = 1, \dots, K_c$ **do**

    1. Update $\mathbf{a}_k$ by solving (3.27) with the gradient descent method.

    2. Update $\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{r}_{ij})$ and $\boldsymbol{\mathcal{K}}(\mathbf{r}_{ij}, \mathbf{Y}^c)$ using (3.24) and (3.25),

    respectively.

  end

  Update the coefficient matrix $\mathbf{X}$ as described in section 3.4.3.

end

return $\mathbf{A}_c$.

3.4.4 Connection to the Traditional Dictionary Learning

It is interesting to note that first part of our cost $\mathcal{J}$ in (3.19) is identical to the traditional dictionary learning cost in the feature space [60], [10], when there is only one instance in each bag. Let this first part of the cost be denoted by $\mathcal{J}_1$. By

setting $M_i = 1, \forall i$ it becomes,

$$\begin{aligned}\mathcal{J}_1 &= -\sum_{i:l_i=c} \log\Big(1 - \prod_{j=1}^{M_i}(1-p_{ij})\Big) = -\sum_{i:l_i=c} \log p_{i1} \\ &= -\sum_{i:l_i=c} \log\exp\big(-\|\mathbf{\Phi}(\mathbf{y}_{i1}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{i1}\|_2^2\big) \\ &= \sum_{i:l_i=c} \|\mathbf{\Phi}(\mathbf{y}_{i1}) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_{i1}\|_2^2. \end{aligned} \tag{3.38}$$

Hence, in the case of one instance per bag, our problem formulation can also be viewed as a discriminative dictionary learning approach where the first part $\mathcal{J}_1$ ensures that the instances are well represented by the dictionary of the corresponding class, and the second part of the cost $\mathcal{J}$ in (3.19) ensures that the samples of the non-c^{th} classes are not represented well by the dictionary $\mathbf{A}_c$.

3.5 Classification

Having computed the dictionaries $\mathbf{A}_c, c = 1, \ldots, C$, for all the classes using method summarized in Algorithm 3, we combine them before computing the sparse codes for learning a classification model. We denote the combined dictionary in the feature space as $\mathbf{\Phi}(\tilde{\mathbf{Y}})\mathbf{A}$, where

$$\mathbf{\Phi}(\tilde{\mathbf{Y}}) := [\mathbf{\Phi}(\mathbf{Y}^1), \ldots \mathbf{\Phi}(\mathbf{Y}^C)]$$

and

$$\mathbf{A} := \begin{bmatrix} \mathbf{A}_1, & & \\ & \ddots & \\ & & \mathbf{A}_C \end{bmatrix}.$$

This is same as concatenating the dictionaries in feature space, i.e.,

$$\mathbf{\Phi}(\mathbf{D}) = [\mathbf{\Phi}(\mathbf{D}_1), \ldots, \mathbf{\Phi}(\mathbf{D}_C)].$$

We compute the sparse coefficients of all the training instances on the combined dictionary by solving the following problem [67]

$$\mathbf{x}_{ij} = \arg\min_{\mathbf{z}} \|\mathbf{\Phi}(\mathbf{y}_{ij}) - \mathbf{\Phi}(\tilde{\mathbf{Y}})\mathbf{A}\mathbf{z}\|_2^2 + \lambda\|\mathbf{z}\|_1. \tag{3.39}$$

We then compute the probability p_{ij} of this instance by (3.23) after replacing $\mathbf{A}_c$ by $\mathbf{A}$ and $\mathbf{Y}^c$ by $\tilde{\mathbf{Y}}$. The sparse representation of the training bags $\mathbf{Y}_i$ is obtained as the weighted combination of the sparse coefficients of its instances. For example, the sparse representation of the i^{th} training bag, denote by $\mathbf{x}_i$ is computed as

$$\mathbf{x}_i = \sum_{j=1}^{M_i} p_{ij}\mathbf{x}_{ij}.$$

Once we obtain the sparse codes for the training bags, any classification algorithm can be used to classify the samples. In this chapter, we utilize an SVM for classification.

Instance Classification: If the task is to classify the individual instances, we propose to use reconstruction error for classificatino. Given a test sample $\mathbf{y}_t$, we compute the sparse coefficient $\mathbf{x}_t$ on dictionary $\mathbf{A}$. The class of the test instance is given by,

$$\text{class of } \mathbf{y}_t = \arg\min_{c} \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y}^c)\mathbf{A}_c\mathbf{x}_t^c\|_2^2, \tag{3.40}$$

where $\mathbf{x}_t^c$ part of $\mathbf{x}_t$ corresponding to dictinary $\mathbf{A}_c$.

3.6 Experimental Results

In this section, we first analyze our algorithm using a synthetic dataset to gain additional insights. We then evaluate our method on popular datasets for MIL like the Tiger, Fox, Elephant [63], Musk [21] and the Corel dataset [57]. Furthermore, we employ the proposed method for the pain detection task [76]. In our previous studies based on kernel dictionary learning [40], [42], we have found that the polynomial kernel performs well on various image classification problems. As a result, we used a polynomial kernel of degree 4 in our experiments. Several methods have been proposed in the literature for optimizing the choice of kernel and kernel parameters such as cross validation and multiple kernel learning [77]. However, these methods tend to make the optimization problem very complex and time consuming. We have included two baselines using two different discriminative dictionary algorithms to compute sparse codes, followed by the SVM for classification. We used the DKSVD [29] method and the LCKSVD [30] method instead of the proposed dictionary learning algorithm. Since these discriminative dictionary algorithms need labels for each training sample, we assigned the label of the bag to each training instance. The classification on the sparse code was done similar to the proposed method by learning a SVM on the bag features. We denote these methods by DKSVD* and LC-KSVD* in the classification tables.

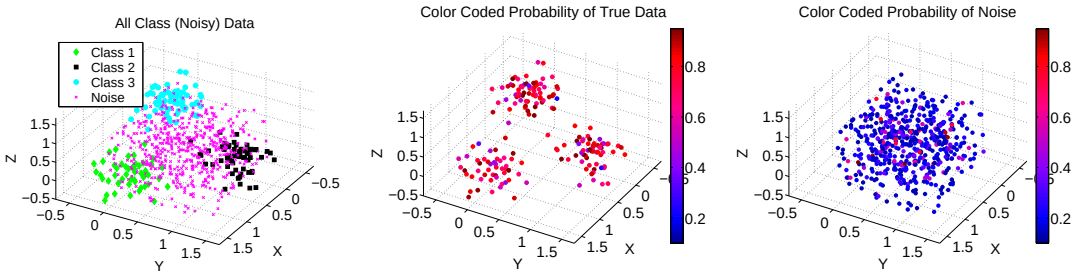
3.6.1 Synthetic Experiment

We analyze our algorithm using a three class synthetic dataset. We create 50 bags for each class by first drawing one sample per bag from three different Gaussian distributions (one for each class) with means $[1, 0, 0]^T$, $[0, 1, 0]^T$ and $[0, 0, 1]^T$ and following covariance matrix

$$\begin{bmatrix} 0.051 & 0.01 & 0.01 \\ 0.01 & 0.051 & 0.01 \\ 0.01 & 0.01 & 0.051 \end{bmatrix}.$$

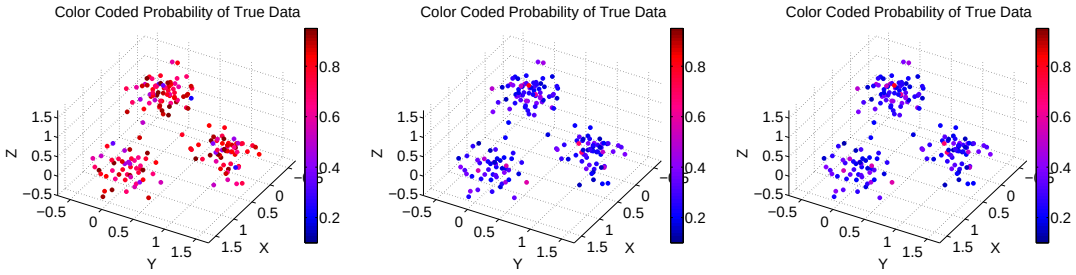
Then, in each bag, we add 3 more samples from uniformly distributed noise as shown in Figure 3.4(a). After learning the dictionary for each class using noisy bags, we project all the instances on the dictionaries of their respective classes and compute the probability for each instance using (3.18). The color coded probabilities are displayed in Figure 3.4(b) and (c). As can be observed from this figure, the instances from Gaussian distributions have higher probabilities (depicted by red color) and instances from noise distributions have lower probabilities (blue color). This clearly demonstrates that the proposed method learns the true representations of each class and reconstructs them well, despite the presence of multiple noise samples in each bag. Furthermore, the method does not learn the structure in the noise samples and hence gives high reconstruction errors for them.

To compare it with other dictionary learning methods we computed the probabilities of true positive samples on DKSVD [29] and LC-KSVD [30] dictionaries. These probabilities are plotted and compared with the proposed method in Figure



(a) (b) (c)

Figure 3.4: Demonstrating the probabilities of instances after projecting them onto their respective dictionaries: (a) Original noisy bags. Each bag contains one instance from a Gaussian distribution of its class and 3 instances from the uniformly distributed noise. (b) Color coded probabilities of instances from three Gaussian distributions, (c) Color coded probabilities of the uniformly distributed noise. Note that the probabilities of true instances are much higher than that of the noise.



(a) (b) (c)

Figure 3.5: Comparing the probabilities of instances from three Gaussian distributions, corresponding to three different classes. (a) Probabilities using GD-MIL, (b) Probabilities using DKSVD, (c) Probabilities using LC-KSVD. Note that the proposed method is able to handle well the noise in bags while the discriminative dictionaries are not able to represent the classes well.

3.5. Although the synthetic data seems simple, note that labels are very noisy and it is hard to learn a discriminative dictionary with so much noise. The traditional discriminative dictionary methods try to represent every sample (instead of just one) in each bag, the atoms are not very discriminative due to noise. We tried different values of the discriminative parameters, but it was hard to compute a dictionary with discriminative atoms. On the contrary, the proposed method tries to represent only one instance from each bag using softmax functions which can handle the noisy

bags effectively.

Atoms are learned by the proposed method to capture the common structure across all the bags of a class, at the same time suppressing the structure of other classes. Thus, the linear combination of these atoms will reconstruct the true samples of the class where as the background or noise is subdued. Since the classifier is learned over the reconstructed signal, it can discriminate the classes well despite the presence of noisy samples in the training bags.

3.6.2 MIL Benchmark Datasets

In this section, we evaluate the proposed approach on benchmark MIL datasets namely Tiger, Elephant and Fox introduced by [63], and the Musk1 and Musk2 proposed by [21]. Each of the Tiger, Elephant, and Fox datasets have 100 positive and 100 negative bags. A positive bag corresponds to the true image of an animal and negative bags are randomly drawn from the pool of other animals. The instances in each bag are created by segmenting the images. Color, texture, and shape features are used as described by [63]. The Musk1 and Musk2 datasets are publicly available datasets that were introduced in drug activity problem proposed by [21]. A bag in these datasets represent a drug molecule that can be represented by multiple features corresponding to different low-energy conformations. We use the same features and experimental set up as used by [63] and compare our results in Table 3.2. The numbers in the table for the competing methods, except PPMM, have been quoted from [56]. In this experiment, dictionaries are learned with 40 atoms

per class. The sparsity parameter $\lambda = 0.001$ and regularization parameter $\alpha = 0.01$ were used for dictionary learning for all the datasets. These two parameters were found using 5-fold cross-validation. Since many competing algorithms in Table 3.2 use NOR model, for fair comparison, we also use the same model to report the classification accuracies. We believe that the main reason why our method performs better is that we learn dictionary in such a way that the learned atoms can represent well the commonalities among the bags of the same class while they result in high reconstruction error for the non-common structure. By translating these reconstruction error into probabilities we are able to reduce the effect of the background of each image while computing the bag features.

Algorithms	Elephant	Fox	Tiger	Musk1	Musk2
mi-SVM [63]	82	58	79	87	84
MI-SVM [63]	81	59	84	78	84
MILES [57]	81	62	80	88	83
SIL-SVM	85	53	77	88	87
AW-SVM [78]	82	64	83	86	84
AL-SVM [78]	79	63	78	86	83
EM-DD [54]	78	56	72	85	85
MILBoost-NOR [72]	73	58	56	71	61
MIForests [56]	84	64	82	85	82
DKSVD* [29]	72	59	78	87	88
LC-KSVD* [30]	82	63	72	84	88
GD-MIL	**89**	**69**	**91**	**93**	**92**

Table 3.2: Average accuracy of five random splits on the benchmark datasets.

3.6.3 Corel Dataset

The Corel dataset consists of 20 object categories with 100 images per category. These images are taken from CD-ROMs published by the COREL Corporation. Each image is segmented into regions and each region is then called an instance [57].

The regions of an image can greatly vary depending on its complexity. We use the same instance features as used by [57] and report our result in Table 3.3. The numbers for the competing methods have been quoted from [57]. Here, we perform two categorization tasks: first on 10 object categories (corel-1000) and then on all the 20 object categories (corel-2000). For corel-1000 task, we analyze the class accuracy for each category using the confusion matrix in Figure 3.6. Each column in the confusion matrix corresponds to the predicted accuracy of the test samples. As we can see from the figure, class 2 ('Beach') is confused mostly with class 9 ('Mountains and glaciers') which is possibly due to their similar appearances. In both tasks the sparsity parameter is set equal to $\lambda = 0.001$, and $\alpha = 0.001$. Dictionaries are learned with 40 atoms per class. As before, λ and α were selected by 5-fold cross-validation.

Algorithms	1000-Image Dataset	2000-Image Dataset 2
MILES [57]	$82.6 : [81.4, 83.7]$	$68.7 : [67.3, 70.1]$
MI-SVM [63]	$74.7 : [74.1, 75.3]$	$54.6 : [53.1, 56.1]$
DD-SVM [55]	$81.5 : [78.5, 84.5]$	$67.5 : [66.1, 68.9]$
k-means-SVM [79]	$69.8 : [67.9, 71.7]$	$52.3 : [51.6, 52.9]$
DKSVD* [29]	$80.1 : [79.4, 80.8]$	$64.7 : [63.1, 66.6]$
LC-KSVD* [30]	$76.4 : [75.2, 77.6]$	$61.1 : [59.9, 62.2]$
GD-MIL	$\mathbf{84.3} : [83.1, 85.5]$	$\mathbf{72.6} : [71.5, 73.7]$

Table 3.3: Average accuracy along with the 95 percent confidence interval over five random test sets of Corel Dataset.

Furthermore, to study the effect of dictionary size on classification accuracy, we plot accuracy vs number of atoms for one of the splits of the corel1000 experiment in Figure 3.7. As can be seen from this plot that the results are not very sensitive when the number of atoms range from 30 to 45. Experiments have shown that increasing the number of atoms beyond 50 generally decreases the performance. This is not surprising because as more atoms are retained, the representation gets more exact,

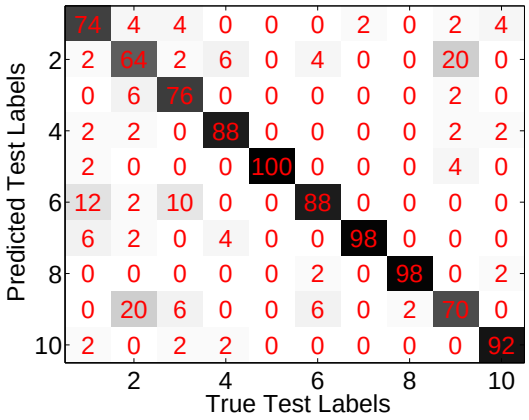


Figure 3.6: Confusion matrix for one of the splits of Corel-1000 image dataset.

and it has to deal with all the noise present in the data. Whereas with the fewer number of dictionary atoms, a more accurate description of the internal structure of the class is captured and robustness to noise is realized [26, 80, 81]. A similar trend is also observed with the other datasets.

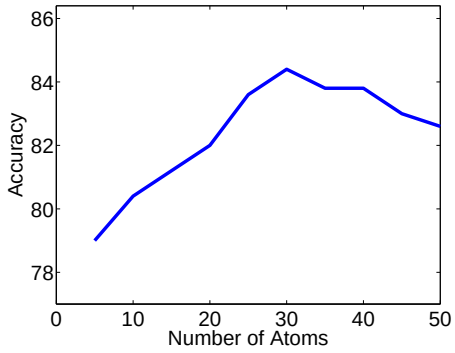


Figure 3.7: Classification accuracy vs number of atoms for corel1000 dataset.

3.6.4 Pain detection

In the final set of experiments, we address an important issue of detecting pain from a video sequence that has a very useful application in medical care. In certain scenarios, patient may not able to communicate his pain through verbal means or

does not know when to call for help due to his inability to judge the severity of the pain. For example, in the case of a child care or after an operation it is convenient to monitor the patient through a camera and alert the nurse when patient is in pain. We use image data from the UNBC-McMaster Pain Shoulder Archive as proposed by [76]. This dataset consists of 200 video sequences from 25 subjects suffering from shoulder pain due to various medical conditions. Each frame in a video sequence contains the face of the subject with varying expressions indicating the degree of pain he or she is experiencing due to various active and passive movements of their limbs. Each video sequence has been rated with Observer Pain Intensity (OPI) index ranging from $0-5$, with 0 being no pain and 5 being maximum pain. Following the protocols proposed by [73, 82, 83] the video sequences were divided into two categories : (1) 'pain' category or positive class with OPI rating greater than or equal to 3, (2) 'no-pain' category or negative class with OPI rating equal to 0. The sequences with intermediate ratings of 1 and 2 were omitted as per the protocol. Also, we included only those subjects that have atleast one positive class video and one negative class video sequence. This resulted in 146 video sequences from 22 subjects. The goal is to predict the class of a given video sequence of an unseen subject.

Many approaches have been proposed in literature to address this problem. [83] use active appearance model (AAA) to decouple shape and appearance parameters from face images. Based on the AAM features frames were clustered into multiple groups using K-means. Each of these clusters was given to train a SVM classifier for pain detection. At the testing time, the score of each video frame was predicted

using the learned SVM and then average score was used to predict the class of the video sequence. [82] use the AAM-SVM based approach as the baseline and improve its performance by compressing the image signal in spatial domain. An MIL based approach for pain detection was recently proposed by [73] where each video sequence was segmented into multiple segments of contiguous frames and each segment was considered an instance and the whole video sequence was considered a bag under MIL setting. An off-the-shelf MIL algorithm was applied to predict the label of the video sequence.

Similar to the approach taken by [73], we divide each video sequence into different segments. In order to do that, first a spatial pyramid feature is computed for each frame by max pooling the multi-scale dense SIFT features. The video sequence is divided into multiple segments by following the approach proposed by [84] where an image is segmented into many clusters using multiple stable segmentation. The segments are obtained by varying the parameters of a normalized cut. In the case of a video sequence, the weight matrix for the normalized cut is defined to capture similarity between frames. To restrict the segments to contain only contiguous frames, the similarity between each frame was defined to incorporate the distance between the time index of two frames along with their feature similarity. Recall that each cluster of frames is treated as an instance under MIL setting. Hence, the spatial pyramid features of each frame within a segment are max-pooled to compute the instance feature.

