Application of sparsity and metric learning based methods in classification problems

Submitted in partial fulfillment of the requirements
for the degree of

MASTER OF TECHNOLOGY
(Control & Computing)

by

Digbalay Bose
(143070026)

under the guidance of
Prof. Subhasis Chaudhuri

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

June 2016

Dissertation Approval Certificate

This dissertation entitled **Application of sparsity and metric learning based methods in classification problems** by **Digbalay Bose** (Roll No: 143070026) is approved for the degree of **Master of Technology** in Electrical Engineering with specialization in **Control and Computing** from **Indian Institute of Technology Bombay**, India.

Examiners

____

____

Supervisor

____

Chairman

____

Date: 24 JUNE 2016

Place: EE, IIT BOMBAY

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Digbalay Bose

DIGBALAY BOSE
143070026
Department of EE
IIT BOMBAY

24 June 2016.

Acknowledgement

I would like to express my sincere gratitude towards my guide Prof. Subhasis Chaudhuri for his constant support and guidance. His continuous motivation and immense knowledge helped me during the course of the research work. It was a great experience working under his esteemed guidance. I am also very grateful to Prof. Vinay P. Namboodiri, CSE Department, IIT Kanpur for providing valuable suggestions during the course of the project. I would also like to express my gratitude towards all the faculty members of IIT Bombay, whose teachings helped me in my project work on various occasions.

I would like to thank Mr. Avik Hati for helping me during the simulations and providing technical inputs. I would also thank Ms. Neha Bhargava for the helpful discussions, which helped me in clarifying many doubts. I am also very grateful to the other fellow members in Vision and Image Processing Lab namely Amit More, Dr. Amit Bhardwaj, Vineet Gokhale for their suggestions and providing me an excellent work environment.

Lastly but not the least I would like to thank my parents for their constant encouragement and love all through.

DIGBALAY BOSE

24th June 2016

Abstract

Classification problem is one of the standard problems in the machine learning domain, whose main goal is to determine the class of a new observation when a training set consisting of data members (having known class memberships)is given. Common yet challenging tasks under the broad class of classification problems include facial recognition and object recognition. Effective facial recognition involves the ability to identify a particular person , in spite of variations in lighting, pose, appearances etc , thus making the problem hard in real life scenario. Object recognition involves identifying objects in images and videos across different scales and viewpoints, which makes it a difficult problem for automatic systems. Further the problem becomes increasingly challenging when the different categories of images are highly correlated in visual sense and have finer differences between them.

Here we have considered both the facial and object recognition problems by considering two approaches i.e. sparsity and(i.e. basic Sparse Representation based Classification and Dictionary Learning techniques) and metric learning based methods. Firstly, sparse representation based classification was applied in the cases of facial recognition of two standard datasets i.e. Extended Yale B and AR and the effect of various features on the corresponding recognition accuracies have been analyzed. Also the sparse classification scheme was applied in the case of crowd videos (CUHK dataset). In both the cases of crowd video classification and facial recognition , comparisons have been done with SVM and K-nearest neighbor. Further, in the domain of sparsity based methods, dictionary learning schemes and their modifications have been discussed for classification problems. In case of object recognition, the problem of classifying visually correlated categories was considered from the Imagenet database. The correlations among different categories were exploited to obtain a hierarchical model i.e. a tree based structure . The structure thus obtained was coupled with Large margin nearest neighbor framework to obtain a Mahalnobis metric at each non leaf node of the tree, which was further utilized in a top down energy based classification scheme of the test data. For hyperparameters tuning of the proposed scheme two approaches were investigated namely Grid Search and Bayesian Optimization. Effect of features used were analyzed on the performance of the proposed scheme. Further another scheme was proposed for utilizing the obtained hierarchical structure , in which Label Consistent KSVD , a dictionary learning scheme for joint classifier and dictionary learning was applied to each non leaf node of the tree. In order to improve the classification performance, the training and test samples are passed through a pre-processing step where the Kernel matrix is sampled using Nystrom's method followed by an eigen vector decomposition to obtain a set of mapped samples(or virtual samples) which can be used by any dictionary learning technique. It is seen that this pre-processing step helps in improving the classification accuracy by enhancing the discrimination among the classes.

Contents

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**

1.1 Related Work 3

1.2 Organization of Thesis 5

2 Sparse Representation based Classification(SRC) **7**

2.1 Sparse Model for classification 7

2.2 SRC classification scheme 9

2.3 Geometrical Explanation 9

2.4 Results 10

2.4.1 Facial Recognition 11

2.4.2 Crowd Video Classification 15

3 Dictionary Learning **17**

3.1 Dictionary Learning Methods 17

3.1.1 Method of Optimal Directions 18

3.1.2 KSVD algorithm 18

3.2 Dictionary Learning for Classification approaches 20

3.2.1 Dictionary Learning approaches 20

3.2.2 Class specific KSVD learning 21

4 Object Recognition of Visually Correlated Categories using Hierarchial model **24**

4.1 Problem Statement 24

4.2 Distance metric learning for Large margin nearest neighbor classification . 24

4.2.1 Distance Metric Learning 25

4.2.2 Model 26

4.3 Distance metric learning for hierarchial classification of visually correlated categories 29

4.3.1 Spectral Clustering 30

4.3.2 Self Tuning Spectral Clustering 31

4.3.3 Spectral Clustering based Hierarchial Organization of Visually Correlated Categories 33

4.3.4 Node specific Large Margin Nearest neighbor 39

4.3.5 Classification scheme 40

4.3.6 Experiments 42

4.4 Label Consistent KSVD based hierarchial classification of visually correlated categories 58

4.4.1 Label Consistent KSVD 58

4.4.2 Node specific dictionary learning 60

4.4.3 Classification scheme 60

4.4.4 Preprocessing using Linearized Kernel Dictionary Learning 61

4.4.5 Results 64

5 Conclusions and Future work 65

A Appendix 1 67

B Appendix 2 72

List of Figures

1.1 **Two types of recognition problems: Identification of specific objects, broad object categories**) (*Image Courtesy: Visual Object Recognition , Kristen Graumann, Bastian Leibe ,Morgan and Claypool publishers 2011*) 2

2.1 **The intersection of the** l_p **ball and the plane** $y = Tx$ **(Top left(p=2),Top right(p=1),bottom left(p=1),bottom right(p=0.7)**) (*Image Courtesy: Sparse and Redundant Representation, Michael Elad, Springer, December, 2009*) 10

2.2 **Recognition rates on Extended YaleB database for varying feature dimensions** 12

2.3 **Comparison of the accuracies of SRC,SVM,K-NN classifier for feature space dimension 504(Extended YaleB database)** 13

2.4 **Recognition rates on AR database for varying feature dimensions** 13

2.5 **Comparison of the accuracies of SRC,SVM,K-NN classifier for feature space dimension 504(AR database)** 14

2.6 **Confusion matrix plot for the CUHK database classification using SRC** 16

2.7 **Variation of Classification accuracy with changing percentage of data in training set** 16

3.1 **Variation of recognition accuracy with the number of dictionary atoms of Yale database** 22

3.2 **Variation of recognition accuracy with the number of dictionary atoms of AR database** 23

4.1 **Diagram indicating the neighborhood of one training sample x_i before and after training . After optimization of the distance metric the 3 target neighbors lie within a smaller radius around the training sample x_i and the impostors(differently labelled inputs) lie outside the region of smaller radius by some specific margin value.**) (*Distance metric learning for Large Margin Nearest Neighbor Classification, Killian Q.Weinberger Lawrence.K.Saul, JMLR 2009*) 27

4.2 **Flow diagram for SIFT+LLC feature extraction scheme** 43

4.3 **Flow diagram for SIFT+BOW feature extraction scheme** 43

4.4 **Effect of cropping on image of keyboard class** 44

4.5 **Effect of cropping on image of cat class** 44

4.6 **Optimal tree structure($tree_2$) in terms of the average classification accuracy. Each node has a set of classes associated with it and the classes are mentioned by class numbers . [2,3] means hound and dog are grouped together in a node. Mapping from class number to actual classes can be found in Table 4.4** 52

4.7 **Convergence plot of the root node of $tree_2$** 53

4.8 **Convergence plot of the 1st node of level 2 of $tree_2$** 53

4.9 **Convergence plot of the 2nd node of level 2 of $tree_2$** 54

4.10 **Convergence plot of the 6th node of level 3 of $tree_2$** 54

4.11 **Convergence plot of the 8th node of level 3 of $tree_2$** 55

4.12 **Confusion matrix of the 11 classes obtained using tree structure $tree_2$(Best accuracies shown here). Features used: SIFT+LLC on uncropped images** 56

4.13 **Tree structure obtained when the number of nearest neighbor $neighbor_{num}$ was 7(For class numbers refer to Table 4.4)** 57

List of Tables

2.1 Comparsion of accuracy results of SRC,KNN,SVM 15

4.1 Hyperparameter ranges for Bayesian optimization 46

4.2 Comparison between SIFT+BOW and SIFT+LLC on uncropped images 47

4.3 Different groupings of the classes in the third levels of 4 different tree structures and their number of occurrences ([5,6] here refers to the grouping of classes 5 and 6) 48

4.4 Class numbers and associated classes 49

4.5 Comparison of the classification accuracies of 4 different tree structures mentioned in Table 4.3 49

4.6 Avg accuracy of two runs of the tree structure $tree_2$ compared with the accuracies reported for FDDL ScSPM, jDL 50

4.7 Avg accuracy of two runs of the tree structure $tree_2$ w.r.t. the accuracies reported for FDDL and ScSPM 50

4.8 Comparsion between the average accuracies of mean based traversal of the tree and maximum margin based learning +tree traversal 51

4.9 Comparison of the average classification accuracies for tree structures for both 1 and 2 in case of grid search and bayesian optimization 51

4.10 Training times associated with each non-leaf node(of $tree_2$) in seconds after the hyperparameters are fixed using bayesian optimization 52

4.11 **Comparison of the testing accuracies between different kernels and the best hyperparameter settings corresponding to each of them** 64

B.1 **Optimal hyperparameters obtained for the nodes of the tree structure $tree_2$ over two runs of bayesian optimization** 72

B.2 **Optimal hyperparameters for the entire tree structure obtained using grid search** 73

Chapter 1

Introduction

Classification problem is one of the classic problems in machine learning domain and it aims at determining the class or category of an unknown sample when the classes of certain given samples called training data are known apriori. There are multiple examples in our daily lives where the need of classification arises e.g. email classification into spam and non spam , diagnosis of a patient , fraud identification etc. More specifically, classification problems fall under the domain of supervised learning problems where the goal is to design a classifier i.e. a mathematical function based on the training operation on a set of correctly labelled samples.

Common types of problems encountered are binary and multiclass classification problems where in the first one the task is to assign the new sample into one of the given classes and the 2nd case involves picking up the particular class among multiple classes (greater than 2).Linear classifers are capable of separating data into disjoint classes if the given data is linearly separable . In cases , when the data points are not linearly separated , non linear classifiers learn complex decision boundaries for separation of the different classes. Some commonly used linear classifiers are Linear Discriminant Analysis, Naive Bayes ,Logistic Regression, perceptron etc. Examples of non-linear classifiers include K-nearest neighbor, SVM with associated kernels etc.

Examples of challenging classification problems in the computer vision domain include facial recognition, object recognition , action classification etc. Facial recognition aims at determining the class or identity of a person , when a database of images are provided and is extensively used in security/surveillance domain. The problem becomes challenging when the database available at hand consists of very few images (which are captured at certain public locations like airports, shopping malls etc) and the recognition process is required to be performed real time. Object recognition is another such difficult task .Humans are able to recognize multitude of objects including familiar and newly encountered ones with minimal effort. This is possible for humans even in the presence of wide variations in shape,color and texture. But for machines accurate object recognition is still a challenging task. Two types of object recognition problems are considered by researchers i.e. determination of the generic category of the object or its specific category. Specific cases involve identification of prominent buildings like Eiffel Tower etc. whereas generic category determination aims at identifying different instances of particular categories like buildings, cars, dogs etc. The differences are shown in Fig 1.1 .

Object recognition pipeline involves gathering training images of the specific classes and determining a model that can make proper predictions for new instances of those classes. Further finer tasks involve bounding box detection around the objects followed by their segmentation. The problem of object recognition is further challenging when the objects from different categories are visually similar to each other. The differences among such visually similar categories are quite subtle and objects across different categories share some common features. For example, the cat category when compared with computer is radically different but when compared with margay(a form of wild cat native to Central America) the differences are quite subtle.

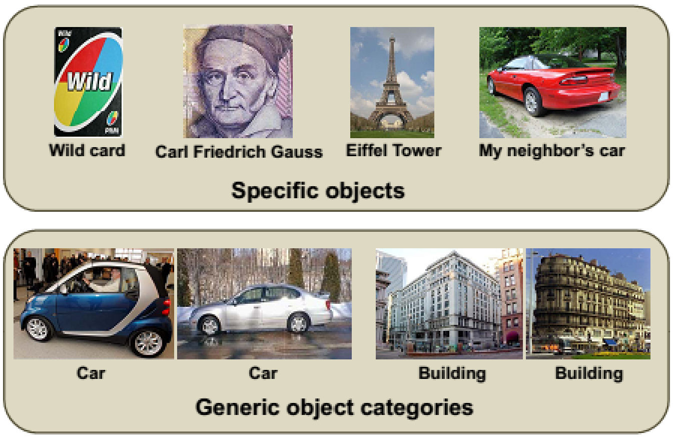


Figure 1.1: **Two types of recognition problems: Identification of specific objects, broad object categories**) (*Image Courtesy: Visual Object Recognition , Kristen Graumann, Bastian Leibe ,Morgan and Claypool publishers 2011*)

The problems that have been tackled are facial recognition, crowd video classification and object recognition . Crowd video classification is an interesting application , whose goal is to classify a set of videos on the basis of activities of the crowd . This classification of the groups is preceded by group identification in the videos along with their activities. The approaches considered are based on sparse dictionary learning and distance metric learning. In case of object recognition we consider the specific cases where certain categories have strong visual correlation among themselves and whether classification of the categories can be done after grouping them in a hierarchial model.

1.1 Related Work

Sparse representation which has proved to be useful in compressing high dimensional signals has been recently used for classification purposes. Wright *et al.* [1] has showed that the sparse representation of the test sample w.r.t a matrix consisting of the training samples stacked as columns followed by minimum residual computation gave competitive performance in the domain of facial recognition. The advent of dictionary learning methods , where a compact dictionary is learnt from the training samples itself has been adapted for classification purposes in recent works like DKSVD [2],LCKSVD [3], FDDL [4] etc. A detailed explanation of the dictionary learning techniques including the variants for classification purposes is given in chapter 2.

In case of object recognition proper image descriptors play an important role in capturing the relevant visual cues. The feature descriptors can be grouped into two types i.e. window based representation where the region of interest (as specified by the window) is summarized by a single descriptor and part based representation where separate descriptors are combined along with geometric layout. SIFT descriptor devised by Lowe[5] used Difference of Gaussian(DoG) at multiple scales for keypoints localization followed by extraction of the descriptors around the keypoints. For determining the descriptor around a keypoint a neighborhood of 16×16 is considered which are further divided into 16 sub-blocks each having size 4×4. Within each such sub-block a 8 bin histogram is created which after concatenating for 16 sub-blocks gives the 128 dimensional keypoint descriptor. The illumination invariance is finally obtained by normalizing the descriptor to unit length. An efficient alternative to SIFT was proposed by Bay *et al.*[6] in the form of SURF (Speeded up Robust Features)which utilized Hessian-Laplace based detector of the regions of interest and approximated the effect of derivative filter kernels using 2D box filters . The SURF descriptor obtains summary statistics for derivatives in both x and y direction resulting in a 64 dimensional length vector. Further improved version of SURF was proposed by Cornelis *et.al* [7] which harnessed the processing power of the GPUs to perform feature extraction for a 640 × 480 image at frame rates upto 200Hz.

In order to obtain a single feature vector for the entire image using the descriptors computed , the standard approach that is followed is known as Bag of Words(BOW) model. The BOW model proposed by Csurka *et al.* [8] creates a visual dictionary (learnt from the descriptors using K-means) and use it to provide a summary of descriptors within the image in the form of a 1-D histogram. This BOW model has shown to be quite effective in certain object recognition tasks like PASCAL VOC challenge by Everingham *et al.*[9] and Scene recognition by Fei-Fei *et.al* [10]. Since BOW is an orderless representation of an image Lazebnik *et al.* [11] extended it further by computing a spatial pyramid representation of the image and concatenating the BOW histogram in each bin of the pyramid to obtain a pyramid feature vector for the entire image. In order to use a linear SVM for classification , Yang *et al.* [12] developed an extension of the Spatial Pyramid Scheme called ScSPM by using a sparse coding scheme for learning the vocabulary (instead of using Kmeans) and max pooling the sparse codes across multiple scales in the spatial pyramid to obtain feature vector representation of the images. The computational complexity of ScSPm was tackled in Wang *et al.*[13] work , where instead of sparse codes Locality constrained linear coding was used followed by similar max-pooling strategy to obtain the feature vectors.

With the advent of the advanced feature extraction techniques, many state of the art algorithms have been proposed for object recognition problems. Attempts have been made to improve the standard K-NN classification scheme by incorporating distance metric learning in Chopra *et al.* [14], Goldberger *et al.* [15] et.al. Weinberger *et al.* [16] took the idea of distance metric learning further by obtaining a Mahalnobis distance metric and testing on problems like facial recognition, handwriting recognition, text categorization etc.(Details of this algorithm are included in the section 4.2). Recent efforts have been made in the direction of large scale visual recognition challenge , made popular by the large scale Image database known as Imagenet [17], which contains around 14 million images for 22K categories of varying types. Krizhevsky *et al.* [18] trained a deep convolutional neural network model for classifying the 1.2 million images in the ImageNet LSVRC 2010 challenge. Bengio *et al.*[19] proposed a tree structure based classifier by minimizing overall tree loss and applied it to Imagenet dataset. Deng *et al.*[20] also proposed a label tree scheme which simultaneously learnt the tree structure and classifiers per node and obtained balanced trees as compared to Bengio's scheme.

With the increasing popularity of the supervised dictionary learning schemes as listed in the previous chapter, Zhou *et al.* [21] proposed a joint Dictionary learning scheme with the aim of classifying visually correlated categories. In order to tackle visually correlated categories the first step involved detection of the groups among the image classes. (This was performed using standard clustering techniques where each cluster was a disjoint group). The identification of the groups was followed by learning of two specific type of dictionaries i.e. the group dictionary and the dictionaries coressponding to the classes within the group. The group dictionary was responsible for learning the shared features among the classes in the group. The learning phase was followed by classification of test samples which followed a two stage process of identifying the particular group followed by its specific class.

1.2 Organization of Thesis

The entire thesis is organized in 5 chapters. The first chapter gives a brief introduction about the problems considered and the related works in the respective domains. The second chapter explains the basic Sparse Representation based classification(SRC) in detail followed by its application in facial recognition , crowd video classification .Comparison of SRC has been done with the state of the art classifiers like Support Vector Machines , K-nearest neighbor. The third chapter gives an idea about the dictionary learning techniques and its applicability in classification problems. In the fourth chapter the problem of object recognition of group of visually correlated categories was considered. In order to exploit the visual correlations among the categories Self Tuning Spectral Clustering was used in a recursive manner to organize the categories in a hierarchial fashion(tree structure). The hierarchial organization was exploited using a large margin nearest neighbor classification scheme at the location of each tree node followed by an energy based classification model. Apart from a nearest neighbor classification scheme , a node specific variant of joint classifier and dictionary learning was also suggested. The final chapter concludes the work by providing directions for future work.

Chapter 2

Sparse Representation based Classification(SRC)

Sparse representation based classification provides new insight into a general classification algorithm based on sparse representation which can be used for variety of classification problems. The details of the algorithm are described below followed by applications in facial recognition and crowd video classification.

2.1 Sparse Model for classification

Facial recognition problem considers each image as a matrix of dimensions $m \times n$. For the purpose of processing each image is converted into a vector of dimensions $mn \times 1$. For a facial recognition problem with C classes the vectors associated with the images can be stacked together columnwise to obtain a submatrix $T_c = [I_{c,1}, I_{c,2}, I_{c,3}, ..., I_{c,n_c}]$, where $I_{c,j}$ is the j^{th} image associated with the c^{th} class. This T_c can be considered as the training set for the c^{th} class. The overall training set $T \in R^{mn \times N}$ is then given by :

$$T = [T_1, T_2, T_3, ..T_c..., T_C] \tag{2.1}$$

Given that sufficient number of samples are available for each class , the test sample y belonging to the c^{th} class can be expressed in terms of the linear combination of the members of the c^{th} class i.e. by the columns of T_c. The test sample y can be expressed as :

$$y = \sum_{i=1}^{n_c} x_{c,i} I_{c,i} \tag{2.2}$$

This linear representation of y can be further extended to the training samples associated with the overall training set T and can be written as :

$$x = [0, ..., 0, x_{c,1}, x_{c,2}, x_{c,3}, ...x_{c,n_c}, 0.....0] \tag{2.3}$$

Here the zero entries correspond to the entries associated with classes other than the c^{th} class. Using the above representation of x the test image y can be expressed as :

$$y = Tx \tag{2.4}$$

Thus given a test sample y the entire training set T can be used to obtain x , which is sparse in nature. The identity of the test image y can be determined using the nature of x. The system of equations given by (2.4) is undetermined in nature since for the matrix T we have $mn < N$ .Considering that the matrix T has full rank , an infinite number of solutions can exist.

