A Dissertation Report On

MACHINE LEARNING BASED FACE RECOGNITION USING PARTICLE SWARM OPTIMIZATION

Submitted in partial fulfilment of the requirements

For the award of the degree of

Master of Technology

In

Software Engineering

By

RAJEEV BHATT

(Roll No. 2K13/SWE/13)

Under the guidance of

Dr. O.P. Verma

(Prof. and Head, Department of CSE)

Delhi Technological University, Delhi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

2013-2015

DELHI TECHNOLOGICAL UNIVERSITY

CERTIFICATE

This is to certify that the project report entitled **Machine Learning based Face Recognition using Particle Swarm Optimization** is a bona fide record of work carried out by Rajeev Bhatt (2K13/SWE/13) under my guidance and supervision, during the academic session 2013-2015 in partial fulfilment of the requirement for the degree of Master of Technology in Software Engineering from Delhi Technological University, Delhi.

Dr. O.P. Verma

Professor and Head,

Department of Computer Science and Engineering

Delhi Technological University

Delhi-110042

DELHI TECHNOLOGICAL UNIVERSITY

ACKNOWLEDGEMENT

With due regards, I hereby take this opportunity to acknowledge a lot of people who have supported me with their words and deeds in completion of my research work as part of this course of Master of Technology in Software Engineering.

To start with I would like to thank the almighty for being with me in each and every step of my life. Next, I thank my parents and family for their encouragement and persistent support.

I would like to express my deepest sense of gratitude and indebtedness to my guide and motivator, **Dr. O.P. Verma**, Professor and Head, Department of Computer Science and Engineering, Delhi Technological University for his valuable guidance and support in all the phases from conceptualization to final completion of the project.

I humbly extend my grateful appreciation to all the faculties and PhD. Scholars of Computer Science and Engineering Department, Delhi Technological University who have enlightened me during my project. Last but not the least; I would like to thank all the people directly and indirectly involved in successfully completion of this project.

Rajeev Bhatt

Roll No. 2K13/SWE/13

ABSTRACT

Machine Learning is an interdisciplinary field that has its applications in various fields. Machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Face recognition has so many important applications that researchers have been doing a lot of work for decades to improve accuracy and speed of the face recognition algorithms. We present a systematic comparison of machine learning methods applied to the problem of Face Recognition. We report results on a series of experiments comparing recognition using minimum distance classifier, support vector machines (SVM) and minimal complexity machines (MCM). We also explored feature selection techniques by the use of Particle Swarm Optimization in addition to Principle Component Analysis. Best results were obtained by selecting features using PCA, PSO and doing classification with Minimal Complexity Machines. The system operates in real-time and obtained 96.6% accurate results on the Cohn-Kanade expression dataset.

Table of Contents

List of Figures

List of Tables

CHAPTER-1

INTRODUCTION

Face recognition is such a natural activity, which a person unnoticeably does many number of times in his daily life. Since the 1960s, researchers are doing a lot of work on face recognition [1]. However, the accuracy and speed of the face recognition algorithms still remain a great challenge to the researchers. Now a day's face recognition is used in various applications like national security, Surveillance, drivers licence verification, General identity verification tools, "Smart Card" applications, human-computer interface and many other [2].

Coming to the technicality face-recognition is a classification problem in which each person denotes a class and we have to classify which face belongs to which person or class. The face recognition process can be subdivided into two parts: 1- feature extraction 2- classification.

Feature extraction: - feature extraction can be thinking of as a dimensionality reduction procedure. Each image has some unique characteristics. One obvious feature that we can think of, for an image is the pixel intensities. But it would not be wise to use pixel intensities as our features because the number of features would be very large to represent an image. In feature extraction, we try to obtain as less as possible features, which can be sufficient enough to represent an image. In others words sufficient enough to differentiate an image with all other images. One most common technique for dimensionality reduction is PCA.

Classification: classification is a supervised learning [3] technique, which give a discrete value output for an unknown input. The output denotes the class label. In our case i.e. the problem of face recognition the output would be person name or a digit which uniquely represents a person. Many classification techniques [4] are there, which can be applied such as logistic regression [5], neural networks [6] support vector machines [7], minimum distance classifier [8], [9] etc.

1.1 Motivation

The motivation comes from the fact that if PCA has an efficient dimensionality reduction capacity, can we apply it on an image? If face recognition is a classification problem why should we not apply machine learning classifiers to solve face recognition?

PCA performs feature extraction by mapping image space onto Eigen faces [10] i.e. Set of images are transformed into set of Eigen vectors. These Eigen vectors are principal components. And For each image, there is a set of weights where each weight corresponds to a principal component. Therefore, each image in image space can be exactly represented by a linear combination of principal components and weights. Weight can be defined as a participation of a principal component to an image. Since the principal components are

common to all images, it's actually the weights which separate one image from another. Therefore, we choose these weights as a representative of an image i.e. features of an image.

PSO is a nature motivated optimization algorithm proposed by Kennedy and Eberhart which iteratively improves a solution with the help of improving the fitness i.e. the quality of the solution. Each particle denotes a probable and modifiable solution. Group of particles is called swarm. We have used PSO to modify the weights that we got after applying PCA.

The first classifier that we used for face recognition is minimum distance classifier, which calculates Euclidean distance of test image to all other images in the sample. The class of the 'sample having minimum distance' is considered as output class. The Euclidean distance is calculated as square root of the squared sum of differences between weights. The distance between two images can be given as.

$$\|I_1 - I_2\| = \sqrt{\sum_{i=1}^{k} (w\text{-}_1^i - w_2^i)^2}$$

The second classifier that we used for face recognition is Support Vector Machines (SVM). SVM is a classifier that separates the classes by an optimum hyperplane. The hyperplane does separation such that there exist no point which lies on a distance greater than d_{min}, where d_{min} is maximum possible distance of a point from hyperplane.