We have followed the protocol used by previous work to report the total classification rate computed at Equal Error Rate (EER) on the receiver operation curve

(ROC). Our results are summarized in Table 3.4 which were conducted using a leave-one-subject-out cross validation strategy. The numbers for the competing methods have been quoted from [73]. For each split of training and testing data, training data contained video sequences from all but one subject while the testing data contained the video sequences from the left out subject. Thus, there was no overlap between subjects in training and testing video sequences. In this experiment, we learned dictionaries with 40 atoms, sparsity parameter λ was set equal to 0.001 and discriminative parameter was set equal to 1. This parameters were slightly optimized for the performance on one of the splits (i.e. for one subject) and then the same parameters were used for all the data splits. Since the bag size varies a lot in this dataset, we use the GM model to reduce bias of bag size.

Algorithms	Accuracy (at EER)
ML-SVM$_{\text{avg}}$	70.75
ML-SVM$_{\text{max}}$	76.19
[83]	81.21
[83] (shown by [82])	68.31
[82]	80.99
[73]	83.7
DKSVD* [29]	77.01
LC-KSVD* [30]	79.34
GD-MIL	**88.18**

Table 3.4: Classification accuracy (at EER) on pain dataset [76].

To qualitatively evaluate our method, we compute the frame score from instance probabilities using the approach proposed by [73]. Let the set of frames that constitute feature $\mathbf{y}_{ij}$ be denoted by s_{ij}. The instance probability p_{ij} is distributed to all the frames contained in s_{ij} by employing a Hamming window. If a frame belongs to multiple segments, then its score is computed as the maximum from all

the segments. If the k^{th} frame in the i^{th} video sequence is denoted by f_i^k, its score $p_{f_i^k}$ is computed as,

$$p_{f_i^k} = \max_j (w(s_{ij}) * p_{ij} | f_i^k \in s_{ij}), \tag{3.41}$$

where w is a hamming window function centered at segment s_{ij} and $*$ is scalar multiplication. We plot these scores for multiple subject in Figure 3.8 along with face image to display facial expressions of key frames. Along with our score, we also plot the Prkachin and Solomon Pain Intensity (PSPI) score, described by [76], for each frame. In Figure 3.8(a), we show an instance of multiple pain occurrences in the video sequence. We are able to accurately localize the pain as shown by key face images as well as the corresponding PSPI score. In Figures 3.8(b) and (c), we plot the frame scores of a video sequence where it is localized at just one place. In Figure 3.8(b), the PSPI score is small compared to our frame. However, we can see a facial expression that corresponds to significant pain. Figure 3.8(d) displays a case where the intensity of pain around the frame index 300 is predicted much more than around frame index 100. Even facial expressions around frame index 300 seem to indicate less pain. However, we believe that the detection of pain with high intensity around frame 300 is due to large head movements. We provide multiple video sequences in the supplementary material to support our claim.

3.6.5 USPS digit experiment

We evaluate our method on the USPS digit dataset and provide detailed analysis of this experiment to gain some meaningful insights regarding our method. The

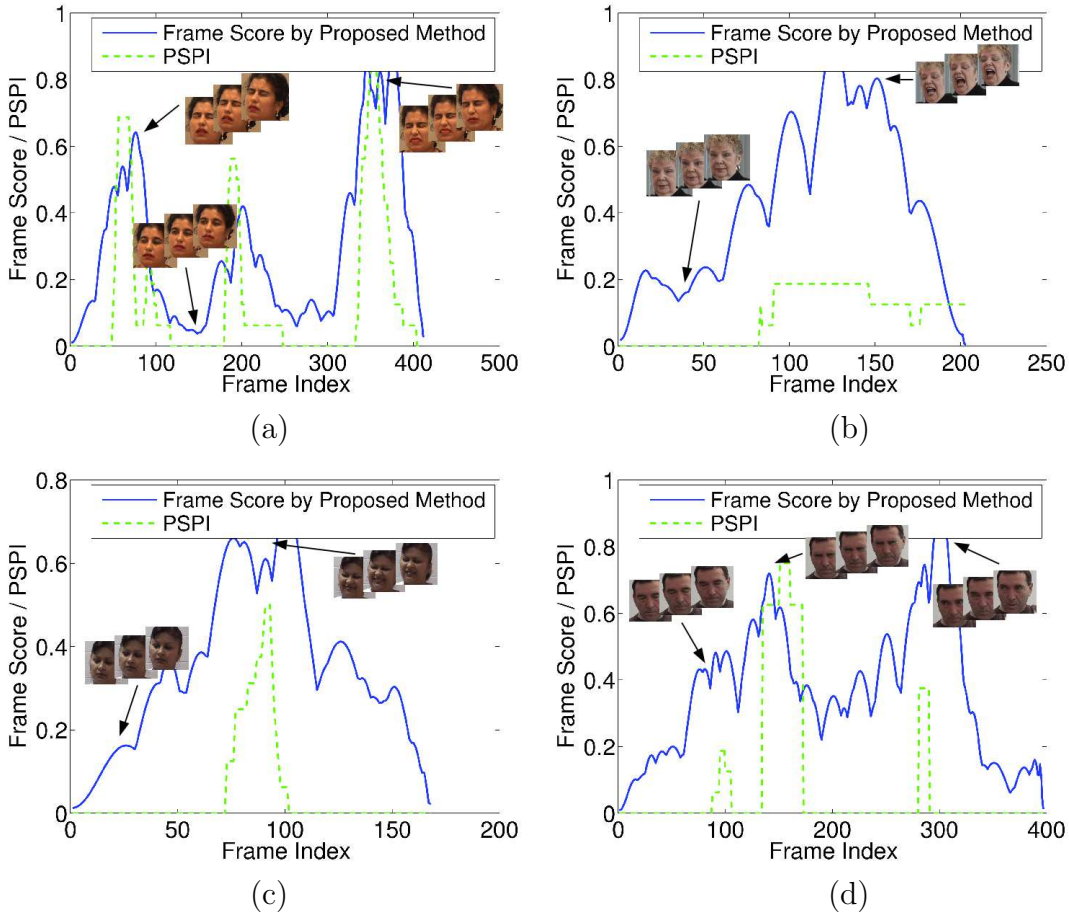


Figure 3.8: Frame score of multiple video sequences and comparison with PSPI rating. Please see text for details.

USPS digit dataset consists of total 9298 hand written digit images from 0 to 9. Each digit image is of size 16×16 pixels and raw pixels are used as features for all the methods compared in this chapter. To evaluate our method for the multi-class setting, we create 50 training bags for each class. Each training bag of class c consists of 4 instances out of which one is from the c^{th} class while the remaining 3 are randomly chosen from the other classes. Our test data consists of 2000 samples, 200 from each class. Furthermore, for a fair comparison with the other dictionary learning algorithms that do no use an explicit SVM, classification of digits is performed using the reconstruction error. Without learning the common structure present in

the positive class, reconstruction error-based classification method would not work well. As a result, a good classification accuracy suggests that the dictionary of each class would have learned the common internal structure present in the positive bags. Note that in this experiment, we evaluated our method only on the instances because a test bag with samples from different classes would have an ambiguity in the ground truth class label.

We compare our method with three discriminative dictionary based algorithms - DKSVD [29], LC-KSVD [30] and FDDL [32] and one MIL-based algorithm mi-SVM [63] with polynomial kernel of degree 4, the same as our method. For the discriminative dictionary learning algorithms, each training instance is given the class of the bag. As can be seen from Table 3.5, these algorithms do not perform well because the labels are very noisy. To gain additional insight, we plot the pre-images of the dictionary atoms of the GD-MIL method in Figure 3.9(a) and compare them with the dictionary atoms of the FDDL method in Figure 3.9(b). The pre-image of $\Phi(\mathbf{Y})\mathbf{a}_k$ is obtained by seeking a vector $\mathbf{d}_k \in \mathbb{R}^d$ in the input space that minimizes the cost function $\|\Phi(\mathbf{d}_k) - \Phi(\mathbf{Y})\mathbf{a}_k\|_2$. Due to various noise effects and the generally non-invertible mapping Φ, the exact pre-image does not always exist. However, the approximated pre-image can be reconstructed without venturing into the feature space using the techniques described in [52]. Note that the DKSVD method and the LC-KSVD method do not label the dictionary atoms, hence, we compare our method only with the FDDL method. However, we believe without considering the noise in bags, it is difficult to learn the common structure present in positive class. As can be seen from Figure 3.9, our dictionary atoms look very

similar to the digits for the corresponding classes, compared to the FDDL dictionary atoms that look very noisy due to the label noise.

Algorithms	Accuracy (%)
DKSVD [29]	56.4
LCKSVD [30]	37.4
FDDL [32]	44.4
mi-SVM [63]	78.7
GD-MIL	**83.4**

Table 3.5: Classification accuracy (%) on the USPS digit dataset.

(a) (b)

Figure 3.9: Visualization of the dictionary atoms learned on the USPS digit dataset. (a) Dictionary atoms of the GD-MIL method and (b) the FDDL method. Each row corresponds to the dictionary atoms of a class, i.e. digits from 0 to 9.

Furthermore, to compute the "upper bound" of the proposed method, we compute the classification accuracy of the three dictionary learning algorithms in the absence of any label noise. That is, noisy labels from the positive bags are removed before learning the dictionaries. The performance of the dictionary learning algorithms without any label noise has been presented in Table 3.6. As can be seen from this table, our algorithm is able to perform quiet close to this empirical "upper bound" despite significant label noise.

Algorithms	Accuracy (%)
DKSVD [29]	86.7
LCKSVD [30]	84.9
FDDL [32]	87.2

Table 3.6: Classification accuracy (%) on the USPS digit dataset without the label noise. This can be considered as an empirical "upper bound" for the proposed method.

3.6.6 MSR2 Action Recognition

The MSR2 action dataset has in total 54 video sequences and each video sequence consists of one or more of the following three actions: (1) Clapping, (2) Hand Waving and (3) Boxing. We randomly select 27 videos for training and the remaining ones for testing. Each action sample is a spatio-temporal cuboid and the most of the video sequences have just one or two such action cuboids per class. For each action cuboid, we added two more cuboids with the same spatial co-ordinates overlapped by 50% in the temporal dimension. Most of the bags of class c contained 2 action cuboids of the c^{th} class and 1 from a different class. The exact number of instances in each bag varies depending on the action cuboids of its class present in the video sequence. To compute the features for each action cuboid, we use bag-of-words of dense spatial temporal interest points (STIP) features [85]. Similar to the USPS digit experiment, we compare our method with three discriminative dictionary learning algorithms and one MIL algorithm in Table 3.7. As we can see from this table, the performance improves significantly by considering the MIL structure of the bag instead of relying only on the discriminative capability of the dictionary learning algorithm. Furthermore, we also compute the classification accuracy without any label noise in the training bags in Table 3.8. As can be seen, the accuracy of the

proposed method is within 4% of this "upper bound".

Algorithms	Accuracy (%)
DKSVD [29]	65.9
LCKSVD [30]	71.8
FDDL [32]	72.3
mi-SVM [63]	75.7
GD-MIL	**79.3**

Table 3.7: Classification accuracy (%) on the MSR2 action dataset.

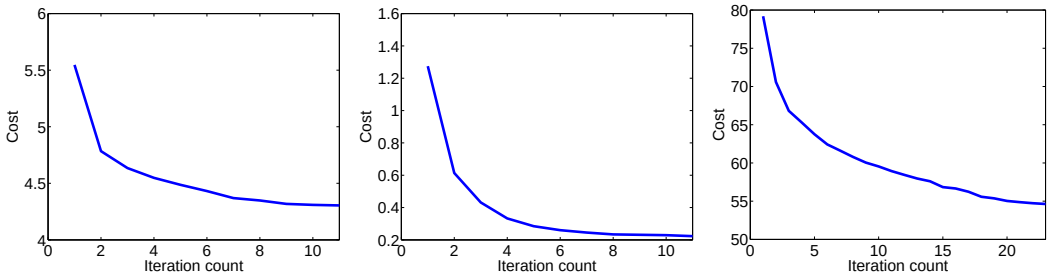
Algorithms	Accuracy (%)
DKSVD [29]	82.6
LCKSVD [30]	83.1
FDDL [32]	80.5

Table 3.8: Classification accuracy (%) on the MSR2 action dataset without label noise. This can be considered as an empirical "upper bound" for the proposed method.

3.6.7 Timing and Convergence of the proposed method

As summarized in Algorithm 3, the proposed algorithm iteratively updates the dictionary and the coefficient matrix. Updating a dictionary involves minimizing a smooth function while a coefficient matrix is updated by minimizing a smooth cost along with the ℓ_1 regularizer. Hence, a legitimate question of convergence of the cost arises. To show the empirical convergence of our method, we plot the cost in (3.19) as a function of iterations for some of the experiments with different datasets in Figure 3.10. As can be seen from these figures, the proposed method converges in a few iterations.

The training time depends on the number of atoms and the training data. We implemented our method in MATLAB on a 8 core computer with 8GB RAM. The code can be made more efficient by implementing it in C/C++. With the



(a) (b) (c)

Figure 3.10: Empirical convergence of cost for multiple experiments. (a) Tiger dataset (b) Corel1000 dataset (c) Pain dataset.

current implementation in MATLAB, the training and testing times on the USPS digits experiment are given in Table 3.9. We compare the proposed method with the kernel mi-SVM method which uses a highly optimized C/C++ implementation of the SVM library. In our method, the main computation time is taken by the gradient descent algorithm for the atoms update step. Note that compared to the mi-SVM algorithm, our method is efficient at the test time.

Algorithms	Training Time (sec)	Test Time (sec)
mi-SVM	442	8.2
GD-MIL	784	2.4

Table 3.9: Timing comparisons of the proposed method and the mi-SVM method on the USPS digit dataset.

3.7 Conclusion

We proposed a general diverse density-based dictionary learning method for multiple instance learning. Two DD-based approaches were proposed for learning dictionaries. It was shown that special case of our method reduces to a novel discriminative dictionary learning formulation. Furthermore, the non-linear extension of dictionary learning for MIL were presented. An efficient algorithm was proposed

for updating each atom of the dictionary and sparse coefficients of the instances. Experiments on the standard MIL datasets and a pain dataset demonstrate the effectiveness of the proposed method.

Chapter 4: Multiple Kernel Learning for Sparse Representation-based Classification

4.1 Introduction

It has been shown that sparse representation works well in many inverse problems where the original signal $\mathbf{y}_t$ needs to be reconstructed as accurately as possible, such as denoising, deconvolution and image inpainting. Sparse representation framework has also been used for signal classification tasks [17], [18], [2], [19], [20]. In particular, Sparse Representation-based Classification (SRC) algorithm [18] has gained a lot of attraction in recent years. This is mainly due to the fact that it is robust to noise and occlusion [18], [86].

The SRC method is based on finding a linear representation of the data. However, linear representations are almost always inadequate for representing non-linear structures of the data which arise in many practical applications. To deal with this problem, non-linear SRC methods have been proposed in the literature [69], [87]. These algorithms essentially map the non-linear data into high-dimensional feature space using the kernel trick so that data of the same distribution are easily grouped together and are linearly separable. This may also allow one to easily find the sparse

representation of data and significantly reduce the reconstruction error [60], [88]. Kernel SRC methods have shown to produce better classification results than the traditional SRC.

Kernel SRC methods [69], [87] require the use of a predetermined kernel function such as the polynomial kernel or the Gaussian kernel. Selection of the kernel function and its parameters is an important issue in training when kernel SRC methods are used for classification. In general, cross validation is used to choose the best kernel function among a set of kernel functions. Recently, Multiple Kernel Learning (MKL) methods that allow one to use multiple kernels instead of using a specific kernel function have been proposed in the literature [89].

In this chapter, we propose a kernel sparse representation-based classification method based on MKL where multiple kernel functions are combined to obtained a better solution. Our method uses a two step training method using the SRC as the base learner. At each iteration, first the combination function parameters are updated while fixing the base learner parameters, and then the base learner parameters are updated while fixing the combination function parameters. These two steps are repeated until convergence. Fig. 4.1 presents an overview of our method.

4.1.1 Organization of the chapter

This chapter is organized as follows. In Section 4.2, we review some related work on SRC, kernel SRC and MKL. Details of our MKL-based SRC method are

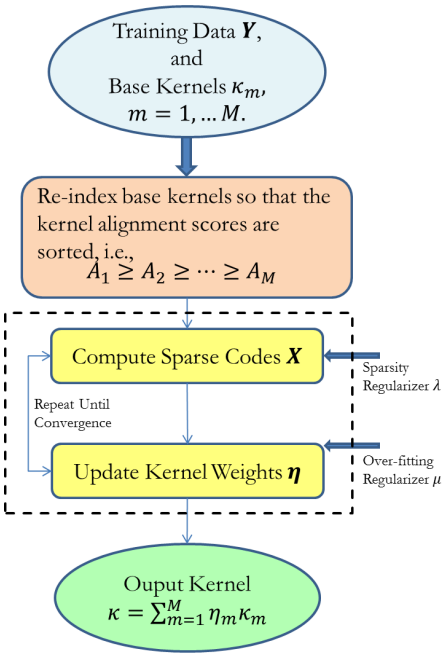


Figure 4.1: Overview of the proposed method.

given in Section 4.3. Experimental results are presented in Section 4.4 and Section 4.5 concludes the chapter with a brief summary and discussion.

4.2 Background

In this section, we review some related work on SRC, kernel SRC and MKL.

