DECISION TREE CLASSIFIER

:==:

A weekly lab report submitted by Group No : 3 for the degree of

B.Tech

in

Computer Science and Engineering

विश्वजीवनामृतं ज्ञानम्

Atal Bihari Vajpayee-

Indian Institute of Information Technology and Management, Gwalior

(An Institute of National Importance)

DOS: March 3, 2020

:==:

Members of Group No : 3

2017BCS-009, 2017BCS-025

2017BCS-013, 2017BCS-028

2017BCS-017

Machine Learning Lab

Instructor:

Dr. Sunil Kumar

Contents

1 Introduction to Machine Learning **1**

1.1 Problem statement : 2

1.2 Learning objectives : 2

1.3 Implementation methodology : 2

1.3.1 Description of inputs : 3

1.3.2 Mathematical model of the experiment : 3

1.4 Experimental outputs : 4

1.5 Discussion and Conclusions : 7

1 Introduction to Machine Learning

1. Machine learning is an application of AI which through which the systems learn and make their own decisions.

2. Machine learning focuses on the idea of training the computers with the help of data.

3. It is the study of algorithms and techniques which can make the computers learn easily and accurately.

4.These are of 4 types:

i. Supervised learning: In this, data has labels or classes specified.

ii. Unsupervised learning: In this, target values are not given. We have to find similarity and then make classes.

iii. Reinforcement learning: It is about taking suitable action to maximize reward in a particular situation.

1.1 Problem statement :

Digit Recognition using PCA and KNN.

1.2 Learning objectives :

Objective is to build a classifier for digit recognition using PCA, then inverse transform of PCA, and applying KNN for final classification.

1.3 Implementation methodology :

We took the famous MNIST dataset and reshaped it to 28x28 image.
Then we normalized the images.
We applied PCA on the resulting images with no. of components=8, it means dimensions will be reduced by 20% and it will return the Eigenfaces that have the 80% of the variation in the dataset.
We applied inverse PCA on the training images and it resulted into the previous correlated images with 20% variation as we can not return to the exact original state.
Then we hot encoded the labels i.e., changing them to an array of 0s and 1s.
We splitted our dataset into 20% testing and 80% training data.
After that we applied KNN on PCA transformed data with and checked the accuracy with different neighbouring components.

1.3.1 Description of inputs :

1. We took the famous MNIST dataset which has 784 columns as pixel values and 785th column as label.

2. Labels are numbers from 0 to 9.

3. Pixel values are from 0 to 255.

4. Total samples in the dataset are about 60,000.

1.3.2 Mathematical model of the experiment :

The central idea of Principal Component Analysis (PCA) is to reduce the dimensions of a large correlated data while retaining as much variation as possible.

Mathematics behind PCA:

1. Take a dataset of d+1 dimensions and ignore the labels, considering only d dimensions.

2. Compute the mean of every column.

3. Compute the covriance matrix of the whole dataset.

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n} (Xi - \bar{x})(Yi - \bar{y})$$

4. Compute the eigenvectors and corresponding eigenvalues.

i. Intuitively, an eigenvector is a vector whose direction remains unchanged when a linear transformation is applied to it.

ii. Let A be a square matrix, v a vector and λ a scalar that satisfies Av =

λv, then λ is called eigenvalue associated with eigenvector v of A.

The eigenvalues of A are roots of the characteristic equation:

$$\det(A-\lambda I) = 0$$

5. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a d x k dimensional matrix W.

6. Use this d x k eigenvector matrix to transform the samples onto the new subspace.

7. The eigenvector corresponding to the largest eigenvalue is the 1st principal component.

Mathematics behind KNN:

1. K-nearest neighbour(KNN) classifier is based on the distance of query example with data points of different classes.

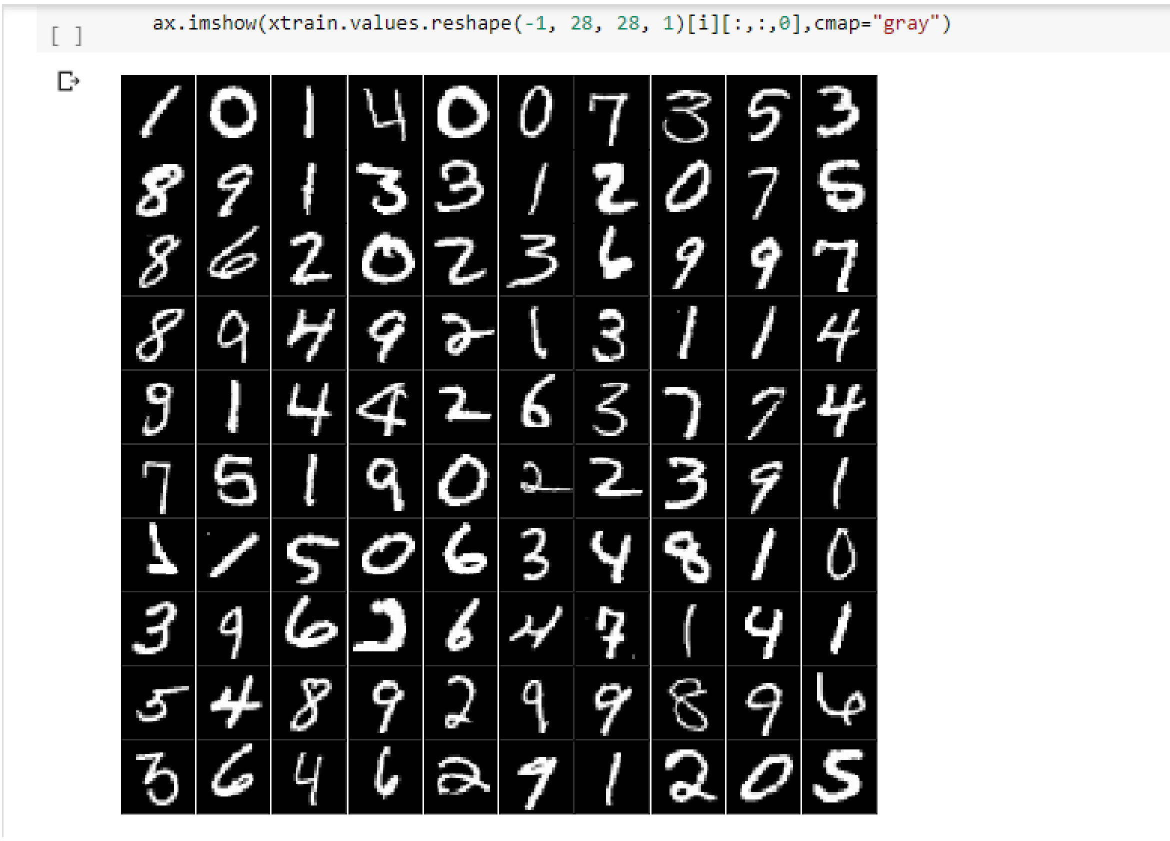
2. K is the number of neighbours to consider.

3. Most popular distance is Euclidean distance:

dist((x, y), (a, b)) = $\sqrt[2]{}[(x-a)]+\sqrt[2]{}[(x-b)]$.

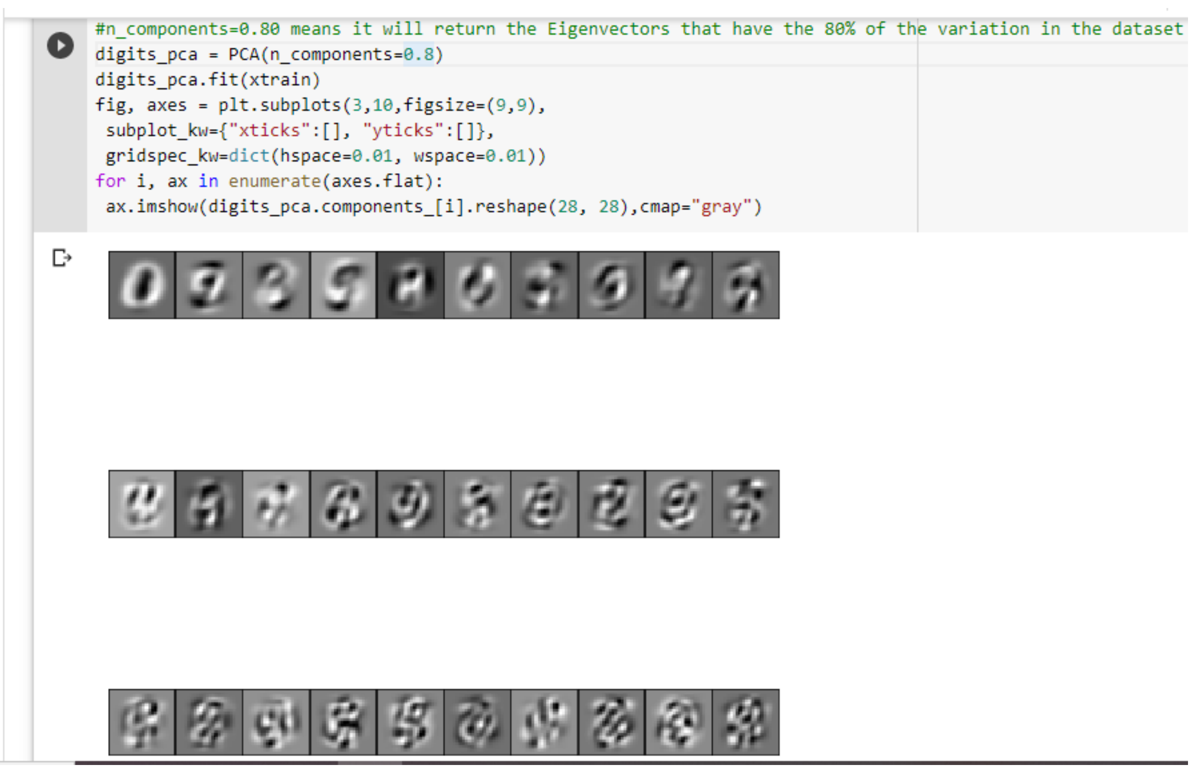
1.4 Experimental outputs :