The third classifier that we used for face recognition is MCM ( ). MCM is also a hyperplane classifier just like SVM .the only difference is it generalizes the hyperplane or solves the problem of over fitting [11] by providing a bound on VC [12-13] dimension. We used MCM in order to improve the results obtained using SVM.

1.2 Related Work

Researchers have been doing a lot of research in the area of face recognition for many decades. Marian Stewart Bartlett, Gwen Littlewort, Claudia Lainscsek, Ian Fasel, and Javier Movellan have analysed "machine learning methods for fully automatic recognition of facial expressions and facial actions [14]". Xiaoguang Lu Dept. of Computer Science & Engineering has done image analysis for face recognition on the basis of appearance based and model-based schemes [15] . E. Garc´ıa Amaro, M.A. Nuno-Maganda and M. Morales-Sandoval ˜ Polytechnic University of Victoria has worked on "Evaluation of Machine Learning Techniques for Face Detection and Recognition"[16]

S.Sakthivel, Dr.R.Lakshmipathi, M.A.Manikandan worked on "Improving the Performance of Machine Learning based Face Recognition Algorithm with Multiple Weighted Facial Attribute Sets"[17] Guodong Guo, Stan Z. Li, and Kapluk Chan School of Electrical and Electronic Engineering have worked on" Face Recognition by Support Vector Machines"[18]

1.3 Problem Statement

Face recognition process is subdivided in phases feature extraction and classification .we can apply machine learning techniques in both phases. PCA for feature extraction and minimum distance classifier / SVM / MCM for classification. Moreover in extraction phase we have applied PSO for improving the features that we got after applying PCA. Therefore, the problem statement can be.

"**MACHINE LEARNING BASED FACE RECOGNITION USING PARTICLE SWARM OPTIMIZATION**"

1.4 Scope of the Work

The work aims at efficient face recognition with the help of machine learning techniques such as minimum distance classifier, support vector machines, and it improved version namely . The algorithms that we used for our work are following:

1.4.1 PCA+PSO+MDC

PCA+PSO for feature extraction and MDC for classification

1.4.2 PCA+PSO+SVM

PCA+PSO for feature extraction and MDC for classification

1.4.3 PCA+PSO+MCM

PCA+PSO for feature extraction and MDC for classification.

1.5 Organization of the Report

The report is organized as follows:

Chapter2 Describes the literature view of related works of face recognition machine learning. Its gives detailed description of all the technique that we are going to apply in our proposed work.

Chapter3 we have proposed our approaches for face recognition.

Chapter4. Experimental set-up for the model and results are discussed.

Chapter5 Concludes the discussion with the observed results and future work.

CHAPTER-2

LITERATURE REVIEW

In this chapter, we are going to explain several techniques that can be used in the process of face recognition. Following techniques can be used:

- Principal Component Analysis
- Particle swarm optimization
- Minimum distance Classifier
- Support Vector Machines
-

2.1 Principal Component Analysis (PCA):

Principal component analysis is basically used for dimensionality reduction. The main task of PCA is to analyse the data and to find and identify the patterns in it. Finding out the patterns in dataset helps us to reduce the dimensionality of dataset with minimal loss of information.

Principal component analysis is basically used to project a feature space of higher dimensionality onto a smaller subspace that represents our data "well". The motive behind reducing the dimensionality is to reduce the computational cost needed to evaluate the features.

Principal Component Analysis is a statistical procedure that illustrates the covariance structure of a set of variables. In particular it allows us to identify the principal directions in which the data varies.

For example, in figure 2.1, we represented two variable dataset in X-Y coordinate system. Using PCA we found out the principal directions in which the data varies i.e. the U-axis. The second important we have V-axis orthogonal to U-axis. If we place the $U - V$ axis system at the mean of the data it gives us a compact representation. Now we map each point of X-Y coordinate system into corresponding U-V system. Mapping the data points, makes the data de-correlated, means the co-variance between U and V becomes zero.

For a given dataset, PCA finds the principal components. These principal components are the axis or the principal directions along which the data varies. In figure1 U-V axis system are the principal components.

As we see in Figure 2.2, there are two principal axis U and V. We can also see that V component of all the data points are close to zero. This V component might have come in data points due to some external factor for example due to noise. Thus in the U-V axis system

we can represent the data set by one variable U and discard v. This is how we reduce the dimensionality of the data set. And hence the dimensionality in this case is reduced by 1.

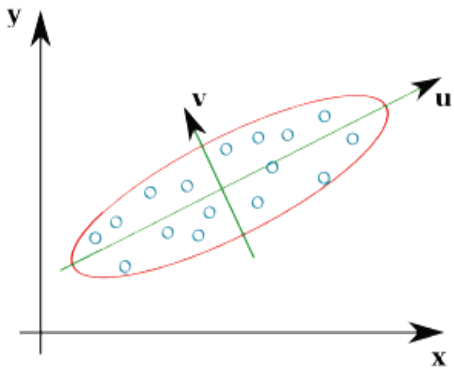

Figure 2.1: PCA for data representation

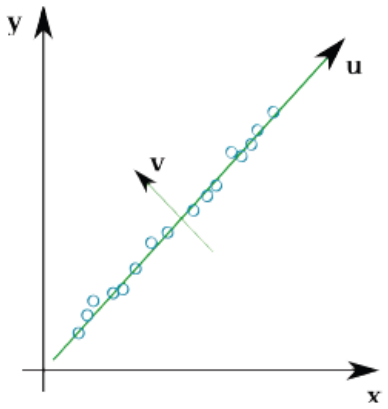

Figure 2.2: PCA for dimension reduction

General algorithm for PCA:

1. Take the entire dataset consisting n samples each of having the dimension d making d*n matrix
2. Compute the d-dimensional mean vector of the d*n matrix
3. Compute the co-variance matrix of the d*n matrix (i.e. of the entire dataset)
4. Calculate the Eigenvectors and corresponding Eigen values of the covariance matrix obtained.
5. Sort the Eigen vectors by decreasing Eigen values and choose k Eigen vectors of higher Eigen values. Each Eigen vector is of d-dimension, hence forming a d*k dimensional matrix W.
6. Use this *d* x *k* Eigenvector matrix to transform samples onto the new subspace. This can be summarized by the mathematical equation:
   $y = W^T * x$
   (where **x** is a *d* x 1 -dimensional vector represents one sample, and *y* is the transformed **k** x 1 -dimensional sample in the new subspace.)