Conventional approach involves obtaining the solution having minimum l_2 norm. The minimum l_2 norm solution is obtained using the following expression:

$$x^* = argmin\|x\|_2 \quad subject\ to \quad y = Tx \tag{2.5}$$

The closed form solution of the problem is obtained as $x^* = T^t(TT^t)^{-1}y$. The minimum l_2 norm solution can be dense in nature due to which large number of entries might correspond to other classes , thus making it difficult to determine the identity of the test image y. Since the test vector can be represented using training samples from the same class the solution vector x as seen in (2.3) is sparse. So the solution having minimum l_0 norm is the desired one.The minimum norm solution is obtained by

$$x^* = argmin\|x\|_0 \quad subject\ to \quad y = Tx \tag{2.6}$$

In case that there is a solution x of the system of equations $y = Tx$ such that it satisfies $\|x\|_0 < 1/2(1 + 1/\mu(A))$ where ($\mu(A)$ denotes the maximum normalized dot product between the columns of A) then this solution is the sparsest possible [22]. But the above problem is NP hard in nature. It has been shown in [23] that if the solution to (2.6) is sufficiently sparse in nature then a good approximation to the solution of the l_0 minimization problem can be obtained using the following l_1 minimization problem.This holds in a stable manner in case of noisy data too.

$$x^* = argmin\|x\|_1 \quad subject\ to \quad y = Tx \tag{2.7}$$

In presence of noise factor e the above problem can be modified as:

$$y = Tx + e \tag{2.8}$$

So the modified optimization problem ,where $\|e\| \leq \epsilon$ is given as :

$$x^{*} = argmin\|x\|_1 \quad subject\ to \quad \|Tx - y\| \leq \epsilon \tag{2.9}$$

2.2 SRC classification scheme

The minimum l_1 norm solution provides a sparse representation of the test vector in terms of columns of the overall training set T. Using (2.9) the sparse representation of the test vector y is obtained and passed to a function $\delta_i(x)$ which is defined as :

$$\delta_i(x)_{=}[0, ..., 0, x_{i,1}, x_{i,2}, x_{i,3}, ...x_{i,n_i}, 0.....0]^t \tag{2.10}$$

Thus the function $\delta_i(x)$ contains the coefficients of x corresponding to class i. The reconstruction error of y w.r.t class i is then determined by

$$r_i(y) = \|y - T\delta_i(x)\|_2 \tag{2.11}$$

The class of the test vector is then given by $class(y) = argmin_i r_i(y)$. Steps of the SRC Algorithm are summarized below:

Algorithm 1 SRC algorithm

```
1.Input: Test vector $y$ , Training matrix $T$ with the columns $l_2$ normalized .
2. Solve the $l_1$ minimization problem:
$x^{*} = argmin\|x\|_1 \quad subject\ to \quad \|Tx - y\| \leq \epsilon$
3.Compute the reconstruction error w.r.t different classes $r_i(y) = \|y - T\delta i(x)\|_2$ for $i = 1, 2, ...C$
4.Output: Class of y=$argmin_i\ r_i(y)$.
```

2.3 Geometrical Explanation

The $l_p(p > 0)$ norm of the vector $x = [x_1, x_2, ...x_n]^T$ is defined as :

$$\|x\|_p = (\sum_{i=1}^{n} |x|^p)^{1/p} \tag{2.12}$$

Putting $p = 0$ the l_0 norm is defined as the number of non-zero entries in the vector x. Considering the problem being taken into account in SRC(Sparse Representation based Classification) the problem being solved is of the following form P_p.

$$(P_p): \ min\|x\|_p \ subject \ to \ Tx = y \tag{2.13}$$

The geometrical explanation for success of the (l_p $p \leq 1$) norm minimization in recovering sparse solutions can be attributed to the following figure: The constraints given by $Tx = y$ forms a hyperplane which reduces to a line in 2 dimensions. In case of obtaining the minimum l_p norm solution the l_p ball($\|x\|_p \leq c$) is expanded until it meets the hyperplane . As seen from the figure ,the $l_p(p \leq 1)$ norm balls intersect the hyperplane at the corners of the ball , particularly at one coordinate axis, thus promoting sparse solutions. For $p \geq 1$ the point of intersection is not on the axis which can be seen from the figure 2.1 for $p = 1.5$ and $p = 2$.

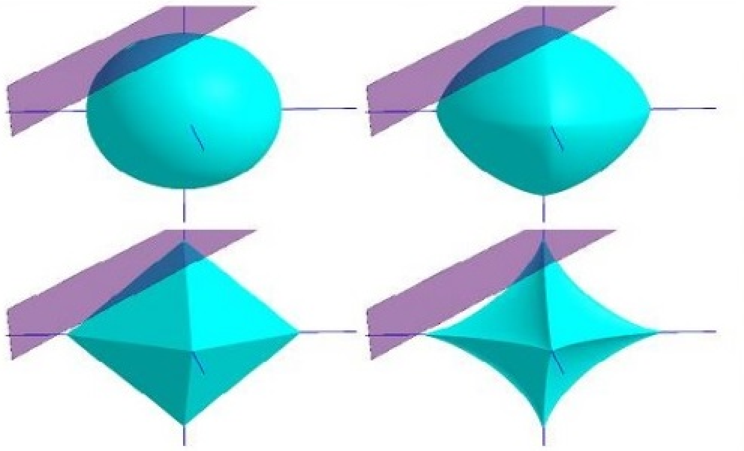


Figure 2.1: **The intersection of the l_p ball and the plane $y = Tx$ (Top left(p=2),Top right(p=1),bottom left(p=1),bottom right(p=0.7))** (*Image Courtesy: Sparse and Redundant Representation, Michael Elad, Springer, December, 2009*)

2.4 Results

The L1-minimization has been performed using L1-magic [24] and the simulations have been carried out using MATLAB. For implementing SVM LibSVM [25] pacakge has been used with one vs all setting.For k-NN classifier the number of nearest neighbors was chosen as 5. Two applications were tested i.e.facial recognition and crowd video classification.

2.4.1 Facial Recognition

The effectiveness of the SRC algorithm has been tested on the two standard Facial databases i.e. Yale Database and AR database. Specific emphasis has been given to the importance of the feature extraction schemes in case of the facial databases. Comparison has been done with SVM and K-NN(K-nearest neighbor classifier)

Feature extraction and SRC

In this section SRC algorithm's performance has been investigated w.r.t different feature extraction schemes like commonly used Eigenfaces and two unconventional schemes i.e. randomfaces and downsampled images. The images stacked as columns in the training set T result in high dimensions(like for AR database ,mn for the training set T is 19800). Dimensionality reduction is done by pre-multiplying using a matrix $R \in \mathbb{R}^{d \times mn}$.Then the training set is modified to $T^* = RT$. The correct solution is recovered using :

$$x^* = argmin\|x\|_1 \quad subject\ to \quad \|RTx - y\| \leq \epsilon \tag{2.14}$$

Details about the feature extraction processes are listed below:

1. **Randomfaces**

   For random faces a transform matrix R is generated where the entries are sampled from zero mean normal distribution and each row is normalized to unit length. The row vectors can be considered as d random faces.

2. **Eigenfaces**

   Standard PCA technique has been applied to reduce the dimensionality of the problem to d. The eigenvalues and eigenvectors of the covariance matrix of the training set T were computed and the topmost d eigenvalues and eigenvectors were selected to obtain a set of d eigenfaces given by $Projmat = [v_1, v_2, ..., v_d]$.The projection of the data into low dimensional subspace is obtained using $R = Projmat^t$.

3. **Downsampled features**

   Downsampled features are obtained by downsampling the image at a specific interval. For an interval of n the nth samples across the row and columns are selected resulting in a reduced dimension for the training set images.

The recognition rates for different feature extraction schemes have been plotted with the different feature dimensions for both AR and Yale database.

Extended YaleB Database Results:

Extended YaleB database consists of 2414 images of 38 individuals withe each image having a size of 192×168. For each class 70% of the images were selected as training set and rest were considered test images .As listed above the various feature extraction schemes have been applied with the dimensions being considered as 30, 56, 120 and 504. The recognition rates for different feature extraction schemes have been plotted with the different feature dimensions for the Extended YaleB database. SRC's performance has been compared with SVM and K-NN classifier and the results have been shown in Fig 2.3 for feature space dimension of 504(in each of the feature extraction).SRC clearly outperforms SVM and K-NN in all the three cases considered.

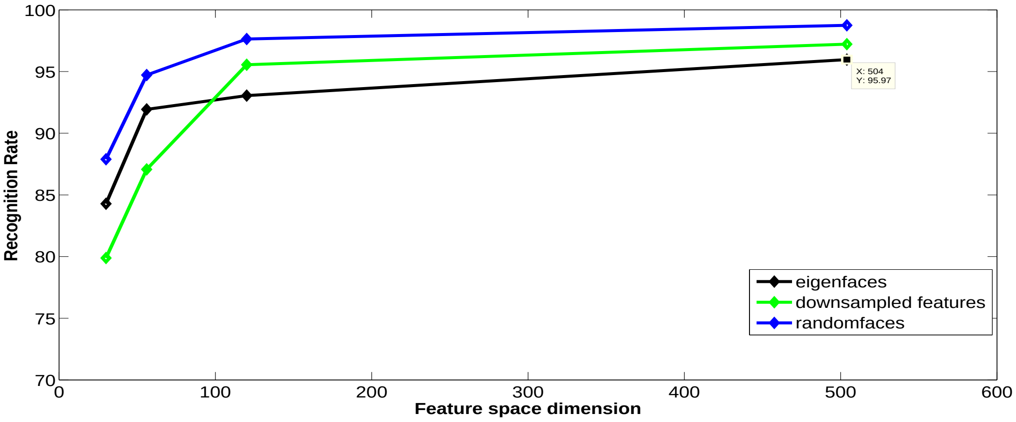


Figure 2.2: **Recognition rates on Extended YaleB database for varying feature dimensions**

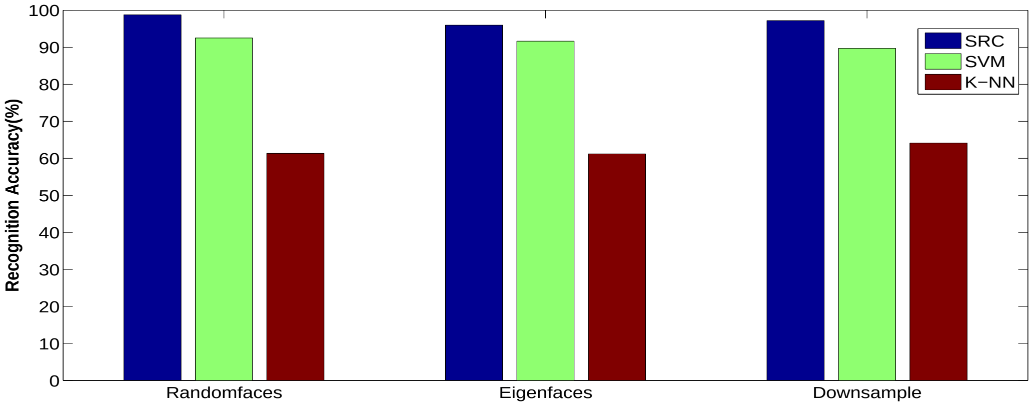


Figure 2.3: **Comparison of the accuracies of SRC,SVM,K-NN classifier for feature space dimension 504(Extended YaleB database)**

AR Database Results:

AR database [26] consists of 2600 images of 100 individuals(50 Male and 50 Female) with each image having a size of 120 × 165. For each class 70% of the images were selected as training set and rest were considered test images .As listed above the various feature extraction schemes have been applied with the dimensions being considered as 30, 54, 130 and 540. The recognition rates for different feature extraction schemes have been plotted with the different feature dimensions for the AR database. SRC's performance has been compared with SVM and K-NN classifier (in case of AR database) and the results have been shown in Fig 2.5 .SRC's better performance w.r.t SVM and K-NN can be seen in all the three cases considered.

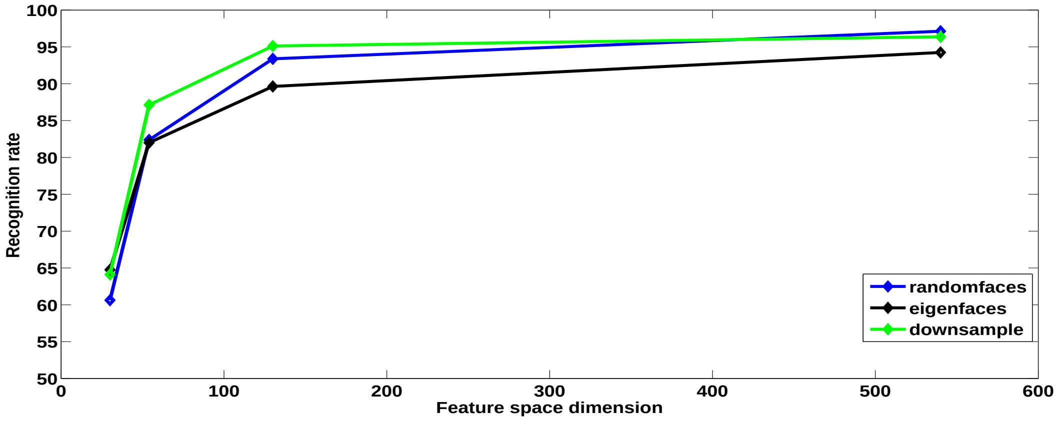


Figure 2.4: **Recognition rates on AR database for varying feature dimensions**

Some observations about the feature extraction processes and performance of SRC :

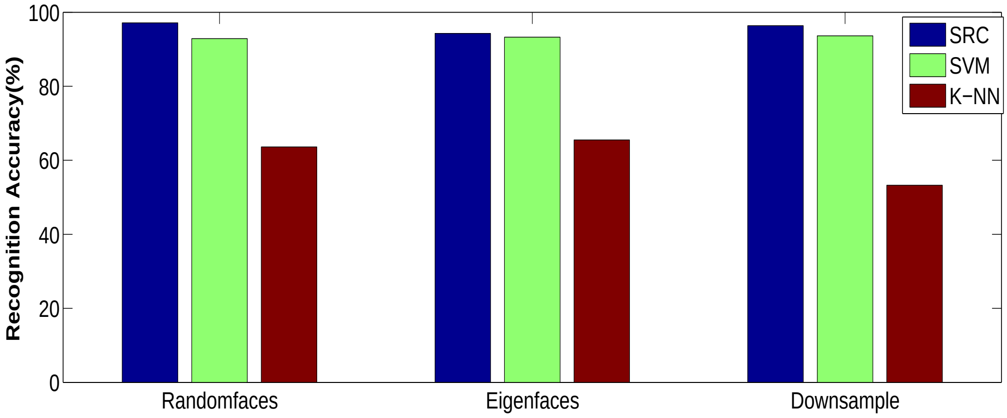


Figure 2.5: **Comparison of the accuracies of SRC,SVM,K-NN classifier for feature space dimension 504(AR database)**

- Random faces are less structured as compared to eigen-faces, fisher faces or downsampled features. It is possible to obtain high accuracy for facial recognition even after using randomly selected features instead of carefully selecting different facial features.

- The performance of the random projections can be attributed to the bound given on the number of measurements of the feature extraction matrix R. In [27] it has been proved that if the solution x has $k \ll N$ number of non-zero entries , then with high probability the correct solution x can be obtained if the dimension d of the matrix R satisfies the following constraint:

$$d \geq 2\ k\ log(N/d) \tag{2.15}$$

- Correctness of the solution x will ensure similar performance for all feature extraction techniques used. Dimension d when exceeds the bounds given by (2.15) will ensure similar performance. This can be seen in the figure Fig 2.2.

- So random faces can be considered as an alternative for the other feature extraction schemes like eigen faces, fisher faces which are computationally expensive. The computation of the random matrix can be done only once and doesnot depend on the training set.

2.4.2 Crowd Video Classification

Here SRC framework has been applied to the case of classification of crowd videos using the CUHK crowd dataset [28]. CUHK dataset consists of 474 videos and 8 classes consisting of the following :

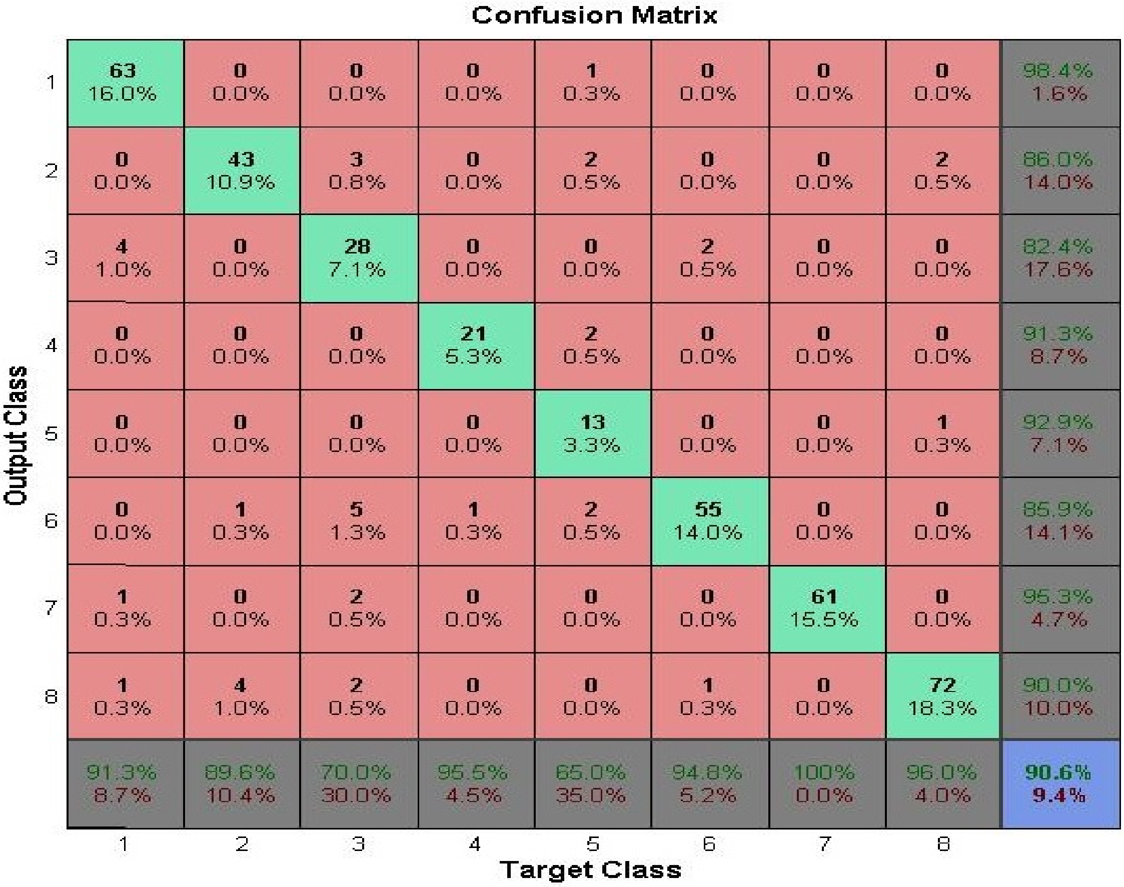
- Mixed crowd
- Well organized crowd following mainstream
- Not well organized crowd following mainstream
- Crowd merge
- Crowd split
- Crowd crossing in opposite directions
- Intervened escalator traffic
- Smooth escalator traffic

Detection of the groups were followed by extraction of group level features like group density ,histogram informations of direction of the groups and their activities .The group level features were used to obtain feature vectors for the video clips. Using a 70 -30 % split for the training and test data, the results of SRC were compared with SVM and K-NN classifier. The confusion matrix plot for the SRC's performance has been given below: From the confusion matrix it can be seen that the classification accuracy for the class 5 is low as compared to others due to the insufficient number of training examples associated with class 5 . In the following figure the classification accuracies have been plotted with varying sizes of the training data set.

SRC	SVM	K-NN
90.58 %	84.14 %	82.72 %

Table 2.1: **Comparsion of accuracy results of SRC,KNN,SVM**

From the figure(2.7) it can be inferred that as the size of training sample decreases it is difficult for the test samples to be represented properly by the set of training samples, thereby reducing the classification accuracy.