4.2.1 Sparse Representation-based Classification

Suppose that we are given C distinct classes and a set of N_c training images per class. We identify an $l \times p$ grayscale image as a d-dimensional vector which can be obtained by stacking its columns. Let $\mathbf{Y}_c = [\mathbf{y}_1^c, \ldots, \mathbf{y}_{N_c}^c] \in \mathbb{R}^{d \times N_c}$ be the matrix of training images from the cth class. Define a new matrix, $\mathbf{Y}$, as the concatenation

of training samples from all the classes as

$$\begin{aligned}\mathbf{Y} &= [\mathbf{Y}_1, \ldots, \mathbf{Y}_C] \in \mathbb{R}^{d \times N} \\ &= [\mathbf{y}_1^1, \ldots, \mathbf{y}_{N_1}^1 | \mathbf{y}_1^2, \ldots, \mathbf{y}_{N_2}^2 | \ldots\ldots | \mathbf{y}_1^C, \ldots, \mathbf{y}_{N_C}^C] \\ &\triangleq [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N],\end{aligned}$$

where $N = \sum_c N_c$. We consider an observation vector $\mathbf{y}_t \in \mathbb{R}^d$ of unknown class as a linear combination of the training vectors as

$$\mathbf{y}_t = \sum_{c=1}^{C} \sum_{i=1}^{N_c} x_i^c \mathbf{y}_i^c \tag{4.1}$$

with coefficients $x_i^c \in \mathbb{R}$. The above equation can be more compactly written as

$$\mathbf{y}_t = \mathbf{Y}\mathbf{x}, \tag{4.2}$$

where

$$\begin{aligned}\mathbf{x} &= [x_1^1, \ldots, x_{N_1}^1 | x_1^2, \ldots, x_{N_2}^2 | \ldots | x_1^C, \ldots, x_{N_C}^C]^T \\ &\triangleq [x_1, x_2, \ldots, x_N]^T\end{aligned} \tag{4.3}$$

and $.^T$ denotes the transposition operation. One can make an assumption that given sufficient training samples of the cth class, $\mathbf{Y}_c$, any new test image $\mathbf{y}_t \in \mathbb{R}^d$ that belongs to the same class will approximately lie in the linear span of the training samples from the class c. This implies that most of the coefficients not associated with class c in (4.3) will be close to zero. As a result, assuming that observations are noisy, one can recover this sparse vector by solving the following optimization problem,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \qquad \text{subject to } \|\mathbf{y}_t - \mathbf{Y}\mathbf{x}\|_2 \leq \epsilon \tag{4.4}$$

or equivalently the following formulation,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{y}_t - \mathbf{Y}\mathbf{x}\|_2 + \lambda\|\mathbf{x}\|_1, \tag{4.5}$$

where λ is a parameter. The sparse code $\mathbf{x}_t$ can then be used to determine the class of $\mathbf{y}_t$ by computing the following error for each class,

$$e_c = \|\mathbf{y}_t - \mathbf{Y}_c\mathbf{x}_t^c\|_2, \tag{4.6}$$

where, $\mathbf{x}_t^c$ is the part of coefficient vector $\mathbf{x}_t$ that corresponds to $\mathbf{Y}_c$. Finally, the class c^* that is associated to the test sample $\mathbf{y}_t$, can be declared as the one that produces the smallest approximation error

$$c^* = \text{class of } \mathbf{y}_t = \arg\min_c e_c. \tag{4.7}$$

The SRC method was originally proposed for face biometric in [18]. It was then extended for cancelable iris biometric in [86] and for automatic target recognition in [90].

4.2.2 Kernel SRC

Many types of descriptors in computer vision such as the spatial pyramid descriptor and the region covariance descriptor have intrinsic nonlinear similarity measure functions. This has motivated researchers to develop non-linear kernel sparse representations for object representation and classification [69], [87], [60], [91], [92], [61], [42], [41].

In kernel SRC, essentially the idea is to map data in the high dimensional feature space and solve (4.5) using the kernel trick [52]. Let $\mathbf{\Phi} : \mathbb{R}^d \rightarrow G$ be a non-

linear mapping from d-dimensional space into a dot product space G. A non-linear SRC can be performed by solving the following optimization problem,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{4.8}$$

where

$$\mathbf{\Phi}(\mathbf{Y}) \triangleq [\mathbf{\Phi}(\mathbf{y}_1^1), \cdots, \mathbf{\Phi}(\mathbf{y}_{N_1}^1)| \cdots |\mathbf{\Phi}(\mathbf{y}_1^C), \cdots, \mathbf{\Phi}(\mathbf{y}_{N_C}^C)].$$

Denote the first term of (4.8) by $\mathcal{E}_\kappa$ as follows

$$\begin{aligned}
\mathcal{E}_\kappa(\mathbf{x}; \mathbf{Y}, \mathbf{y}_t) &= \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{x}\|_2^2 \\
&= \mathbf{\Phi}(\mathbf{y}_t)^T\mathbf{\Phi}(\mathbf{y}_t) + \mathbf{x}^T\mathbf{\Phi}(\mathbf{Y})^T\mathbf{\Phi}(\mathbf{Y})\mathbf{x} \\
&\quad - 2\mathbf{\Phi}(\mathbf{y}_t)^T\mathbf{\Phi}(\mathbf{Y})\mathbf{x} \\
&= \boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{y}_t) + \mathbf{x}^T\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})\mathbf{x} \\
&\quad - 2\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{Y})\mathbf{x},
\end{aligned} \tag{4.9}$$

where $\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y}) \in \mathbb{R}^{N\times N}$ is a positive semidefinite kernel Gram matrix whose elements are computed as

$$\begin{aligned}
[\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})]_{i,j} &= [\langle\mathbf{\Phi}(\mathbf{Y}), \mathbf{\Phi}(\mathbf{Y})\rangle]_{i,j} \\
&= \mathbf{\Phi}(\mathbf{y}_i)^T\mathbf{\Phi}(\mathbf{y}_j) = \kappa(\mathbf{y}_i, \mathbf{y}_j),
\end{aligned}$$

$\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{y}_t) = \kappa(\mathbf{y}_t, \mathbf{y}_t)$, and

$$\boldsymbol{\mathcal{K}}(\mathbf{y}_t, \mathbf{Y}) \triangleq [\kappa(\mathbf{y}_t, \mathbf{y}_1), \kappa(\mathbf{y}_t, \mathbf{y}_2), \cdots, \kappa(\mathbf{y}_t, \mathbf{y}_N)] \in \mathbb{R}^{1\times N}.$$

Here, $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the kernel function.

It is apparent that the objective function is feasible since it only involves a matrix of finite dimension. Furthermore, the computation of $\boldsymbol{\mathcal{K}}$ only requires dot

products. Therefore, we are able to employ Mercer kernel functions to compute these dot products without carrying out the mapping $\mathbf{\Phi}$. Some commonly used kernels include polynomial kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \langle (\mathbf{x}, \mathbf{y} \rangle + a)^b$$

and Gaussian kernels

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{c}\right),$$

where a, b and c are the parameters. Note that κ in the subscript of $\mathcal{E}_\kappa$ stresses on the fact that the error term depends on the choice of the kernel function. With the above definitions, the kernel version of the SRC optimization problem in (4.5) can be written as,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \mathcal{E}_\kappa(\mathbf{x}; \mathbf{Y}, \mathbf{y}_t) + \lambda \|\mathbf{x}\|_1. \tag{4.10}$$

One can solve the optimization problem (4.10) by modifying the LARS algorithm [93]. In the case, when ℓ_1-norm is replaced by the ℓ_0-norm in (4.10), kernel orthogonal matching pursuit algorithm can be used to solve the resulting optimization problem [91], [60], [94].

4.2.3 Multiple Kernel Learning

In order to achieve good performance in object classification tasks, it is important to combine inputs from various image features. The large margin classifiers as well as many other classifiers in computer vision are constructed based on similarity measures between samples (or kernels). Finding appropriate feature combinations

entails designing good kernel functions among a set of candidate kernels. One way to achieve this is by finding positive mixtures of predetermined base kernels. MKL is a theoretically and technically very attractive way of determining the mixing weights of multiple kernels [89], [95], [96], [97], [98], [99]. This method have also been used to combine multiple features as explored by [100] and object detection [101]. MKL learns the kernel weights and the classifier simultaneously.

Let $\kappa_1, \cdots, \kappa_M$ be a set of base kernel functions that would be used to compute kernel matrices. In the MKL framework, linear combinations of the base kernels are considered

$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \sum_{m=1}^{M} \eta_m \kappa_m(\mathbf{y}_i, \mathbf{y}_j),$$

and the mixing coefficients η_m are learned together with the model parameters, so as to maximize the classification ability [89]. Various MKL algorithms have been proposed in the literature that essentially differ in the training method, the base learner, the functional form or the learning method [89].

For example, one can obtain a valid kernel by taking the summation or multiplication of M valid kernels

$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \sum_{m=1}^{M} \kappa_m(\mathbf{y}_i, \mathbf{y}_j)$$
$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \prod_{m=1}^{M} \kappa_m(\mathbf{y}_i, \mathbf{y}_j).$$

One can also select the kernel weights based on the performance of each kernel. The following rule is proposed in [102] for the selection of kernel weights

$$\eta_m = \frac{\xi_m - \delta}{\sum_{l=1}^{M}(\xi_l - \delta)},$$

where δ is the threshold that should be less than or equal to the minimum of the accuracies obtained from single-kernel learners and ξ_m is the accuracy obtained using only $\boldsymbol{\mathcal{K}}_m$.

The notion of of kernel alignment which is a measure of similarity between two kernel functions or between a kernel and a target function was introduced in [103]. Let $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$ be the Gram matrices of kernel functions κ_1 and κ_2 for a set $\{\mathbf{y}_i\}_{i=1}^N$ of the inputs. Then, the alignment score A between the kernel matrices $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$ is defined,

$$A(\boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_2) = \frac{\langle \boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_2 \rangle_F}{\sqrt{\langle \boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_1 \rangle_F \langle \boldsymbol{\mathcal{K}}_2, \boldsymbol{\mathcal{K}}_2 \rangle_F}}, \tag{4.11}$$

where,

$$\langle \boldsymbol{\mathcal{K}}_1, \boldsymbol{\mathcal{K}}_2 \rangle_F = \sum_{i=1}^{N} \sum_{j=1}^{N} \kappa_1(\mathbf{y}_i, \mathbf{y}_j) \kappa_2(\mathbf{y}_i, \mathbf{y}_j).$$

One can view kernel alignment as the cosine of the angle between $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$. Kernel alignment can be used to select the kernel weights. In [104], the following approach is used for selecting the kernel weights

$$\eta_m = \frac{A(\boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}d})}{\sum_{l=1}^{M} A(\boldsymbol{\mathcal{K}}_l, \boldsymbol{\mathcal{K}d})} \quad \forall m,$$

where $\boldsymbol{\mathcal{K}d} \in \mathbb{R}^{N \times N}$ is the ideal kernel matrix whose elements are defined as follows

$$[\boldsymbol{\mathcal{K}d}]_{(i,j)} = \begin{cases} 1, & \text{if } \mathbf{y}_i \in \text{class } c \text{ and } \mathbf{y}_j \in \text{class } c \\ 0, & \text{otherwise.} \end{cases} \tag{4.12}$$

In other words, $\boldsymbol{\mathcal{K}d}$ is a block diagonal matrix which has 1's where rows and columns correspond to the same class and 0's everywhere else. Suppose that we are given 3 classes and 2 samples per class, e.g., $\mathbf{Y}_{ex} = [\mathbf{y}_1^1, \mathbf{y}_2^1, \mathbf{y}_1^2, \mathbf{y}_2^2, \mathbf{y}_1^3, \mathbf{y}_2^3]$, then the resulting ideal matrix is the following block diagonal matrix

$$\boldsymbol{\mathcal{K}}\boldsymbol{d}(\mathbf{Y}_{ex}, \mathbf{Y}_{ex}) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{4.13}$$

A method for updating Gram matrices based on optimizing the alignment $A(\boldsymbol{\mathcal{K}}, \boldsymbol{\mathcal{K}}\boldsymbol{d})$ was proposed in [103], where the definition of the ideal kernel was slightly different. Many other methods have been proposed in the literature that use kernel alignment or a variation of kernel alignment for learning the kernel weights. See [89] for an excellent survey of different MKL algorithms.

4.3 Multiple Kernel Learning for SRC

In this section, we first present our formulation for Multiple Kernel Learning for SRC (MKL-SRC). We then present the details of the optimization algorithm.

4.3.1 Problem Formulation

If we use the training matrix $\mathbf{Y}$ to predict the class of a training sample, then the sparse code will always be all zeros but a single 1 at the location corresponding to the training sample under consideration. This sparse code will always correctly classify all the training samples and, hence, will not help in computing the optimal kernel. In order to avoid this degenerate case, we set the corresponding column of $\mathbf{Y}$ to $\mathbf{0}$ before computing the sparse code. This can be done as follows

$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \mathcal{E}_{\kappa}(\mathbf{x}_i; \tilde{\mathbf{Y}}_i, \mathbf{y}_i) + \lambda \|\mathbf{x}_i\|_1, \tag{4.14}$$

where,

$$\tilde{\mathbf{Y}}_i = [\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathbf{0}, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_N], \tag{4.15}$$

and $\mathbf{0}$ is a d-dimensional vector with zeros as its entries. We stack up all the sparse vectors $\mathbf{x}_i$ in columns of a matrix $\mathbf{X}$, i.e., $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times N}$. Now, in order to learn the optimal kernel κ, we write it as linear combination of M base kernels as follows

$$\kappa(\mathbf{y}_i, \mathbf{y}_j) = \sum_{m=1}^{M} \eta_m \kappa_m(\mathbf{y}_i, \mathbf{y}_j), \tag{4.16}$$

where η_m is the weight of the mth base kernel and $\sum_{m=1}^{M} \eta_m = 1$. Using (4.16), $\mathcal{E}_\kappa$ can be written as,

$$\begin{aligned}
\mathcal{E}(\mathbf{x}, \boldsymbol{\eta}; \tilde{\mathbf{Y}}_i, \mathbf{y}_i) &= \sum_m \eta_m \boldsymbol{\mathcal{K}}_m(\mathbf{y}_i, \mathbf{y}_i) \\
&\quad + \mathbf{x}^T \Big(\sum_m \eta_m \boldsymbol{\mathcal{K}}_m(\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}}) \Big) \mathbf{x} - 2 \sum_m \eta_m \boldsymbol{\mathcal{K}}_m(\mathbf{y}_i, \tilde{\mathbf{Y}}) \mathbf{x} \\
&= \sum_m \eta_m \Big(\boldsymbol{\mathcal{K}}_m(\mathbf{y}_i, \mathbf{y}_i) + \mathbf{x}^T \boldsymbol{\mathcal{K}}_m(\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}}) \mathbf{x} \\
&\quad - 2 \boldsymbol{\mathcal{K}}_m(\mathbf{y}_i, \tilde{\mathbf{Y}}) \mathbf{x} \Big),
\end{aligned}$$

where $\boldsymbol{\mathcal{K}}_m(\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}})$ can be computed by setting ith row and ith column of $\boldsymbol{\mathcal{K}}(\mathbf{Y}, \mathbf{Y})$ to zeros, and $\boldsymbol{\mathcal{K}}_m(\mathbf{y}_i, \tilde{\mathbf{Y}})$ by setting ith column of $\boldsymbol{\mathcal{K}}(\mathbf{y}_i, \mathbf{Y})$ to zero. Let the kernel mixing coefficients vector be denoted by $\boldsymbol{\eta}$, i.e., $\boldsymbol{\eta} := \Big[\eta_1, \ldots, \eta_M \Big]^T$. Note that we have dropped the subscript κ and added the variable $\boldsymbol{\eta}$ to stress the dependency of the cost on the coefficients $\boldsymbol{\eta}$. In order to jointly learn the optimal sparse codes $\hat{\mathbf{X}}$ and the kernel function coefficients $\hat{\boldsymbol{\eta}}$, the following MKL optimization problem

needs to be solved,

$$\hat{\mathbf{X}}, \hat{\boldsymbol{\eta}} = \arg\min_{\mathbf{X},\boldsymbol{\eta}} \sum_{i=1}^{N} \mathcal{E}(\mathbf{x}, \boldsymbol{\eta}; \tilde{\mathbf{Y}}_i, \mathbf{y}_i) + \lambda \|\mathbf{x}_i\|_1$$
$$\text{subject to} \quad \sum_{m=1}^{M} \eta_m = 1, \text{and } \boldsymbol{\eta} \geq 0. \tag{4.17}$$

To optimize (4.17), one can alternate between solving for $\mathbf{X}$ with fixed $\boldsymbol{\eta}$ and, then, solving for $\boldsymbol{\eta}$ while keeping $\mathbf{X}$ fixed. With fixed $\boldsymbol{\eta}$, the optimization problem reduces to the standard kernel SRC which can be solved by using LARS [93] type of algorithm. However, while solving for $\boldsymbol{\eta}$ (with fixed $\mathbf{X}$), the problem reduces to a linear programming (LP) problem and has the following two shortcomings:

1. The solution of the optimization problem finds the kernel that reduces the reconstruction error of each sample but does not necessarily classify them in correct classes.

2. The LP finds a solution at the vertex, which, in our problem, lies on the axes. As a result, the optimization chooses just one kernel at each iteration and this choice of kernel keeps changing over iterations. This makes the algorithm very unstable.

In order to avoid these issues, we propose a kernel alignment-based algorithm for kernel learning that focuses on classification error of the training samples. To this end, our goal at the kernel learning stage is to learn the optimal kernel function κ that results in the maximum training classification accuracy while avoiding over-fitting. We first explain how this is done to avoid over-fitting. Then, we describe our algorithm for computing the weights $\boldsymbol{\eta}$.

4.3.2 Ordered Kernel Alignment Scores

We rank each base kernel based on how close the corresponding kernel matrix of the training data is to the ideal kernel matrix $\boldsymbol{\mathcal{K}}\boldsymbol{d} \in \mathbb{R}^{N\times N}$ that we defined in (4.12). To avoid over-fitting, we give preference to a kernel matrix $\boldsymbol{\mathcal{K}}$ that is "closer" to the ideal matrix.

The notion of closeness between two kernel matrices is defined in terms of kernel alignment criterion in (4.11). Kernel alignment score between a base kernel matrix $\boldsymbol{\mathcal{K}}_m$ and the ideal kernel matrix can be computed as

$$A_m(\boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}}\boldsymbol{d}) = \frac{\langle \boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}}\boldsymbol{d} \rangle_F}{N\sqrt{\langle \boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}}_m \rangle_F}}. \tag{4.18}$$

A kernel function κ_m whose corresponding kernel matrix $\boldsymbol{\mathcal{K}}_m$ gives higher alignment score with the ideal kernel matrix, is ranked higher. Without loss of generality (w.l.o.g.) we assume that the alignment scores of the base kernel functions $\kappa_1, \ldots, \kappa_M$ are sorted as follows,

$$A_1 \geq A_2 \geq \cdots \geq A_M. \tag{4.19}$$

The assumption is true w.l.o.g. because if the alignment scores are not sorted, we can re-index the base kernels so that they become sorted. Next, we explain how we compute the weights $\eta_m, m = 1, \ldots, M$, for all the base kernels based on the classification accuracy. Furthermore, we show how the ordering of kernels based on alignment scores helps to avoid over-fitting.