2.2 Particle Swarm Optimization:

Particle swarm optimization (PSO) is a population based stochastic optimization technique inspired by social behaviour of bird flocking. It was developed by Dr.Eberhart and Dr.Kennedy in 1995. In PSO, system is initialized by a population of particles representing random solutions. From these random solutions we find a candidate solution and our task is to optimize that candidate solution iteratively generation by generation. The particles moves

in the search space and the position and velocity of particles are updated in each generation. In simple words PSO simulates the behaviour of bird flocking.

For example, a group of birds searching for food in a search space and there is only one piece of food in search space. No bird knows where the food is, but they know how far the food is in each iteration. The birds adopt the strategy to follow the bird which is nearest to the food. The position and velocity of each bird is updated in each iteration. In PSO, each single solution instead of "bird" is called "particle". With each iteration particles move towards the solution. Figure 2.3 illustrates that.

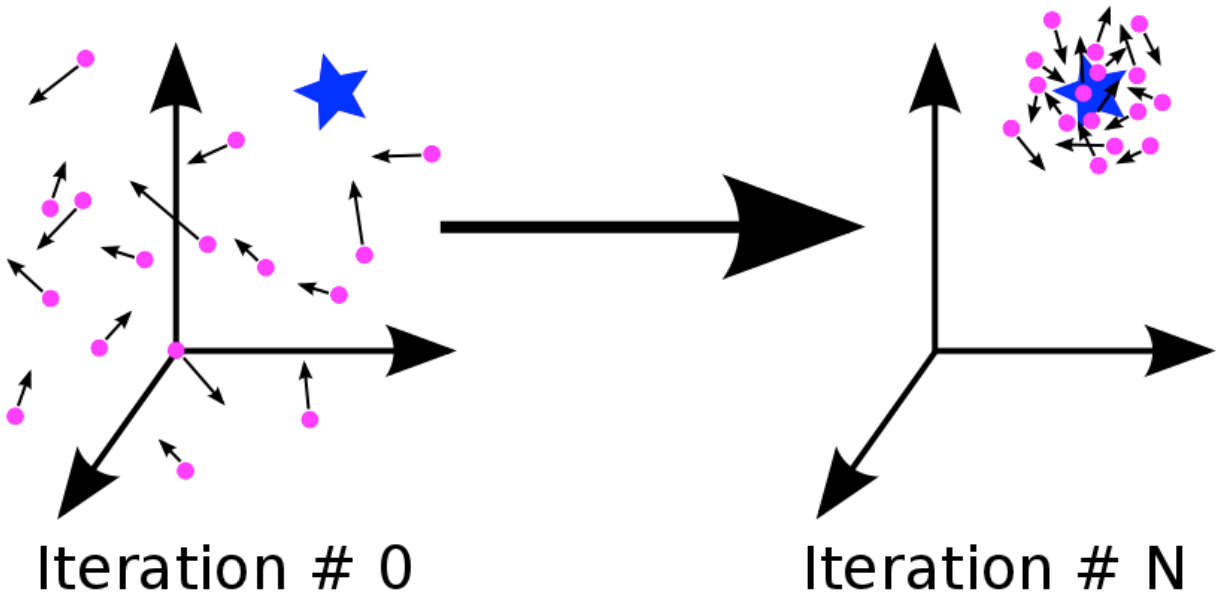

Figure 2.3: Basic nature of PSO

Pink dots represent paricles and the blue star denotes the final solution.

Fitness value of each and every particle is evaluated in each iteration. The particle having the best fitness value said to be global best or gbest and its fitness value as gbest value. The velocity of the particle that is updated in each iteration depends upon the gbest value and the personal best value(pbest) of the particle itself. . When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values gbest and pbest , the particle updates its velocity and positions with following equations(a) and(b) :

1. v[]=v[]+c1*rand()*(pbest[]-present[])+c2*rand()*(gbest-present[]) ---(a)
2. present[]=present[]+v[] ---(b)

- v[] is the particle velocity
- persent[] is the current particle (solution).
- pbest[] and gbest[] are personal best value of the particle and gbest is the global best value among all the particles.

- rand () is a random number between (0,1). c1, c2 are learning factors, usually c1 = c2 = 2.

General Algorithm for PSO:

For each particle
  Initialize particle
END

Do
  For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness value (pBest) in history
      set current value as the new pBest
  End
  Choose the particle with the best fitness value of all the particles as the gBest
  For each particle
    Calculate particle velocity according equation (a)
    Update particle position according equation (b)
  End
While maximum iterations or minimum error criteria is not attained

This is how PSO perform optimization.

2.3 Minimum Distance Classifier

Minimum distance classifier is the simplest classifier which is based on minimum error calculation. Error is calculated as norm or Euclidean distance

Say 'A' and 'B' are 2 vectors of dimension k.

Norm(A,B) = ||A-B|| = Euclidean distance $= \sqrt{\sum_{i=1}^{k}(A^i - B^i)^2}$

Suppose we are given m training examples as (x^1, y^1), (x^2, y^2),..... (x^m, y^m) .x is input and y is the output class. And there is an unknown test sample X_i. We have to determine to which class this test sample belongs to.

For this we will calculate distance of X_i from each training sample .distance if calculated as norm of the vector. And then we will select that training sample which is closest or has minimum error or has minimum Euclidean distance. The output label y of the selected training sample will be our class.