	1	2	3	4	5	6	7	8	
1	63 16.0%	0 0.0%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	0 0.0%	0 0.0%	98.4% 1.6%
2	0 0.0%	43 10.9%	3 0.8%	0 0.0%	2 0.5%	0 0.0%	0 0.0%	2 0.5%	86.0% 14.0%
3	4 1.0%	0 0.0%	28 7.1%	0 0.0%	0 0.0%	2 0.5%	0 0.0%	0 0.0%	82.4% 17.6%
4	0 0.0%	0 0.0%	0 0.0%	21 5.3%	2 0.5%	0 0.0%	0 0.0%	0 0.0%	91.3% 8.7%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	13 3.3%	0 0.0%	0 0.0%	1 0.3%	92.9% 7.1%
6	0 0.0%	1 0.3%	5 1.3%	1 0.3%	2 0.5%	55 14.0%	0 0.0%	0 0.0%	85.9% 14.1%
7	1 0.3%	0 0.0%	2 0.5%	0 0.0%	0 0.0%	0 0.0%	61 15.5%	0 0.0%	95.3% 4.7%
8	1 0.3%	4 1.0%	2 0.5%	0 0.0%	0 0.0%	1 0.3%	0 0.0%	72 18.3%	90.0% 10.0%
	91.3% 8.7%	89.6% 10.4%	70.0% 30.0%	95.5% 4.5%	65.0% 35.0%	94.8% 5.2%	100% 0.0%	96.0% 4.0%	90.6% 9.4%

Figure 2.6: **Confusion matrix plot for the CUHK database classification using SRC**

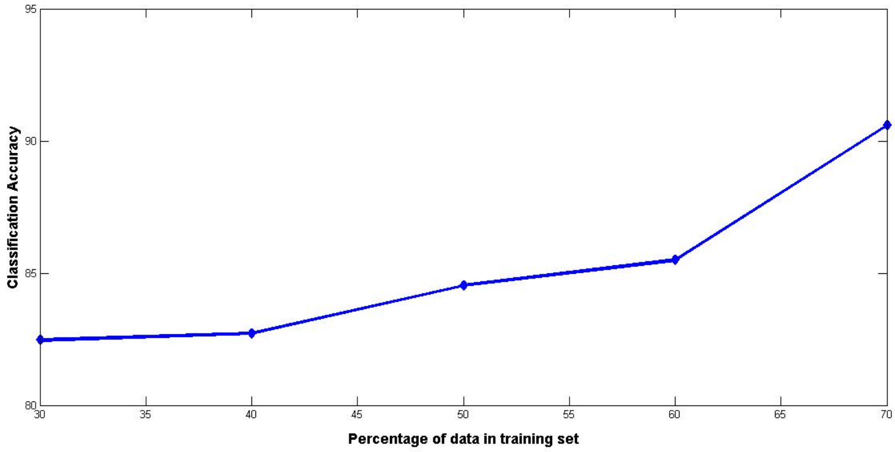


Figure 2.7: **Variation of Classification accuracy with changing percentage of data in training set**

Chapter 3

Dictionary Learning

The large amount of information being captured every second results in a huge amount of high dimensional information .But most of the information is redundant and relevant information lies in a lower dimensionality subspace. In certain cases the number of instances is much more than the dimensionality of each such instance leading to an overcomplete representation.

For such overcomplete representation, Dictionary learning aims at determining the proper representation of the data sets by the usage of reduced dimensionality subspaces.In case of dictionary learning sparsity constraint plays a key role as they help in identification of the most relevant representations of the observed data. This section gives an overview of the standard dictionary learning algorithms and certain variants suited for classification purposes.

3.1 Dictionary Learning Methods

The earliest work in the field of dictionary learning involves probabilistic methods such as maximum likelihood (ML) dictionary learning , maximum a posteriori (MAP) dictionary learning etc. These algorithms iteratively solve the dictionary learning problem and use two step optimization process. Each iteration involves computing sparse coefficients for the input signals and updating the dictionary. Another approach is known as the method of optimal directions (MOD) which came after MAP and MP based algorithms and was inspired by the two step optimization used by these algorithms. Recently KSVD algorithm proposed by Aharon *et al.* [29] based on the generalization of the K-Means algorithm has resulted in lot of applications in classification,denoising etc.

3.1.1 Method of Optimal Directions

MOD algorithm is an iterative algorithm with alternating sparse coding and dictionary update stages.For updating the dictionary at each stage the frobenius norm of the error is minimized where the reconstruction error is given by:

$$\|E\|_F^2 = \|Y - DX\|_F^2 \tag{3.1}$$

In order to obtain the updated dictionary D the derivative of (3.1) is done w.r.t D which results in the following expression:

$$D^{(n+1)} = Y(X^n)^t(X^n(X^n)^t)^{-1} \tag{3.2}$$

3.1.2 KSVD algorithm

KSVD algorithm is based on the following objective function:

$$\begin{aligned} &\underset{D}{\text{minimize}} && \|Y - DX\|_F^2 \\ &\text{subject to} && \|x_i\|_0 \leq k_0, \ \forall i. \end{aligned} \tag{3.3}$$

KSVD Algorithm details:

KSVD algorithm is based on two stages :

- **Sparse Coding stage** In the sparse coding phase the dictionary D is kept fixed and the problem reduces to just sparse coding on the coefficients. The error term in (3.3) can be simplified as:

$$\|Y - DX\|_F^2 = \sum_{i=1}^{N} \|y_i - Dx_i\|_2^2 \tag{3.4}$$

Following the above simplifications there are N decoupled problems given as:

$$\begin{aligned} &\underset{x_i}{\text{minimize}} && \|y_i - Dx_i\|_2^2 \\ &\text{subject to} && \|x_i\|_0 \leq k_0, \quad i = 1, 2....N. \end{aligned} \tag{3.5}$$

This problem of sparse coding is solved using OMP(Orthgonal Matching Pursuit)

- **Dictionary Update** The dictionary update step of KSVD algorithm is its main part where the dictionary columns(atoms) are updated on a column basis (i.e. one at a time). The reconstruction error in dictionary learning phase can be written as :

$$\|Y - DX\|_F^2 = \left\|Y - \sum_{i=1}^{K} d_i x_T^i\right\|_F^2 = \left\|(Y - \sum_{i=1, i\neq k}^{K} d_i x_T^i) - d_k x_T^k\right\|_F^2 \tag{3.6}$$

The above expression can be further simplified to

$$\left\|(Y - \sum_{i=1, i\neq k}^{K} d_i x_T^i) - d_k x_T^k\right\|_F^2 = \|E_k - d_k x_T^k\|_F^2 \tag{3.7}$$

The matrix E_k is basically the error for all the examples when the kth atom is removed. SVD can be obtained to find d_k and x_k^T as SVD will find the closest rank-1 matrices which will minimize the error in (4.7). But such an update of x_k^T will ensure that x_k^T is filled. In order to simplify the problem a group of indices called ω_k is defined, which points to the examples in Y that use the dictionary atom d_k i.,e. where the entries in x_k^T are nonzero.Hence

$$\omega_k = \{i | 1 \leq i \leq K, x_T^k(i) \neq 0\} \tag{3.8}$$

Defining Ω_k as a matrix of size $N \times |\omega_k|$ with ones on the $(\omega_k(i), i)$ entries and zeros elsewhere. Using this matrix Ω_k the following are obtained :

$$x_k^R = x_T^k \Omega_k, \quad Y_R^k = Y\Omega_k, \quad E_R^k = E\Omega_k \tag{3.9}$$

x_k^R shrinks the vector x_k^T by keeping only the non-zero entries. Similarly Y_k^R creates a matrix consisting of only those columns of Y which uses the d_k. Now the the following function needs to be minimized

$$\|E_k\Omega_k - d_k x_T^k \Omega_k\|_F^2 = \|E_k^R - d_k x_R^k\|_F^2 \tag{3.10}$$

Now the SVD of the matrix E_k^R is computed$(U\Delta V^t)$. Updating of the dictionary atom and the coefficient vector is done by $d_k = u_1, x_R^k = v_1\Delta(1,1)$. Thus the columns of the dictionary remain normalized and support either remains the same or gets smaller.

Since K-Means clustering algorithm involves K computations of the mean here in KSVD the updating of the columns is done K number of times in a column wise fashion.

3.2 Dictionary Learning for Classification approaches

3.2.1 Dictionary Learning approaches

Dictionary learning algorithms primarily aim at minimizing the reconstruction error and not suited for classification and hence can be termed as unsupervised schemes. Advantage of the dictionary learning scheme can be verified by the simple functioning of the SRC algorithm in the previous chapter where the dictionary is composed of all the training images of different classes stacked together in a column wise manner.The SRC algorithm even though it performs well may result in a large dictionary when the number of training images increase in size.

By exploiting the class labels the dictionary learning schemes can be adapted for classification purposes. The strategies adopted are listed below:

1. **Learning class specific dictionaries**:

   This scheme aims at making the dictionaries compact and discriminative(w.r.t each class) by learning class specific dictionaries. Yang *et al.* [30] proposed a scheme of Metaface Learning to class specific dictionaries .Further improvements were made by Ramirez *et al.* [31] ,who merged the class specific dictionary learning scheme along with an incoherence term to promote independence among the dictionaries. This scheme lacks the benefit of making the representation coefficients discriminative.

2. **Learning shared dictionary and supervised updating of the coefficients**: In this scheme instead of learning class specific dictionaries a shared dictionary is being learnt and discrimination power of the coefficients are being improved. Mairal *et al.* [32] proposed a supervised dictionary learning framework by imposing a logistic loss function on the sparse coefficients. A label consistency term for the coefficients has been added to KSVD framework along with joint learning of the classifier in LCKSVD [3]. The benefits of discriminative dictionaries are lost here.

3. **Learning class specific dictionaries and supervised updating of coefficients**:

   This scheme has the advantages of both the scenarios i.e. the compact class specific dictionaries and discriminative nature of the coefficients .FDDL [4] learns both class specific dictionaries and imposes discrimination on the coefficients by minimizing within class scattering and maximizing across class scattering. DLCOPAR [33] apart from learning both class specific dictionaries and discriminative coefficients also preserve the common atoms among the classes by maintaining a shared dictionary.

Thus the general dictionary learning framework can be summarized as follows by incorporating all the constraints:

$$\begin{gathered}(D, X, W) = \underset{D,X,W}{arg\min} C(H, Y, D, X) + \eta f(W, X, H) + \lambda_x h_X(X) + \lambda_W h_W(W) \\ \text{subject to } constraint\ on\ the\ columns\ of D\end{gathered} \tag{3.11}$$

where H represents the label matrix of the set of training samples Y , W represents the matrix of the classifier parameters , X the sparse coefficients . Here $C(H, Y, D, X)$is the conventional dictionary learning framework and includes the reconstruction cost associated with the dictionary learning framework. $f(W, X, H)$ is the discrimination term on the coefficients X and includes the classifier training term . $h_X(X)$ includes the sparsity constraints on the coefficients and $h_W(W)$ the constraints on the classifier parameters.

3.2.2 Class specific KSVD learning

In this section a KSVD based class specific dictionary learning approach has been tested for varying dictionary sizes.

KSVD can be utilized to learn class specific dictionaries by applying the dictionary learning framework on the training data associated with each class.

Algorithm 2 Class specific KSVD based dictionary learning algorithm

```
Input: Test vector $y$ , Training matrix $T$ , total dictionary size $dictsize$ , class number $classnumber$.
$dict\_class\_size = round((dictsize)/classnumber)$
$Dict\_tot = []$
for $c = 1 : classnumber$ do
    $Dict\_i = KSVD(T\_i)$
                ▷ ($T\_i$ is the submatrix obtained from T associated with $ith$ class )
    $Dict\_tot = [Dict\_tot\ Dict\_i]$
end for
```

Algorithm 3 Class specific KSVD -part 2

```
9: Solve the $l_1$ minimization problem: $x^* = argmin\|x\|_1 \quad subject\ to \quad \|Dict\_tot\ x - y\| \leq \epsilon$
10: Compute the reconstruction error w.r.t different classes $r_i(y) = \|y - Dict\_tot\ \delta_i(x)\|_2$ for $i = 1, 2, ...C$
11: **Output:** Class of y=$argmin_i\ r_i(y)$.
```

Here $\delta_i(x)$ is the same used in Algorithm 1 and defined in Eq (2.10).

Results

In this section the variation of recognition accuracies for the facial recognition problem of Extended Yale B database with the varying sizes of dictionaries has been considered. Since Extended Yale B database has 38 classes the number of dictionary atoms have been varied in the range $[380, 570, 760, 950, 1140]$ i.e. $[10, 15, 20, 25, 30]$ atoms per dictionary using randomly projected features having dimension 120. In case of AR database which has 100 classes the number of dictionary atoms have been varied in the range $[600, 1000, 1400, 1800]$ i.e. $[6, 10, 14, 18]$ atoms per class dictionary.The resulting variation of the recognition accuracies have been given in the following figures:

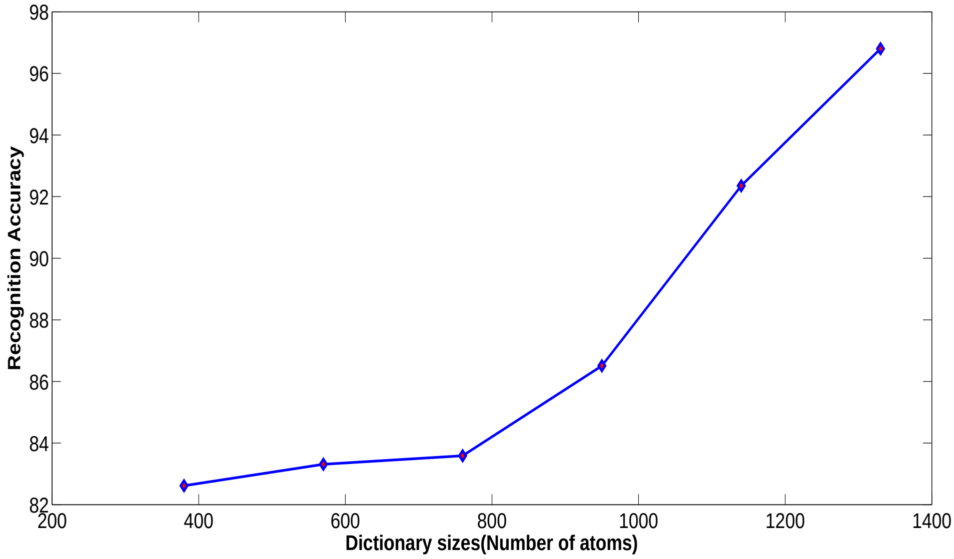


Figure 3.1: **Variation of recognition accuracy with the number of dictionary atoms of Yale database**

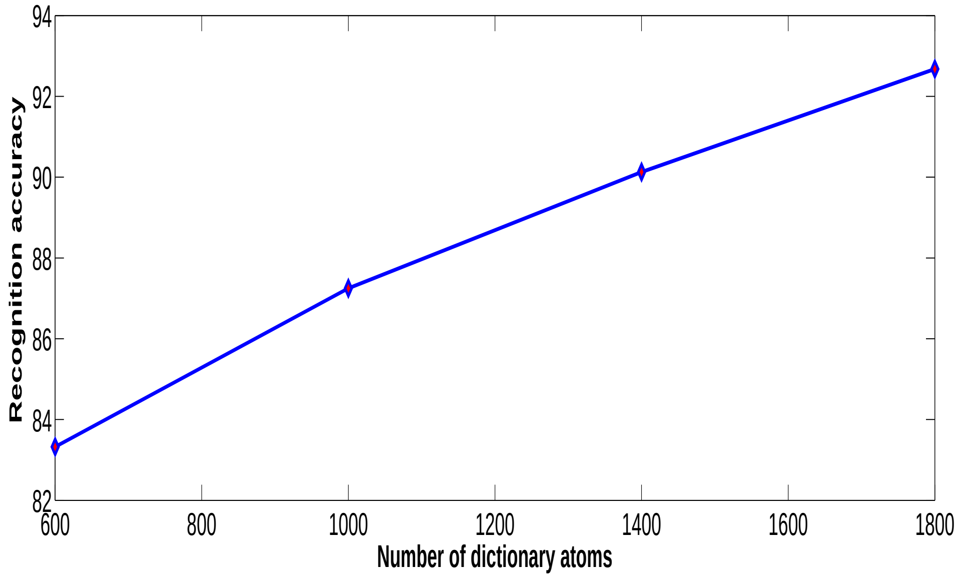


Figure 3.2: **Variation of recognition accuracy with the number of dictionary atoms of AR database**

Some general observations regarding the class specific KSVD dictionary learning:

- This approach takes into account learning of the class specific dictionaries without putting any additional discrimination constraints on the sparse coefficients.
- From the figure it can been that as the number of atoms in the dictionary increase the recognition accuracy increases . But with increase in the number of dictionary atoms the solver faces some problem in finding the correct solutions due to the large number of atoms in the dictionary.
- KSVD is a heuristic algorithm and thus the dictionary obtained for the entire set of samples in the database might not be the optimal one for representing the entire set of samples. Hence the performance remains inferior as compared to SRC algorithm

Chapter 4

Object Recognition of Visually Correlated Categories using Hierarchial model

4.1 Problem Statement

Given a set of image categories $C_1, C_2, C_3, \ldots C_k$ having visual correlation among themselves , goal is to exploit the visual correlations in order to automatically group the classes in a hierarchial model followed by utilizing the model for classification.

4.2 Distance metric learning for Large margin nearest neighbor classification

Distance metric learning for large margin classification , proposed by Killian Weinberger [16] aims at learning a Mahalnobis distance metric for improving the scheme of K nearest neighbor . The Mahalnobis metric learnt acts as a global transformation of the input space before the k-NN classification scheme is used. Further in practice it has been shown that instead of k-NN classification scheme after Mahalnobis metric learning , energy based classification gives better test results.

4.2.1 Distance Metric Learning

Introduction

A mapping denoted by $D : \mathscr{X} \times \mathscr{X} \rightarrow R_0^+$ over a vector space $\mathscr{X}$ is called a metric if vectors $\forall x_i, x_j, x_k \in \mathscr{X}$ the following properties are satisfied:

1. $D(x_i, x_j) + D(x_j, x_k) \geqslant D(x_i, x_k)$
2. $D(x_i, x_j) \geqslant 0$
3. $D(x_i, x_j) = D(x_j, x_i)$
4. $D(x_i, x_j) = 0 \Leftrightarrow x_j = x_i$

If the mapping is such that the first three conditions ae satisfied but not the fourth then the metric is termed as pseudometric. After performing a linear transformation given by $x^{'} = \mathbf{L}x$ the distances are computed as :

$$D_{\mathbf{L}}(x_i, x_j) = \|\mathbf{L}(x_i - x_j)\|_2^2 \tag{4.1}$$

In order to show that the above equation determines a valid metric the matrix $\mathbf{L}$ must have full rank. The above equation can be modified to obtain the following expression :

$$D_{\mathbf{L}}(x_i, x_j) = \|\mathbf{L}(x_i - x_j)\|_2^2 = (\mathbf{L}(x_i - x_j))^T(\mathbf{L}(x_i - x_j)) = (x_i - x_j)\mathbf{L}^T\mathbf{L}(x_i - x_j) \tag{4.2}$$

Denoting $\mathbf{L}^T\mathbf{L} = \mathbf{M}$ the squared distances in terms of matrix $\mathbf{M}$ is given by :

$$D_{\mathbf{M}}(x_i, x_j) = (x_i - x_j)^T\mathbf{M}(x_i - x_j) \tag{4.3}$$

The matrix M is positive semi definite and the metric defined above is called as Mahalnobis distance metric. The Mahalnobis distance metric can be either parameterized by $\mathbf{M}$ or $\mathbf{L}$. Using $\mathbf{L}$ the matrix $\mathbf{M}$ can be determined uniquely but from $\mathbf{M}$ the matrix $\mathbf{L}$ can be recovered upto a rotation. In other words if we consider $L_1 = RL$ where R is an orthogonal matrix then we can say for a positive semidefinite matrix $\mathbf{M}$ the following:

$$\mathbf{M} = {L_1}^T L_1 = (RL)^T(RL) = L^T R^T R L = L^T L \tag{4.4}$$

Hence the distance computed remains unchanged in equation (4.2) even if the matrix $\mathbf{L}$ is recovered upto a rotation matrix R. Thus there are two possible approaches for distance metric learning where either the linear transformation $\mathbf{L}$ or positive semidefinite matrix $\mathbf{M}$ can be estimated.

4.2.2 Model

Basic idea

The basic premise of the algorithm is to improve the K-NN classification scheme by ensuring that a training sample x_i shares its label y_i with its k nearest neighbors and the training samples with different labels must be widely separated. A linear transformation must be learnt such that the training samples after transformation must satisfy these properties. For a particular training sample xi two sets of training samples are maintained:

- **Target Neighbors:**
  Target neighbors are determined beforehand for each training sample x_i. Target neighbors are obtained by determining those k-nearest neighbors of a training sample x_i which share the same label as x_i. The target neighbors are those which are desired to be closer to x_i. These target neighbors establish a boundary around x_i that should not be invaded by training samples having different labels.

