Use of Subspaces in Face Recognition

Sahil Garg
Bachelor of Technology, Computer Science- 2019
Indian Institute of Technology, Mandi

***Abstract*— Mathematics never fail to influence lives of engineers and researchers. From its utility in life of a common person, ranging from household expenses' calculations to the high maths included in stock market; it always find a way to come into the work of researchers. The following article highlights Maths (particularly, Linear Algebra) and its usage in facial recognition. The given report focuses on the subspace perspective of a person's face and the idea of all other images of that person lying in the vicinity of same subspace. This idea of subspace along with other techniques of Machine Learning have been used in order to classify images of Yale B Face dataset, as discussed in the following text.**

I. Introduction

In this paper, we are going to address the use of subspaces in face recognition. The given paper will mostly deal with the problem of classification of images of different persons - a supervised learning technique. In the end, we will give a basic initiation to the approach which may be used in clustering the dataset - an unsupervised learning technique.

Whenever, an image is addressed, we are required to tackle with a high-dimensional data. This high-dimensionality of data makes it difficult for the programmer to work on it and gather the insights. Generally, this data lies in a low-dimensional subspace instead of lying in the whole high-dimensional vector space [1]. This approach will be used in the following paper to get an easily workable data. Later, the data will be worked upon, using some Linear Algebra techniques, to classify data.

II. Dimension Reduction Techniques

Linear Algebra has wide variety of concepts embedded in it. Many of these concepts are either directly or indirectly linked to Machine Learning techniques.
As discussed in section I, we are required to reduce dimension of data most of the times. There are particularly two methods that can be applied to get low memory data, which holds major and most variant features of it.

A. Principal Component Analysis

Usually called as PCA, the given technique focuses on calculating covariance matrix of data. Suppose data matrix,

$$\boldsymbol{D} = \{x_n\}$$

where

$$x_n \in \Re^d$$

and *n* ranges from 1 to N & N is the total number of samples in it, then covariance matrix of it is given by

$$\mathbf{C} = \mathbf{X}^T\mathbf{X}$$

where $\mathbf{X}$ is the data matrix $\mathbf{D}$ centered around $\mathbf{0}$ mean.
Computing the eigenvalues of $\mathbf{C}$ will give

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq ... \geq \lambda_d$$

with their corresponding eigenvectors

$$\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, ..., \mathbf{q}_d$$

The dimension of x_n can be reduced from d to l with i^{th} principal component of $\mathrm{x}_n{}'$ given by

$$a_{ni} = \mathbf{q}_i{}^T\mathbf{x}_n$$

This brings us to a new vector $\mathrm{x}_n{}' \in \Re^l$ with altogether a new orthogonal basis of $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$, ..., $\mathbf{q}_l$
The magnitude of eigenvalues represent the amount of strength or variance data holds in the direction of corresponding eigenvector.

B. Singular Value Decomposition

Popularly called as SVD [2], this technique decomposes the data matrix **D** (centered around 0 mean) to a matix product of 3 matrices

$$\mathbf{D} = \mathbf{U}\Sigma\mathbf{V}^T$$

where columns of **V** are eigenvectors of $\mathbf{D}^T\mathbf{D}$ and columns of **U** are eigenvectors of $\mathbf{D}\mathbf{D}^T$.
Σ is a diagonal (can be rectangular as well) matrix containing singular values σ_i of **D** which are indeed corresponding square root of eigenvalues of $\mathbf{D}\mathbf{D}^T$ or $\mathbf{D}^T\mathbf{D}$. Note that both these products have same eigenvalues. We can get a reduced dimensioned data matrix in this by simply multiplying **U** and Σ. The new matrix $\mathbf{D}'$ is given by

$$\mathbf{D}' = \mathbf{U}\Sigma$$

where we will use only upto *l* values of Σ ,i.e.,

$$\sigma_1, \sigma_2, \sigma_3, ..., \sigma_l$$

The corresponding orthogonal basis of $\mathbf{D}'$ will be

$$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, ..., \mathbf{v}_l$$

where these are first *l* columns of **V**.

III. Projection Matrix

The main concept that we will be using in our analysis will be of Projection Matrix [4] **P**. Suppose we want to find a solution for

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

then we can easily find *x* if **b** lies in columnspace of *A*. But what happens when number of rows in **A** are greater than number of columns. Then we can't find a solution for **x** as **b** would lie in a subspace of dimension greater than that of **C(A)** (**C** represents column space).
Here, we can only find a projection **p** of **b** onto **C(A)** such that **p** is nearest to **b**. Since, **p** would be nearest to **b** only when[3]

$$\mathbf{b}\text{-}\mathbf{p} \perp \mathbf{C(A)}$$

therefore, if **Aw = p**, then

$$\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{w}) = O$$

Solving this, we will get

$$\mathbf{p} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$$

This matrix **P** can project any vector **b** in vector space to the subspace **C(A)** with an error of

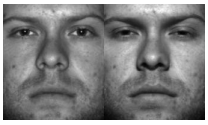
$$\mathbf{e} = \mathbf{b} - \mathbf{P}\mathbf{b}$$