4.3.3 Computing Kernel Function Weights $\boldsymbol{\eta}$

Our MKL-SRC method alternates between learning sparse coefficients $\mathbf{X}$ and kernel function weights $\boldsymbol{\eta}$. Given sparse codes, we predict the labels of all the training samples. Let h_i be the predicted label of $\mathbf{y}_i$ using the current kernel function. To determine the prediction accuracy on the training samples, we define boolean variables $z_i \in \{0, 1\}, i = 1, \ldots, N$, that is set to 1 if the predicted labels of $\mathbf{y}_i$ is correct and 0 otherwise. That is,

$$z_i = \begin{cases} 1, & \text{if } h_i = l_i \\ 0, & \text{otherwise.} \end{cases} \tag{4.20}$$

We update the kernel weights $\boldsymbol{\eta}$ by adding a kernel that can help to classify the samples correctly where $z_i = 0$. We pre-compute the predicted labels of all the training samples for each base kernel. Let g_i^m be the predicted label of the ith sample with the mth base kernel and let,

$$z_i^m = \begin{cases} 1, & \text{if } g_i^m = l_i \\ 0, & \text{otherwise.} \end{cases} \tag{4.21}$$

We choose the base kernel m if its prediction error is the smallest among all the base kernel functions. At the same time, to avoid over-fitting, we want this chosen kernel to have high alignment score. This ensures that we do not choose a kernel just based on its training classification accuracy. In other words, taking the alignment score of the kernel functions into consideration ensures generalization capability of the classifier and, thus, avoids over-fitting. For the miss-classified samples, let the

accuracy for the mth base kernels be defined as

$$c_m = \frac{\sum_{\{i:z_i=0\}} z_i^m}{\sum_{i=1}^{N}(1-z_i)}. \tag{4.22}$$

We choose a kernel κ_{m^*}, if it gives the best accuracy among all the kernels that have lower alignment score than κ_{m^*}, and its accuracy c_{m^*} is at least better by μ than the accuracy of any kernel function that has higher alignment scores than that of κ_{m^*}. Formally,

$$\text{choose } k_{m^*} \text{ such that } \begin{cases} c_{m^*} \geq c_m, & \text{if } m \leq m^* \\ c_{m^*} \geq c_m + \mu, & \text{if } m > m^*. \end{cases} \tag{4.23}$$

The parameter μ controls the over-fitting by favoring a kernel function that has higher kernel alignment score. Note that the above choice of m^* in (4.23) gives the preference to the kernels with higher alignment score because they are assumed to be sorted in decreasing order. After choosing the kernel κ_{m^*}, we adjust the weights of the kernel functions in proportion to their respective accuracies. In order to compute the weights of the kernel functions, we consider only those samples whose labels are incorrectly predicted by either the current kernel or the chosen kernel. By the current kernel we mean the linear combination of all the kernel functions in the previous iteration. Hence, the weight of the new kernel is given by,

$$w_{newKernel} = \frac{\sum_{i=1}^{N}(z_i^{m^*}) \wedge (1-z_i)}{\sum_{i=1}^{N}(1-z_i) \vee (1-z_i^{m^*})}, \tag{4.24}$$

where $\wedge$ is a logical 'AND' operator and $\vee$ is a logical 'OR' operator. The numerator in (4.24) counts the number of samples where the new kernel predicts correct label while the current kernel does not. Similarly, the denominator counts the number

of samples where either the current kernel or the new kernel does not predict the correct label. Likewise, the weight for the current kernel is computed as,

$$w_{currKernel} = \frac{\sum_{i=1}^{N}(1 - z_i^{m^*}) \wedge (z_i)}{\sum_{i=1}^{N}(1 - z_i) \vee (1 - z_i^{m^*})} \tag{4.25}$$

where, the numerator counts the number of samples whose labels are correctly predicted by the current kernel but not by the new kernel. The current kernel is the linear combination of the kernels in the previous iteration. Let $\boldsymbol{\eta}^t = [\eta_1^t, \ldots, \eta_M^t]$ be the kernel weights at the tth iteration. Then, the weights for the $(t+1)$th iteration are computed as follows

$$\eta_m^{t+1} = \begin{cases} w_{newKernel} & \text{if } m = m^* \\ \eta_m^t * w_{currKernel} & \text{otherwise.} \end{cases} \tag{4.26}$$

The kernel weights are initialized such that all the weight is given to the kernel with highest alignment score, i.e. $\eta_1^0 = 1$ at the start of the first iteration. Finally, we divide each kernel weight η_m by the sum of all the weights

$$\eta_m \leftarrow \frac{\eta_m}{s}, \quad \text{where} \quad s = \sum_{m=1}^{M} \eta_m. \tag{4.27}$$

As an example, consider an illustrative example of 10 samples as shown in Fig. 4.2. The current kernel predicts correct labels of samples $\{1, 2, 3, 6, 7, 9, 10\}$, while the chosen kernel κ_{m^*} predicts correct class of the samples $\{1, 2, 5, 8, 9\}$. Since samples $\{1, 2, 9\}$ are predicted correctly by both the kernels, we consider samples $\{3, 4, 5, 6, 7, 8, 10\}$. Out of these 7 samples, current kernel predicts 4 correctly while, the new kernel predicts 2 correctly. Hence, $w_{currKernel} = 4/7$ and $w_{newKernel} = 2/7$.

Our approach for learning kernel weights is summarized in Algorithm 4.

Algorithm 4: Multiple Kernel Learning for SRC

Input: Data samples $\mathbf{Y}$, labels $\mathbf{l}$, kernel functions κ_m, parameters λ, μ, maximum iteration count T, ϵ_0.

Output: Kernel function weights $\boldsymbol{\eta}$.

For each kernel function κ_m and sample $\mathbf{y}_i$ compute the predicted label g_i^m, by computing the sparse code using (4.14).

Initialize $\epsilon_1 \leftarrow \epsilon_0 + 1, t \leftarrow 0$, and compute kernel matrices $\boldsymbol{\mathcal{K}}_m(\mathbf{Y}, \mathbf{Y})$.

while $t \leq T$ *and* $\epsilon_1 \geq \epsilon_0$ **do**

  for $i = 1, \ldots N$ **do**

    Compute $\boldsymbol{\mathcal{K}}_m(\tilde{\mathbf{Y}}_i, \tilde{\mathbf{Y}}_i)$ by setting the ith row and the ith column of $\boldsymbol{\mathcal{K}}_m(\mathbf{Y}, \mathbf{Y})$ to 0.

    Compute the sparse code $\mathbf{x}_i$ using (4.14).

    Compute the predicted label h_i using $\mathbf{x}_i$.

  end

  Update $\eta_m^t, \quad \forall m = 1, \ldots, M$ using (4.26).

  Compute the sum of all weights $s = \sum_{m=1}^{M} \eta_m$.

  $\eta_m \leftarrow \eta_m / s, \quad \forall m = 1, \ldots, M.$

  Set $\epsilon_1 \leftarrow \|\boldsymbol{\eta}^{t-1} - \boldsymbol{\eta}^t\|_2$.

  $t \leftarrow t + 1$

end

return $\boldsymbol{\eta}$

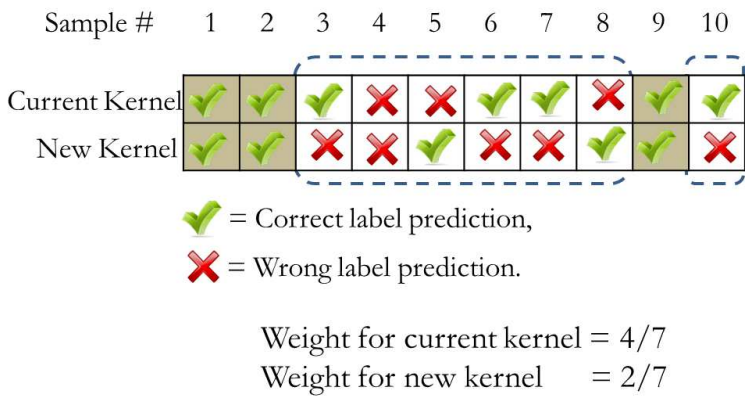


Figure 4.2: Updating kernel weights in each iteration.

4.3.4 Classification

In order to predict the class of a test sample $\mathbf{y}_t$, we compute the sparse code by optimizing the following problem,

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \mathcal{E}_\kappa(\mathbf{x}; \mathbf{Y}, \mathbf{y}_t) \tag{4.28}$$

where, the learned linear weights $\boldsymbol{\eta}$ are used to compute the kernel function κ. Then, the error for each class is computed as

$$e_c^\kappa = \mathcal{E}_\kappa(\mathbf{x}_t^c; \mathbf{Y}, \mathbf{y}_t). \tag{4.29}$$

Finally, the class of the sample $\mathbf{y}_t$ is the one that results in the minimum error

$$c_t^* = \text{class of } \mathbf{y}_t = \arg\min_c e_c^\kappa. \tag{4.30}$$

4.4 Experimental Results

In this section, we present several experimental results demonstrating the effectiveness of the proposed MKL-SRC method for classification tasks on both synthetic

and real datasets. In particular, we present classification results on the Caltech101 object dataset [105], University of Washington RGB-D dataset [106] and gender recognition on the AR Face dataset [107]. We compare the results of our method with that of SVM, linear SRC [18], kernel SRC [69], [87], and a multiple kernel learning algorithm based on SVM (SVM-MKL) [108].

For all the experiments, we use a total of 50 base kernels $\kappa_m(\mathbf{y}_i, \mathbf{y}_j)$ which are described below

1. Two linear kernels, $\mathbf{y}_i^T \mathbf{y}_j$ and $(1 + \mathbf{y}_i^T \mathbf{y}_j)$.

2. Fifteen polynomial kernels, $(a + \mathbf{y}_i^T \mathbf{y}_j)^b$ of degree $b = 2, 3, 4$, and constant $a = 0.5, 1.0, 1.5, 2.0, 2.5$.

3. Ten tangent hyperbolic kernels, $\tanh(c1 + c2 * (\mathbf{y}_i^T \mathbf{y}_j))$ with $(c1, c2) = (0.1, 1)$, $(0.2, 1)$, $(0.3, 1)$, $(0.4, 1)$, $(0.5, 1)$, $(0.5, 0.2)$, $(0.5, 0.4)$, $(0.5, 0.6)$, $(0.5, 0.8)$, $(0.5, 1)$.

4. One histogram intersection kernel.

5. Remaining 22 Gaussian kernels, $\exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{c}\right)$, with c from 0.1 to 2.2 in the steps of 0.1.

As for the parameter selection, we have only two parameters: sparsity regularizer λ and over-fitting regularizer μ. In all of our experiments, we set $\lambda = 0.01$ and $\mu = 0.05$ except for synthetic data where μ is set to 0.2.

4.4.1 Analysis on Synthetic Data

We thoroughly evaluate the basic behavior of the proposed MKL-SRC method on two two-dimensional synthetic data sets. In both of the synthetic experiments, we generate two classes of two dimensional data from the Gaussian distributions, with different means but the same covariance matrices. In the first experiment, the mean of class 1 is set to $[1,1]^T$ and that of class 2 to $[2,2]^T$. The covariance matrix for both of the classes is $\begin{bmatrix} 0.51 & 0.049 \\ 0.049 & 0.51 \end{bmatrix}$. This generates the samples which are approximately co-linear as shown in Fig. 4.3(a). To generate the test data in this experiment, we change the covariance matrix to $\begin{bmatrix} 0.51 & 0.01 \\ 0.01 & 0.51 \end{bmatrix}$ which results in the data as shown in Fig. 4.3(b). It is known that, with co-linear data, SRC algorithm does not work well because both classes can be represented almost equally well by the training samples of either class [69]. This results in a biased decision boundary as shown in Fig. 4.3(c) and gives very poor accuracy of only 62.40%. In order to remove the co-linearity of the data, one can represent the samples in a high dimensional feature space and perform classification using the kernel SRC. However, choosing a kernel based on classification accuracy of the training data alone, can result in over fitting and non-optimal choice of kernel, as shown in Fig. 4.3(d). The kernel is non optimal because it does not take into the consideration of the fact that test data might be slightly different from the training data, and hence results in over-fitting. On the other hand, the proposed method uses the alignment score of the kernel matrices and computes the best composite kernel that gives good alignment score

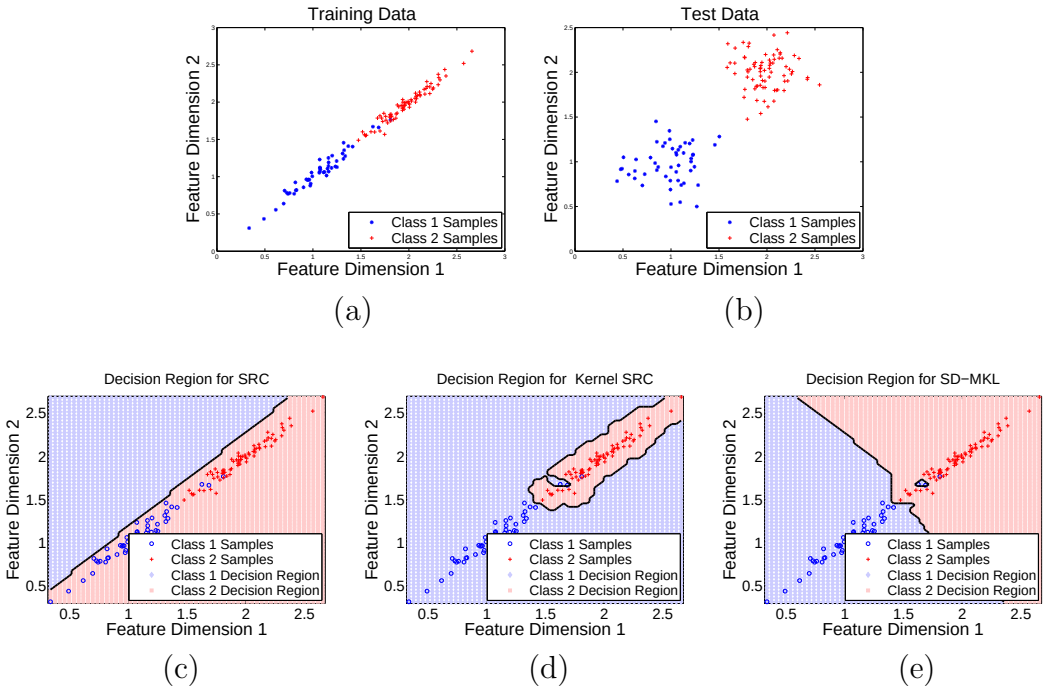


Figure 4.3: First synthetic dataset. (a) Training Data. (b) Test Data. Decision boundary with (c) SRC, (d) kernel SRC and (e) MKL-SRC.

as well as good training accuracy. The result is a classifier that can generalize well on slightly different test data as shown in Fig. 4.3(e) and results in 100% accuracy.

In the second synthetic experiment, we generate class 1 data from a Gaussian distribution with mean $[1, 1]^T$ and covariance matrix $\begin{bmatrix} 0.07 & 0.00 \\ 0.00 & 0.07 \end{bmatrix}$ and the second class using a Gaussian distribution with mean $[2, 2]^T$ and the same covariance matrix. The test data for class 1 is generated from the same Gaussian distribution as the training data, however, we slightly change the mean of class 2 for test data to $[2, 2.5]^T$. The training and the test data for this experiment are shown in Fig. 4.4(a) and (b), respectively. We show the decision boundary for SRC, kernel SRC and the proposed method in Fig. 4.4(c), (d) and (e), respectively. This experiment shows

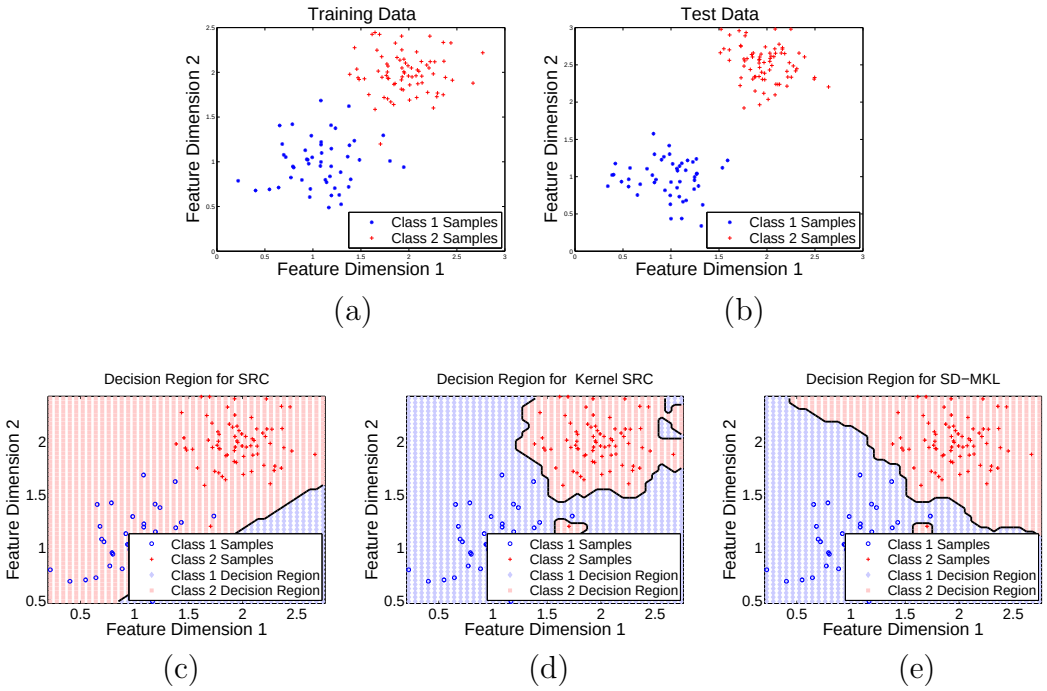


Figure 4.4: Second synthetic dataset. (a) Training Data, (b) Test Data. Decision boundary with (c) SRC, (d) kernel SRC, (e) MKL-SRC.

that learning a kernel that avoids over-fitting can result in better decision boundaries and hence better classification accuracy. Classification results on the synthetic data are summarized in Table 4.1.

	SRC	Kernel SRC	MKL-SRC
Synthetic data 1	62.40	80.00	**100.0**
Synthetic data 2	63.20	69.60	**99.20**

Table 4.1: Accuracy (%) on the synthetic data in Fig. 4.3 and 4.4.

4.4.2 Object Recognition

We perform the first set of object recognition experiments on the Caltech-101 database [105]. The Caltech101 dataset contains 102 categories including one background class. Each category has about 40 to 80 images and most of the categories have about 50 images. The images have been downloaded from the internet us-

ing Google search engine (www.google.com). The database contains a diverse and challenging set of images from buildings, musical instruments, animals and natural scenes, etc.

To show the appropriateness of sparsity in our application, we plot sparse coefficients when a test sample is represented as a sparse linear combination of training samples in the feature space. In particular, we randomly select five classes from the Caltech101 dataset to form a training matrix $\mathbf{Y}$ with fifteen samples from each class. Then, given a test sample $\mathbf{y}_t$ corresponding to one of the five classes from the Caltech101 dataset, we solve the following problem in the feature space using the polynomial kernel of degree two

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} \|\mathbf{\Phi}(\mathbf{y}_t) - \mathbf{\Phi}(\mathbf{Y})\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1. \tag{4.31}$$

We repeat this procedure fifteen times with different test samples corresponding to each class and take the average of sparse codes. We plot these spare representations in Figure 4.5. From this figure, we see that most of the coefficients are clustered around the class corresponding to the training samples. Furthermore, we ran multiple experiments with different kernel functions and obtained similar results. This essentially shows that on average the data used in our application does have a sparse representation in the feature space.