2.4 Support Vector Machines

SVM is an optimum, binary classifier. It provides unbiased and large margin hyperplane as separating boundary. It means SVM decision boundary has some larger minimum distance from any of the training example. Figure 2.4 illustrates the same:-

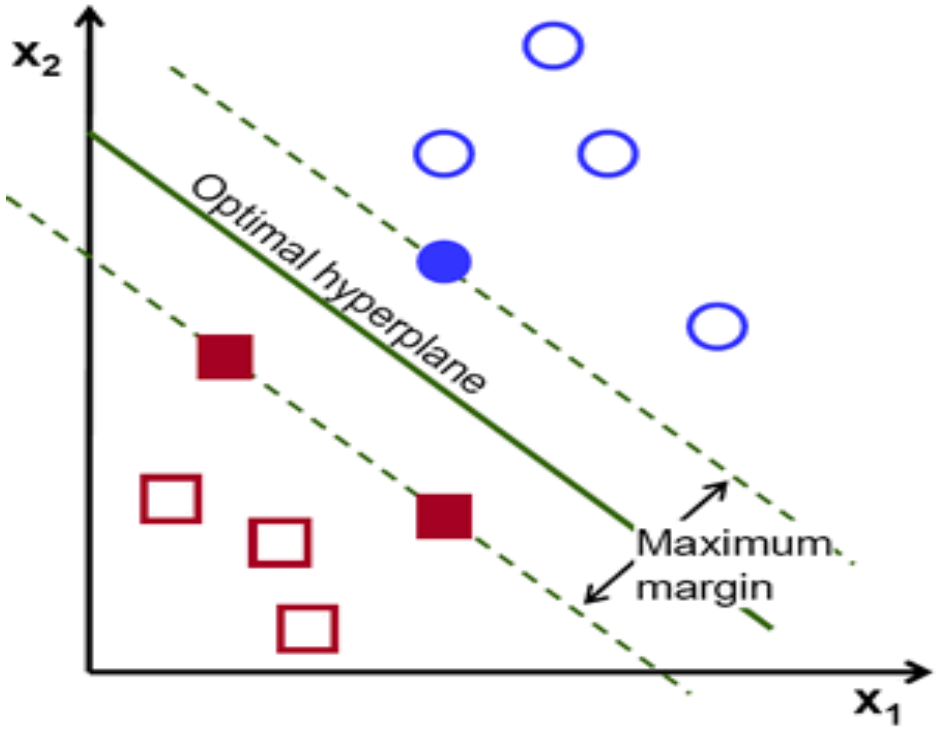

Figure 2.4: SVM : a Large Margin Classifier

SVM as a linear classifier

Say equation of hyperplane is

$g(X) = \Theta^T X + b$,

Where Θ is a vector perpendicular to the hyperplane and b is position of hyperplane in D dimensional space.

If X1 is on the positive side of hyperplane g(x1) will be greater than 0 i.e. $\Theta^T X + b > 0$ and output class label will be y=1.

If X1 is on the negative side of hyperplane g(x1) will be less than 0. i.e. $\Theta^T X + b < 0$ and output class label will be y=0.

Cost function for SVM

$J(\Theta) = c * \sum_{i=1}^{m}[y^i(-\log h(x^i)) + (1 - y^i)(-log\,(1 - h(x^i)))] + \frac{1}{2}\sum_{i=1}^{n}\Theta_j^2$

m = number of training examples, n= number of independent variables or number of dimensions.

c is a constant. If c is very large outliers will not be ignored. In practice we choose smaller c and ignore few outliers.

y^i is the output corresponding to i^{th} training example . y is either 0 or 1.

$h(x)= \frac{1}{1+e^{-\Theta^T X}}$

if y= 1, we want $h(x) \approx 1$, $\Theta^T X >> 0$

if y=0 , we want $h(x) \approx 0$, $\Theta^T X << 0$

First term of cost function $J(\Theta)$ for support vector machine can be approximately represented by following graphs(Figure 2.5) . Left one is for y=1 class and right one is for y=0 class. X-axis of graph is $\Theta^T X$ and Y- axis is first term of the cost function

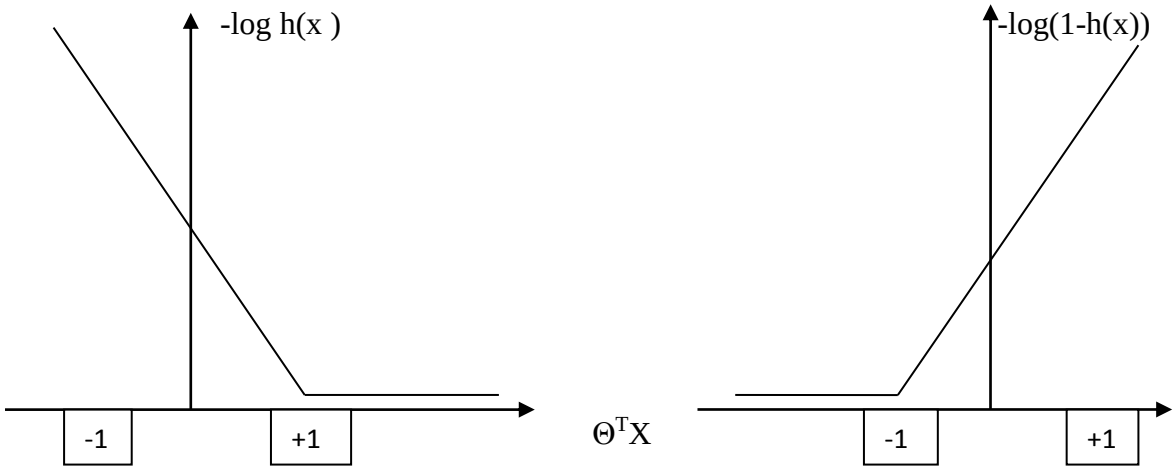

Figure 2.5: Cost Function of SVM

From the above graphs it is very clear that

if y=1, we want $\Theta^T X >= 1$ (not just >=0) and if y=0, we want $\Theta^T X <= 1$ (not just < 0) so that at least first term in the cost function would be very close to 0. This converts SVM into following optimization problem:-

$\min_{\Theta} \frac{1}{2} \sum_{i=1}^{m} \Theta_j^2$

Such that $\Theta^T X >= 1$,if y=1 and

$\Theta^T X <= 1$, if y=0

If X_i is a support vector $|| \Theta^T X_i|| = 1$

SVM with Kernels

Unfortunately, we often have datasets that have no separating hyperplane. We need to move to a non-linear solution, Ideally, we'd like to map the data into a feature space in which we can form a separating hyperplane.