- **Impostors:**
  Impostors refer to those training samples which have different labels than x_i but intrude the perimeter set by the target neighbors of x_i . The number of impostors should be minimized in the learning process.

In order to ensure that classification is robust a stringent criteria is imposed in terms of a margin around the decision boundary. The idea of margin is incorporated in the definition of imposters in the following way:
Given a training sample x_i with label y_i,its target neighbor and impostor are denoted by x_j and x_l respectively. Then for the impostor x_l (which has different label $y_l \neq y_i$) the following inequality holds:

$$\|\mathbf{L}(x_i - x_l)\|_2^2 \leq \|\mathbf{L}(x_i - x_j)\|_2^2 + 1 \tag{4.5}$$

Thus any sample which has a different label than x_i and invades the perimeter set by target neighbor x_j plus 1 is considered an impostor. Thus the main goal of the learning is to push the impostors away from the perimeter set by the target neighbor after using a linear transformation **L**. The following figure highlights the scenario before and after the learning process . Before learning the impostors denoted by the red and blue marked samples lie within the perimeter and after using the learnt linear transformation the margin between the impostors and perimeter has increased . The goal of learning the linear transformation is to push the impostors out of the perimeter set by the target neighbors.

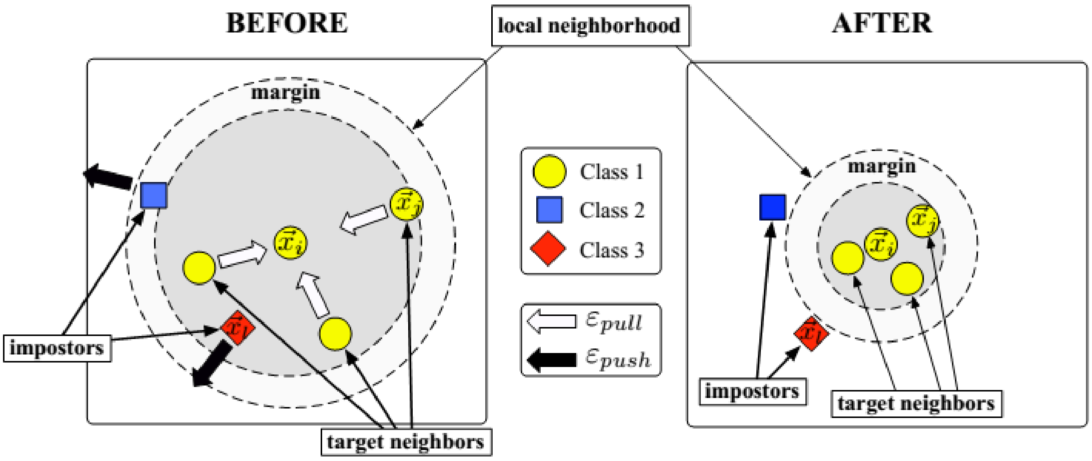


Figure 4.1: **Diagram indicating the neighborhood of one training sample x_i before and after training . After optimization of the distance metric the 3 target neighbors lie within a smaller radius around the training sample x_i and the impostors(differently labelled inputs) lie outside the region of smaller radius by some specific margin value.**) (*Distance metric learning for Large Margin Nearest Neighbor Classification, Killian Q.Weinberger Lawrence.K.Saul, JMLR 2009*)

Optimization model

The loss function for LMNN classification consists of two terms which have the following roles:

- **Pull Component** The pull component aims to attract the target neighbors closer to the training sample x_i. The pull component is given by the following:

$$Cost_{pull}(\mathbf{L}) = \sum_{j \rightsquigarrow i} \|\mathbf{L}(x_i - x_j)\|_2^2 \tag{4.6}$$

Here $j \rightsquigarrow i$ indicates that x_j is the target neighbor of x_i. Thus in the above mentioned component the linear transformation matrix L should be such that in the transformed space the distance of the input sample x_i from its target neighbors x_j must be minimized. Here the distances between all the similarly labelled samples are not minimized but only the k-nearest neighbor similarly labelled samples are considered. Since the force F and potential energy function V are related by the following relation :

$$F = -\nabla V \tag{4.7}$$

So the gradient of the term $Cost_{pull}(\mathbf{L})$ generates a pull force in the direction of decreasing values of the pull component $Cost_{pull}(\mathbf{L})$.

- **Push Component** The push component aims at penalizing the distances between a training sample and the differently labelled inputs . Whenever the constraint given by (4.6) is violated a non-zero value of loss is added by this term. The loss function is given by:

$$Cost_{push}(\mathbf{L}) = \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - z_{il}) max(0, \|\mathbf{L}(x_i - x_j)\|_2^2 + 1 - \|\mathbf{L}(x_i - x_l)\|_2^2) \tag{4.8}$$

z_{il} is a indicator variable such that when the labels $y_i = y_l$ then $z_{il} = 1$ otherwise $z_{il} = 0$. The term $max(0, p)$ considered in the cost function is the standard hinge loss term and the violations in the margin given by (4.6) are determined by the hinge loss term. Whenever a differently labelled example x_l invades the perimeter set by the target neighbors i.e. satisfies the inequality in (4.6) then the term $max(0, \|\mathbf{L}(x_i - x_j)\|_2^2 + 1 - \|\mathbf{L}(x_i - x_l)\|_2^2)$ is nonzero thus adding to the loss function .If the differently labelled sample lies at a safe distance then there is no contribution to the loss function. Since the $Cost_{push}(\mathbf{L})$ term contains the hinge loss term which is non differentiable , the sub-gradient is computed which provides the required pushing force to remove the impostors away from the perimeter set by the target neighbors.

Hence the overall cost function is given by :

$$Cost(L) = (1 - \alpha) Cost_{pull}(\mathbf{L}) + \alpha Cost_{push}(\mathbf{L}) \tag{4.9}$$

Since we have from (4.3), $\|\mathbf{L}(x_i - x_j)\|_2^2 = (x_i - x_j)^T \mathbf{M}(x_i - x_j)$ and $(x_i - x_j)^T \mathbf{M}(x_i - x_j) = D_M(x_i, x_j)$, we have an alternative version of the cost function in(4.9)

$$Cost(M) = (1-\alpha) \sum_{i,j \rightsquigarrow i} D_M(x_i, x_j) + (\alpha) \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - z_{il}) max(0, 1 + D_M(x_i, x_j) - D_M(x_i, x_l)) \tag{4.10}$$

Energy based Classification algorithm

Energy based classification relies on the loss function for classifying the test example . For classification purpose a test sample x_t is considered as a training sample and assigned a label y_t out of all possible labels . For all possible labels assigned to the test sample the loss function given by (4.10) is evaluated and the label giving the least energy is considered as the predicted label. From equation(4.10) the loss function for a single test example is given by removing the outer summation and considering the first two terms .The first term takes into account the distance of the test sample from its target neighbors and the second term considers the hinge loss over all the impostors (i.e. differently labelled examples) that intrude the perimeter set by the target neighbors of x_t. The sum of two terms is given by :

$$Cost_{11} = (1-\alpha)\sum_{j \rightsquigarrow t} D_M(x_t, x_j) + \alpha \sum_{j \rightsquigarrow t,l} (1-z_{il}) max(0, 1+D_M(x_t, x_j) - D_M(x_t, x_l)) \quad (4.11)$$

The third term takes into account the fact that the test sample x_t when considered as a training sample with a label y_t will act as impostors for other training samples . Thus from the second term in (4.10) the impostor nature of x_t is obtained in the following term :

$$Cost_3 = \sum_i \sum_{j \rightsquigarrow i} (1 - z_{it}) max(0, 1 + D_M(x_i, x_j) - D_M(x_i, x_t)) \quad (4.12)$$

Thus the predicted label of x_t given by y_{pred} is the one which minimizes the sum of $Cost_{12}$ and $Cost_3$.

$$y_{pred} = \underset{y_t}{argmin}(Cost_{12} + Cost_3) \quad (4.13)$$

4.3 Distance metric learning for hierarchial classification of visually correlated categories

Here in this section the proposed algorithm in which Large Margin Nearest neighbor classification has been used on the top of a hierarchial model obtained by recursive application of Self Tuning Spectral Clustering. For the purpose of applying Large Margin Nearest Neighbor classification whenever a non leaf node is considered then its children are assigned coarse class labels for the purpose of learning Mahalnobis metric M at the parent node which is later used for discriminating among the children while traversing down from the root node to the leaf node.

4.3.1 Spectral Clustering

Spectral clustering method [34] is a data clustering technique that utilizes the eigenvalues associated with the similarity matrix of the underlying graph structure of the data points to partition into set of disjoint clusters. It has gained immense popularity among the computer vision community primarily due to its ease of implementation and its ability to outperform traditional clustering methods like k-means. Given a set of points $X = (x_1, x_2, x_3, \ldots x_n)$ the basic steps of the spectral clustering on the set X into k clusters can be summarized as below:

- Design the affinity matrix S such that the following holds

$$\begin{aligned} S_{ij} &= exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}) \quad & i \neq j \\ &= 0 & i = j \end{aligned} \tag{4.14}$$

- Define the D to be the diagonal matrix such that :

$$D(i,i) = \sum_{j=1}^{n} S_{ij} \tag{4.15}$$

- Construct the matrix $L = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$.
- Obtain the k largest eigenvectors associated with the matrix L and stack them columnwise to obtain the matrix $E = [e_1, e_2, e_3, \ldots, e_k]$
- Obtain the matrix Y from E by normalizing each row to have unit norm.

$$Y_{ij} = \frac{E_{ij}}{\sum_j E_{ij}^2} \tag{4.16}$$

- Treat each row of Y as a member of the k dimensional space R^k and cluster the rows into disjoint groups using k means algorithm.
- If the *ith* row of the matrix Y was assigned to cluster j then assign the datapoint x_i to cluster j.

The parameter σ^2 determines how rapidly the similarity between two data points x_i and x_j falls off with the distance between them.

4.3.2 Self Tuning Spectral Clustering

Standard spectral clustering technique involves specifying the number of groups/clusters beforehand. Many applications necessitate the determination of cluster numbers automatically since the manual specification of cluster number might not be an optimal choice. Self Tuning version of Spectral Clustering proposed by Manor and Perona [35] addresses the issue of automatically selecting the number of clusters from the data along with handling of the data having multiple scales.

Scale Determination

The scale parameter σ determines the measure of similarity between two data points. Standard techniques for determining σ involve checking for multiple values and selecting that particular value which gives the least amount of distortion of clusters (obtained using the rows of matrix Y which are determined by (4.16)). This method is far from optimal since it involves lot of computation time and a single value of σ might not give the best clustering results when the data has multiple scales i.e. a tight cluster consisting of smaller number of points might exist within a background cluster. A scaling parameter σ_i was determined for each data point x_i such that the distance from x_i to x_j as seen by x_i is given by $\frac{d(x_i,x_j)}{\sigma_i}$. Conversely the distance seen by x_j is given by $\frac{d(x_j,x_i)}{\sigma_j}$. Thus similarity between two points x_i and x_j is computed by :

$$\begin{aligned} S_{ij} &= exp(-\frac{d^2(x_i,x_j)}{\sigma_i\sigma_j}) \quad i \neq j \\ &= 0 \qquad\qquad\qquad\qquad i = j \end{aligned} \tag{4.17}$$

Given the neighborhood of a particular point x_i the local statistics around that point x_i is captured by the local scale parameter associated with that point i.e. σ_i. The computation of σ_i is done by considering the distance to the k^{th} neighbor of x_i.

$$\sigma_i = d(x_i, x_k) \tag{4.18}$$

The selection of the parameter k is discussed later by putting emphasis on the problem at hand.

Determining number of clusters

The technique of automatically determining the number of clusters depends on the structure of the eigen vectors associated with the matrix $L = D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$ where the matrices D and S are defined in (4.15) and (4.17) respectively. Considering the ideal case when the clusters are infinitely apart from each other then the affinity matrix S and the laplacian matrix L have a block diagonal structure as shown below with the number of blocks equal to the number of clusters K.

$$S = \begin{bmatrix} S_1 & 0 & \dots & 0 \\ 0 & S_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & S_K \end{bmatrix}$$

$$L = \begin{bmatrix} L_1 & 0 & \dots & 0 \\ 0 & L_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & L_K \end{bmatrix}$$

For construction of the matrix E defined in (4.17) the first K eigen vectors of matrix L are considered. Due to the block diagonal structure of matrix L the overall eigenvalues are obtained by the union of the eigenvalues of the individual blocks and the eigenvectors are obtained after stacking the eigenvectors of the individual blocks with zeros in appropriate locations. Hence the structure of the matrix E is given as follows:

$$E = \begin{bmatrix} e^1 & 0 & \dots & 0 \\ 0 & e^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^K \end{bmatrix}$$

Here e^k is the eigenvector associated with the sub-matrix L_k , which coressponds to cluster k. Since the eigenvectors are mutually orthogonal it is possible to pick up another set such that $E^{'} = ER$, where $R \in \mathbf{R^{K \times K}}$ and is an orthogonal matrix. Thus given a set of eigen vectors we can find a rotation matrix R such that $E^{'} = ER$ has rows with single non-zero entries. The quality of alignment of the set of eigenvectors by a rotation matrix into a coordinate system (having single non zero entry in a row) is given by:

$$Cost_{rotate} = \sum_{i=1}^{n} \sum_{j=1}^{K} \frac{Mrot_{ij}^2}{m_i^2} \tag{4.19}$$

Here $Mrot = ER$ and $m_i = \underset{j}{max}\ Mrot_{ij}$ and E is the matrix containing the set of eigenvectors. For a particular number of clusters i.e. K the rotation matrix is recovered that provides the best possible alignment with the standard basis system. The optimal number of clusters is selected to be the one which provides the minimum value of the cost function in (4.19). The detailed steps involved in automatic cluster number determination and scale selection are listed below:

- For each datapoint x_i determine the local scale parameter using (4.18).
- The determination of local scale parameters is followed by computation of the affinity matrix S, whose entries are given by (4.17).
- Compute the entries in the matrix D using $D(i,i) = \sum_{j=1}^{n} S_{ij}$ and use it to compute the normalized laplacian matrix $L = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$.
- Considering K to be the largest possible cluster number , extract the K largest eigen vectors of the matrix L to obtain the set $E = [e_1, e_2, \ldots, e_K]$.
- For each k upto the largest cluster number K , find the rotation R which aligns the set of eigenvectors with the standard basis system and note down the associated value of cost function mentioned in (4.19).
- Determine the best cluster number $clust_{best}$ to be the largest cluster number having minimum cost function value.
- Consider the aligned top $clust_{best}$ eigenvectors and assign the original data point x_i to cluster k if $Mrot_{ik}^2 = \underset{j}{max}\ Mrot_{ij}^2$.

4.3.3 Spectral Clustering based Hierarchial Organization of Visually Correlated Categories

The first step in our proposed method is to obtain a hierarchial organization of the given image categories in a tree like structure. One possible alternative was to apply top down hierarchial clustering where one starts with the entire data in the root node and uses a flat clustering technique like k-means at each level to split the clusters in one level into finer clusters. But such technique involves pre-specifying the number of clusters into which a particular group of data points must be split. Our goal is to automatically learn the number of clusters from the given datapoints (datapoints here refer to the images of the different image categories) and apply it in a recursive manner to determine the underlying tree structure coressponding to the given set of image categories. Since Self Tuning Spectral Clustering can determine the optimal number of clusters in the data based on the quality of the alignment with the standard basis system , it was used to determine the underlying tree structure.

The self tuning spectral clustering routine was applied recursively from root node until the number of leaf nodes were equal to the number of classes. While obtaining clusters from set of image categories ,instead of using the entire dataset of images as a whole ,a representative member was selected from each category. The representative member of a particular image category is computed by taking the mean of the feature vectors associated with that particular class. Given the set of feature vectors $[f_{i1}, f_{i2}, \dots f_{iN}]$ associated with *ith* class the representative member is computed as:

$$rep_i = \frac{\sum_{j=1}^{N} f_{ij}}{N} \tag{4.20}$$

Further each such vector rep_i is normalized to have unit l_2 norm. The self tuning clustering routine is carried out on these set of representative members .

The parameter which decided the split for a particular node(set of datapoints) into set of child nodes(clusters) was the neighbor number associated with the datapoints. The role of the neighbor number has been discussed in the results section.

The important steps in the basic spectral clustering routine(**SelfTuneSpectral**) and tree construction(**BuildTree**) are detailed in the following pseudo codes:

Algorithm 4 SelfTuneSpectral-Spectral Clustering routine

```
Input: $data$(data to be clustered)
$clustrange$(range of cluster numbers to be considered )
$neighbor_{num}$(the nearest neighbor parameter used for computing local scale statistics)
Output: $bestclust$(gives the cluster grouping with the best alignment quality)
$clustdata$(holds the cluster number coressponding to the data given )
$numclust$(the number of clusters returned)
$Distmat = pairdist(data, data)$
                    ▷ Computes the pairwise euclidean distance between the data points
$Ascaled = scaledist(Distmat, neighbor_{num})$
                                                  ▷ Computes the scaled affinity matrix
$[clust_{RLS}, bestind, Q] = cluster_{rotate}(Ascaled, clustrange)$
    ▷ Cluster rotate function which returns the quality of alignment corresponding to
each cluster number choice and the best cluster grouping found so far
$bestclust = clust_{RLS}\{bestind\}$
                          ▷ the best cluster groupings of the datapoints obtained so far
$numclust \leftarrow size(bestclust, 2)$                              ▷ the number of clusters
for $i = 1 : numclust$ do
    $tmpset \leftarrow bestclust\{i\}$                                      ▷ $i^{th} cluster$
    $clustdata(tmpset) \leftarrow i$            ▷ Assigning cluster $i$ to the datapoints in $tmpset$
end for
```

Algorithm 5 Buildtree

```
**Inputs:** $Trdat$ (Training data), $numclass$ (Number of classes)
**Initialization:** $leafnumber \leftarrow 0$, $level \leftarrow 1$, $labelval\{level\}\{1\} \leftarrow [1, 2, \ldots, numclass]$, $clustinit \leftarrow 1$, $numnode\{level\} \leftarrow 1$, $idlevel\{level\} \leftarrow 0$, $childnum \leftarrow \varnothing$, $idtot \leftarrow \varnothing$
**while** $(leafnumber \neq numclass)$ **do**
    **if** $(level == 1)$ **then**
        $labelacc = labelval\{level\}\{1\}$
        $[numchild, labelchild] =$ **splitdivide**$(TrDat, labelacc)$
        $numnode\{level+1\} \leftarrow numchild$, $childnum\{level+1\} \leftarrow numchild$
        $[id] =$ **checkleaf**$(labelchild, numchild)$
        $[idtot] \leftarrow [idtot, id]$, $idlevel\{level+1\} \leftarrow id$
        **for** $j = 1 : numchild$ **do**
            $labelval\{level+1\}\{j\} = labelchild\{j\}$
        **end for**
    **else**
        $nl \leftarrow numnode\{level\}$, $idtemp \leftarrow idlevel\{level\}$, $idcurrlevel \leftarrow \varnothing$
        **for** $j = 1 : nl$ **do**
            **if** $(idtemp(j) == 0)$ **then**
                $labelacc = labelval\{level\}\{j\}$
                $[numchild, labelchild] =$ **splitdivide**$(TrDat, labelacc)$
                $sizel[j] = numchild$
                $[id] =$ **checkleaf**$(labelchild, numchild)$
                $[idcurrlevel] \leftarrow [idcurrlevel, id]$, $[idtot] \leftarrow [idtot, id]$
                **if** $(j == 1)$ **then**
                    **for** $k = 1 : numchild$ **do**
                        $labelval\{level+1\}\{k\} = labelchild\{k\}$
                    **end for**
                    $start \leftarrow numchild$
```

Algorithm 6 Buildtree-Part2

```
                else
                    $lb \leftarrow start + 1$, $ub \leftarrow start + numchild$
                    for $k = lb : ub$ do
                        $labelval\{level + 1\}\{k\} = labelchild\{k - start\}$
                    end for
                    $start = start + numchild$
                end if
            else
                if $(j == 1)$ then
                    $start \leftarrow 0$
                else
                    $start \leftarrow start$
                end if
            end if
        end for
        $idl\{level + 1\} \leftarrow idcurrlevel$, $leafnum \leftarrow leafnum + \sum_i idcurrlevel[i]$
        $numnode\{level + 1\} = \sum_i sizel[i]$, $childnum\{level\} = sizel$
    end if
end while
```

In the above pseudocode $labelval\{level\}\{j\}$ includes the class labels associated with the jth node in the current level of the tree which is denoted by $level$. $numnode\{level\}$ determines the number of the nodes in the current level. $childnum\{level\}$ holds the number of child nodes coressponding to all the nodes in the current level. The **splitdivide** routine uses the Spectral Clustering routine i.e. **SelfTuneSpectral**(mentioned in Algorithm 2) to split the current node into child nodes and returns the number of child nodes in $numchild$ and the class labels associated with the child nodes in $labelchild$. While implementing the tree structure a parent set $Parent$ was maintained for each node of the tree ($Parent(i)$ holds the parent node of the ith node in the tree). An edge $(Parent(i), i)$ was added to the edge set E. In the pseudo code of $BuildTree$ I have omitted the part of Edge determination due to lack of space .