Following the common experimental set up on this dataset [30], we train on j images, where $j \in \{5, 10, 15, 20, 25, 30\}$, and test on the rest. For fair comparison, we use the same spatial pyramid features as used in [30]. Table 4.2 shows the comparison of our classification accuracy with the state-of-the-art. Note that

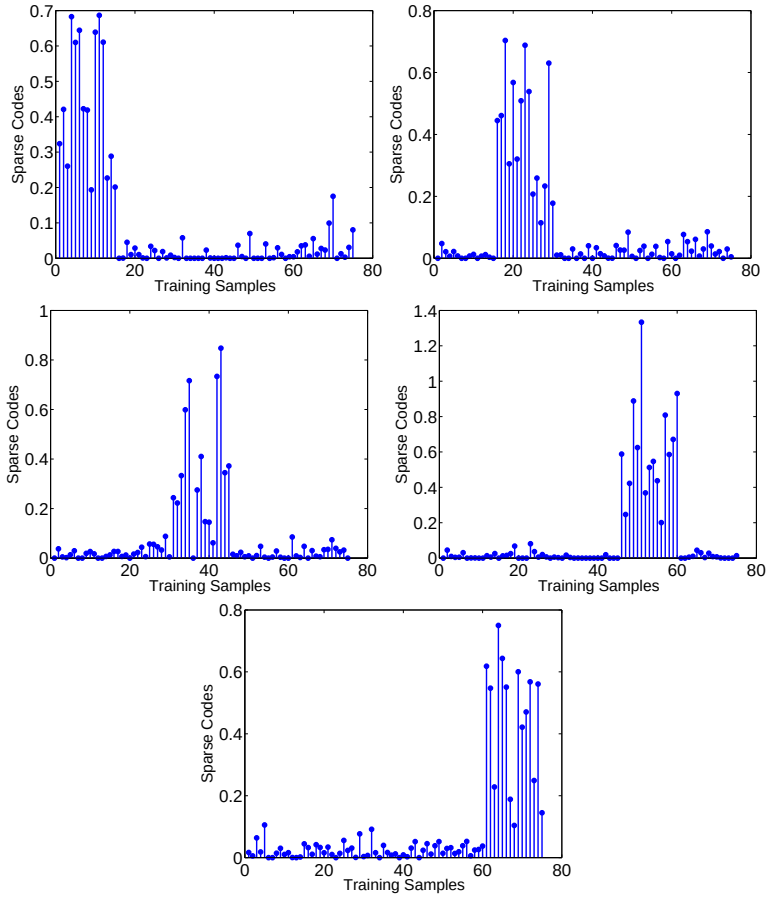


Figure 4.5: Sparse coefficients corresponding to five classes from the Caltech101 dataset in the feature space.

Number of training samples	5	10	15	20	25	30
Malik [109]	46.6	55.8	59.1	62.0	–	66.20
Lazebnik [110]	–	–	56.4	–	–	64.6
Griffin [111]	44.2	54.5	59.0	63.3	65.8	67.60
Irani [112]	–	–	65.0	–	–	70.40
Grauman [113]	–	–	61.0	–	–	69.10
Venkatesh [114]	–	–	42.0	–	–	–
Gemert [115]	–	–	–	–	–	64.16
Yang [116]	–	–	67.0	–	–	73.20
Wang [117]	51.15	59.77	65.43	67.74	70.16	73.44
K-SVD [118]	49.8	59.8	65.2	68.7	71.0	73.2
D-KSVD [29]	49.6	59.5	65.1	68.6	71.1	73.0
LC-KSVD1 [30]	53.5	61.9	66.8	70.3	72.1	73.40
LC-KSVD2 [30]	54.0	63.1	67.1	70.5	72.3	73.40
SVM-MKL [108]	51.2	62.4	67.1	69.8	72.7	74.6
Kernel SRC	50.4	60.8	66.5	69.2	72.0	74.1
SRC [18]	48.8	60.1	64.9	67.7	69.2	70.7
MKL-SRC	**54.6**	**64.9**	**69.3**	**72.0**	**74.2**	**75.7**

Table 4.2: Accuracy (%) on the Caltech 101 object recognition dataset.

our method performs significantly better than the linear SRC. Furthermore, it is interesting that our method outperforms the other discriminative approaches such as LC-KSVD and D-KSVD.

In Fig. 4.6, we plot the learned kernel weights for the experiment when 30 samples per class are used for training. As can be seen from this plot, our method is able to learn the optimal combinations of base kernels directly from data.

To further analyze our results, we plot the confusion matrix in Fig. 4.7(a) and per class accuracy in Fig. 4.7(b) in the case when 30 samples per class are used for training. There are in total 11 classes that result in 100% accuracy. These images are shown in Fig. 4.8.

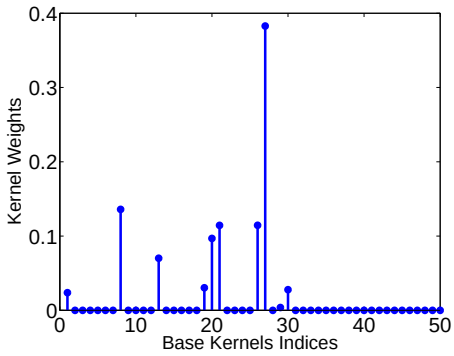
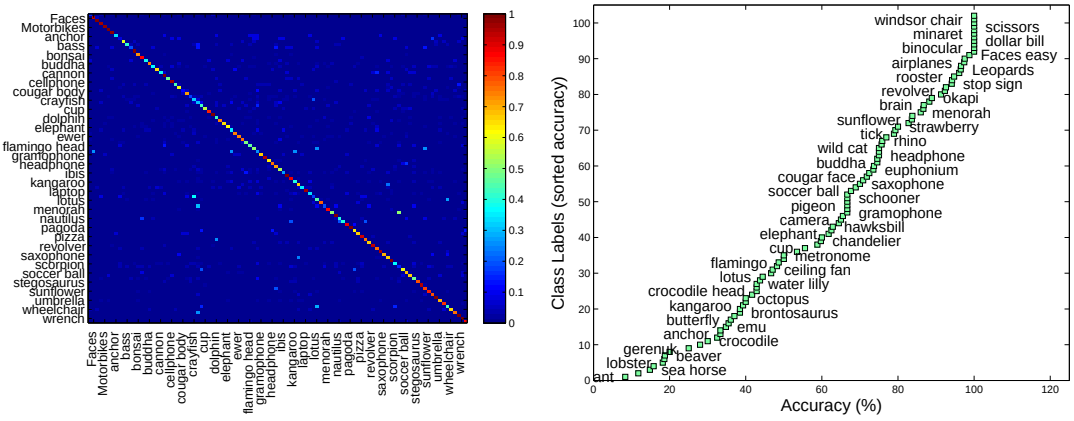


Figure 4.6: Learned kernel weights for the Caltech101 (for 30 training samples).



(a) (b)

Figure 4.7: Results on the Caltech 101 object dataset. (a) Confusion matrix. (b) Per class accuracy.

Figure 4.8: Example images from 11 categories of the Caltech101 dataset that achieve 100% accuracy. Category names (from left to right):accordion, binocular, car_side, dollar_bill, inline_skate, minaret, pagoda, scissors, trilobite, windsor_chair, wrench

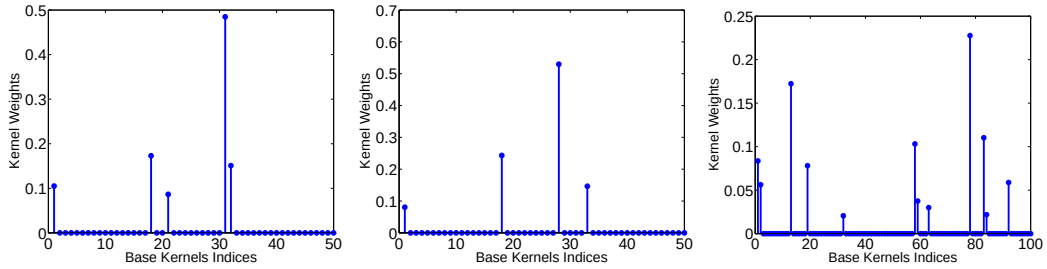
4.4.3 Object Recognition using Intensity and Depth Data

Recently, there has been a growing interest in using both the intensity and the depth data for computer vision algorithms. For example, with Microsoft's Kinect camera one can capture videos of both color as well as corresponding depth data. The purpose of this experiment is to demonstrate that our algorithm can naturally be extended to the multi-modal features. We use the same set of base kernels for intensity as well as depth images to compute the kernel matrices. This can be viewed as a single modality but with twice as many number of base kernels.

In this experiment, we use the RGB-D dataset of University of Washington [106] which consists of 51 categories. Few examples of pairs of color and depth images from this dataset are shown in Fig. 4.9. Most of the depth images are noisy

as shown in the second row of the figure. Hence, we apply a recursive median filter to remove the missing values. Processed images are shown in the third row of Fig. 4.9.

Figure 4.9: Example images of the RGBD dataset. First row shows the intensity images, second row displays the corresponding depth images, and the third row is the denoised version of the second row after applying the recursive median filter.



(a) (b) (c)

Figure 4.10: Learned kernel weights for RGBD dataset: (a) Intensity data. (b) Depth data. (c) Intensity and depth data.

We test our algorithm on the subset of the dataset by randomly selecting 10 images for training and 10 images for testing from each category. As is common in object recognition, we use bag-of-words (BoW) features for intensity as well as depth images. For the intensity images, we use BoW of 128 dimensional SIFT features [119] and 1000 clusters using the k-means algorithm for computing the bags. For the depth features, we compute the dense spin image [120] features on the depth data at each 3D point. We use the radius of 0.1 to compute neighbors at

each point, and bin size of 16 to compute the spin images. Finally, BoW features are computed for each depth image with 1000 bags computed using the k-means algorithm.

Comparison of all the methods using the intensity features alone are shown in the second column of Table 4.3. As can be seen from this column, our method performs more than 4% better than linear SRC. In this experiment, kernel SRC with non-optimal choice of the kernel performs a little worse than linear SRC. Next, we perform the same experiment on the depth data alone. Results of this experiment are summarized in the third column of Table 4.3. Our method performs slightly better than the other methods on the depth data as well. Next, we demonstrate that multiple features can improve the performance of the classifier. To this purpose, we compute BoW on HOG features and HSV color histograms of intensity images and evaluate our algorithm against SRC and kernel SRC. Again, our algorithm can naturally be extended to incorporate multiple features by computing kernel matrices for each of these features. With 3 features and 50 base kernels for each of them, we have total 150 kernel matrices and learned $\boldsymbol{\eta}$ is 150 dimensional weight vector. The accuracy with these features combined have been shown in the parentheses in Table 4.3.

Finally, we demonstrate how our algorithm can be extended when both the intensity and the depth features are available. For the non MKL based methods, we concatenate the intensity and the depth features for classification. This is equivalent to giving equal weights to both of the features. On the other hand, when we look at the kernel matrices from both the modalities, we can view them as twice as

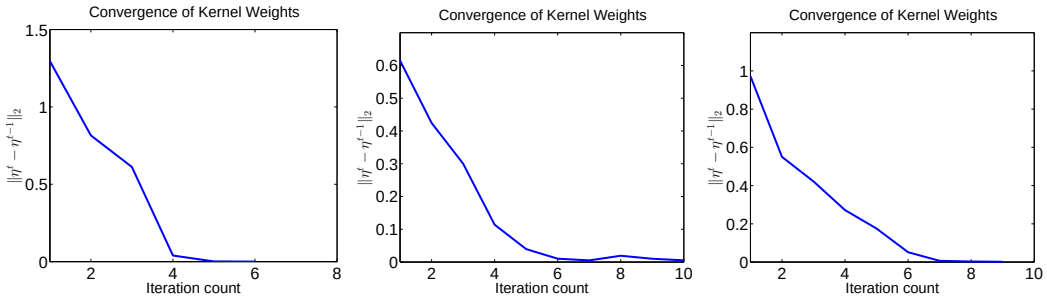
Algorithms	Intensity features	Depth features	Intensity and depth features
SVM	71.96	74.90	84.11
NN	69.80	75.88	86.08
SVM-MKL [108]	72.75	78.24	86.07
SRC	70.00(72.16)	79.80	86.27
Kernel SRC	69.80(72.54)	80.00	86.60
MKL-SRC	**74.12**(**75.88**)	**81.37**	**87.65**

Table 4.3: Accuracy (%) on the RGB-depth object dataset. In parentheses, we show the accuracy for multiple features (see text for details).

many base kernel matrices. Since in our case we use 50 base kernels, combined intensity and depth features results in 100 base kernel matrices. Learning weights for each kernel matrix automatically learns the optimal weights for the modalities. To further elaborate this point, we show the kernel weights in Fig. 4.10(a)-(c) for the intensity, depth, and their combination, respectively. Fig. 4.10(c) has 100 base kernel indices, out of which first 50 correspond to the intensity feature base kernels while the last 50 are for the depth feature base kernels. As can be seen from this figure, the weight given to the depth feature base kernels is more than the weights for the intensity features. This can be explained by the observation that the depth feature alone results in higher accuracy than the intensity feature. Results of the combined features are shown in the fourth column of Table 4.3. Again, our method performs better than the other methods on this combined dataset.

4.4.4 Gender Recognition

In the final set of experiments, we evaluate our algorithm on the gender recognition task. Towards this purpose, we use the AR Face dataset [107] that consists



(a) (b) (c)

Figure 4.11: Convergence of kernel weights ($\|\boldsymbol{\eta}^t - \boldsymbol{\eta}^{t-1}\|_2$): (a) Caltech 101 dataset. (b) RGBD dataset. (c) Gender recognition on AR dataset.

of 126 individuals with frontal faces captured in two sessions with different illuminations, expressions and occlusions. We choose 50 male subjects and 50 female subjects and 14 faces per subject from both sessions. Next, we train our algorithm, with first 25 males subjects and 25 female subjects and test our method with the remaining 25 male and 25 female subjects. The feature dimension was reduced to 300 using the principle component analysis. Comparison of our method with that of different methods is summarized in Table 4.4. Note that our method not only outperforms linear and non-linear SRC but it perform better than the state-of-the-art discriminative dictionary learning method such as [29].

SVM	NN	SVM-MKL [108]	DKSVD [29]	Kernel SRC	SRC	MKL-SRC
92.4	90.7	93.1	86.1	94.1	93.0	**95.4**

Table 4.4: Accuracy (%) on the gender recognition task using the AR face dataset.

4.4.5 On the Convergence of the Proposed Method

The proposed method iterates until the kernel weights converge or the maximum number of iterations are reached. So, a natural question about the convergence

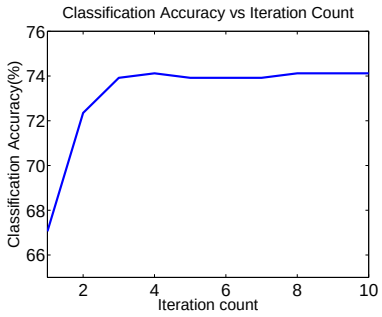


Figure 4.12: Improvement of classification accuracy over iterations (RGBD dataset using intensity features).

of the algorithm arises. Do the kernel weights converge and whether the classification accuracy actually improves over iterations? To answer these questions, let us closely look at what happens at each iteration. At every iteration, a new kernel is added only if it can correctly predict the labels of the subset of those samples that are incorrectly predicted using current kernel. In case of multiple choices, we pick the kernel with maximum accuracy. Intuitively, adding this new kernel should complement the current kernel and improve the accuracy. However, it is likely that the new kernel might wrongly predict the labels of those samples which are correctly predicted by the current kernel. Hence, we adjust the weight of the current kernel and the new kernel in proportion to the number of correctly predicted samples as explained in section 4.3 and illustrated in Fig. 4.2. Although, there is no theoretical guarantee that the combination of two kernels should improve upon the individual kernels, we empirically observe that combining the kernel as proposed improves the overall classification accuracy at each iteration, as shown in Fig. 4.12 for RGB-D dataset.

Note that the kernel coefficients are updated at each iteration only if a new

kernel complements the current kernel by correctly predicting the labels of the those samples where current kernel fails. As we add more kernels, the number of correctly predicted samples increases which, intuitively, results in reduced scope of further gain in later iterations. We can imagine that eventually, no kernel can correctly predict any significant subset of training data which is not already done by current kernel. This intuition is corroborated with experimental evaluation on all the three datasets. In Fig. 4.11 we plot $\|\boldsymbol{\eta}^t - \boldsymbol{\eta}^{t+1}\|_2$ and observe the quick convergence of kernel weights.

4.5 Conclusion

The SRC method works by computing the sparse coefficients of a test sample directly from the training data and does not require any prior training. However, for most of the applications, training can be useful provided that there is no overfitting. In this chapter, we have introduced a training stage to SRC that can learn the optimal kernel and improve the classification performance of SRC. The resulting algorithm alternates between learning sparse codes and kernel function weights.

Even though, in this chapter, we used a linear SRC as a base learner for MKL, it is possible to learn discriminative SRC and kernel weights simultaneously by adapting a discriminative SRC in our formulation. One can also adapt dictionary learning methods in our MKL formulation. It remains an interesting topic for future work to develop and analyze the accuracy of a discriminative dictionary learning-based MKL algorithm for classification.

Chapter 5: Class Consistent Multimodal Learning

5.1 Introduction

Combining information from multiple sources - multiple sensor modalities or multiple feature channels applied to a single sensor modality - is generally advantageous for recognition problems. For example, a self-driving car can better navigate its environment using multiple sensors including color cameras, depth sensors, inertial sensors, etc. Using both color and depth cameras, instead of either, can significantly improve the performance of computer vision tasks such as object categorization, detection, tracking, segmentation and others ([106], [121], [122]). In biometrics, fingerprints from multiple fingers can be used, or fingerprint and iris can be combined to determine identity. In this chapter, we consider classification by fusing information from multiple modalities and present an algorithm for multi-modal fusion by enforcing the intuitive constraint that the predicted class label should be consistent across all modalities.