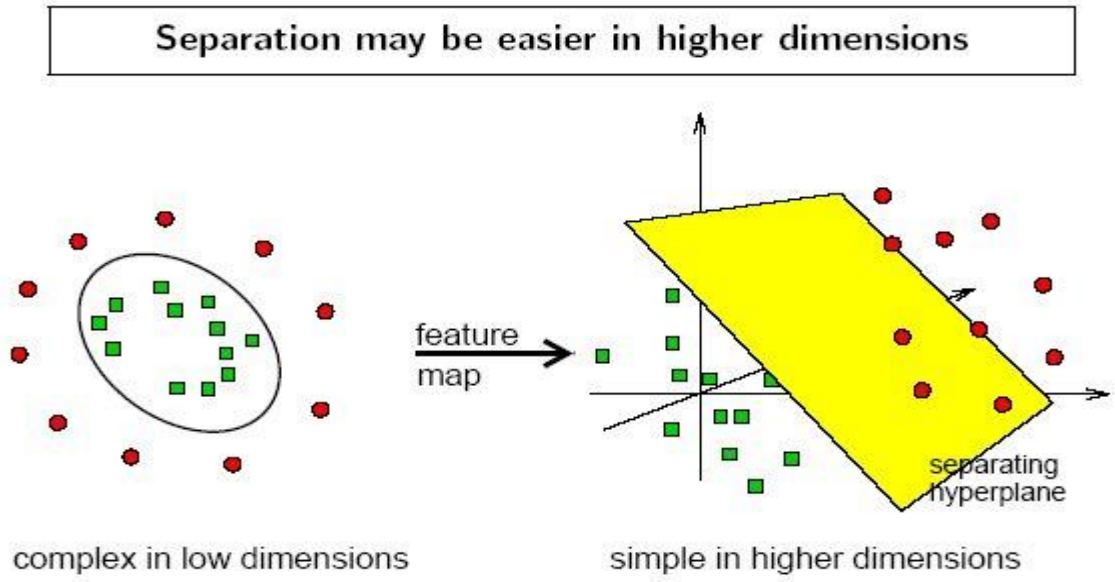

Figure 2.6: Basic Concept of Kernel

The most common algorithm for kernel SVM is following:-

Similarity between two points 'x' and 'l' is defined by function

Similarity(x,l) = $e^{\frac{-||x-l||^2}{2\sigma^2}}$ eq-1

Given m training examples as (x^1,y^1),(x^2,y^2),.....,(x^m,y^m) .

Choose $l^1=x^1,l^2=x^2,.....l^m=x^m$

For each training example x^i, y^i ,find similarity according to eq-1

f_1^i =similarity(x^i,l^1)

f_2^i =similarity(x^i,l^2)

.

.

f_m^i= =similarity(x^i,l^m)

hypothesis: Given x, compute features f

predict y=1 if $\Theta^T f >=0$

SVM training:

$$\min_{\Theta} c * \sum_{i=1}^{m} [y^i \text{cost}_1(\Theta^T f^i) + (1 - y^i)\text{cost}_0(\Theta^T f^i)] + \frac{1}{2} \sum_{i=1}^{n} \Theta_j^2$$

Here m will be equal to n.

Where $\text{Cost}_1(\Theta^T f) = -\log\left(\frac{1}{1+e^{-\Theta^T f}}\right)$ and $\text{Cost}_0(\Theta^T f) = -\log\left(1 - \frac{1}{1+e^{-\Theta^T f}}\right)$

CHAPTER-3

PROPOSED APPROOACH

This chapter discusses the model that we have used for Face Recognition. It will explain all three approaches that we have proposed.

3.1 Face Recognition model

Face recognition process is subdivided in 2 parts 1-Feature extraction 2-Classification. Our model is explained in following figure 3.1.