Algorithm 7 splitdivide

```
Inputs: Traindat(Training data associated with current node), labelacc(Set of labels associated with current node)
Outputs: numchild, labelchild
Initialization: $TrDatnode \leftarrow \varnothing$ , $meanset \leftarrow \varnothing$
if $size(labelchild, 2) == 2$ then
    $numchild \leftarrow size(labelchild, 2)$
    $labelchild\{1\} \leftarrow labelacc(1), labelchild\{2\} \leftarrow labelacc(2)$
    for $i = 1 : numchild$ do
        $TrDatNode \leftarrow [TrDatNode, Traindat_{labelacc(i)}]$
    end for
else
    $numclass \leftarrow size(labelchild, 2)$
    for $i = 1 : numclass$ do
        $n_i = size(Traindat_{labelacc(i)}, 2)$
        $meanset \leftarrow [meanset, \frac{\sum_i Traindat_{labelacc(i)}}{n_i}]$
    end for
    $clustrange \leftarrow [2, \dots, numclass]$
    $[bestclust, clustdata, numchild] = \textbf{SelfTuneSpectral}(meanset, clustrange)$
                                        ▷ bestclust(i) holds the datamember ids of the ith cluster
                        ▷ clustdata(i) holds the cluster number coressponding to the ith datamember
    for $i = 1 : numchild$ do
        $indnum \leftarrow bestclust\{i\}$
        $trtemp \leftarrow \varnothing$
        for $j = 1 : size(indnum, 2)$ do
            $index \leftarrow indnum(j)$
            $trtemp \leftarrow [trtemp, TrainDat_{labelacc(index)}]$
        end for
        $TrDatNode \leftarrow [TrDatNode, trtemp]$
    end for
```

Algorithm 8 splitdivide

```
    for $i = 1 : numclust$ do
        $findex \leftarrow \varnothing$
        for $j = 1 : size(clustdata, 2)$ do
            if $clustdata(j) == i$ then
                $findex \leftarrow [findex, j]$
            end if
        end for
        $labelchild\{i\} = labelacc(findex)$
    end for
end if
```

Algorithm 9 checkleaf

```
Inputs: $numchild$(Number of children associated with current node), $labelchild$(Set of labels associated with child nodes)
Outputs: $id$ ($id[i] = 1$ means the ith child node is leaf or otherwise)
for $i = 1 : numchild$ do
    $tmplb \leftarrow labelchild\{i\}$
    if $size(tmplb, 2) == 1$ then
        $id[i] = 1$
    else
        $id[i] = 0$
    end if
end for
```

The **splitdivide** function splits the current node considered into *numchild* number of child nodes and returns the class labels associated with the child nodes in *labelchild*. The **checkleaf** function determines which child nodes are leaf nodes i.e. have one constituent class. For the *ith* child node $id[i] = 1$ means the child is a leaf node.

4.3.4 Node specific Large Margin Nearest neighbor

The organization of the different classes of image in a hierarchial tree structure is followed by learning of a Mahalnobis metric at each non-leaf node. Given a non leaf node n at level l containing the set of classes $Class_n = (C_1, C_2, C_3 \dots C_n)$ and the set of classes associated with the ith child node of n denoted by $child_i$, then the following property must be satisfied:

$$Class_n = \bigcup_i child_i \tag{4.21}$$

Since the traversal from the non leaf parent node proceeds down to a particular child node in a hierarchial manner before ending at a leaf node, a suitable criteria must be maintained at the parent node for selecting the appropriate child node. Considering the ith child node , a broad class label is assigned to it such that for k child nodes , there are k possible broad classes available for selection at the location of the parent node. Thus the selection of the appropriate child node is cast as a multi class classification problem at the parent node. After the assignment of the broad class labels to the child nodes ,a Mahalnobis metric $\mathbf{M}$ is learnt at the parent node. The main idea behind learning a Mahalnobis metric is to ensure that the number of impostors in the perimeter of the considered training sample(the perimeter being set up by its target neighbors) is minimized after learning the metric. The impostors refer to the training samples having different broad class labels i.e. part of different child nodes , but invading the perimeter set by the target neighbors. In this case the target neighbors of a training sample indicate the $\mathbf{k}$ nearest neighbors among the members of the child node, that the training sample is a part of.

For the given hierarchial structure the Mahalnobis metric $\mathbf{M}$ is learnt at each non-leaf node. Thus the formulation for learning $\mathbf{M_{node}}$ at a non leaf node is given by minimizing the following function:

$$\begin{aligned} Cost(\mathbf{M_{node}}) &= (1-\alpha) \sum_{i,j \rightsquigarrow i} D_{\mathbf{M_{node}}}(x_i, x_j) \\ &+ \alpha \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - z_{il}) max(0, 1 + D_{\mathbf{M_{node}}}(x_i, x_j) - D_{\mathbf{M_{node}}}(x_i, x_l)) \end{aligned} \tag{4.22}$$

where $D_{\mathbf{M_{node}}}(x_i, x_j) = (x_i - x_j)^T \mathbf{M_{node}} (x_i - x_j)$.

4.3.5 Classification scheme

The classification of a test sample y occurs in a top-down fashion starting from the root node and traversing the non leaf nodes until a terminal node is reached. The class associated with the terminal node is considered as the predicted class for the test sample.

The selection of child node to traverse from the parent node is done on the basis of the energy classification model as described in Eq (4.14). The test sample y is assigned the broad class labels of the child nodes and for each assigned class label the loss mentioned in Eq (4.14) is evaluated by using the Mahlanobis metric $\mathbf{M_{node}}$ learnt at the parent node. That particular child node is selected in the next level whose broad class label when assigned to the test sample results in minimum value of the considered loss function. The classification algorithm is listed below:

Algorithm 10 Classification algorithm-part 2

```
**Input:** test sample $x_t$, tree parameters $T = (Nodemark, E, Mset, Labelset)$
$Nodemark$ : numbering associated with tree nodes at each level.
$E$ : Set of edges associated with the tree.
$Mset$ : Set of Mahalnobis metric $M$ associated with the nodes of the tree.
$Labelset$ :Set of labels associated with the tree nodes.
**Output:** predicted label $y_{pred}$ of the test sample $y$
$level \leftarrow 1$                                  ▷ current level of the tree
$index \leftarrow 1$                                  ▷ root node accessed
$labelcurr \leftarrow Labelset\{index\}$              ▷ set of labels of the root node
while ($|labelcurr| \neq 1$) do
    $M_{node} \leftarrow Mset\{index\}$
    $classc = \underset{\{y_c:(index,c)\in E\}}{argmin} ((1-\alpha)\sum_{j\rightsquigarrow t} D_{M_{node}}(x_t, x_j) + \alpha\sum_{j\rightsquigarrow t,l}(1-z_{il})max(0, 1 + D_{M_{node}}(x_t, x_j) - D_{M_{node}}(x_t, x_l)) + \alpha\sum_{i,j\rightsquigarrow i}(1 - z_{it})max(0, 1 + D_{M_{node}}(x_i, x_j) - D_{M_{node}}(x_t, x_i)))$
    ▷ $y_c$ is the broad class label associated with the $c^{th}$ child node of the current node marked as $index$.
▷ $classc$ is the broad class label associated with the child node which gives minimum value of loss function
    $level \leftarrow level + 1$
    $index = f(classc, level)$
    $labelcurr \leftarrow Labelset\{index\}$
end while
$y_{pred} \leftarrow labelcurr$
```

4.3.6 Experiments

For experiments we consider the Imagenet Database [17] which consists of multiple visually correlated image categories.In the imagenet database the categories are organized according to WordNet hierarchy[36]-[37]. Each meaningful word in the wordnet hierarchy is termed as synset and for each sysnet an average number of 1000 images are used to describe the synset. The Imagenet Hierarchy has the following overall statistics:

- Number of non-empty synsets: 21841
- Number of images: 14,197,122
- Number of images with bounding box annotations:1,034,908

Each image category is associated with a unique synset id. There are multiple high level categories like amphibian, reptile, animal, appliance, bird etc which are further categorized into multiple subcategories. For our experiments we considered two groups of highly visually correlated classes which are distinct from each other but correlated within themselves. The details are given below:

- **Group 1:**
  Dog(Synset id: n02084071), **Hound**(Synset id: n02087551), **Whippet**(Synset id: n02091134), **Cat**(Synset id: n02121620), **Margay**(Synset id: n02126640)

- **Group 2:**
  Computer Monitor(Synset id: n03085219), **Computer Screen**(Synset id: n03086502), **Desktop Computer**(Synset id: n03180011), **Keyboard**(Synset id: n03614007), **Laptop**(Synset id: n03642806), **Television**(Synset id: n04404412)

From both the groups two sets of images are considered i.e. uncropped and cropped . The cropping of the images to obtain the object parts were done using the bounding boxes provided in the form of XML files . The XML files were parsed using the PASCAL Development Toolkit. Total number of images which had the bounding boxes are 5854. After cropping the images the multiple objects in the images were saved and the number of images increased to 6313. The feature extraction schemes were performed on both cropped and uncropped images . Two sets of feature extraction schemes were compared i.e. Dense SIFT+ Bag of Words model and Dense SIFT +Locality Constrained Coding+Spatial

Pyramid framework. In training phase the toolbox provided by Mark Schmidt [38] has been used for optimization with the underlying algorithm being LBFGS. The toolbox has been provided as a part of the basic LMNN code in Weinberger's site `http://www.cs.cornell.edu/~kilian/code/code.html`.

Feature extraction pipeline

The feature extraction steps for both SIFT+LLC scheme and SIFT+Bag of Words are highlighted in the following flow diagrams:

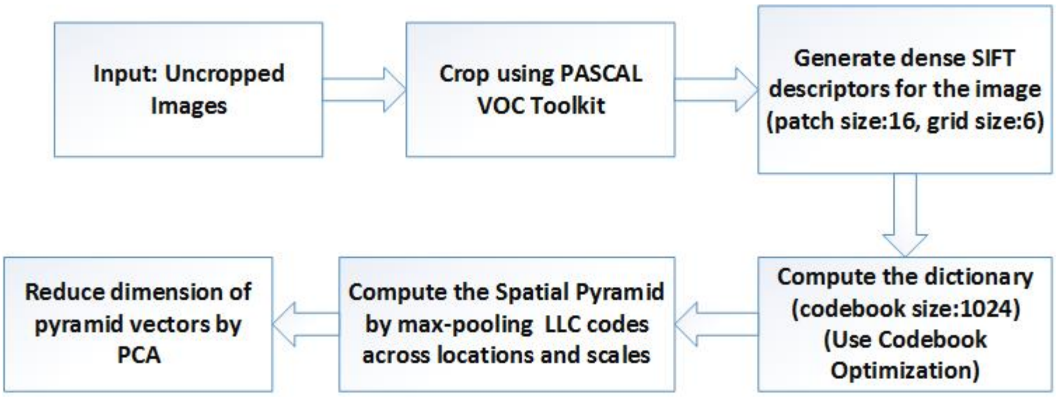

Input: Uncropped Images
Crop using PASCAL VOC Toolkit
Generate dense SIFT descriptors for the image (patch size:16, grid size:6)
Compute the dictionary (codebook size:1024) (Use Codebook Optimization)
Compute the Spatial Pyramid by max-pooling LLC codes across locations and scales
Reduce dimension of pyramid vectors by PCA

Figure 4.2: **Flow diagram for SIFT+LLC feature extraction scheme**

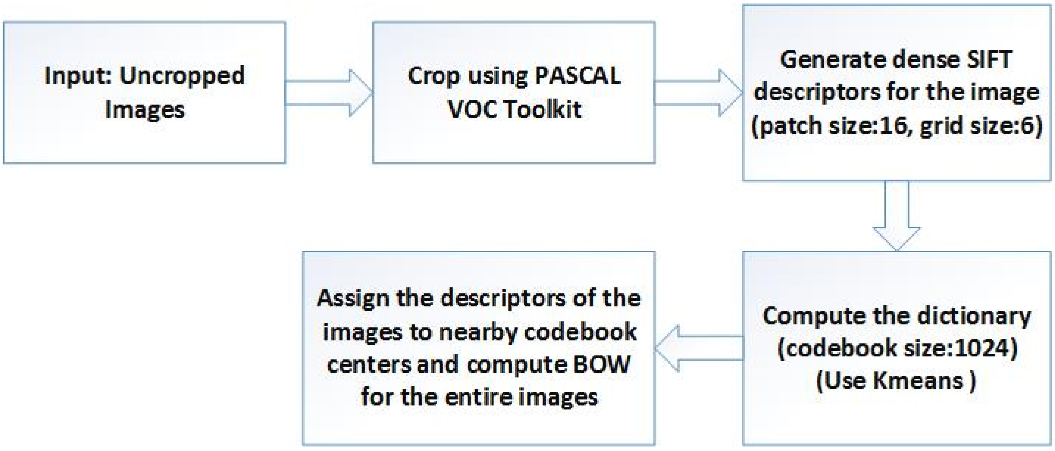
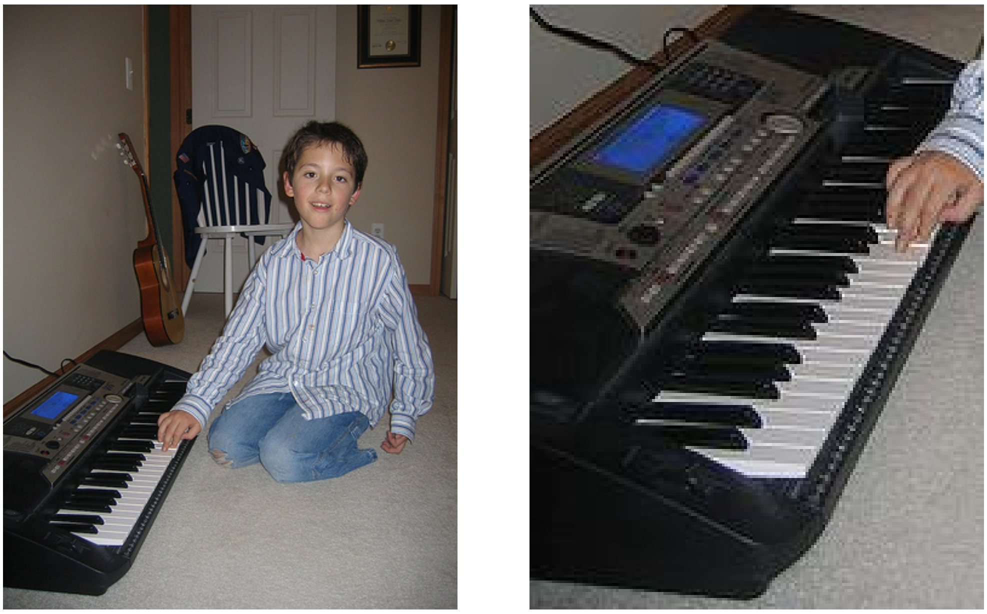

Input: Uncropped Images
Crop using PASCAL VOC Toolkit
Generate dense SIFT descriptors for the image (patch size:16, grid size:6)
Compute the dictionary (codebook size:1024) (Use Kmeans)
Assign the descriptors of the images to nearby codebook centers and compute BOW for the entire images

Figure 4.3: **Flow diagram for SIFT+BOW feature extraction scheme**

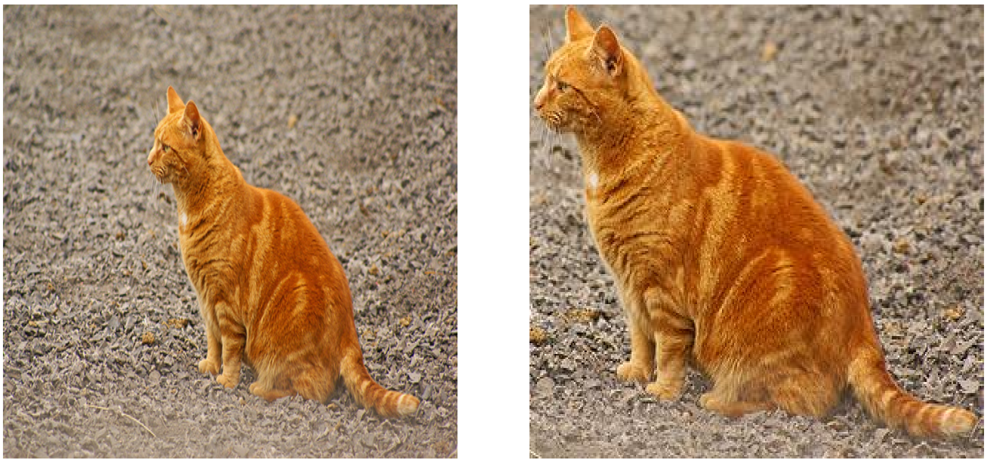
In the codebook optimization step LLC code for each descriptor is computed along with the optimized codebook. Further in case of standard BOW representation of the image PCA was not used because the dimension of each histogram vector was 1024. In case of SIFT+LLC scheme PCA was used because dimension of each pyramid vector for 3 level pyramid was 21504. After reduction using PCA(retaining 95% variance)the dimension of the feature vectors associated with cropped images was reduced to 4313. The same technique is used for uncropped images only exception being the absence of the 2nd cropping stage.



(a) An image of keyboard class before cropping

(b) Same image after cropping

Figure 4.4: **Effect of cropping on image of keyboard class**



(a) An image of cat class before cropping

(b) Same image after cropping

Figure 4.5: **Effect of cropping on image of cat class**

Hyperparamter tuning

The set of hyperparameters that were required to be tuned were :

- K(number of target neighbors)
- outdim(number of rows of the Linear transformation matrix **L**)
- maxiter (maximum number of iterations required for training)

For hyperparameter tuning two strategies were used i.e Bayesian Optimization and grid search. The α used in Eq (4.10) was taken to be the default value used in the Large Margin Nearest Neighbor algorithm i.e. 0.5. The comparative analysis of the two schemes are given in this section. The entire pseudocode which involves finding optimal hyperparameters for each node of the tree has been included in Appendix A.

Bayesian Optimization

Bayesian optimization [39] is a global optimization strategy used for functions which have a black box kind of nature. It is used in those cases when the function evaluations are quite expensive or the functions donot have a closed form expression (derivatives dont exist or the function is non-convex in nature. Bayesian optimization proceeds by placing a prior distribution over the functions, which considers assumptions about the function to be optimized like smoothness.

Gaussian process is used as prior distribution over functions . Any gausssian process defined by $f : X \to R$ has the property that a finite set of N points in X has an associated multivariate Gaussian distribution in R^N. Any Gaussian process is defined by its mean and covariance function i.e. $m : X \to R$ and $K : X \times X \to R$. Given a set of datapoints $x_n \in X$ and a noisy estimate of the function values at those datapoints i.e. $y_n = f(x_n) + \eta$ where $\eta \in N(0, v)$, then using these estimates and the prior distribution of the functions the GP posterior is updated . The updated GP model is used to determine the nature of acquisition function $A : X \to R^+$ and the point maximizing the acquisition function is considered as the point for next evaluation. Thus the optimization of the function f is reduced to maximization of the acquisition function in each iteration. Popular choice of acquisition function is the Expected improvement criteria which provides a proper tradeoff between exploration and exploitation in the search space. The expected improvement criteria is given by:

$$A_{EI}(x:\{x_n, y_n\}, \theta) = \sigma(x:\{x_n, y_n\}, \theta)(\gamma(x)\phi(\gamma(x)) + N(\gamma(x); 0, 1)) \tag{4.23}$$

Here $\gamma(x)$ is given by :

$$\gamma(x) = \frac{f(x_{best}) - (\mu(x:\{x_n, y_n\}, \theta))}{\sigma(x:\{x_n, y_n\}, \theta)} \tag{4.24}$$

Here $\mu(x:\{x_n, y_n\}, \theta)$ and $\sigma^2(x:\{x_n, y_n\}, \theta)$ are the model parameters of the Gaussian process and $\phi(x)$ determines the cumulative distribution of the standard normal distribution.

In our setting the function returning the validation error for a particular setting of hyperparameters is considered for bayesian optimization since this function doesnot have an exact closed form expression. The space of hyperparamters was initially sampled uniformly using Sobol sequence and the expected improvement criteria was used for acquisition function. So in each iteration of the Bayesian Optimization the hyperparameters maximizing the expected improvement criteria was selected . The mean function considered was the constant mean function and the covariance function used was the Automatic Relevance determination Squared Exponential Kernel. This selection of hyperparamters using bayesian optimization was performed for node specific case of the underlying tree structure. The ranges of the parameters considered were :

Hyperparameters	Range
K	[1,15]
outdim	[2, number of rows in training sample]
maxiter	[10,200]

Table 4.1: **Hyperparameter ranges for Bayesian optimization**

The number of bayesian optimization iterations were varied between 5 and 20 depending upon the experiments.