Fusing multiple modalities for classification has been explored in many computer vision applications. These approaches can broadly be divided into three categories: (1) feature level fusion, (2) score level fusion, and (3) decision level fusion. In feature level fusion, features from multiple modalities are combined before feeding

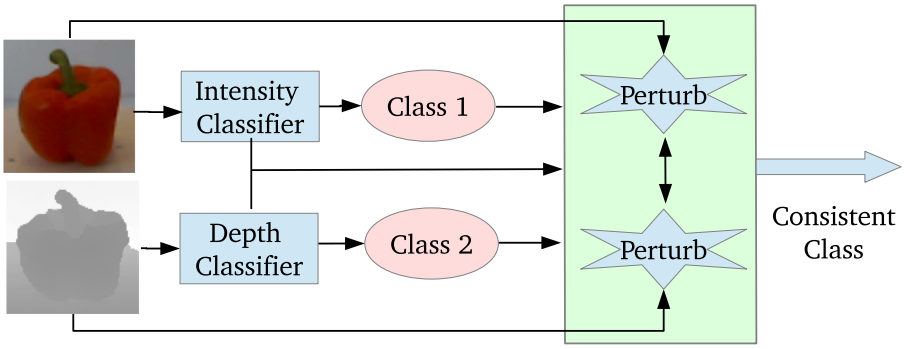


Figure 5.1: Overview of the proposed Class Consistent Multi-Modal (CCMM) fusion. The proposed algorithm perturbs the input feature until all the input modalities predict a consistent class.

them to the decision unit or a classifier, e.g., a support vector machine (SVM) [123]. A straight-forward way of combining the features is to concatenate them, which has been used in biometrics ([124], [125], [126]), object recognition [127], scene classification [128] etc. Feature concatenation preserves the raw information so that the classifier can utilize the correlation among modalities. However, these features are often very high dimensional and hence simple concatenation can be inefficient. Another approach to feature level fusion is multiple kernel learning (MKL), which learns a linear combination of multiple kernels. Finding appropriate feature combinations entails designing good kernel functions among a set of candidate kernels. MKL is a powerful way of determining the mixing weights of multiple kernels [89], [108], [100], [101]. For multi-modal fusion, each modality can be used to form a kernel matrix; an optimal linear combination of kernel matrices translates into optimal feature level fusion. When using only a linear kernel for each modality, the MKL methods are similar to feature concatenation, except that the features from each modality are weighted based on training accuracy. However, in order to make a good decision at test time, it is important that we also determine the quality

of the test features from each modality. For example, from a training database of depth and intensity images, we might conclude that both modalities are equally useful for classification; however, at test time, the depth image may be noisy due to specularity on the object's surface. In such situations, it is useful to make the prediction based on score level or decision level fusion and not rely entirely on training accuracy. Score level fusion can be done by averaging the scores of decision functions and decision level fusion can be performed by taking a majority vote from all the modalities. Recently, [129] proposed a sparse representation-based multi-modal biometric fusion method, which represents the test data by a sparse linear combination of training data, while constraining the observations from different modalities of the test subject to share their sparse representations. Effectively, they regularize the joint sparse coefficient matrix with the $\ell_{\{1,2\}}$ norm, which enforces the test feature to be reconstructed from the training features of the same class. This method is the closest to our approach in that it implicitly enforces that different modalities share a common class at test time. However, [129] does not learn a classification model, and training data for each class needs to be "paired" for each modality. That is, the number of samples in each modality must be the same to enforce the $\ell_{\{1,2\}}$ constraint on the joint sparse coefficient matrix. Furthermore, enforcing row sparsity on the joint sparse coefficient matrix of a test sample makes the method susceptible to the ordering of the training samples within each class. Also, sparse methods are generally slow for large training matrices.

Most algorithms for feature fusion have been developed for continuous features. Recently, with the the availability of large datasets, the need for efficient algorithms

that can work with big data has increased. One way to efficiently process large number of features is to represent each of them as binary features. Binary codes are attractive representations of data for similarity-based search and retrieval purposes, due to their storage efficiency and computational efficacy ([130], [131], [132], [133]). For example, 250 million images can be represented by 64 bit binary codes by employing only 16 GB of memory. Hashing is a common method to convert high dimensional features to binary codes whose Hamming distances preserve the original feature space distances. Although shorter codes are more desirable due to direct implementation in hash tables, longer binary descriptors of data have been shown to be efficient for fast similarity search tasks. For example, [134] proposed a multi-index hashing method, and [135] introduced a branch and bound approach to perform exact k-nearest neighbors search in sub-linear time with long binary codes. To the best of our knowledge, the method presented in this chapter is the first work proposing multi-modal fusion using binary codes. We propose to modify the test features in a way that all the modalities consistently agree on a common class label. We call this approach *class consistent multi-modal* (CCMM) fusion. The key idea, summarized in Fig. 5.1, is to minimize the magnitude of perturbations to feature values for each modality to get to a point where all the modalities are predicting a common class label. We develop this intuition into an optimization problem that can be solved efficiently via quadratic programming for continuous features, and mixed integer programming for binary features . We evaluate this algorithm on several state-of-the-art datasets and results show that the method outperforms previous methods consistently. The contributions of this chapter are as follows,

- We enforce class consistency across all available modalities in a perturbation model to determine the class of multi-modal data item.

- Based on this notion of class consistency, we develop an efficient binary feature fusion algorithm.

5.2 Class Consistent Multi-Modal Fusion (CCMM)

Our method relies on the intuition that when multiple modalities are available, each of them should predict the same class. In case of discrepancy in the prediction of a test sample, we employ a strategy that enforces consistency of the predicted class across modalities. We achieve this consistency by perturbing the test sample from each modality so that their predictions are consistent. This is formally posed as an optimization problem which minimizes the perturbation to satisfy the constraint that all modalities predict the same class label. In what follows, we establish our notation and develop the algorithm, first for continuous features and then, for binary features.

Assume that there are M modalities each with N_m labeled samples where $m = 1, \ldots, M$. Let the data matrix of the m^{th} modality be denoted by $\mathbf{Y}^{(m)} \in \mathbb{R}^{d \times N_m}$, where each column of $\mathbf{Y}^{(m)}$ is a d-dimensional data sample denoted by $\mathbf{y}_i^{(m)} \in \mathbb{R}^d$, for $i = 1, \ldots, N_d$. Let the class label of the i^{th} sample in the m^{th} modality be denoted by $l_i^{(m)} \in \{1, \ldots, C\}$, where C is the number of classes. Note that, for now, we regard the features as continuous; subsequently we adapt our method for binary features.

Let $\mathbf{W}^{(m)} := \begin{bmatrix} \mathbf{w}_1^{(m)} \\ \vdots \\ \mathbf{w}_C^{(m)} \end{bmatrix}$, be the classifier matrix for all categories in modality m, where the c^{th} row vector $\mathbf{w}_c^{(m)} \in \mathbb{R}^{1\times d}$ denotes the parameters of a linear classifier for the c^{th} class, which we refer to as a classification weight vector. These weight vectors are learned in a way that the class of a test sample $\mathbf{y}_p^{(m)}$ can be computed as,

$$\text{class of } \mathbf{y}_p^{(m)} = \arg\max_c \mathbf{w}_c^{(m)} \mathbf{y}_p^{(m)}. \tag{5.1}$$

In our implementation we use an SVM ([136], [137]) to learn these classification weight matrices $\mathbf{W}^{(m)}$ for all modalities.

First, we describe the method for two modalities and then extend it to multiple modalities. Denote a given test sample's two modalities by $\mathbf{y}_p^{(1)}$ and $\mathbf{y}_p^{(2)}$ which by construction belong to the same class. Our goal is to minimize the total perturbation needed to reach the condition that the predicted classes using SVM matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are identical. This is captured in the following optimization problem,

$$\begin{aligned} &\min_{\mathbf{y}^{(1)},\mathbf{y}^{(2)}} \|\mathbf{y}^{(1)} - \mathbf{y}_p^{(1)}\|_2 + \|\mathbf{y}^{(2)} - \mathbf{y}_p^{(2)}\|_2 \\ \text{subject to,} \quad &\arg\max_c \mathbf{w}_c^{(1)} \mathbf{y}^{(1)} = \arg\max_c \mathbf{w}_c^{(2)} \mathbf{y}^{(2)} \end{aligned} \tag{5.2}$$

The optimization problem in (5.2) is non-smooth and non-convex due to the arg max functions. In order to solve it efficiently, we approximate it with a tractable convex problem. To achieve this, we employ an alternative optimization approach. First, we assume that the class predicted by the second modality is correct and optimize for $\mathbf{y}^{(1)}$, and then, we fix the class to the one predicted by the first modality and

optimize for $\mathbf{y}^{(2)}$. When optimizing for the m^{th} modality feature $\mathbf{y}^{(m)}$, the class that is assumed to be correct is called the target class and is denoted by t_m , i.e.,

$$t_1 := \arg\max_c \mathbf{w}_c^{(2)} \mathbf{y}_p^{(2)}, \tag{5.3}$$

and,

$$t_2 := \arg\max_c \mathbf{w}_c^{(1)} \mathbf{y}_p^{(1)}. \tag{5.4}$$

We seek to perturb the feature $\mathbf{y}_p^{(m)}$ so that its predicted class is t_m, which can be achieved by solving the following problem,

$$\begin{aligned} & \min_{\mathbf{y}^{(m)}} \|\mathbf{y}^{(m)} - \mathbf{y}_p^{(m)}\|_2 \\ \text{subject to,} \quad & \arg\max_c \mathbf{w}_c^{(m)} \mathbf{y}^{(m)} = t_m. \end{aligned} \tag{5.5}$$

As explained later, the optimization problem in (5.5) is a quadratic program (QP). Let the solution of (5.5) be denoted by $\tilde{\mathbf{y}}_t^{(m)}$. Finally, the consistent class, denoted by l_p, across both modalities is the target class of the modality that requires the smallest change with respect to the original feature norm, i.e.,

$$l_p = t_{m^*}, \tag{5.6}$$

where,

$$m^* = \arg\min_m \frac{\|\tilde{\mathbf{y}}_p^{(m)} - \mathbf{y}_p^{(m)}\|_2}{\|\mathbf{y}_p^{(m)}\|_2}. \tag{5.7}$$

Next, we describe how the optimization problem in (5.5) can be written as a

quadratic convex program. The constraints in (5.5) can be re-written as,

$$\begin{aligned}
\arg\max_{c} \quad & \mathbf{w}_c^{(m)}\mathbf{y}^{(m)} = t_m \\
& \Rightarrow \mathbf{w}_{t_m}^{(m)}\mathbf{y}^{(m)} \geq \mathbf{w}_i^{(m)}\mathbf{y}^{(m)}, \qquad \forall i \neq t_m \\
& \Rightarrow \mathbf{w}_i^{(m)}\mathbf{y}^{(m)} - \mathbf{w}_{t_m}^{(m)}\mathbf{y}^{(m)} \leq \mathbf{0}, \qquad \forall i \neq t_m \\
& \Rightarrow \mathbf{A}_{t_m}\mathbf{y}^{(m)} \leq \mathbf{0}, \qquad (5.8)
\end{aligned}$$

where, $\mathbf{A}_{t_m}\mathbb{R}^{C-1\times d}$ is a constraint matrix whose rows are computed as, $[\mathbf{A}_{t_m}]_{i,:} = \mathbf{w}_i^{(m)} - \mathbf{w}_{t_m}$. Hence, the problem in (5.5) can be optimized by solving the following QP program,

$$\begin{aligned}
& \min_{\mathbf{y}^{(m)}} \|\mathbf{y}^{(m)} - \mathbf{y}_p^{(m)}\|_2 \\
\text{subject to,} \quad & \mathbf{A}_{t_m}\mathbf{y}^{(m)} \leq \mathbf{0}. \qquad (5.9)
\end{aligned}$$

5.2.1 CCMM for binary features

As stated earlier, binary features are very useful for large scale classification because they require smaller storage space and are efficient for classification. However, the optimization problem in (5.9) has been designed for continuous features. For binary features, if we predict a consistent class by solving this problem, we may not achieve good performance because the solution will not lie in a binary space. In face recognition, for example, a binary feature may represent an image attribute like sunglasses, which could be either present or not present in the image. Hence, we optimize for the binary features over a binary space.

In this sub-section, we assume the features $\mathbf{b}_i^{(m)} \in \{0,1\}^d$ are d-dimensional

binary vectors. Furthermore, data matrices $\mathbf{B}^{(m)} = [\mathbf{b}_1^{(m)}, \ldots, \mathbf{b}_{N_m}^{(m)}] \in \{0,1\}^{d \times N_m}$ are of size $d \times N_m$ with binary elements. As with continuous features, we learn SVM weight matrices for each modality. The major difference in setting up our optimization problem is that, in the case of binary features, we want the solution of the optimization problem to lie in a binary space. We reformulate (5.9) for binary features as:

$$\min_{\mathbf{b}^{(m)}} \|\mathbf{b}^{(m)} - \mathbf{b}_p^{(m)}\|_1$$

subject to,

$$\mathbf{A}_{t_m}\mathbf{b}^{(m)} \leq \mathbf{0}$$

$$\mathbf{b}^{(m)} \in \{0,1\}^d. \tag{5.10}$$

Note that for binary features, we minimize the ℓ_1 instead of the ℓ_2 norm because the former counts the number of places in the binary vector where the solution differs from the input feature. In other words, we minimize the Hamming distance between the input feature and the solution. Minimizing the ℓ_1 norm is a non-smooth function; hence, we use an auxiliary variable $\mathbf{z}$ to make the cost differentiable,

$$\min_{\mathbf{b}^{(m)}, \mathbf{z}} \|\mathbf{z}\|_1$$

subject to,

$$\mathbf{A}_{t_m}\mathbf{b}^{(m)} \leq \mathbf{0}$$

$$\mathbf{z} = \|\mathbf{b}^{(m)} - \mathbf{b}_p^{(m)}\|_1,$$

$$\mathbf{b}^{(m)} \in \{0,1\}^d, \qquad \mathbf{z} \in \{0,1\}^d. \tag{5.11}$$

The ℓ_1 constraints involving $\mathbf{z}$ are difficult to optimize. In order to eliminate them,

we replace them with a set of linear constraints as follows. Let the i^{th} element of vectors $\mathbf{z}$, $\mathbf{b}^{(m)}$, and $\mathbf{b}_p^{(m)}$ be denoted by z_i, $b_i^{(m)}$ and $b_{pi}^{(m)}$, respectively. Next, $\mathbf{z} = \|\mathbf{b}^{(m)} - \mathbf{b}_p^{(m)}\|_1$ can be replaced by the following linear constraints,

$$z_i \geq (b_{pi}^{(m)} - b_i) \tag{5.12}$$

$$z_i \geq -(b_{pi}^{(m)} - b_i). \tag{5.13}$$

Now the optimization problem in (5.11) can be modified as,

$$\min_{\mathbf{b}^{(m)},\mathbf{z}} \sum_{i=1}^{d} z_i$$

subject to,

$$\mathbf{A}_{t_m}\mathbf{b}^{(m)} \leq \mathbf{0}$$

$$\begin{bmatrix} -\mathbf{I}_d & +\mathbf{I}_d \\ -\mathbf{I}_d & -\mathbf{I}_d \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ \mathbf{b}^{(m)} \end{bmatrix} \leq \begin{bmatrix} \mathbf{b}_p^{(m)} \\ -\mathbf{b}_p^{(m)} \end{bmatrix}$$

$$\mathbf{b}^{(m)} \in \{0,1\}^d, \qquad \mathbf{z} \in \{0,1\}^d. \tag{5.14}$$

The optimization in (5.14) is a linear programming problem in $\mathbf{b}^{(m)}, \mathbf{z}$ except for the fact that the solution space is binary. Although this problem can be solved with a mixed integer programming (MIP) solver, we propose an efficient greedy algorithm and, later, empirically demonstrate that the solution of the greedy algorithm is close to that of the MIP solver. In order to solve the problem in (5.14) or (5.10) greedily, we first find a feasible solution, which can simply be one of the training samples from the target class satisfying the constraints of problem (5.10). Now, starting from this feasible solution, we move towards the test sample as much as possible without leaving the feasible region. Let the initial feasible solution be denoted by

$\mathbf{b}_0$ and the running solution, which we will keep updating, be denoted by $\mathbf{b}$. First we initialize $\mathbf{b}$ to $\mathbf{b}_0$. Next, we find all the elements of $\mathbf{b}$ that are different from the test sample $\mathbf{b}_p^{(m)}$. Let this set of bit locations be denoted by $\mathcal{S}$, i.e.,

$$\mathcal{S} := \{i \mid b_i \neq b_{pi}^{(m)}\},$$

where b_i is the i^{th} bit of vector $\mathbf{b}$ and $b_{pi}^{(m)}$ is the i^{th} bit of $\mathbf{b}^{(m)}$. Our goal is to change as many bits from this set $\mathcal{S}$ as possible because every change takes $\mathbf{b}$ one step closer to the test feature $\mathbf{b}_p^{(m)}$. Choosing the optimal subset of bits is an NP-hard problem and, hence, we resort to an approximate greedy method. Next, we present this greedy algorithm that changes one bit at a time from this set $\mathcal{S}$. The solution of the greedy algorithm can further be improved by various MIP solvers; however, empirically we observe that the greedy solution is quite good. In order to select a bit from $\mathcal{S}$ we flip all the bits in $\mathcal{S}$ and compute the following score,

$$s_i = \min_{c \neq t} (\mathbf{w}_t^{(m)} \mathbf{b} - \mathbf{w}_c^{(m)} \mathbf{b}_{\bar{i}}), \tag{5.15}$$

where $\mathbf{b}_{\bar{i}}$ is $\mathbf{b}$ with its i^{th} bit flipped. Note that, since we start from a feasible solution, s_i is bounded below by 0. Recall that $\mathbf{w}_c^{(m)}$ is the SVM weight vector for the c^{th} class of the m^{th} modality. The score s_i is the difference of scores between the target class and its closest one if the i^{th} bit is flipped. In other words, s_i corresponds to the constraint closest to the current solution $\mathbf{b}$ and is most likely to be violated if b's i^{th} bit is changed. This is illustrated in Fig. 5.2. Now, our goal is to change that bit which will keep the current solution in the feasible region as much as possible. Hence, we change that bit of $\mathbf{b}$ that belongs to set $\mathcal{S}$ and for which the solution remains as feasible as possible. Feasibility is measured by the maximum distance of

Algorithm 5: Greedy algorithm to optimize for binary feature

Input: Binary test data $\mathbf{b}_p^{(m)}$, Classification matrices $\mathbf{W}^{(m)}$, a feasible solution $\mathbf{b}_0$, target class t

Output: $\mathbf{b}$

Initialize $\mathbf{b} = \mathbf{b}_0$, $v = 1$.

$\mathcal{S} \leftarrow \{i \mid b_i \neq b_{pi}^{(m)}\}$

while *($v > 0$) AND ($\mathcal{S}$ is not empty)* **do**

1. $\mathbf{b}_{\bar{i}} \leftarrow \mathbf{b}$ with i^{th} bit flipped.

2. $v \leftarrow \max_{i \in \mathcal{S}} \min_{c \neq t} (\mathbf{w}_t^{(m)} \mathbf{b} - \mathbf{w}_c^{(m)} \mathbf{b}_{\bar{i}})$

if $v > 0$ **then**

$j \leftarrow \arg\max_{i \in \mathcal{S}} \min_{c \neq t} (\mathbf{w}_t^{(m)} \mathbf{b} - \mathbf{w}_c^{(m)} \mathbf{b}_{\bar{i}})$

$b_j \leftarrow 1 - b_j$

end

3. $\mathcal{S} \leftarrow \{i \mid b_i \neq b_{pi}^{(m)}\}$

end

return b

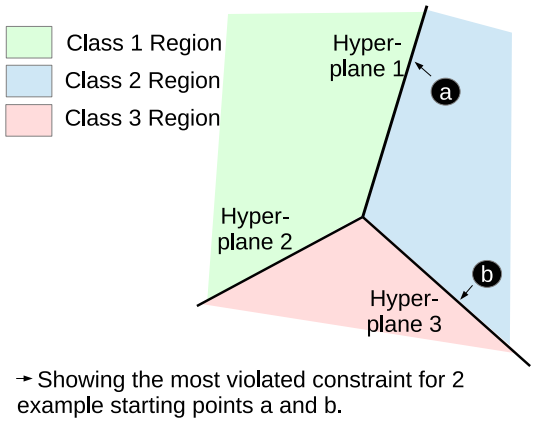


Figure 5.2: Example of most violated constraint. Both solutions a and b lie in target class 2. Since a is likely to move to class 1, hyperplane 1 corresponds to the most violated constraint. Similarly, if the current solution is b, hyperplane 3 corresponds to the most violated constraint.