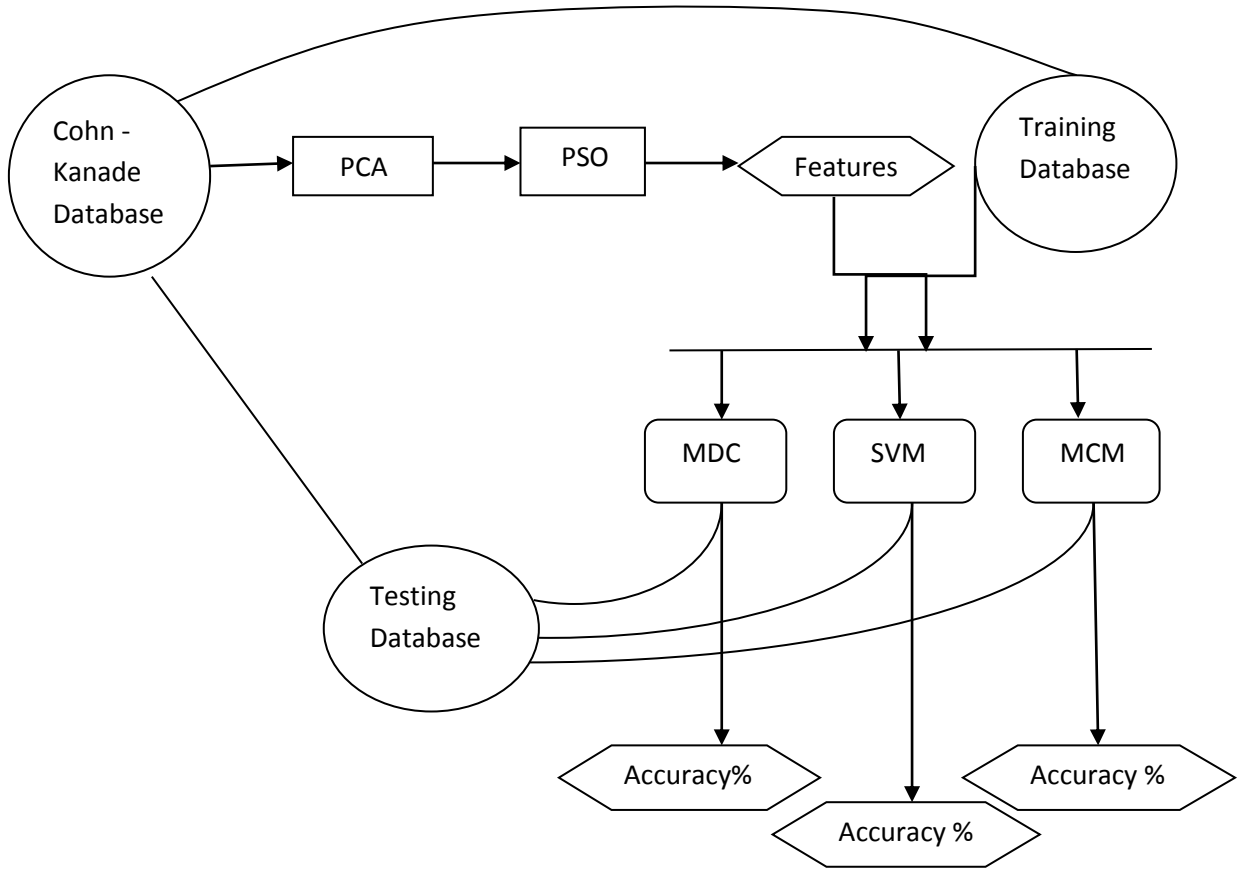

Figure 3.1 Face Recognition Model

Step 1: We take Cohn Kanade Database images for training.

Step 2: Apply PCA to find principal components. Select weights of principal components as features. Take subset of these weights corresponding to some higher Eigen values as final features.

Step 3.Apply PSO to improve the measure of subset features that we got after applying PCA.

Step 4.Train these features with the help of training database using each of the three algorithms MDC, SVM and MCM.

Step 5.Accuracy and Time consumption is calculated through number of simulation and is being compared to one another.

3.2 Feature Extraction

3.2 .1 PCA for Extracting Features

PCA is a variable reduction procedure. Chapter 2 discusses it in detail. Here we are going to discuss how PCA can be applied in Face Recognition problem. Specifically how it can give principal components of face images. For this we don't apply PCA on a single image rather we apply it on whole training set of images. These principal components are called Eigen faces. Eigen faces are like ghostly faces. Each individual face can be represented in terms of a linear combination of Eigen faces. The steps of the algorithm are following:-

1. We have m images in our training set – I_1, I_2,....., I_m. Each image is a N×N matrix of following form:-

   Where $I_i = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}$ a_{11},..... a_{NN} are pixel intensities of the image.

   We convert this image into a N^2×1 vector.

   $T_i = [a_{11}, a_{12}$ ....., $a_{NN}]^T$

2. Find average image of all the m training images.
   Avg_Img = $1/m \sum_{i=1}^{m} T_i$
3. Subtract average image from each of the images.
   $\ddot{I}_i = T_i$ - Avg_Img, for all i=1 to m.
4. Take A as collection of images in step 3. A becomes N^2×M matrix.
   A=[$\ddot{I}_1$, $\ddot{I}_2$, $\ddot{I}_3$, ... $\ddot{I}_M$]
5. X= A^T×A , where A^T denotes transpose of A. X becomes M×M matrix. X is called covariance matrix.
6. Find Eigen vectors of matrix X. M Eigen vectors will be there.
   V=[V_1, V_2, V_3, ..., V_M].
   V becomes M×M matrix.
7. For i =1 to m
   U_i = A×V_i , U_i becomes N^2×1 vector.
8. U= [U_1, U_2, U_3,.....,U_M] , Where U becomes N^2×M matrix. These are called Eigen faces. Take 'k' columns from matrix U corresponding to k highest Eigen values. Store them in matrix ¥.

¥ =[u_1,u_2,.....u_k] , ¥ is N^2×k matrix.

9. Now find weight matrix for each image.
   For image = 1 to m
   For j= 1 to k
   $W_{ji} = U_j^T \times \ddot{I}_i$
   End
   End
10. That's how we get a k×M weight matrix W corresponding to M images in training sample where one column represents features of one image.

In this way PCA provide us features for each image, which are a very less in comparison to original image.

3.2.2 PSO for Improving Representative Features

PSO is a nature based optimization algorithm, which is based on swarm intelligence. PSO iteratively improves the quality of the solution. Position of Particles represents probable and modifiable solutions. Hence the position of particles should be weights .in our algorithm. We are taking 10 particles.

3.2.2.1 Why PSO

We know that each face image can be exactly represented in terms of linear combination of Eigen faces and the weights corresponding to that image. From step 8 of PCA algorithm [3.2] we can see that we require 'm' Eigen faces for exact representation. This implies that we need 'm' weights for exact representation. Instead of m, we take 'k' Eigen faces hence 'k' weights as our representative features. If k<<m, k weights might not be good enough to substitute 'm' weights and represent the image. In our case k=6 and m=100. So we try to modify the value of these k weights with the help of PSO.

3.2.2.2 Steps of Algorithm

Features or weights of an image have 'k' values. In other words, Features of an image is a point in k dimensional space. Since k weights can almost represent an image. The better representative must lie in the neighbourhood of that point in k-dimensional space. Hence we take other particle positions in the neighbourhood of that particle. The steps of the algorithm are following:-

1. We have certain inputs for the PSO algorithm used here. They are
   Position array: It consists of particle positions initially. We represent each particle position by X_1, X_2, X_3…… X_{10}.
   X_1= W
   X_2= W+ 1 %(W)

X_3= W- 1 %(W)

X_4=W+ 2 %(W)

.