1. **Original Images**



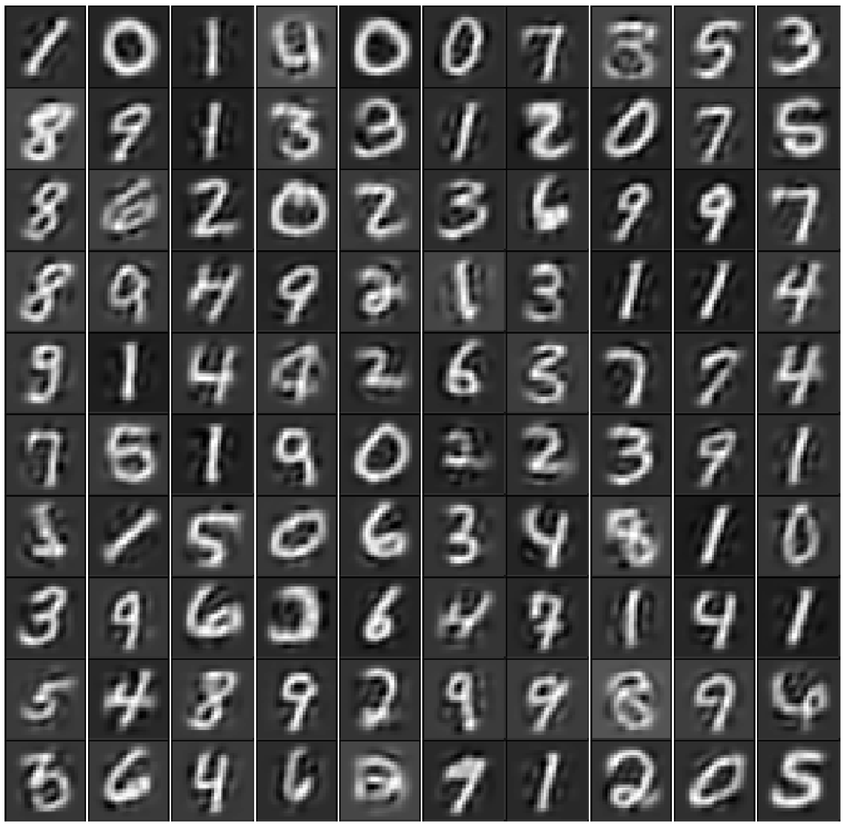
```
[ ]      ax.imshow(xtrain.values.reshape(-1, 28, 28, 1)[i][:,:,0],cmap="gray")
```

2. Eigenfaces



```
#n_components=0.80 means it will return the Eigenvectors that have the 80% of the variation in the dataset
digits_pca = PCA(n_components=0.8)
digits_pca.fit(xtrain)
fig, axes = plt.subplots(3,10,figsize=(9,9),
 subplot_kw={"xticks":[], "yticks":[]},
 gridspec_kw=dict(hspace=0.01, wspace=0.01))
for i, ax in enumerate(axes.flat):
 ax.imshow(digits_pca.components_[i].reshape(28, 28),cmap="gray")
```

3. PCA inverse transformed images



4. Accuracy

```
[ ]  from sklearn.metrics import accuracy_score
     ypred=classifier.predict(xval)
     # ypred=np.argmax(ypred, axis=1)
     # ytrue=np.argmax(yval, axis=1)
     result2 = accuracy_score(yval,ypred)
```

```
[ ]  result2
```

```
0.9489285714285715
```

1.5 Discussion and Conclusions :

1. Accuracy with KNN is 94.89%.

2. PCA reduced the dimensions and increased the accuracy.

3. We applied inverse PCA transform and generated the approximate original space.

4. Applying KNN with 8 neighbours gave an accuracy of 94.89%. We can increase or decrease the neighbours and check if accuracy increases or decreases. And can find the optimal no. of neighbours.

5. We can do the same thing with no. of components of PCA.