$\mathbf{b}$ from all the constraints. We denote the index of the bit to be flipped by j, and it is computed as,

$$j = \arg\max_i s_i. \tag{5.16}$$

All the steps of the greedy algorithm are summarized in Algorithm 5. Having computed the solution to the optimization problem in (5.14), we can compute the consistent class l_p from both modalities based on the perturbation as follows,

$$l_p = t_{m^*}, \tag{5.17}$$

where,

$$m^* = \arg\min_m \frac{\|\tilde{\mathbf{b}}_p^{(m)} - \mathbf{b}_p^{(m)}\|_1}{\|\mathbf{b}_p^{(m)}\|_1}. \tag{5.18}$$

However, we find in our experiments that weighting the perturbations based on the quality of the modalities, improves the performance. This is explained in the next sub-section. Finally, we summarize all the steps to compute the consistent class from two modalities in Algorithm 6.

Algorithm 6: Summary of CCMM for binary features

Input: Binary test data $\mathbf{b}_p^{(m)}$, Classification matrices $\mathbf{W}^{(m)}$, for $m = 1, 2$

Output: Class consistent label l_p

1. Predict class labels $l_p^{(m)} = \arg\max_c w_c^{(m)} b_p^{(m)}$ for individual modalities.

2. Compute target labels as $t_1 = l_p^{(2)}, t_2 = l_p^{(1)}$.

3. Compute $\tilde{b}_p^{(m)}$ by solving optimization problem in (5.14) using greedy Algorithm 5.

4. Predict class consistent label l_p using (5.17) or (5.23).

return l_p

5.2.2 Extension to Multiple Modalities

So far we have described how to predict a consistent class with two modalities. For multiple modalities, we make a set of target labels for each modality. This set, denoted by $\mathcal{Z}$, consists of the labels predicted by all the modalities. Let the number of modalities be denoted by M. We compute the score s_{mc} of the m^{th} modality based on the perturbation needed to predict the c^{th} target class as follows,

$$s_{mc} = \frac{\exp(-\epsilon_{mc}/\sigma)}{\sum_{c \in \mathcal{Z}} \exp(-\epsilon_{mc}/\sigma)}, \tag{5.19}$$

where,

$$\epsilon_{mc} = \frac{\|\tilde{\mathbf{b}}_p^{(m)} - \mathbf{b}_p^{(m)}\|_1}{\|\mathbf{b}_p^{(m_z)}\|_1},$$

and σ is a parameter that controls the sharpness of the distribution of the scores over classes. For each class, we compute the combined score as a weighted sum of

each modality,

$$s_c = \sum_{m=1}^{M} \eta_m * s_{mc}, \tag{5.20}$$

where, η_m is the quality of the m^{th} modality based on the training data. Although, we could have given the same weight to each modality, i.e. set $\eta_m = 1, \forall m$, setting these weights based on training data slightly improves performance. We compute η_m based on the kernel alignment criterion [103] which has been shown to generalize well for unseen test data and has been used for combining multiple kernels [89]. The η_m is computed based on similarity between the the linear kernel matrix of the m^{th} modality, i.e. $\mathbf{B}^{(m)T}B^{(m)}$, and ideal kernel matrix $\boldsymbol{\mathcal{K}d}_m \in \mathbb{R}^{N_m \times N_m}$, defined as,

$$\boldsymbol{\mathcal{K}d}_m(i,j) = \begin{cases} 1, & \text{if} \quad l_i^{(m)} = l_j^{(m)}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.21}$$

The score η_m is computed as,

$$\eta_m = \frac{\langle \boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}d}_m \rangle}{\sqrt{\langle \boldsymbol{\mathcal{K}}_m, \boldsymbol{\mathcal{K}}_m \rangle \langle \boldsymbol{\mathcal{K}d}_m, \boldsymbol{\mathcal{K}d}_m \rangle}}, \tag{5.22}$$

where, $\boldsymbol{\mathcal{K}}_m := \mathbf{B}^{(m)T}B^{(m)}$, and $\langle ., . \rangle$ denotes the dot product between the argument matrices. Finally, the class l_p of the test input is predicted as the one with the maximum score,

$$l_p = \arg\max_c s_c. \tag{5.23}$$

5.3 Experiments

We evaluate the method on publicly available computer vision datasets. First, we use the fusion algorithm to combine image and depth data for object categorization. Next, we apply the method to fuse multiple modalitites for biometrics

applications and demonstrate significant improvement over previous methods. Finally, we combine intensity and semantic features on the Pascal-Sentence dataset. We compare the method to state-of-the-art multimodal fusion methods such as recently proposed sparse multimodal biometric recognition (SMBR) [129], sparse logistic regression (SLR) [138], support vector machine (SVM) [39] , and multiple kernel learning (MKL) [108] algorithms. As SLR and SVM methods cannot handle multiple modalities, [129] explored score-level and decision level fusion to combine modalities. Score level fusion was achieved by adding probability outputs of all the modalities to obtain a final score vector. Classification was performed by choosing the class corresponding to the maximum score. For decision level fusion, the class chosen by the maximum number of modalities was chosen. The score level fusion using SLR is called SLR-Sum and the decision level fusion is called as SLR-Major; for SVM, they are called SVM-Sum and SVM-Major, respectively. The parameter σ is set to 0.01 for all our experiments. We implemented our algorithm in MATLAB and used the Gurobi optimizer [139] for solving MIP and QP problems, for binary features and continuous features, respectively. We observed that the performance of the Gurobi optimizer was similar to the proposed greedy algorithm for biometrics application, and slightly better for object categorization. We plot the normalized difference between the solutions of the Gurobi and the greedy algorithm in Fig. 5.3, for fusing two iris features. As can be seen from this figure, the greedy algorithm's solution is close to that of the Gurobi for most of the test samples. Hence, we report results using only the greedy algorithm to solve the MIP for biometrics application, and report results using the Gurobi optimizer initialized with the greedy solution

for object categorization. Furthermore, for classification or rank-one recognition, we include the top 5 classes into target class set. We used the SVM model learned on the joint features of both the modalities. In the following subsections, we describe each of the datasets and present our results.

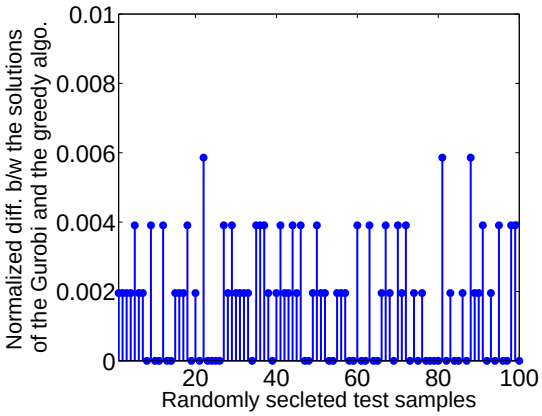


Figure 5.3: The performance of the greedy algorithm compared to the Gurobi MIP solver. For most of the test samples, the difference is less than 0.004 which corresponds to approximately 2 bits. That is, loosely speaking, the greedy algorithm's solution is, on an average, within 2 bits of the sophisticated MIP solver.

	CCMM	SMBR	SLR-Sum	SLR-Major	SVM-Sum	SVM-Major	MKL Fusion
Intensity Features	70.1	64.7	64.3	64.3	70.1	70.1	68.4
Depth Features	64.9	61.1	63.6	63.6	64.9	64.9	61.8
Combined Features	**83.7**	73.5	73.4	71.9	79.1	74.1	77.1

Table 5.1: Classification Accuracy for RGB-D data

5.3.1 RGB-D data

Recently, there has been a growing interest in using both intensity and depth data for computer vision algorithms. For example, with Microsoft's Kinect camera

one can capture videos of both color as well as corresponding depth data. The purpose of this experiment is to evaluate CCMM on binary features computed using color and depth data. We use the RGB-D dataset from the University of Washington [106] which consists of 51 object categories. A few examples of pairs of color and depth images from this dataset are shown in Fig. 5.4. Most of the depth images are noisy. Hence, we apply a recursive median filter to fill in missing values. Processed images are shown in the third row of Fig. 5.4.

Figure 5.4: Example images of the RGBD dataset. First row shows the color images, second row displays the corresponding depth images, and the third row is the denoised version of the second row after applying the recursive median filter.

5.3.2 WVU dataset

	Finger 1	Finger 2	Finger 3	Finger 4	Iris 1	Iris 2
CCMM	67.8	86.9	69.4	89.3	60.5	61.2
SMBR	68.1	88.4	69.2	87.5	60.0	62.1
SLR	67.4	87.9	66.0	87.5	57.1	57.9

Table 5.2: Rank-one recognition of single modalities for WVU data

The WVU biometrics dataset [140] consists of multiple biometrics such as fingerprints, iris, palmprint, hand geometry and voice samples from different subjects.

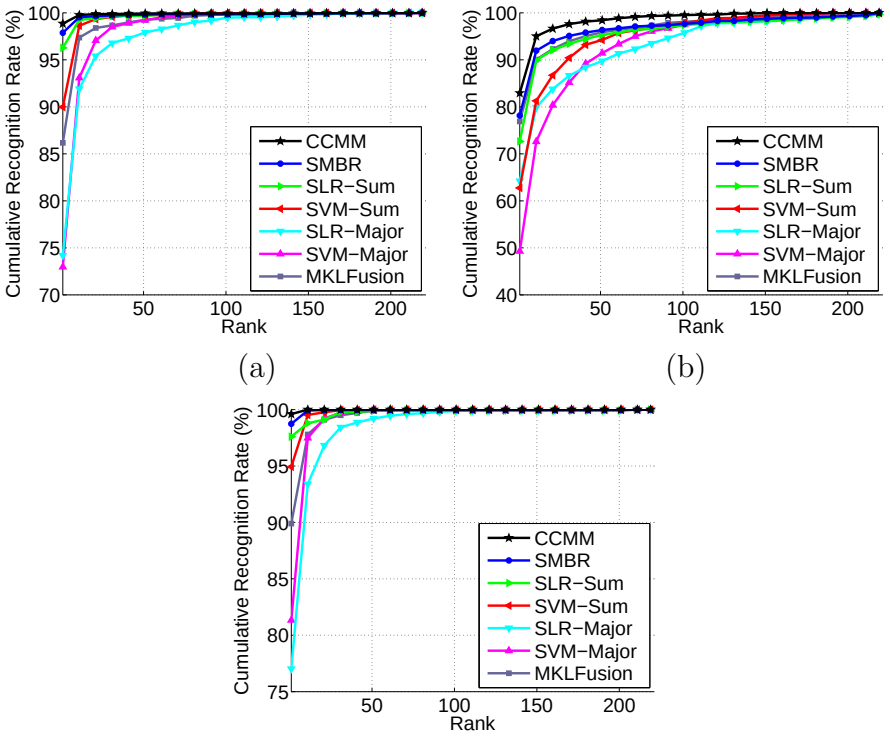


Figure 5.5: The comparison of CMCs for different modality combinations of WVU dataset. (a) Four fingers, (b) Two Irises, and (c) All the modalities (four fingers and two irises).

	CCMM	SMBR	SLR-Sum	SLR-Major	SVM-Sum	SVM-Major	MKL Fusion
4 Fingers	**98.8**	97.9	96.3	74.2	90.0	73.0	86.2
2 Irises	**82.9**	76.5	72.7	64.2	62.8	49.3	76.8
All Modalities	**99.6**	98.7	97.6	84.2	94.9	81.3	89.8

Table 5.3: Comparison of Rank-one recognition performance on WVU dataset for different combinations of modalities

Following the standard setting proposed in [129], we chose iris and fingerprint for testing the method. Furthermore, the evaluation was done on the subset of 219 subjects having samples in both modalities. Some challenging examples of fingerprints and iris images are shown in Fig. 5.6. We used the same Gabor features as in [129] for fingerprints and iris images. Before computing these features a robust pre-processing was applied to the images. Iris images were first segmented using the method proposed in [141], and then, a 25×240 iris template was created using the publicly available code of Masek et al [142]. Fingerprint images were first enhanced using filtering methods, then a core point was detected using algorithms from [143]. Finally, Gabor features were computed around the detected core point. Furthermore, to evaluate our algorithm on binary features, we computed the binary features of size 512 for each of the modalities using the method from [133]. Table 5.2 shows the accuracy of individual modalities. As can be seen, the performance of all the methods is comparable when only a single modality is considered. Next, we evaluate CCMM on various combinations of modalities. Following standard settings in [129], we compare the method on three combinations : (1) All the fingerprints (2) both iris images (3) all modalities. We present the comparison of different multi-

modal fusion algorithms in Table 5.3, which shows that CCMM outperforms the competing algorithms despite the fact that, for individual modalities, the performance of CCMM is lower than SMBR. This demonstrates that forcing the class predictions of multiple modalities to be consistent is useful for multi-modal fusion.

We also compare cumulative match curves (CMC) of different methods with multiple modality combinations in Fig. 5.5. The CMC is a popular tool to analyze the performance of biometric systems [129], [144], [145]. To compute CMCs, the target label set $\mathcal{Z}$ is composed of all the class labels. As can be seen from this figure, the proposed method consistently outperforms the comparison methods.

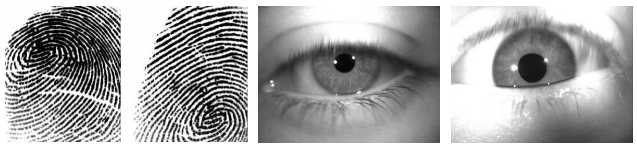


Figure 5.6: Example of challenging fingerprints and iris images from WVU dataset [140]. Many images in the dataset suffer from various artifacts such as blur, occlusion, noise etc.

5.3.3 CASIA Fingerprints dataset

The CASIA Fingerprint Image Database Version 5.0 (or CASIA-FingerprintV5) [146] contains a total of $20,000$ fingerprint images from 500 subjects. Each subject contributed a total of 40 images from 8 fingers, 4 from each hand. Each finger was scanned 5 times and the volunteers were asked to rotate their fingers with various levels of pressure to generate significant intra-class variations. In order to effectively compare all the algorithms, we took the subset of the first 50 subjects. Furthermore, we randomly selected 3 training images per modality for each subject, and

kept the remaining 2 for testing. The results presented in Table 5.4 are the average classification accuracies over 5 random trials. For each fingerprint, we compute the same features as for the WVU dataset. In this experiment, we evaluate the idea of class consistency with continuous features only. Furthermore, we compare CCMM on three combinations of modalities: (1) four fingerprints from the left hand, (2) four fingerprints from the right hand, and (3) all 8 fingerprints. As can be observed from Table 5.4, CCMM is comparable to SMBR when 4 fingers are fused, however it performs slightly better when fusing all 8 fingers.

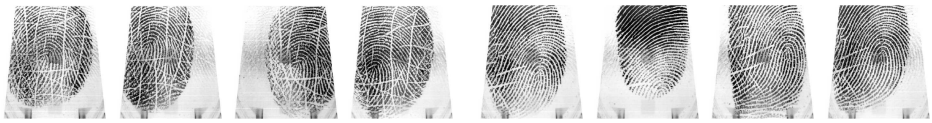


Figure 5.7: Example images of CASIA v5 dataset showing large intra-class variation. The first four images belong to one finger of subject 1 and the last four images are from subject 2.

	CCMM	SMBR	SLR-Sum	SLR-Major	SVM-Sum	SVM-Major	MKL Fusion
Left Fingers	**92.4**	90.4	88.2	83.2	85.8	76.4	81.8
Right Fingers	91.8	**92.2**	90.0	84.6	82.2	74.6	78.8
All Fingers	**97.0**	96.2	95.4	87.8	92.6	83.6	91.2

Table 5.4: Comparison of rank-one recognition performance on multi-modal CASIA fingerprint data

We test CCMM on the subset of the dataset by randomly selecting 15 images for training and 15 images for testing from each category. For the intensity images, we compute gradient based kernel descriptors [147] on 16×16 patches over a dense regular grid with spacing of 8 pixels. With these features, we compute a dictionary

	CCMM	SMBR	SLR-Sum	SLR-Major	SVM-Sum	SVM-Major	MKL Fusion
Intensity Features	66.2	66.2	65.4	65.4	66.2	66.2	67.2
Semantic Features	63.2	69.6	47.0	47.0	63.2	63.2	64.4
Combined Features	**77.2**	75.4	64.2	63.4	76.2	71.2	76.0

Table 5.5: Classification Accuracy for Pascal-Sentence dataset

of 1000 words using k-means. Using this dictionary of visual words, we employed efficient match kernels and used 1×1, 2×2, and 4×4 pyramid sub-regions [148] to compute image level features. For depth features, we compute the shape features over point clouds as described in [149] and gradient kernel descriptor features on the depth image. Similar to intensity features, image level depth features are computed using efficient match kernels over 1×1, 2×2, and 4×4 pyramid sub-regions using a dictionary of 1000 words. Finally, image level intensity and depth features are converted into binary features using the method proposed in [133]. We evaluate CCMM on individual modalities as well as their combination in Table 5.1. From the first two rows of the table, we note that SLR and SMBR methods have lower accuracy than SVM (CCMM is, of course, equivalent to SVM for the single modality case since we employ SVM as the per modality classifier). The reason for this is that these methods do not learn any classification model and rely on a sparse linear combination of training features to represent the test feature. Furthermore, none of the methods take into consideration that the input features are binary. By treating binary features in an appropriate way, CCMM is able to significantly improve the performance using a fusion of depth and intensity features, as seen in the last row

of Table 5.1.

5.3.4 Pascal-Sentence Dataset

This dataset has two modalities - images and sentences. The images in the dataset are collected from PASCAL VOC 2008, which is one of the most popular benchmark datasets for object recognition and detection. For each of the 20 categories of the PASCAL 2008 challenge, 50 images are randomly selected. Each image is annotated with 5 sentences using Amazon's Mechanical Turk. For our task we randomly picked just one sentence for each image. These sentences represent the semantics of the image.

Our image features, following [150], are collections of responses from a variety of detectors, image classifiers and scene classifiers. The details of the image features can be found in [150]. The semantic features are constructed by using word-net semantic similarity with a dictionary of 1,200 words. These are followed by a quantization step that reduces the dimension to 20. The details of the text features are presented in [150]. Finally, the features are converted to binary codes using [133]. We present our result in Table 5.5, which shows that CCMM works better than the competing methods.