.

.

X_{10}= W +5 %(W)

Velocity array: It consists of velocity of each particle and is initialized by zero

Pbest array: It consists of particle best position. It is initialized by X_1, X_2, X_3…….. X_{10}

Gbest array: It consists of the position of best particle of the swarm.

Pfitness: It consists of the best fitness of each particle

Gfitness: It consists of the fitness of best particle

Rand (): it gives a random value between 0 and 1.

2. For each particle

Initialise particle by the inputs mentioned above

End

Do

For each particle

For each image

Reconstruct the image using S[image]= F * (X[particle][image])^T * meanface

//S will be of dimension (N^2 *1)

Fitness= Fitness + |(S[image]- L_i)|

End

If the fitness value is better than the best fitness value (Pfitness) in history

Set current fitness value to new Pfitness and update the Pbest position array with the current position of particle.

End

Choose the particle with the best fitness value of all particles and update the Gbest position with the position of the best particle.

For each particle

Update the velocity array

V[particle] = V[particle] +c_1 * rand () * (Pbest[particle] – X[particle])

+ c_2 * rand ()* (Gbest[]- X[particle])

Update the position array

X[particle]= X[particle]+ V[particle]

End

While max iterations criteria not attained

3. As a result of PSO we will get a Gbest position corresponding to the best particle which will contain the best representative weights. Gbest is our final weight matrix.

4. W = Gbest .

Our Feature Extraction process finishes over here.

3.3 Classification

With respect to face recognition problem, classification means which face belongs to which person. In our thesis we have used three classifiers in machine learning-

1. Minimum distance classifiers
2. Support vector Machines
3.

These classification algorithms are discussed below:-

3.3.1 Minimum Distance Classifier

Minimum distance classifier is based on Euclidian Distance. Euclidean distance between two images is given by

$$\|I_1 - I_2\| = \sqrt{\sum_{i=1}^{k} \left(w_1^i - w_2^i\right)^2} \qquad (1)$$

W_1 are weights corresponding to first image and W_2 are weights corresponding to second image.

Steps of face recognition from minimum distance classifier are following:-

1. Convert N×N input image into a N^2×1 vector. T_{test} is that N^2×1 vector.
2. Apply step 3 of PCA algorithm [3.2.1] i.e.
   $\ddot{I}_{test} = T_{test} - Avg\_Img,$
3. Apply step 9 of PCA algorithm [3.2.1] ie

   For j= 1 to k

   $Test\_W_j = U_j^T \times \ddot{I}_{test}$

   End

   $Test\_W_j$ are weights corresponding to test image.

4. Find Euclidean distance of Test_W to all other weights. And select that image which has minimum Euclidean distance according to Eq-1.

   Initialize Min as infinity.

   For image number =1 to M

   Dist=Euclidean distance (Test_W, W[image number])

If(Dist < Min)

Min =Dist, selected image number = image number;

End

End

5. ‘Selected image number’ is our final image which is closest to test image.
6. From the training database Output the person name to which this ‘selected image number’ belongs to.

That’s how minimum distance classifier recognizes a face.

3.3.1 Support Vector Machines

Support vector machine is a machine learning classification technique which gives an unbiased optimum hyperplane classifier. So it is more robust to noise occurred. Face recognition is a multi class classification problem. And SVM is a binary classifier. We have adopted one vs all technique [21] to do multi class classification from a binary classifier. If there are ‘Y’ classes we need ‘Y-1’ support vector machines.

Training input to support vector machine are weights and class corresponding to those weights .

Input (weights= a column of matrix W)	Output(class lebel)
W[1][1]..... W[K][1]	1
W[1][2]..... W[K][2]	1
. . .	1 . .
W[1][11]..... W[K][11]	2
. . .	2 . .
W[1][21]..... W[K][21]	3
. . .	3 . .

Table 3.1 Training Inputs for SVM

We have used SVM toolkit of MATLB for implementing SVM. Steps of training for multiclass classification by SVM are following:-

1. 'Svmtrain' [22] in a function which provides a hyperplane classifier when we feed training inputs in it.
2. In one vs all approach of multiclass classification; we separate a class from all other classes. For separating class 'X' to from the rest. we feed output label corresponding to class 'X' as 1 and output label corresponding to for all classes except class 'X' as '0'. And call 'svmtrain' to find hyperplanes.
3. We repeat step 2 until we get all the hyperplanes. If we have 'Y number of classes we repeat step 2 'Y-1' times.

Steps of testing for multiclass classification by SVM are following:-

1. Apply step 1 to 3 of minimum distance classifier algo [3.3.1] to transform a test image into its representative weights.
2. Feed these weights into "svmclassify'[23] function corresponding to first hyperplane.
3. For(hyperplane number =1 to 'Y-1')
   If 'svmclassify(hyperplane number)==1
   ;break the for loop
   Else continue checking for other hyperplanes
   End
4. Hyperplane number is our final output class label.

This is how SVM recognizes a face ,to which class or person it belongs.

CHAPTER-4

EXPERIMENTAL SETUP AND RESULTS

This chapter tells us about the experimental setup, required database and the partition of database to act as input in various stages. It also explains facial database, which we used for face recognition problem. This chapter further includes the outcome of our experiment. Time consumed and accuracy of all the classifiers in different conditions is plotted.

4.1 Experimental Setup

In our face recognition algorithm based upon machine learning classifiers, we have implemented all the algorithms namely PCA, PSO, MDC, SVM, MCM in MATLAB. We used standard Cohn -Kanade Database. It consist of 100 images of 4 persons i.e. 25 images per subject. Each image is gray level with size 92*112 pixels [40]. Each image has got different illumination, pose and facial expression. Some of the images from Cohn -Kanade database are given below.