Grid Search

Grid search is the common technique for selecting hyperparameters where one specify a particular range within which the search must be performed. Since here we have 3 hyperparameters and there are multiple non leaf nodes where the hyperparameters must be found, node specific grid search is cumbersome. Instead of node specific grid search hyperparameters were found for the entire tree. The number of target neighbors K was varied in the range $\{3, \ldots, 15\}$ and dimensionality of the linear transformation outdim was considered in the range $\{1000, 1100, \ldots 2000\}$ in steps of 100. The number of maximum iterations maxiter was fixed to 200.

Results

1.Effect of features:

In the first experiment the effect of features on the classification performance was considered. The set of uncropped images were taken and both sets of features were computed i.e. SIFT+BOW and SIFT+LLC. While comparing the effect of the features , the samples in training, test and validation sets were kept same for both cases i.e. the sample indices were kept same in both cases and the only changes were incorporated in the features. Per node bayesian optimization was performed to determine the best set of hyperparameters in each node. Number of bayesian optimization iterations were fixed at 5 and two separate runs were taken.

	SIFT+BOW	SIFT+LLC
Run 1	46.6%	59.6%
Run 2	45.7%	56.7%
Avg accuracy	46.15%	58.15%

Table 4.2: **Comparison between SIFT+BOW and SIFT+LLC on uncropped images**

It can be seen that standard SIFT+BOW features performed poorly as compared with SIFT+LLC on uncropped images. Since the number of uncropped images were 5854 and the number of cropped images were 6313, comparison was not done between two sets.

2. Determination of optimal tree structure

Since self tuning spectral clustering was used(which utilized mean of feature vectors as representative vector for an image category) , depending on the initial split of the data the grouping of the classes in the tree remained fixed upto the 2nd level but underwent change in the third level.

So in order to fix the optimal tree structure in terms of classification accuracy and find the best hyperparameters per node of the tree , the following steps were performed:

(i) Split the entire data(random fashion) intially into a training and test set (80-20 split). Keep the test set separate.

(ii) Split the training data in hand (80-20 split) into training and validation set. Repeat this split 30 times (in a random fashion) and note the occurrences of different tree structures.

(iii) It was observed that out of 30 runs 4 tree structures were repeating and 23 runs out of 30 runs were covered by 4 tree structures.

(iv) The training and validation splits for the 4 tree structures were considered in order to obtain the best set of hyperparameters for each tree. The classification accuracy was recorded on the same test data which was split initially.

Bayesian optimization routine was run for each tree structure and w.r.t. each node . Two runs were taken for each tree structure , with each run consisting of 20 iterations. The total number of samples considered for testing were 1261. The associated tree structures (for the third level) and their number of occurrences among the 30 runs in step (iii) are listed below:

Tree third level groupings	Number of occurrences	Tree Id
[11], [9], [5,6], [1], 4, [2,3], [10], [8], [7]	8	1
[11], [5], [6], [1], [9], [2,3], [10], [4,8], [7]	4	2
[5], [11], [6], [1], [9], [10], [2,3], [4], [8], [7]	7	3
[11], [9], [5,6], [1], [2,3], [10], [4,8], [7]	4	4

Table 4.3: **Different groupings of the classes in the third levels of 4 different tree structures and their number of occurrences ([5,6] here refers to the grouping of classes 5 and 6)**

For reference the class numbers and the associated classes are listed as follows:

Class number	Class
1	cat
2	computer monitor
3	computer screen
4	computer desktop
5	dog
6	hound
7	keyboard
8	laptop
9	margay
10	television
11	whippet

Table 4.4: **Class numbers and associated classes**

The classification accuracies on the same test data were reported for two runs for the 4 different tree structures and for each tree structure the average accuracy was reported. It is seen that the 2nd tree structure $tree_2$ performs the best in terms of classification accuracies. The average accuracies for each class has been listed in the tables (4.6) and (4.7).

	$tree_1$	$tree_2$	$tree_3$	$tree_4$
Run 1	59.6%	69.9%	67.2%	66.7%
Run 2	64.78%	68.3%	68.6%	66.8%
Avg accuracy	62.2%	69.1%	67.9%	66.75%

Table 4.5: **Comparison of the classification accuracies of 4 different tree structures mentioned in Table 4.3**

3. Comparison of Node Specific Maximum margin scheme with other algorithms

Class	Cat	Dog	Hound	Margay	Whippet
Tree+Maximum Margin	**44.95%**	**69.2%**	**47.95%**	**69.3%**	**38.2%**
FDDL	66.26%	44.67%	57.52%	68.06%	62.57%
ScSPM	71.35%	45.43%	58.85%	80.71%	41.63%
jDL	57.14%	59.62%	53.06%	71.67%%	89.29%

Table 4.6: **Avg accuracy of two runs of the tree structure** $tree_2$ **compared with the accuracies reported for FDDL ScSPM, jDL**

Class	Monitor	Screen	Desktop	Keyboard	Laptop	Television
Tree+Maximum Margin	**45.2%**	**58.2%**	**73.5%**	**94.9%**	**75%**	**67.85%**
FDDL	43.75%	43.75%	48.09%	98.04%	41.19%	61.60%
ScSPM	24.58%	39.22%	80.77%	96.65%	54.88%	77.38%
jDL	41.67%	53.85%	83.08%	98.05%	57.14%	81.48%

Table 4.7: **Avg accuracy of two runs of the tree structure** $tree_2$ **w.r.t. the accuracies reported for FDDL and ScSPM**

Here in the above table , comparison have been done with algorithms like FDDL [4], ScSPM [12], jDL [21]. Our accuracies are reported for all the test samples i.e. 1261 whereas the accuracies in case of other algorithms are reported on 944 test samples.

Further comparison has been done for the avg accuracies associated with the tree structures mentioned in Table 4.5 with a simple mean based traversal scheme. In the mean based traversal scheme instead of maximum margin learning at each node , a test sample moves to a child node if its distance to the mean of the members of that child node is minimum among all child nodes. The tree structure is the same as before ,learnt using self tuning spectral clustering:

	$tree_1$	$tree_2$	$tree_3$	$tree_4$
Mean based traversal+ Tree	22.84%	21.41%	19.35%	22.20%
Maximum margin + Tree	62.2%	69.1%	67.9%	66.8%

Table 4.8: **Comparsion between the average accuracies of mean based traversal of the tree and maximum margin based learning +tree traversal**

4. Comparison of grid search and bayesian optimization

Tree Ids	Grid Search	Bayesian Optimization
Tree_1	56.3%	62.2%
Tree_2	58.3%	69.1%

Table 4.9: **Comparison of the average classification accuracies for tree structures for both 1 and 2 in case of grid search and bayesian optimization**

It is seen in case of hyperparamter tuning bayesian optimization in just 20 iterations was able to outperform standard grid search . Moreover the hyperparameter space covered by the Bayesian optimization routine was more exhaustive as compared to the grid search and it was covered in fewer iterations.

3. Training time per node and convergence plots

In this segment the time taken for training at each node of the currently best tree structure has been listed . The nodes for which the training times are listed are the non-leaf nodes . Convergence plots report the variation of the cost function value with iterations for the non leaf node. The training times are listed in seconds and are reported only after the best hyperparameters are found for the nodes using Bayesian optimization.
The tree structure $tree_2$ is shown in Fig (4.6) . The nodes in the tree are marked in level order fashion i.e. the node containing the classes $[1, 5, 6, 9, 11]$ in level 2 is marked node 1 in level 1. Similarly the node containing the classes $[2, 3]$ is marked node 6 in level 2. The notation a/b denotes the node numbered a in level b. The following table notes down the training times for the nodes using the following notation.

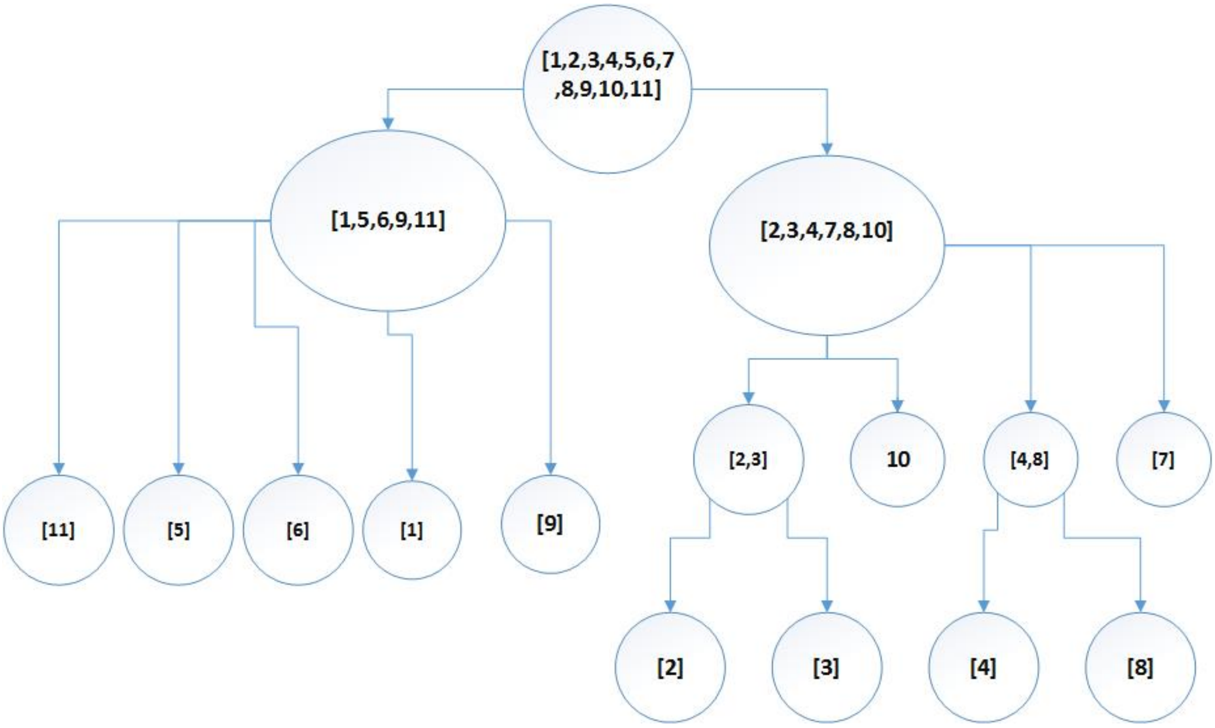


Figure 4.6: **Optimal tree structure($tree_2$) in terms of the average classification accuracy. Each node has a set of classes associated with it and the classes are mentioned by class numbers . [2,3] means hound and dog are grouped together in a node. Mapping from class number to actual classes can be found in Table 4.4**

Node number/level number	Training time (in seconds)
1/1	65.232
1/2	334.363
2/2	119.772
6/3	14.836
8/3	141.338

Table 4.10: **Training times associated with each non-leaf node(of $tree_2$) in seconds after the hyperparameters are fixed using bayesian optimization**

The convergence plots of respective nodes in the tree structure $tree_2$ are given as follows:

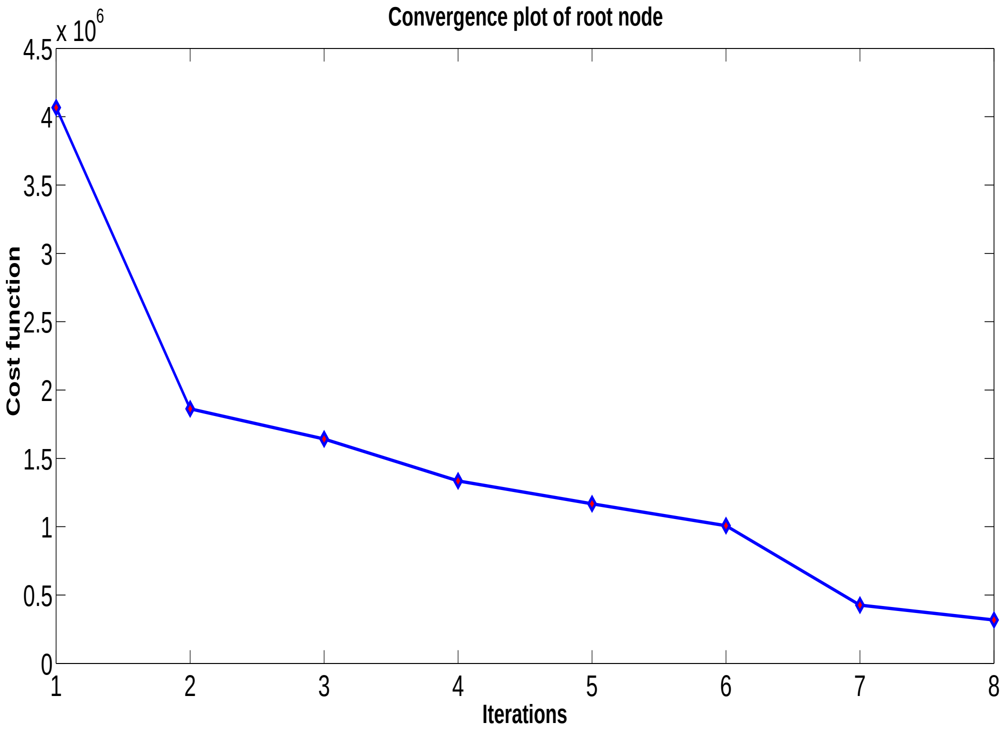


Figure 4.7: **Convergence plot of the root node of** $tree_2$

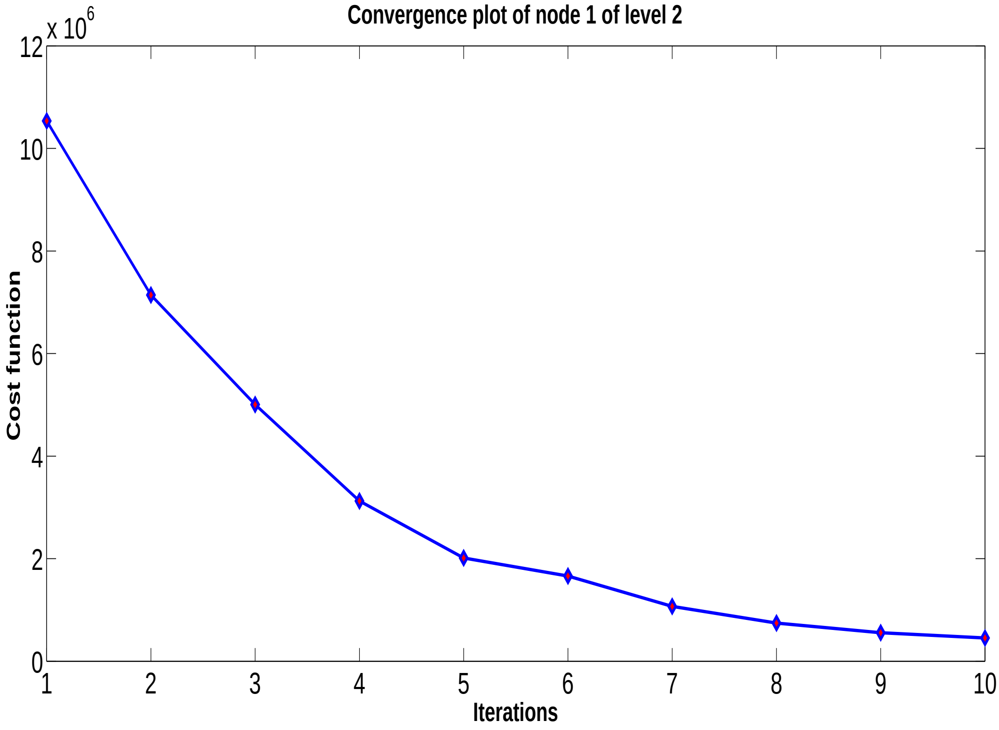


Figure 4.8: **Convergence plot of the 1st node of level 2 of** $tree_2$

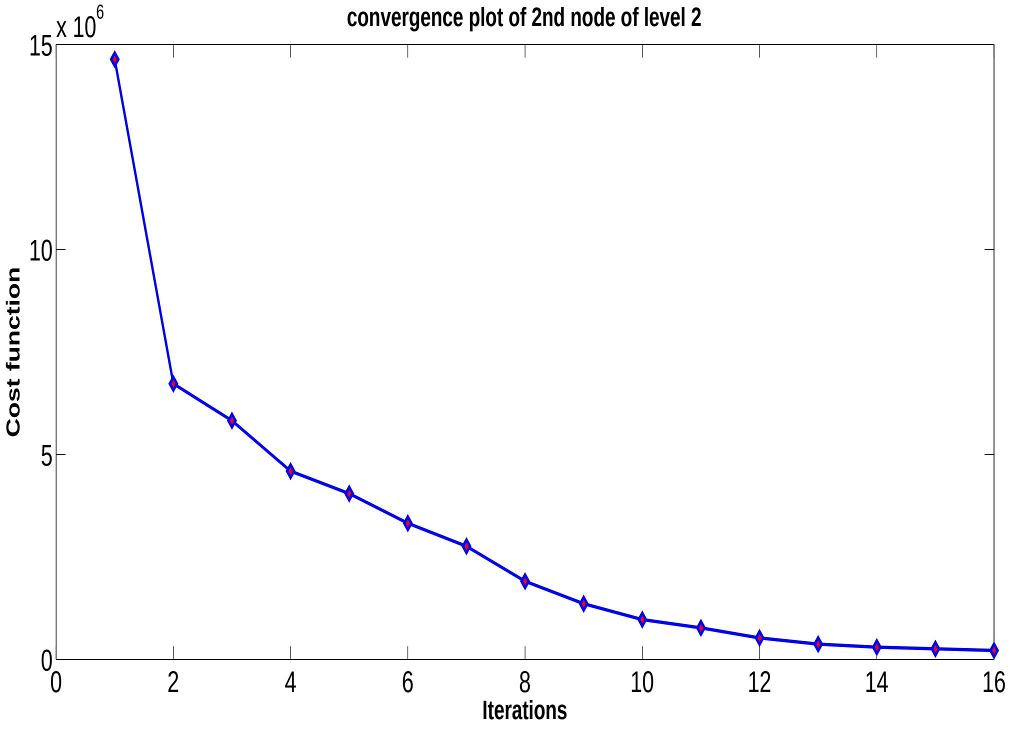


Figure 4.9: **Convergence plot of the 2nd node of level 2 of** $tree_2$

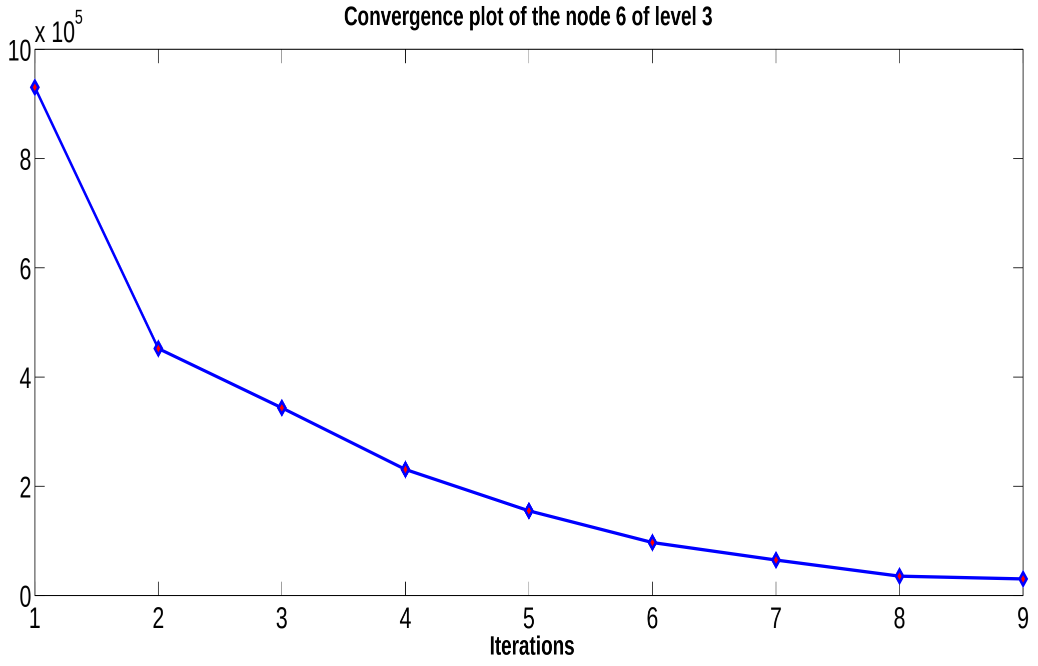


Figure 4.10: **Convergence plot of the 6th node of level 3 of** $tree_2$

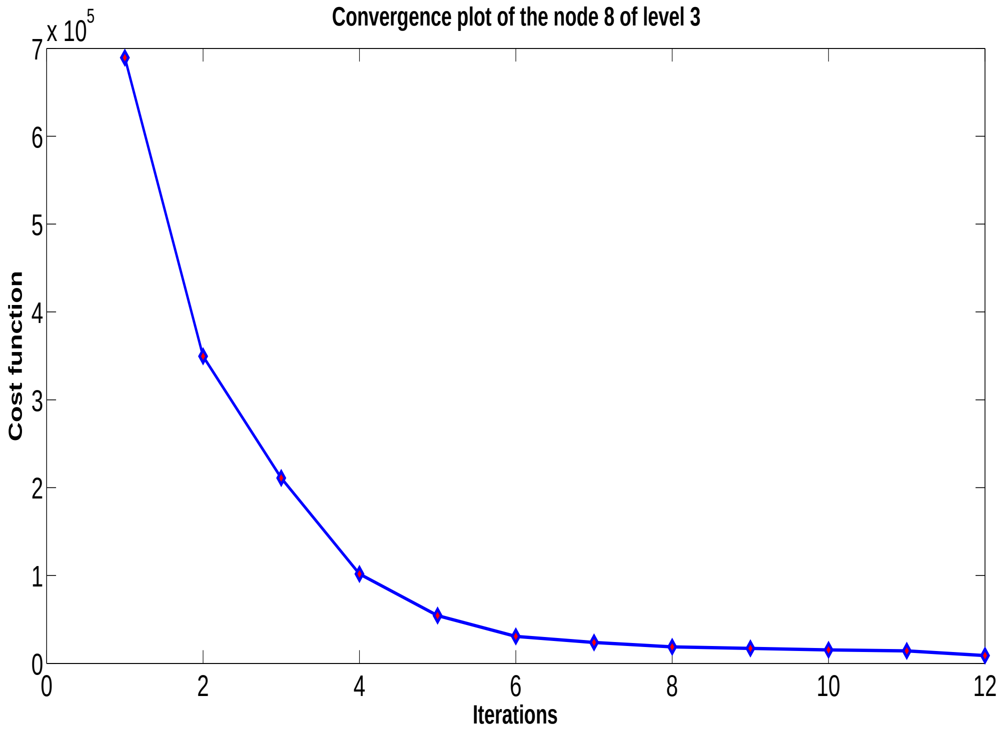
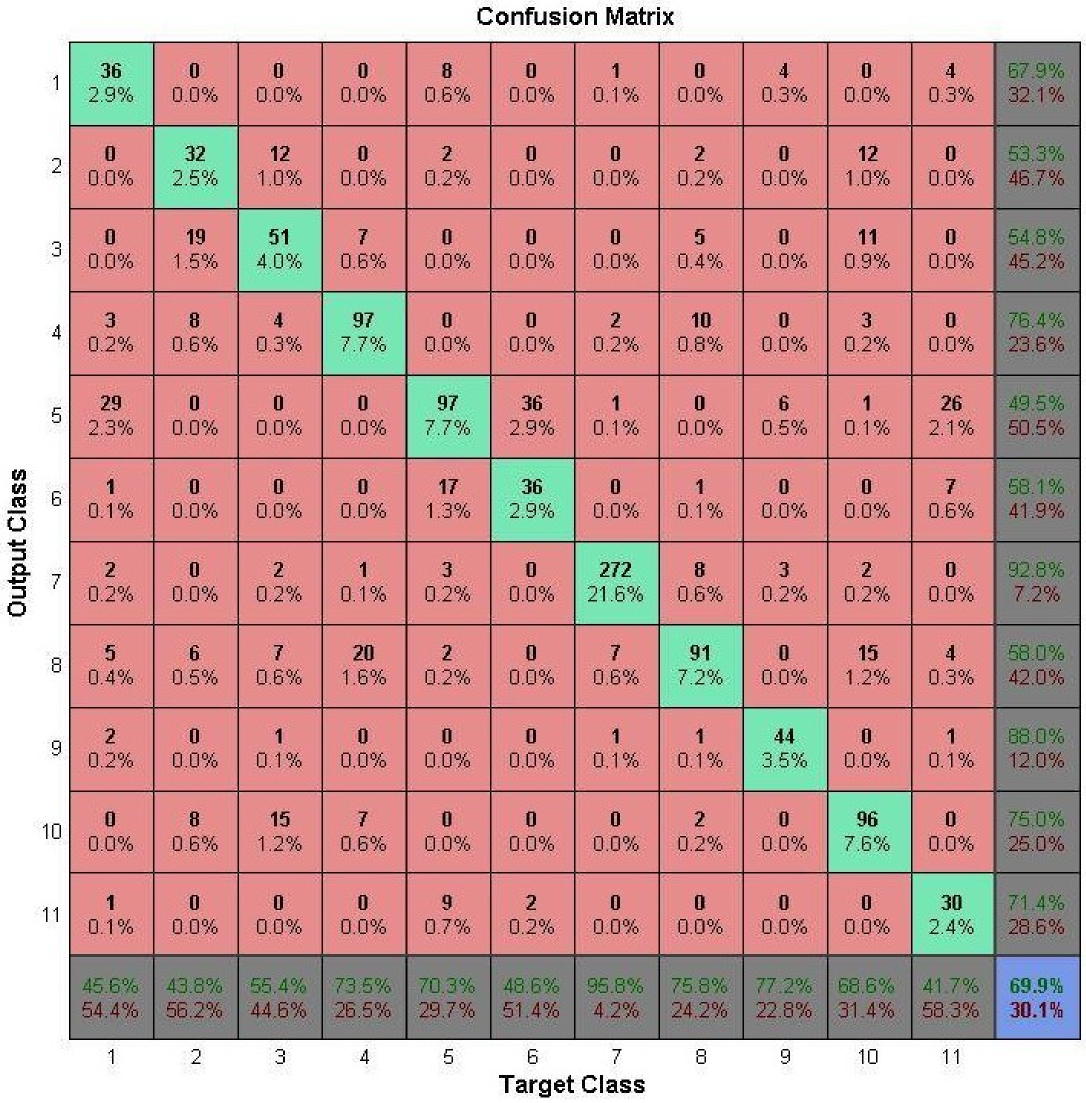


Figure 4.11: **Convergence plot of the 8th node of level 3 of** $tree_2$

Analysis


Confusion Matrix
Output Class
Target Class

	1	2	3	4	5	6	7	8	9	10	11	
1	36 2.9%	0 0.0%	0 0.0%	0 0.0%	8 0.6%	0 0.0%	1 0.1%	0 0.0%	4 0.3%	0 0.0%	4 0.3%	67.9% 32.1%
2	0 0.0%	32 2.5%	12 1.0%	0 0.0%	2 0.2%	0 0.0%	0 0.0%	2 0.2%	0 0.0%	12 1.0%	0 0.0%	53.3% 46.7%
3	0 0.0%	19 1.5%	51 4.0%	7 0.6%	0 0.0%	0 0.0%	0 0.0%	5 0.4%	0 0.0%	11 0.9%	0 0.0%	54.8% 45.2%
4	3 0.2%	8 0.6%	4 0.3%	97 7.7%	0 0.0%	0 0.0%	2 0.2%	10 0.8%	0 0.0%	3 0.2%	0 0.0%	76.4% 23.6%
5	29 2.3%	0 0.0%	0 0.0%	0 0.0%	97 7.7%	36 2.9%	1 0.1%	0 0.0%	6 0.5%	1 0.1%	26 2.1%	49.5% 50.5%
6	1 0.1%	0 0.0%	0 0.0%	0 0.0%	17 1.3%	36 2.9%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	7 0.6%	58.1% 41.9%
7	2 0.2%	0 0.0%	2 0.2%	1 0.1%	3 0.2%	0 0.0%	272 21.6%	8 0.6%	3 0.2%	2 0.2%	0 0.0%	92.8% 7.2%
8	5 0.4%	6 0.5%	7 0.6%	20 1.6%	2 0.2%	0 0.0%	7 0.6%	91 7.2%	0 0.0%	15 1.2%	4 0.3%	58.0% 42.0%
9	2 0.2%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	1 0.1%	44 3.5%	0 0.0%	1 0.1%	88.0% 12.0%
10	0 0.0%	8 0.6%	15 1.2%	7 0.6%	0 0.0%	0 0.0%	0 0.0%	2 0.2%	0 0.0%	96 7.6%	0 0.0%	75.0% 25.0%
11	1 0.1%	0 0.0%	0 0.0%	0 0.0%	9 0.7%	2 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	30 2.4%	71.4% 28.6%
	45.6% 54.4%	43.8% 56.2%	55.4% 44.6%	73.5% 26.5%	70.3% 29.7%	48.6% 51.4%	95.8% 4.2%	75.8% 24.2%	77.2% 22.8%	68.6% 31.4%	41.7% 58.3%	69.9% 30.1%

Figure 4.12: **Confusion matrix of the 11 classes obtained using tree structure** $tree_2$**(Best accuracies shown here). Features used: SIFT+LLC on uncropped images**

From the above figure an idea can be obtained about the confusion among different classes considered . The tree structure had groupings in the third level between classes 2 & 3(monitor and screen) and classes 4 & 8(desktop and laptop). As a result there was significant confusion among the classes. In the 2nd column 19 members who should be part of class 2 are assigned predicted label 3. Similarly in the case of class 3, 12 samples belonging to class 3 are assigned class 2. Same has happened in the case of classes 4 and

8. In case of whippet there is significant confusion with the dog class. As seen from the confusion matrix there is little case of confusion among out of group classes(classes having less visual correlation among themselves) like keyboard and hound.
The variation in the tree structure can be atrributed to the self tuning spectral clustering on the mean of feature vectors of a particular class and the parameter $neighbor_{num}$ mentioned in Algorithm 2. The number of datapoints in the root node of the tree is 11 and it is seen that by fixing the $neighbor_{num}$ to a value greater than 6 resulted in a tree structure with only three levels i.e. the root node , the second level consisting of the two nodes composed of the broader groups i.e. cat/dog and computer, followed by the third level consisting of all leaf nodes.This structure was obtained irrespective of the different initial datasplits considered. The coressponding structure is given below:

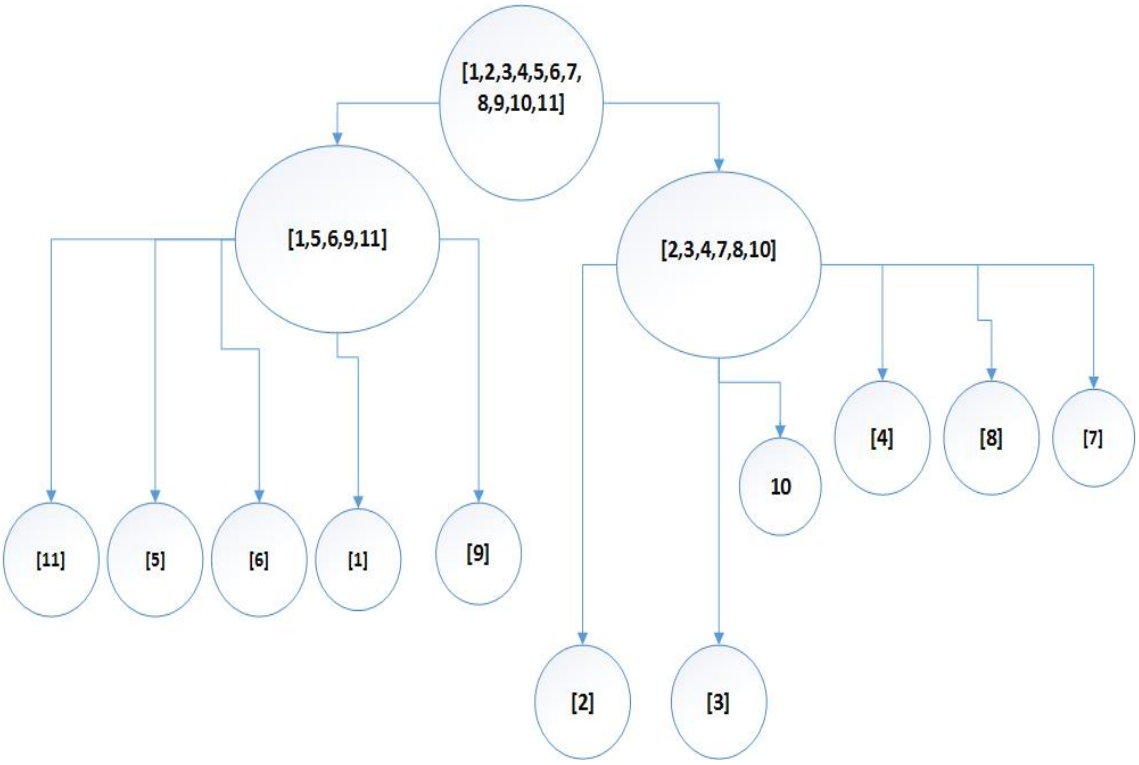


Figure 4.13: **Tree structure obtained when the number of nearest neighbor** $neighbor_{num}$ **was 7(For class numbers refer to Table 4.4)**

In order to obtain groupings at finer levels the parameter $neighbor_{num}$ was varied according to the number of classes. The ranges considered are given below:

- $numclass > 15$**:** $neighbor_{num} = 7$
- $numclass \geqslant 10$ & $numclass < 15$**:** $neighbor_{num} = 5$

- $numclass \geqslant 5$ & $numclass < 10$: $neighbor_{num} = 3$
- $numclass < 5$: $neighbor_{num} = 2$

4.4 Label Consistent KSVD based hierarchial classification of visually correlated categories

In this section the goal is to use the hierarchial model designed by the recursive application of self tuning spectral clustering so that joint learning of classifier and dictionary is done at each non-leaf node. Then for any given test sample the sparse code can be computed at each non leaf node using the corresponding dictionary and feed the sparse code learnt to the classifier associated with that node . The output of the classifier will be considered for selecting the child node for traversal.

4.4.1 Label Consistent KSVD

Since the goal is to learn classifier and dictionary in a joint fashion , Label Consistent KSVD proposed by Jiang [3] is the perfect choice since it learns a dictionary with good discriminative and reconstruction power by imposing a label consistency condition on the sparse codes and learning a linear classifier simultaneously. The linear classifier's performance depends on the discriminative nature of the sparse codes , which is imposed by the label consistency term . Given a set of training samples of C classes, $Y \in R^{P \times N}$, a dictionary $D \in R^{P \times K}$, set of discriminative sparse codes $Q \in R^{K \times N}$, set of label vectors $H \in R^{C \times N}$ and a classifier matrix $W \in R^{C \times K}$, the overall objective function for joint learning of the dictionary, classifier matrix and sparse codes is given by:

$$
\begin{gathered}
< D, T, W, X >= \underset{D,T,W,X}{argmin} \left(\|Y - DX\|_F^2 + \alpha\|Q - TX\|_F^2 + \beta\|H - WX\|_F^2\right) \\
s.t\ \forall i \|x_i\|_0 \leq Thresh
\end{gathered} \tag{4.25}
$$

Here α and β are the relative weights assigned to the label consistency terms and classifier terms respectively. $Thresh$ is the sparsity threshold considered. If dictionary $D = [d_1, d_2, d_3]$ and $Y = [y_1, y_2, y_3]$ are the dictionary atoms and the set of training samples , then if d_1, y_1 are from class 1 , d_2, y_2 are from class 2 and d_3, y_3 are from class

3, then the Q matrix is given by:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

For a particular discriminative sparse code q_i the nonzero value occurs at q_{ij} where the training sample y_i and a dictionary atom d_j has the same label.

Optimization model

The equation (4.25) can be rewritten as :

$$< D, T, W, X >= \underset{D,T,W,X}{argmin} \left\| \begin{pmatrix} Y \\ \sqrt{\alpha}\ Q \\ \sqrt{\beta}\ H \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\alpha}\ T \\ \sqrt{\beta}\ W \end{pmatrix} X \right\|_F^2 \tag{4.27}$$
$$s.t\ \forall i \|x_i\|_0 \leq Thresh$$

Denoting $\begin{pmatrix} D \\ \sqrt{\alpha}\ T \\ \sqrt{\beta}\ W \end{pmatrix} = D^{'}$ and $\begin{pmatrix} Y \\ \sqrt{\alpha}\ Q \\ \sqrt{\beta}\ H \end{pmatrix} = Y^{'}$ the above equation can be recast as

$$< D^{'}, X >= \underset{D^{'},X}{argmin} \|Y^{'} - D^{'}X\|_F^2 \tag{4.28}$$
$$s.t\ \forall i \|x_i\|_0 \leq Thresh$$

The above equation can be solved using KSVD, where the update of the sparse code and dictionary atom is done in the same fashion as chapter 3, section(3.1.2).

LCKSVD Classification Scheme

After obtaining $D^{'}$ and X where $D^{'} = \begin{pmatrix} D \\ \sqrt{\alpha}\ T \\ , \sqrt{\beta}\ W \end{pmatrix}$ from (4.28) it is not possible to use D, T, W directly since the columns of D, T, W are jointly normalized i.e. for the ith column of the respective matrices we have $\| \begin{pmatrix} d_i \\ \sqrt{\alpha}\ t_i \\ , \sqrt{\beta}\ w_i \end{pmatrix} \|_2 = 1$. For any test sample y we have the following :

$$y = Dx = \sum_{i=1}^{K} d_i x_i = \sum_{i=1}^{K} x_i \|d_i\|_2 \frac{d_i}{\|d_i\|_2} = \sum_{i=1}^{K} \hat{x}_i \hat{d}_i \tag{4.29}$$

Here $\hat{d_i}$ is the ith column of the normalized dictionary D_{norm}. Also for the classifier parameters W we have

$$l = Wx = \sum_{i=1}^{K} W_i x_i = \sum_{i=1}^{K} x_i \|d_i\|_2 \frac{W_i}{\|d_i\|_2} = \sum_{i=1}^{K} \hat{x_i}\hat{W_i} \tag{4.30}$$

where $\hat{W_i}$ is the ith column of the normalized classifier parameters W_{norm}. Thus after computation of the $D^{'}$ from (4.28) the normalized dictionary and classifier parameters must be computed using the following equation:

$$\begin{aligned} D_{norm} =& \{\frac{d_1}{\|d_1\|_2}, \ldots, \frac{d_K}{\|d_K\|_2}\} \\ W_{norm} =& \{\frac{w_1}{\|d_1\|_2}, \ldots, \frac{w_K}{\|d_K\|_2}\} \end{aligned} \tag{4.31}$$

4.4.2 Node specific dictionary learning

This step of node specific dictionary learning is preceded by the construction of the hierarchial tree like structure which is constructed using Self tuning spectral clustering ans the steps mentioned are same as that considered in Algorithm 4(Build Tree). Then for a particular non leaf node , its children are assigned broad class labels and the LCKSVD routine is used for learning a dictionary D_{node}, transformation matrix T_{node}, classifier parameters W_{node} . Thus for each non leaf node we have a multi-class classification problem with each broad class associated with a particular child node.

$$\begin{aligned} < D_{node}, T_{node}, W_{node}, X_{node} >= \underset{D_{node}, T_{node}, W_{node}, X_{node}}{argmin} \left\| \begin{pmatrix} Y_{node} \\ \sqrt{\alpha}\ Q_{node} \\ \sqrt{\beta}\ H_{node} \end{pmatrix} - \begin{pmatrix} D_{node} \\ \sqrt{\alpha}\ T_{node} \\ \sqrt{\beta}\ W_{node} \end{pmatrix} X_{node} \right\|_F^2 \\ s.t\ \forall i \|x_{node} i\|_0 \leq Thresh \end{aligned} \tag{4.32}$$

4.4.3 Classification scheme

The classification of the test sample y occurs in a similar top down fashion as mentioned in the Algorithm 9. Instead of following an energy based method as mentioned in Algorithm 9 here the sparse code of the test sample is computed w.r.t. the node's dictionary D_{node} which is given as input to the classifier (parameters denoted by W_{node}) and the entry where the largest value occurs is considered as the broad class label (i.e. the next node for traversal). Note in this case W_{node} and D_{node} must be normalized according to (4.31) before being used for the test sample y.

Algorithm 11 Classification algorithm-LCKSVDtree

```
Input: test sample $x_t$, tree parameters $T = (Nodemark, Dset, Wset, Labelset)$
$Nodemark$ : numbering associated with tree nodes at each level.
$Dset$ : Set of dictionaries associated with the tree.
$Wset$ : Set of classifier parameters $W_{node}$ associated with the nodes of the tree.
$Labelset$ :Set of labels associated with the tree nodes.
Output: predicted label $y_{pred}$ of the test sample $y$
$level \leftarrow 1$                                   ▷ current level of the tree
$index \leftarrow 1$                                   ▷ root node accessed
$labelcurr \leftarrow Labelset\{index\}$               ▷ set of labels of the root node
while ($|labelcurr| \neq 1$) do
    $D_{node} \leftarrow Dset\{index\}$, $W_{node} \leftarrow Wset\{index\}$
    $spcode = compute_{sparse}(D_{node}, x_t)$
    $Hvect \leftarrow W_{node} * spcode$
    $classc \leftarrow \underset{j}{argmax}\ Hvect(j)$
            ▷ $classc$ is the broad class label associated with the child node which gives
    maximum entry in $Hvect$
    $level \leftarrow level + 1$
    $index = f(classc, level)$
    $labelcurr \leftarrow Labelset\{index\}$
end while
$y_{pred} \leftarrow labelcurr$
```

$compute_{sparse}$ is any sparse coding algorithm like OMP . $f(classc, level)$ function returns the node-number associated in the current level coressponding to the broad class label *classc*.