5.4 Conclusion

We described a multi-modal fusion algorithm- CCMM- based on class consistency and demonstrated that it works significantly better than previous methods

for binary features. The main idea was to perturb the input modalities until they predict the same class. To compute the score for a target class, we took weighted average of scores from all the modalities. Although, the proposed algorithm used linear classification models, the class consistent prediction can be explored with other classification models too.

Chapter 6: Summary and Directions for Future Work

6.1 Summary

In this dissertation, we discussed sparse representation and dictionary-based methods for visual classification, when the data is partially or weakly labeled. In the first part, we addressed the problem of labeling the data and proposed a dictionary-based algorithm for partially labeled data. In the second part, we proposed a dictionary-based solution for multiple instance learning problem. Non-linear extensions for both the algorithms were developed using the kernel trick. The choice of kernel for non-linear classification models is often made with cross validation. To learn an optimal kernel, in the third part of the dissertation, we developed a sparse representation-based classification algorithm using multiple kernel learning. In the final chapter, we developed an algorithm for fusing multiple modalities using linear classification models.

6.2 Directions for Future Work

As we discussed in Chapter 2, unlabeled data can be used to improve the classification model by learning a probability distribution over classes. However, one is

often confronted with the situations where information about a sample is available from multiple sources. For example, description of an image might be available in the form of intensity as well as depth image or the identity of a person may be determined by his face, fingerprints, iris signature etc. In such cases, we can learn independent dictionaries for each modality and fuse their score to determine the class information of a test sample. However, instead of using unlabeled data independently, we can use it simultaneously to improve all the dictionary models. For example, if a fingerprint can confirm the identity of a person with high probability, the corresponding face image, which alone may not be very informative, can be used to improve the face dictionaries. This can be done by maintaining probability distributions for all the modalities and updating them jointly. In Chapter 3, a novel dictionary-based algorithm for multiple instance learning was presented. Although this algorithm was applied for classification, it can also be applied for object detection. In Chapter 4, we developed a multiple kernel learning approach for SRC. Note that the sparse codes are learned over the training samples. This method can be used jointly with dictionary learning. The perturbation model developed in Chapter 5 can be applied to many classification models. Although, the proposed algorithm works with linear weight vector, similar optimized problem can be formulated with dictionary learning.

Bibliography

[1] R. Rubinstein, A.M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045 –1057, june 2010.

[2] 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*, 98(6):1031 –1044, june 2010.

[3] M. Elad, M.A.T. Figueiredo, and Y. Ma. On the role of sparse and redundant representations in image processing. *Proceedings of the IEEE*, 98(6):972 –982, june 2010.

[4] Michael Elad. *Sparse and Redundant Representations - From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[5] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comp.*, 20(1):33–61, 1998.

[6] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *1993 Conference Record of the 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[7] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Info. Theory*, 50(10):2231–2242, Oct. 2004.

[8] Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[9] K. Engan, S. O. Aase, and J. H. Husoy. Method of optimal directions for frame design. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1999.

[10] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[11] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, pages 3736 – 3745, December 2006.

[12] M. Protter and M. Elad. Image sequence denoising via sparse and redundant representations. *IEEE Transactions on Image Processing*, 18:27–35, January 2009.

[13] W. Dong, X. Li, L. Zhang, and G. Shi. Sparsity-based image denoising via dictionary learning and structural clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[14] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19:2861–2873, November 2010.

[15] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *IEEE Transactions on Image Processing*, 21:3467–3478, August 2012.

[16] A. Adler, Y. Hel-Or, and M. Elad. A shrinkage learning approach for single image super-resolution with overcomplete representations. In *European Conference on Computer Vision(ECCV)*, 2010.

[17] K. Etemand and R. Chellappa. Separability-based multiscale basis selection and feature extraction for signal and image classification. *IEEE Transactions on Image Processing*, 7(10):1453–1465, Oct. 1998.

[18] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. 2008.

[19] V. M. Patel and R. Chellappa. Sparse representations, compressive sensing and dictionaries for pattern recognition. In *Asian Conference on Pattern Recognition (ACPR)*, 2011.

[20] V. M. Patel, R. Chellappa, and M. Tistarelli. Sparse representations and random projections for robust and cancelable biometrics. *International Conference on Control, Automation, Robotics and Vision*, pages 1–6, Guangzhou, China, Dec. 2010.

[21] T. G. Dietterich and R. H. Lathrop. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

[22] O. Maron and T. Pérez. A Framework for Multiple-Instance Learning. In *Neural Information Processing Systems*, 1998.

[23] X. Zhu and A. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan & Claypool Publishers, 2009.

[24] V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear svms. In *ACM SIGIR*, 2006.

[25] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. Adaptive computation and machine learning. MIT Press, 2006.

[26] F. Rodriguez and G. Sapiro. Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries. *Tech. Report, University of Minnesota*, Dec. 2007.

[27] K. Huang and S. Aviyente. Sparse representation for signal classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[28] M. Ranzato, F. Haung, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[29] Q. Zhang and B. Li. Discriminative k-svd for dictionary learning in face recognition. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[30] Z. Jiang, Z. Lin, and L. S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[31] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, April 2012.

[32] M. Yang, L. Zhang, X. Feng, and D. Zhang. Fisher discrimination dictionary learning for sparse representation. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[33] G. Zhang, Z. Jiang, and L. S. Davis. Online semi-supervised discriminative dictionary learning for sparse representation. In *Asian Conference on Computer Vision*, 2012.

[34] P. Sprechmann and G. Sapiro. Dictionary learning and sparse coding for unsupervised clustering. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[35] I. Ramirez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[36] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[37] Y. Chen, C. S. Sastry, V. M. Patel, P. J. Phillips, and R. Chellappa. In-plane rotation and scale invariant clustering using dictionaries. *IEEE Transactions on Image Processing*, 22(6):2166–2180, June 2013.

[38] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT. ACM, 1998.

[39] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[40] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa. Design of non-linear kernel dictionaries for object recognition. *IEEE Transactions on Image Processing*, 22(12):5123–5135,, Dec. 2013.

[41] A. Shrivastava, H. V. Nguyen, V. M. Patel, and R. Chellappa. Design of non-linear discriminative dictionaries for image classification. In *Asian Conference on Computer Vision*, 2012.

[42] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa. Sparse embedding: A framework for sparsity promoting dimensionality reduction. In *European Conference on Computer Vision (ECCV)*, 2012.

[43] Y. Chen, V. M. Patel, Jaishanker K. Pillai, Rama Chellappa, and P. Jonathon Phillips. Dictionary learning from ambiguously labeled data. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[44] L. Rosasco, A. Verri, M. Santoro, S. Mosci, and S. Villa. Iterative projection methods for structured sparsity regularization. *MIT-CSAIL-TR-2009-050, CBCL-282*, 2009.

[45] T. Cour, B. Sapp, and B. Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12:1225–1261, 2011.

[46] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16:550–554, May 1994.

[47] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing shock graphs. In *IEEE International Conference on Computer Vision (ICCV)*, 2001.

[48] T. Cour, B. Sapp, and B. Taskar. Annotated faces on tv dataset.

[49] T. Cour, B. Sapp, C. Jordan, and B. Taskar. Learning from ambiguously labeled images. *Journal of Machine Learning (JMLR)*, 2011.

[50] A. Shrivastava, J. K. Pillai, V. M. Patel, and R. Chellappa. Learning discriminative dictionaries with partially labeled data. In *IEEE International Conference on Image Processing (ICIP)*, 2012.

[51] M. Chen, K. Q. Weinberger, and Y. Chen. Automatic feature decomposition for co-training. In *IEEE International Conference on Machine Learning (ICML)*, 2011.

[52] B. Scholkopf and A. J. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[53] T. Leung, Y. Song, and J. Zhang. Handling label noise in video classification via multiple instance learning. In *International Conference on Computer Vision*, 2011.

[54] Q. Zhang and S. A. Goldman. EM-DD: An improved multiple-instance learning technique. In *Neural Information Processing Systems*, 2001.

[55] Y. Chen and J. Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

[56] C. Leistner, A. Safari, and H. Bischof. MIForests: Multiple-instance learning with randomized trees. In *European Conference on Computer Vision*, 2010.

[57] Y. Chen, J. Bi, and J. Z. Wang. MILES: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1931–1947, 2006.

[58] V. M. Patel and R. Chellappa. *Sparse representations and compressive sensing for imaging and vision*. SpringerBriefs, 2013.

[59] Hyun Oh Song, Stefan Zickler, Tim Althoff, Ross Girshick, Mario Fritz, Christopher Geyer, Pedro Felzenszwalb, and Trevor Darrell. Sparselet models for efficient multiclass object detection. In *European Conference on Computer Vision*, 2012.

[60] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa. Kernel dictionary learning. In *International Conference on Acoustics, Speech, and Signal Processing*, 2012.

[61] S. Gao, I. W. Tsang, and L.-T. Chia. Kernel sparse representation for image classification and face recognition. In *European Conference on Computer Vision*, 2010.

[62] M. Harandi, C. Sanderson, R. Hartley, and B. Lovell. Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In *European Conference on Computer Vision*, 2012.

[63] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Neural Information Processing Systems*, 2003.

[64] J. Huo, Y. Gao, W. Yang, and H. Yin. Abnormal event detection via multi-instance dictionary learning. In *International Conference on Intelligent Data Engineering and Automated Learning*, 2012.

[65] X. Wang, B. Wang, X. Bai, W. Liu, and Z. Tu. Max-margin multiple-instance dictionary learning. In *International Conference on Machine Learning*, 2013.

[66] Bruno A. Olshausen and David J. Fieldt. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[67] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. *International Conference on Machine Learning (ICPR)*, 2009.

[68] Q. Qiu, V. M. Patel, and R. Chellappa. Information-theoretic dictionary learning for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[69] Li Zhang, Wei-Da Zhou, Pei-Chann Chang, Jing Liu, Zhe Yan, Ting Wang, and Fan-Zhang Li. Kernel sparse representation-based classifier. *IEEE Transactions on Signal Processing*, 60(4):1684 –1695, april 2012.

[70] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition*, 2009.

[71] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.

[72] P. A. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *Neural Information Processing Systems*, 2005.

[73] K. Sikka, A. Dhall, and M. Bartlett. Weakly supervised pain localization using multiple instance learning. In *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2013.

[74] Mark Schmidt, Glenn Fung, and Rómer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *European Conference on Machine Learning*, pages 286–297, 2007.

[75] Mark Schmidt, Glenn Fung, and Romer Rosales. Optimization methods for l1-regularization. *UBC Technical Report TR-2009-19*, 2009.

[76] P. Lucey, J.F. Cohn, K.M. Prkachin, P.E. Solomon, and I. Matthews. Painful data: The UNBC-McMaster shoulder pain expression archive database. In *International Conference on Automatic Face Gesture Recognition and Workshops*, 2011.

[77] A. Shrivastava, V. M. Patel, and R. Chellappa. Multiple kernel learning for sparse representation-based classification. *IEEE Transactions on Image Processing*, 23(7):3013–3024, July 2014.

[78] P. V. Gehler and O. Chapelle. Deterministic annealing for multiple-instance learning. In *International Conference on Artificial Intelligence and Statistics*, pages 123–130, 2007.

[79] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[80] P.J. Phillips. Matching pursuit filters applied to face identification. *IEEE Transactions on Image Processing*, 7(8):1150–1164, Aug 1998.

[81] E. Kokiopoulou and P. Frossard. Semantic coding by supervised dimensionality reduction. *IEEE Trans. Multimedia*, 10(5):806–818, Aug. 2008.

[82] P. Lucey, J. Howlett, J. F. Cohn, S. Lucey, S. Sridharan, and Z Ambadar. Improving pain recognition through better utilization of temporal information. In *International Conference on Auditory-Visual Speech Processing*, 2008.

[83] Ahmed Bilal Ashraf, Simon Lucey, Jeffrey F. Cohn, Tsuhan Chen, Zara Ambadar, Kenneth M. Prkachin, and Patricia E. Solomon. The painful face - pain expression recognition using active appearance models. *Image and Vision Computing*, 27(12):1788–1796, nov 2009.

[84] Carolina Galleguillos, Boris Babenko, Andrew Rabinovich, and Serge Belongie. Weakly supervised object localization with stable segmentations. In *European Conference on Computer Vision*, 2008.

[85] I Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[86] J. K. Pillai, V. M. Patel, R. Chellappa, and N. Ratha. Secure and robust iris recognition using random projections and sparse representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1877–1893, Sept. 2011.

[87] S. Gao, I. W.-H. Tsang, and L.-T. Chia. Sparse representation with kernels. *IEEE Transactions on Image Processing*, 22(2):423 –434, feb. 2013.

[88] Hanchao Qi and Shannon Hughes. Using the kernel trick in compressive sensing: Accurate signal recovery from fewer measurements. In *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, pages 3940 –3943, may 2011.

[89] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, July 2011.

[90] V. M. Patel, N. M. Nasrabadi, and R. Chellappa. Sparsity-motivated automatic target recognition. *Applied Optics*, 50(10), April 2011.

[91] Hanxi Li, Yongsheng Gao, and Jun Sun. Fast kernel sparse representation. In *International Conference on Digital Image Computing Techniques and Applications*, pages 72 –77, Dec. 2011.

[92] X.-T. Yuan and S. Yan. Visual classification with multi-task joint sparse representation. In *Computer Vision and Pattern Recognition*, 2010.

[93] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[94] Y. Chen, N. M. Nasrabadi, and T. D. Tran. Hyperspectral image classification via kernel sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 51(1):217 –231, jan. 2013.

[95] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[96] Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *J. Mach. Learn. Res.*, 6:1099–1125, December 2005.

[97] Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *J. Mach. Learn. Res.*, 6:1043–1071, December 2005.

[98] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 1191–1198, 2007.

[99] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *International Conference on Machine Learning*, pages 1175–1182, 2010.

[100] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *International Conference on Computer Vision (ICCV)*, 2007.

[101] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *International Conference on Computer Vision (ICCV)*, 2009.

[102] H. Tanabe, Tu Bao Ho, Canh Hao Nguyen, and S. Kawasaki. Simple but effective methods for combining kernels in computational biology. In *IEEE International Conference on Research, Innovation and Vision for the Future*, pages 71 –78, july 2008.

[103] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S. Kandola. On kernel-target alignment. In *Neural Information Processing Systems*, pages 367–373, 2001.

[104] Shibin Qiu and Terran Lane. A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 6(2):190–199, April 2009.

[105] L. Fei-Fei, R. Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. 2004.

[106] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse Distance Learning for Object Recognition Combining RGB and Depth Information. In *IEEE International Conference on on Robotics and Automation (ICRA)*, 2011.

[107] A.M. Martinez and R. Benavente. The ar face database. *CVC Technical Report No. 24,*, June 1998.

[108] A. Rakotomamonjy, U. Rouen, F. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[109] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[110] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[111] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. In *CIT Technical Report 7694*, 2007.

[112] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[113] Prateek Jain, Brian Kulis, and Kristen Grauman. Fast image search for learned metrics. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[114] Duc-Son Pham and Svetha Venkatesh. Joint learning and dictionary construction for pattern recognition. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[115] Jan C. van Gemert, Jan-Mark Geusebroek, Cor J. Veenman, and Arnold W. M. Smeulders. Kernel codebooks for scene categorization. In *European Conference on Computer Vision (ECCV)*, 2008.

[116] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[117] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[118] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: Design of dictionaries for sparse representation. In *IN: PROCEEDINGS OF SPARS'05*, pages 9–12, 2005.

[119] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[120] Andrew Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433 – 449, May 1999.

[121] Wongun Choi, C. Pantofaru, and S. Savarese. Detecting and tracking people using an RGB-D camera via multiple detector fusion. In *IEEE International Conference on Computer Vision Workshops*, 2011.

[122] A.K. Mishra, A. Shrivastava, and Y. Aloimonos. Segmenting "simple" objects using RGB-D. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[123] Pradeep K. Atrey, M. Anwar Hossain, Abdulmotaleb El-Saddik, and Mohan S. Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Multimedia Systems*, 16(6):345–379, 2010.

[124] A. Rattani, D.R. Kisku, M. Bicego, and M. Tistarelli. Feature level fusion of face and fingerprint biometrics. In *IEEE International Conference on Biometrics: Theory, Applications, and Systems*, 2007.

[125] X. Zhou and B. Bhanu. Feature fusion of face and gait for human recognition at a distance in video. In *International Conference on Pattern Recognition*, 2006.

[126] Arun Ross A and Rohin Govindarajan B. Feature level fusion using hand and face biometrics. *Proceedings of the SPIE*, 5779:196–204, Mar. 2005.

[127] K. Lai, Liefeng Bo, Xiaofeng Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE International Conference on Robotics and Automation*, 2011.

[128] Xiaofeng Ren, Liefeng Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[129] S. Shekhar, V.M. Patel, N.M. Nasrabadi, and R. Chellappa. Joint sparse representation for robust multimodal biometrics recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):113–126, Jan 2014.

[130] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104, June 2012.

[131] M. Rastegari, J. Choi, S. Fakhraei, H. Daume III, and L. S. Davis. Predictable dual-view hashing. In *IEEE International Conference on Machine Learning (ICML)*, 2013.

[132] Yunchao Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[133] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *European Conference on Computer Vision (ECCV)*, 2012.

[134] M. Norouzi, A. Punjani, and D.J. Fleet. Fast search in hamming space with multi-index hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[135] M. Rastegari, Chen Fang, and L. Torresani. Scalable object-class retrieval with approximate and top-k ranking. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[136] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, Mar 2002.

[137] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[138] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, June 2005.

[139] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2014.

[140] S Crihalmeanu, A. Ross, S. Schuckers, and L. Hornak. A protocol for multibiometric data acquisition, storage and dissemination. In *Technical Report, WVU, Lane Department of Computer Science and Electrical Engineering*, 2007.

[141] S.J. Pundlik, D.L. Woodard, and S.T. Birchfield. Non-ideal iris segmentation using graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2008.

[142] L. Masek and P. Kovesi. Matlab source code for biometric identification system based on iris patterns. *The University of Western Australia, Tech. Rep.*, 2003.

[143] A.K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching. *IEEE Transactions on Image Processing*, 9(5):846–859, May 2000.

[144] K.I. Chang, K.W. Bowyer, and P.J. Flynn. An evaluation of multimodal 2d+3d face biometrics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(4):619–624, April 2005.

[145] R.M. Bolle, J.H. Connell, S. Pankanti, N.K. Ratha, and A.W. Senior. The relation between the roc curve and the cmc. In *IEEE Workshop on Automatic Identification Advanced Technologies*, Oct 2005.

[146] CASIA-FingerprintV5, http://biometrics.idealtest.org/.

[147] L. Bo, X. Ren, and D. Fox. Kernel Descriptors for Visual Recognition. In *Advances in Neural Information Processing Systems (NIPS)*, December 2010.

[148] L. Bo and C. Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Advances in Neural Information Processing Systems (NIPS)*, December 2009.

[149] Liefeng Bo, Xiaofeng Ren, and D. Fox. Depth kernel descriptors for object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[150] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. In *European Conference on Computer Vision (ECCV)*, 2010.