Lesser is the error, more closer is **b**. The matrix **P** holds two properties:

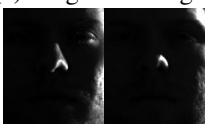
- $\mathbf{P}^T = \mathbf{P}$
- $\mathbf{P}^2 = \mathbf{P}$

IV. Classification Using Projection Matrix

We will be using the standard *Yale Face Database B* which has images of 39 different persons with each one of them having 64 different illuminated samples. However, for our ease, we will be working with only first 10 classes, i.e., only initial 10 persons' images. These are grayscale images which means that each image is a 2-D matrix of pixels. The dimension of each 2-D matrix is 192 x 168. We will reshape each matrix to a 1-D array lying in $\Re^{32256}$
The set of 64 images for each person has a few images which are even very hard to classify from naked eye. For example,



(a) Brightest Images



(b) Darkest Images

Fig. 1: Images from first class label

But still, we will be using them in our analysis. Of course, it can affect our accuracy but our model should be prepared for worst cases as well.

A. Distributing Data For Training and Testing Phases

Since, the data is at different illuminations, it might happen that we take only dark side images for the training phase and would try from the trained classifier to predict for testing phase. But, it would be fatal to do in such a way. It will train

our classifiers badly, leading to poor classification. Instead, we will be making three bins for each class, where first bin will have brightest images of all, second has somewhat less brighter and the third one with darkest images of all. We have distributed 21, 21, 22 images to respective 3 bins (you may do some other way as well). Then we will take 80% of data from each bin for each class as training one, and rest for testing phase.
This will definitely make our classifier more anti-fragile to illumination variances present in data.

B. Reducing Dimension of Data

Let **A** be data matrix, containing train data. We will try to reduce dimension of it, using any one of the techniques discussed in section II.
Since, each image lies in $\Re^{32256}$, it will be catastrophic to calculate covariance matrix of dimension 32256 x 32256, which may bring us to memory error. Therefore, it is more reasonable to use Singular Value Decomposition to reduce dimension of data from 32256 to some *l*. For this SVD to be applicable on data matrix, we are firstly required to subtract its mean from it, to make it centered around zero mean.
Thereafter, we will be using the same mean vector and orthogonal basis of training data matrix to reduce dimension of test data.
This *l* can be chosen to an optimum value after analysing the magnitude of singular values obtained. More the singular value is, more is the information contained in it. However, we will be giving results over a range of values of *l*.

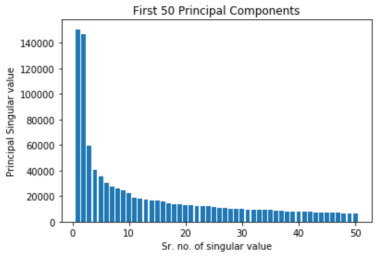


Fig. 2: First 50 Singular Values of Train Data Matrix

After all of this, we get our data to a reduced dimension of *l* from a very high dimension of 32256.

C. Modelling the Classifier

We are going to train a one-versus-all classifier. Basically, we will be taking the set of all data samples belonging to a particular class in **A** (Notice, A is the reduced dimensioned train data matrix) and make a matrix $\mathbf{M}_i$ which has all the samples of i^{th} class as its columns. Thus, $\mathbf{M}_i$ will be a *l* x 52 dimensional matrix (52 is the number of data samples taken as train data from each class out of those 64 images). We will now find a projection matrix $\mathbf{P}_i$ such that it will project any vector **b** onto $\mathbf{C}(\mathbf{M}_i)$. We are assuming here that all the columns of $\mathbf{M}_i$ are independent due to some noise or errors present in data. We can also make them to be independent as discussed in later sections IV-F.
After all of this, we will have a projection matrix $\mathbf{P}_i$ for each class *i*. These are verily the one-versus-all classifiers for the given classes.

D. Prediction on Test Data

Given a sample **b** from the test data, we will try to project it onto the approximated subspace of a class *i* and will try to find out the class which gives us minimum error between **b** and its projection $\mathbf{p}_i$ and that class is given by

$$Class\ label\ for\ \mathbf{b} = argmin_i\{\mathbf{b} - \mathbf{P}_i\mathbf{b}\}$$

E. Results of Experiment

We have covered a range of values of *l* for which we have tried to make pedictions. One thing to notice here is that we can't reduce *l* further from 52 as it will make more columns in our matrix $\mathbf{M}_i$ and concept of projection matrix would fail. For *l* = 52 to 200, we have tried to find out our model predictions on the test data. We have found that at *l* = 134, we have got a maximum accuracy of 91.67%.

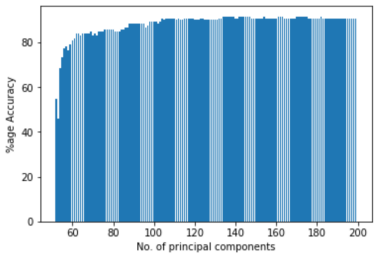


Fig. 3: Accuracy for *l* = 52 to 200

Notice one more thing here, that the accuracy plot tends to saturate at larger values of l, which makes us to recall the fact that after a certain l, singular values tend to zero or becomes negligible, thus containing meagre information about data.