4.4.4 Preprocessing using Linearized Kernel Dictionary Learning

Golts and Elad [40] proposed the LKDL (Linearized Kernel Dictionary Learning) with the aim of handling a large Kernel matrix (in case of dictionary learning) effectively by providing an effective approximation followed by preprocessing of the training data to create virtual samples , which are used as inputs to standard dictionary learning algorithms.

Kernel matrix approximation

Considering the set of samples to be denoted by $Y \in R^{K \times N}$ and the associated kernel matrix is given by $\mathcal{K} \in R^{N \times N}$. Considering that the Mercer's criteria of positive semi-definitenes are satisfied and the feature map is given by $\phi(.)$, a particular entry of the kernel matrix is given by:

$$\mathcal{K}(i,j) =< \phi(y_i), \phi(y_j) > \tag{4.33}$$

Considering $\phi(y_i) = f_i$ where f_i can be considered to be the feature vector associated with y_i in the mapped space. Thus the entire kernel matrix can be represented as follows:

$$\mathcal{K} = \mathcal{F}^T \mathcal{F} \tag{4.34}$$

$\mathcal{F} \in R^{p \times N}$ where p is the dimension of the feature space and f_i can be considered as virtual sample associated with the input sample y_i. In order to obtain an expression for H the following steps are followed:

- Obtain eigenvector decomposition of $\mathcal{K}$ i.e. $\mathcal{K} = V \sum V^T$ where U is a diagonal matrix consisting of the eigenvalues in descending order V contains the corresponding orthonormal eigenvectors.

- Considering $\mathcal{F}^T\mathcal{F} = V \sum V^T$, an approximation of $\mathcal{F}$ is given by $\mathcal{F} = \sum^{1/2} V^T = \sum^{-1/2} V^T K$

- Hence, for a particular input signal y the corresponding feature map(in p dimension) is given by $y \rightarrow \sum^{-1/2} V^T(\mathcal{K}(y, y_1), \mathcal{K}(y, y_2), \ldots \mathcal{K}(y, y_N))$.

Nystrom's sampling method and creation of virtual samples

Nystrom method is a standard technique for obtaining a low rank approximation of a large matrix using a subset of its columns. The approximation depends upon the sampling strategy used for selecting the subset. In terms of the kernel matrix $\mathcal{K}$, the usage of the Nystrom method is detailed as follows:

- For the kernel matrix $\mathcal{K}$ sample c columns uniformly without replacement to generate the matrix $\mathcal{C} \in R^{N \times c}$

- Considering $\mathcal{W}$ to be the kernel matrix composed of the intersection of the chosen c columns with c columns and $\mathcal{B}$ composed of the remaining $(N-c) \times (N-c)$ submatrix, the matrices $\mathcal{C}$ and $\mathcal{K}$ can be written after certain permutations:

$$\mathcal{C} = \begin{bmatrix} \mathcal{W} \\ \mathcal{S} \end{bmatrix}, \mathcal{K} = \begin{bmatrix} \mathcal{W} & \mathcal{S}^T \\ \mathcal{S} & \mathcal{B} \end{bmatrix} \tag{4.35}$$

- Nystrom approximation of K is given as:

$$\mathcal{K} \approx C\mathcal{W}^\dagger C^T \tag{4.36}$$

- Since $\mathcal{W}$ is symmetric and psd hence $\mathcal{W} = U \wedge U^T$ by eigenvector decomposition. Further $\mathcal{W}^\dagger = U \wedge^\dagger U^T$.

- Using $\mathcal{K} = \mathcal{F}^T\mathcal{F} = C\mathcal{W}^\dagger C^T$ and $\mathcal{W}^\dagger = U\wedge^\dagger U^T$ $\mathcal{F}$ can be derived as $\mathcal{F} = (\wedge^\dagger)^{\frac{1}{2}} U^T \mathcal{C}^T$. The rank-k approximation is given by

$$\mathcal{F}_k = (\wedge_k^\dagger)^{\frac{1}{2}} U_k^T \mathcal{C}^T. \tag{4.37}$$

For a given training data at hand i.e. $Traindat$ the above steps must be performed to create samples for the training data . The steps involved in preprocessing the training data $Traindat$ are listed below:

- Subsample the training set to obtain a set of c columns given by $Train_{sub} = [Traindat_{R_1}, Traindat_{R_2}, \ldots Traindat_{R_c}]$

- Compute the $\mathcal{C}$ matrix by computing the kernel between each pair of column in $Train_{sub}$ and $Traindat$

- Compute the kernel between the columns in $Train_{sub}$ to obtain $\mathcal{W}$.

- Use the rank-k eigen decomposition of $\mathcal{W}$ to obtain the final expression of $\mathcal{F}_k$ in (4.37), which is used as virtual samples for training set $Traindat$.

- For a particular test sample y_{test} using the pre-computed $(\wedge_k^\dagger)$ and U_k^T in (4.37), the test sample is mapped as $f_{test} = (\wedge_k^\dagger) U_k^T [\mathcal{K}(y_{test}, Traindat_{R_1}), \ldots \mathcal{K}(y_{test}, Traindat_{R_c})]$

The preprocessing phase has been used for both the training and test sets in our case and the related results are reported.

4.4.5 Results

The preprocessing on the training set is done for each node in order to obtain a set of virtual samples corresponding to each node . Sampling ratio i.e. $\frac{c}{N}$ is set to 1 i.e. subsampling is not performed . Dictionary size is set to 400 and the sparsity threshold was fixed at 30. The dimension of each virtual sample or the number of rows in F_k is fixed to 500. For the purpose of comparison the tree structure $tree_2$ mentioned in the Table 4.3 is considered and the performance without kernel pre-processing is compared with that of polynomial kernel ($degree = 2, c = 0$) and gaussian kernel ($sigma = 1$). In each case the performance at the best optimal hyperparameters are reported.

Kernels	$\sqrt{\alpha}$	$\sqrt{\beta}$	Testing accuracy
Without kernel preprocessing	2×10^{-3}	0.1	62.61 %
Gaussian($\sigma = 1$)	1×10^{-2}	0.1	63.84 %
Polynomial(degree=2, c=0)	2×10^{-3}	0.1	63.414 %

Table 4.11: **Comparison of the testing accuracies between different kernels and the best hyperparameter settings corresponding to each of them**

Using Gaussian kernel the discrimination of the training samples increased thus resulting in increase in the testing accuracy by 1.2 %. The hyperparameters $\sqrt{\alpha}$ and $\sqrt{\beta}$ are obtained for each case using a grid search technique in the range $[1 \times 10^{-4}, 10]$ in steps of 10.

Chapter 5

Conclusions and Future work

In this work we have mainly considered the two specific classification problems i.e. facial recognition and object recognition. In case of facial recognition the sparsity based approach was implemented and its analysis was done w.r.t different feature extraction schemes and its performance was compared with state of the art classifiers like K-NN , Support Vector Machine. Significant stress has been given to the problem of object recognition in chapter 4 where a hierarchial model for organizing visually correlated categories was proposed . The model proposed was based on exploiting the visual correlation among the classes and used a particular variant of the Spectral Clustering scheme to automatically determine the number of groups among the image categories. The organization was followed by adaptation of the Large margin nearest neighbor classification scheme for the hierarchial structure learnt earlier . Detailed analysis of the feature extraction schemes used were given. Further efforts were made to obtain the best set of hyperparameters for each node using Bayesian Optimization routine. It was seen that node specific large margin nearest neighbor performed significantly well when the hyperparameters of the nodes are obtained using Bayesian optimization instead of grid search. The node specific maximum margin framework was followed by the idea of joint learning of classifier and dictionary in each node of the tree . The state of the art dictionary learning algorithm LCKSVD was used for the joint learning purpose . Further the idea of using virtual training samples for dictionary learning was investigated in the same proposed framework.

Future work involves checking the effect of different kernels and sampling strategies while obtaining the virtual samples for the proposed hierarchial dictionary learning scheme. Also in the node specific maximum margin scheme, one approach for tuning hyperparameters that can be investigated is the application of the Bayesian optimization over the entire tree i.e. one set of hyperparameters for the entire tree structure. Further the organization of the image categories on the basis of visual correlations can be replaced using the confusion among the classes as a metric i.e. spectral clustering techniques can be applied to the confusion matrices learnt per node using classifiers. In this way classes which have a tendency to be confused among themselves will be grouped together in subsequent levels. Since the model has been tested on only 11 classes of the Imagenet database , future goal will be to test it on bigger sets having multiple visually correlated classes , and determine the hierarchial structure for classification.

Appendix A

Appendix 1

Entire pseudocode for the node specific maximum margin learning(HSpecLMNN) with setting of optimal hyperparameters is given .

Algorithm 12 HSpecLMNN

```
**Inputs:** $Trdat$ (Training data), $Valdat$(Validation data), $numclass$(number of classes )
**Initialization:** $leafnumber \leftarrow 0$ , $level \leftarrow 1$, $labelval\{level\}\{1\} \leftarrow [1, 2, \ldots, numclass]$, $clustinit \leftarrow 1$ $Kcell \leftarrow \varnothing$ , $Lcell \leftarrow \varnothing$, $Mcell \leftarrow \varnothing$, $numnode\{level\} \leftarrow 1$, $idlevel\{level\} \leftarrow 0$, $childnum \leftarrow \varnothing$, $idtot \leftarrow \varnothing$
while ($leafnumber \neq numclass$) do
    if ($level == 1$) then
        $labelacc = labelval\{level\}\{1\}$
        $[numchild, M, L, K, labelchild] = $ LMNNcdiv$(TrDat, Valdat, labelacc)$
        $Mcell\{level\}\{1\} \leftarrow M$ , $Lcell\{level\}\{1\} \leftarrow L$ , $Kcell\{level\}\{1\} \leftarrow K$
        $numnode\{level + 1\} \leftarrow numchild$, $childnum\{level + 1\} \leftarrow numchild$
        $[id] = $ checkleaf$(labelchild, numchild)$
        $[idtot] \leftarrow [idtot, id]$, $idlevel\{level + 1\} \leftarrow id$
        for $j = 1 : numchild$ do
            $labelval\{level + 1\}\{j\} = labelchild\{j\}$
        end for
```

Algorithm 13 HSpecLMNN - Part 2

```
    else
        $nl \leftarrow numnode\{level\}$, $idtemp \leftarrow idlevel\{level\}$, $idcurrlevel \leftarrow \varnothing$
        for $j = 1 : nl$ do
            if $(idtemp(j) == 0)$ then
                $labelacc = labelval\{level\}\{j\}$
                $[numchild, M, L, K, labelchild] = \textbf{LMNNcdiv}(TrDat, Valdat, labelacc)$
                $Mcell\{level\}\{j\} \leftarrow M$ , $Lcell\{level\}\{j\} \leftarrow L$ , $Kcell\{level\}\{j\} \leftarrow K$
                $sizel[j] = numchild$
                $[id] = \textbf{checkleaf}(labelchild, numchild)$
                if $(j == 1)$ then
                    for $k = 1 : numchild$ do
                        $labelval\{level + 1\}\{k\} = labelchild\{k\}$
                    end for
                    $start \leftarrow numchild$
                else
                    $lb \leftarrow start + 1$, $ub \leftarrow start + numchild$
                    for $k = lb : ub$ do
                        $labelval\{level + 1\}\{k\} = labelchild\{k - start\}$
                    end for
                    $start = start + numchild$
                end if
            else
                if $(j == 1)$ then
                    $start \leftarrow 0$
                else
                    $start \leftarrow start$
                end if
            end if
        end for
```

Algorithm 14 HSpecLMNN - Part 3

```
        $idl\{level+1\} \leftarrow idcurrlevel$
        $leafnum \leftarrow leafnum + \sum_i idcurrlevel[i]$
        $numnode\{level+1\} = \sum_i sizel[i]$
        $childnum\{level\} = sizel$
    end if
end while
```

label{*level*}{*j*} holds the set of class labels associated with the *jth* child at current level *level*. *numnode*{*level*} holds the holds the number of nodes in the tree at current level *level*. *childnum*{*level*} determines the number of child members of the nodes in the current level. **LMNNCdiv** function is responsible for carrying out the split of the current tree node considered and returns the learnt Mahalnobis distance metric **M**, the matrix of linear transformation **L** and the number of target neighbors **K**. The **checkleaf** function operates on the child nodes of a particular parent node and returns an identity array such that a boolean value of 1 indicates that the node is a leaf node and 0 indicates it is a non-leaf node.

The pseudo codes for **LMNNCdiv** and **checkleaf** are given as follows:

Algorithm 15 LMNNCdiv

```
Inputs:$Traindat$(Training data associated with current node), $Valdat$(Validation data associated with current node)$labelacc$(Set of labels associated with current node)
Outputs: $numchild$, $M$, $L$, $K$, $labelchild$
Initialization: $TrDatnode \leftarrow \varnothing$ , $ValDatnode \leftarrow \varnothing$, $meanset \leftarrow \varnothing$ $TotDatNode \leftarrow \varnothing$
if $size(labelchild, 2) == 2$ then
    $numchild \leftarrow size(labelchild, 2)$
    $labelchild\{1\} \leftarrow labelacc(1), labelchild\{2\} \leftarrow labelacc(2)$
    for $i = 1 : numchild$ do
        $TrDatNode \leftarrow [TrDatNode, Traindat_{labelacc(i)}]$
        $ValDatNode \leftarrow [ValDatNode, Valdat_{labelacc(i)}]$
        $TotDatNode \leftarrow [TotDatNod, Traindat_{labelacc(i)}, Valdat_{labelacc(i)}]$
    end for
```

Algorithm 16 LMNNCdiv-Part 2

```
$[K, outdim, maxiter]$ = **findbestparams**($TrDatNode, ValDatNode$)
$[L]$ = **trainLMNN**($TotDatNode, K, outdim, maxiter$)
$M \leftarrow L^T L$
$numclass \leftarrow size(labelchild, 2)$
**for** $i = 1 : numclass$ **do**
    $n_i = size(Traindat_{labelacc(i)}, 2)$
    $meanset \leftarrow [meanset, \frac{\sum_i Traindat_{labelacc(i)}}{n_i}]$
**end for**
$clustrange \leftarrow [2, \ldots, numclass]$
$[bestclust, clustdata, numchild]$ = **SelfTuneSpectral**($meanset, clustrange$)
                                        ▷ *bestclust(i)* holds the datamember ids of the *ith* cluster
                    ▷ *clustdata(i)* holds the cluster number coressponding to the *ith* datamember
**for** $i = 1 : numchild$ **do**
    $indnum \leftarrow bestclust\{i\}$
    $trtemp \leftarrow \varnothing$ $vartemp \leftarrow \varnothing$
    **for** $j = 1 : size(indnum, 2)$ **do**
        $index \leftarrow indnum(j)$
        $trtemp \leftarrow [trtemp, TrainDat_{labelacc(index)}]$
        $valtemp \leftarrow [valtemp, ValDat_{labelacc(index)}]$
    **end for**
    $TrDatNode \leftarrow [TrDatNode, trtemp]$
    $ValDatNode \leftarrow [ValDatNode, valtemp]$
**end for**
$[K, outdim, maxiter]$ = **findbestparams**($TrDatNode, ValDatNode$)
$[L]$ = **trainLMNN**($TotDatNode, K, outdim, maxiter$)
$M \leftarrow L^T L$
**for** $i = 1 : numclust$ **do**
    $findex \leftarrow \varnothing$
    **for** $j = 1 : size(clustdata, 2)$ **do**
        **if** $clustdata(j) == i$ **then**
            $findex \leftarrow [findex, j]$
        **end if**
```

Algorithm 17 LMNNCdiv -Part 3

```
            end for
            $labelchild\{i\} = labelacc(findex)$
        end for
    end if
```

The function **findbestparams** employs bayesian optimization technique for finding the best set of hyperparameters for each node . Here the number of target neighbors coressponding to the coarse classification performed at the specific node i.e. K , the dimension of the linear transformation matrix L and the number of maximum iterations *maxiter* are treated as hyperparameters. The **trainLMNN** function trains the entire model using the best set of hyperparameters found by the bayesian optimization routine.

Algorithm 18 checkleaf

```
    Inputs: $numchild$(Number of children associated with current node), $labelchild$(Set of labels associated with child nodes)
    Outputs: $id$ ($id[i] = 1$ means the ith child node is leaf or otherwise)
    for $i = 1 : numchild$ do
        $tmplb \leftarrow labelchild\{i\}$
        if $size(tmplb, 2) == 1$ then
            $id[i] = 1$
        else
            $id[i] = 0$
        end if
    end for
```

Appendix B

Appendix 2

The optimal hyperparameters found for each node of the tree structure $tree_2$ using bayesian optimization :

Level number/ Node number	run 1	run 2

1/1	K=15,
	outdim=119,
	maxiter=11
		K=12,
		outdim=388,
		maxiter=11

2/1	K=14,
	outdim=4086,
	maxiter=15
		K=7,
		outdim=3331,
		maxiter=28

2/2	K=15,
	outdim=107,
	maxiter=20
		K=14,
		outdim=3930,
		maxiter=20

3/6	K=12,
	outdim=84,
	maxiter=13
		K=12,
		outdim=4117,
		maxiter=13

3/8	K=8,
	outdim=3951,
	maxiter=17
		K=10,
		outdim=4058,
		maxiter=12

Table B.1: **Optimal hyperparameters obtained for the nodes of the tree structure $tree_2$ over two runs of bayesian optimization**

K	outdim
4	1600

Table B.2: **Optimal hyperparameters for the entire tree structure obtained using grid search**

Here the number of maximum iterations was fixed at 200.

Bibliography

[1] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[2] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2010, pp. 2691–2698.

[3] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent k-svd: Learning a discriminative dictionary for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2651–2664, 2013.

[4] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Sparse representation based fisher discrimination dictionary learning for image classification," *International Journal of Computer Vision*, vol. 109, no. 3, pp. 209–232, 2014.

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

[6] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision.* Springer, 2006, pp. 404–417.

[7] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008.* IEEE, 2008, pp. 1–8.

[8] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[9] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[10] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 524–531.

[11] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 2169–2178.

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

[13] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 3360–3367.

[14] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546.

[15] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, 2004, pp. 513–520.

[16] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, 2005, pp. 1473–1480.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition, 2009.* IEEE, 2009, pp. 248–255.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Advances in Neural Information Processing Systems*, 2010, pp. 163–171.

[20] J. Deng, S. Satheesh, A. C. Berg, and F. Li, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *Advances in Neural Information Processing Systems*, 2011, pp. 567–575.

[21] N. Zhou, Y. Shen, J. Peng, and J. Fan, "Learning inter-related visual dictionary for object recognition," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2012, pp. 3490–3497.

[22] M. elad, *Sparse and Redundant Representations: from theory to applications in signal and image processing.* Springer, 2009.

[23] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution," *Communications on pure and applied mathematics*, vol. 59, no. 6, pp. 797–829, 2006.

[24] E. Candes and J. Romberg, "l1-magic: Recovery of sparse signals via convex programming," 2005.

[25] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[26] A. M. Martinez, "The ar face database," *CVC Technical Report*, vol. 24, 1998.

[27] D. Donoho and J. Tanner, "Counting faces of randomly projected polytopes when the projection radically lowers dimension," *Journal of the American Mathematical Society*, vol. 22, no. 1, pp. 1–53, 2009.

[28] J. Shao, C. Change Loy, and X. Wang, "Scene-independent group profiling in crowd," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2219–2226.

[29] M. Aharon, M. Elad, and A. Bruckstein, "Svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[30] M. Yang, L. Zhang, J. Yang, and D. Zhang, "Metaface learning for sparse representation based face recognition," in *2010 IEEE International Conference on Image Processing*. IEEE, 2010, pp. 1601–1604.

[31] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 3501–3508.

[32] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Advances in neural information processing systems*, 2009, pp. 1033–1040.

[33] S. Kong and D. Wang, "A dictionary learning approach for classification: separating the particularity and the commonality," in *European Conference on Computer Vision*. Springer, 2012, pp. 186–199.

[34] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 849–856.

[35] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," 2005.

[36] I. Feinerer and K. Hornik, *wordnet: WordNet Interface*, 2016, r package version 0.1-11. [Online]. Available: https://CRAN.R-project.org/package=wordnet

[37] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[38] M. Schmidt, "minfunc toolbox," *http://www.di.ens.fr/ mschmidt/Software/minFunc.html.*

[39] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[40] A. Golts and M. Elad, "Linearized kernel dictionary learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 726–739, 2016.