Figure 4.1 Cohn –Kanade database

For our experiment, we use at training time 40 images. As we have 4 persons i.e. 10 images correspond to each subject. We divide our whole database into 2 parts: train data and test data. We have used 20 images as our test data. We train the system through train data. Test_data is used to test our train model and give results in the form of Accuracy.

5.2 Performance Analysis

Performance of our algorithm is measure by 2 parameters:-

1. Accuracy
2. Time

Accuracy of a classifier is defined as

$$\frac{Number\ of\ classified\ examples}{number\ of\ classified\ examples + number\ of\ misclassified\ examples} \times 100$$

5.3 Results

In it various graphs are being plotted to show results and analysis of various algorithms is being done, through contrasting and comparing by various elements like time consumption, accuracy.

Feature Extraction Phase

First of all results of feature selection phase are explained. In feature selection phase we apply two algorithm PCA followed by PCA. We run 10 simulations for calculating time taken by both the algorithms. This is showed in following table.

Simulation number	1	2	3	4	5
Time PCA (in seconds)	2.4768	5.7565	6.1867	4.6342	5.9641
Time PSO (in seconds)	20.4119	17.8845	19.4263	18.7786	18.5202

Table 4.1 Times Taken by PCA, PSO in Different Simulations

Simulation number	6	7	8	9	10
Time PCA (in seconds)	6.7289	3.0964	1.7905	1.7625	1.7340

Time PSO (in seconds)	17.3193	5.6511	5.7015	5.7603	5.7086

Table 4.2 Times Taken by PCA, PSO in Different Simulations

Classification Phase

We have used 3 machine learning classifiers for face recognition namely MDC, SVM, MCM. SVM and MCM require training while MCM does not require any training. Training is a onetime process. Once a model is trained, testing can be performed any number of times. 10 simulations have been performed. Following table shows the time taken by SVM and MCM during training:-

Simulation number	1	2	3	4	5
Time SVM (in seconds)	.1325	.1524	.0601	.1308	.2354

Table 4.3 Times Taken by SVM, MCM Training in Different Simulations

Simulation number	6	7	8	9	10
Time SVM (in seconds)	.2214	1.4730	.0306	.0292	.2729

Table 4.4: Times Taken by SVM, MCM Training in Different Simulations

After training is performed, testing is performed by all the classifiers. Following line graph compares th time taken by these classifiers in different simulations:-

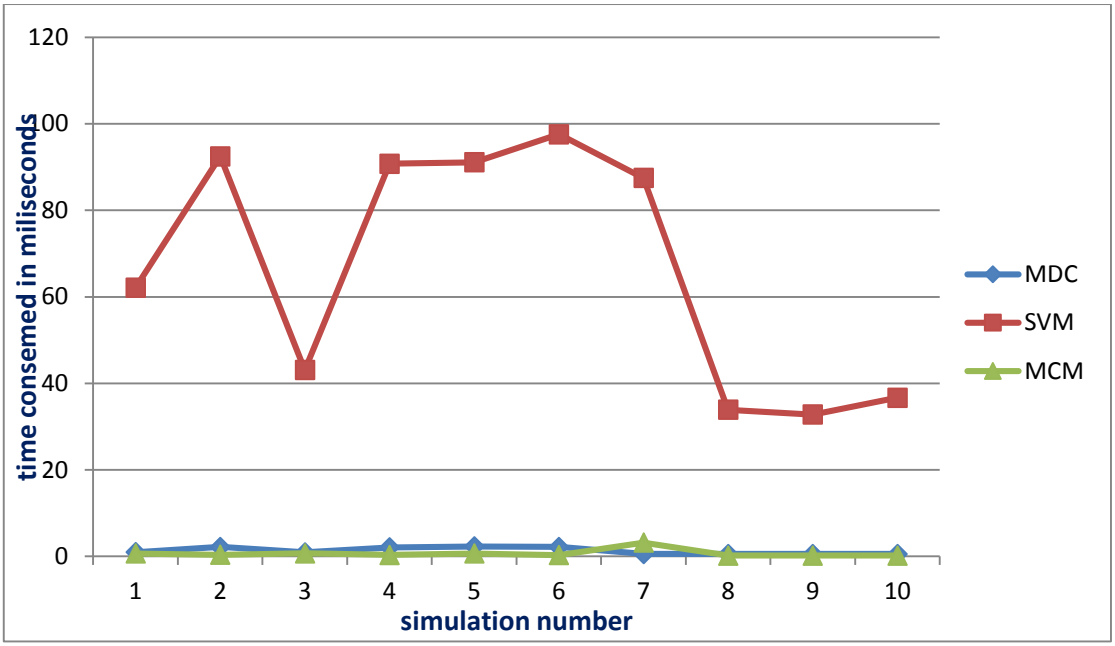

Figure 4.2 Simulation Results of Classifiers

The above line chart (Figure 4.2) shows that time consumed by SVM is far greater than that of MDC and MCM. MCM performs fastest among all. The comparison between MDC and MCM is not clearly expressed by above chart. Following Chart compares MCM and MDC With respect to time consumed.

CHAPTER-5

CONCLUSION AND FUTURE WORK

In this thesis we have we have proposed three algorithms for solving face recognition problem. These are based of Minimum Distance Classifier, Support Vector Machines (SVM), and (MCM). The machine-learning based system presented here can be applied to recognition of any facial expression dimension given a training dataset. MCM based algorithm outperforms the rests in terms of both accuracy and speed..

For face recognition application, it is required to extract features from image and then to develop a training database and then recognize an image. Principal Component Analysis (PCA) is one of the most common technique for feature extraction and is being used widely .We used PCA and further modified it with particle swarm optimization to produce better representative features.

Future Work

We applied PSO in feature extraction phase. We can see the performance variations applying some other nature based algorithms like ant colony optimization or bee colony optimization. We can use swarm intelligence algorithms also in classification phase. For e.g. PSO can be helpful in preventing the exhaustive search in case of Minimum Distance Classifier.