F. Making the Classifier More Space Efficient

In the previous case, we can get our $\mathbf{P}_i$ to be least of dimension 52 x 52. As dimension of projection matrix will be same of dimension of data sample, i.e., if data lies in $\Re^l$, then $\mathbf{P}_i$ will be of l x l. Therefore, we require a lot of space for $\mathbf{P}_i$ to be stored. Now, if we want to reduce its dimension further, then we can apply PCA on $\mathbf{M}_i$ to get only 9 independent columns of it. We have chosen 9 here as dimension of subspace in which images of each person's face lies on some empirical results, established earlier [5]. This will make us free to choose values of l even in range [9-51]. We again tried to find out accuracy for the same test data and found that this time our accuracy decreased (which was expected as we decreased dimension of each class's subspace) and lies close to 80-85%. We have got a maximum accuracy of 85.83% at l = 23.

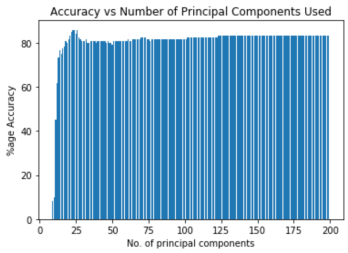


Fig. 4: Accuracy for l = 9 to 200 with PCA applied

V. Principal Angles

For two subspaces $\mathbf{F}$ and $\mathbf{G}$ in $\Re^n$, if dimension of $\mathbf{F}$ and $\mathbf{G}$ are equal to q, then the principal angles between these two subspaces are defined recursively by

$$cos(\theta_k) = \max_{\substack{f \in \mathbf{F}, ||f||_2 = 1 \\ f^T[f_1,\ldots,f_{k-1}]=0}} \max_{\substack{g \in \mathbf{G}, ||g||_2 = 1 \\ g^T[g_1,\ldots,g_{k-1}]=0}} f^T g$$

These principal angles measure the amount of separation between two subspaces. They can easily be found using **QR** factorisation of $\mathbf{F}$ and $\mathbf{G}$ and then using SVD [6].

If $\theta_1 \geq \theta_2 \geq \theta_3 \geq ..... \geq \theta_q$; are the principal angles, then distance between subspaces is given by

$$dist(\mathbf{F},\mathbf{G}) = sin(\theta_q)$$

A. Finding Principal Angles for Different Classes

When the method given in [6] is used to calculate principal angles between subspaces of different class labels, we see that the minimum principal angles θ_i between any pair of classes lies in range [10°,16°]. We have calculated these angles for those reduced 9 columns for each class from that 52 ones at l = 134.

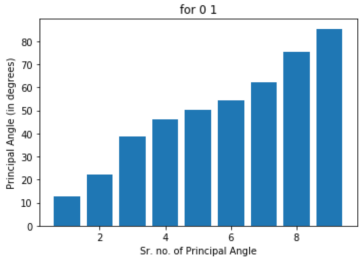


Fig. 5: Principal Angles for Class 0 and 1

These minimum angles range tells us that data of different classes lies very close to each other. Therefore, we are more prone to errors while making a prediction.

If we anyhow manage to transform this data such that each class becomes farthest apart to every other class (basically, a linear transformation and optimisation problem) without loss of data, then we can easily classify data. This will be no less than a revolution, if it happens to be done in near future or so.

If we try to gather different subspaces' images using self-expressiveness [1] of face images, and apply a linear transformation to make different subspaces farthest apart, then we can find astonishing results, even in case of unsupervised learning as well. This field of unsupervised learning can further be explored in coming future with some work already done on it [5] in recent past.

VI. Conclusion and Future Work

We studied the problem of classification of images present at different illuminations using concept of subspaces. Using the projection matrix, we got the best accuracy of 91.67%, which is

still far better than many shallow neural networks applied for the same task. We have tried to come up with an approach that is far more mathematical than neural networks and easy to understand.

For making the model further advance and in a try to get better results, we are investigating the pertinency of Grassmann manifold and some more concepts like that of self expressive layer [5]. Perhaps, in the coming future, we may get the best results in face recognition and even other arenas as well, using this most influential subject - Mathematics, that can take Machine Learning to a whole new level.

REFERENCES

[1] Sparse Subspace Clustering: Algorithm, Theory and Applications
For complete paper, go to given link

[2] Lecture 29: Singular Value Decomposition by Gilbert Strang
For complete lecture, click the link

[3] Hilbert Space Approximation Theorem
For statement and its proof, see this

[4] Lecture 15: Projections onto Subspaces by Gilbert Strang
For complete lecture, click the link

[5] Deep Subspace Clustering Networks by Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, Ian Reid (page 1)
For complete paper, go to given link

[6] Matrix Computations by Gene H. Golub & Charles F. Van Loan (4^{th} edition), page 329-331