CSE 569: Fundamentals of Stat. Learning Project Part1 Report

Ziming Dong[1]

***Abstract*—This is the part 1 of project in Fundamentals of Stat. Learning Course. I tried to implement PCA, Density Estimation, and Bayesian Classification method on subset of MINIST handwritten digit dataset, which only have ”0” and ”1”.**

I. Dataset Introduction

This part of project will use subset of images from MINIST hand written dataset, which contain 60000 training images and 10000 testing images. I only use images for digit ”0” and digit ”1” in this project, and the images have been slightly modified to suit this project. Each images is stored as 28x28 array, we have number of samples in the training set : ”0”: 5983; ”1”: 6742, number of samples in the testing set : ”0”: 980; ”1”: 1135.

II. Task Description

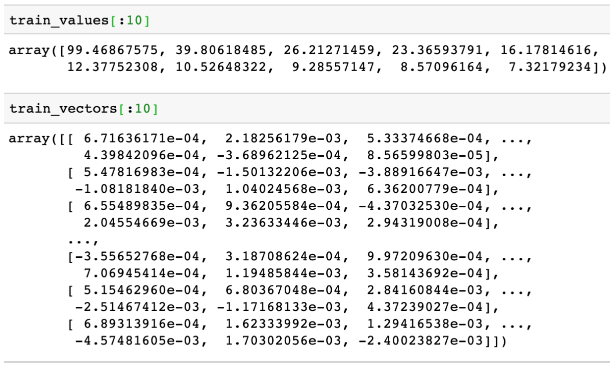
There are five tasks, including feature normalization, dimension reduction using PCA, density estimation and bayesian decision theory for optimal classification. The following subsection shows summary of process and intermediate result for each task.

A. Task 1. Feature normalization (Data conditioning)

For this task, I need to normalize the data before starting any subsequent tasks. Using all the training images (each viewed as a 784-d vector, X = [x1, x2, ..., x784] compute the mean mi and standard deviation (STD) si for each feature xi (remember that we have 784 features) from all the training samples. The mean and STD will be used to normalize all the data samples (training and testing): for each feature xi, in any given sample, the normalized feature will be, yi = (xi - mi)/si.After I loading the mat data, I write the vectorize function which to transfer 3-d sample to 2-D samples, for both training and testing data, the transferred data have the dimension like(number of samples,784). Thus for each samples, we have 784 features now. To do the feature normalization, I calculate the mean and std for each feature, which means the every column in 2-D array. So I will get the calculated mean and std as 2-D array(2,784), the first row will be the 784 means for each training samples, the second row will be the std for each training samples. Now, I can normalize each feature in training and testing dataset by mean and std for each column. The formular will be new-feature=(old-feature-mean)/std. The final result will be 2 2-D arrays, for training examples, I get (12665,784) array, for testing examples, I get (2115,784) array. Both arrays contains normalized values/features.

B. Task 2. PCA using the training samples

The second task asks to implement PCA algorithm manually. I will perform the PCA using the training samples. Acoording to paper [1], I know there are four steps: steps1: feature normalization, I already did in the task1; steps2: computer the covariance matrix, I import the numpy library to use np.cov(array.T) function to get the covariance matrix. The cov matrix shape will be (784,784) since we have 784 features. Step 3: computer eigenvalue and eigenvalue and eigenvector. Eignvalue is the sum of explained variances — and must be equal to 1.I use the np.linalg.eig function to fit the cov matrix to get the values and vectors. The eign values is 784 length array which contains components, the eign vector will be used to calculate project data value later. Lets take a look at first 10 eignvectors and eignvalues to know how they look like



```
train_values[:10]
```

```
array([99.46867575, 39.80618485, 26.21271459, 23.36593791, 16.17814616,
       12.37752308, 10.52648322,  9.28557147,  8.57096164,  7.32179234])
```

```
train_vectors[:10]
```

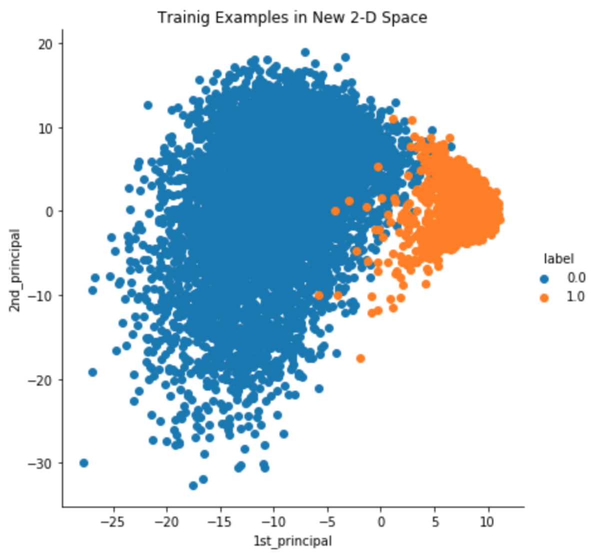
```
array([[ 6.71636171e-04,  2.18256179e-03,  5.33374668e-04, ...,
         4.39842096e-04, -3.68962125e-04,  8.56599803e-05],
       [ 5.47816983e-04, -1.50132206e-03, -3.88916647e-03, ...,
        -1.08181840e-03,  1.04024568e-03,  6.36200779e-04],
       [ 6.55489835e-04,  9.36205584e-04, -4.37032530e-04, ...,
         2.04554669e-03,  3.23633446e-03,  2.94319008e-04],
       ...,
       [-3.55652768e-04,  3.18708624e-04,  9.97209630e-04, ...,
         7.06945414e-04,  1.19485844e-03,  3.58143692e-04],
       [ 5.15462960e-04,  6.80367048e-04,  2.84160844e-03, ...,
        -2.51467412e-03, -1.17168133e-03,  4.37239027e-04],
       [ 6.89313916e-04,  1.62333992e-03,  1.29416538e-03, ...,
        -4.57481605e-03,  1.70302056e-03, -2.40023827e-03]])
```

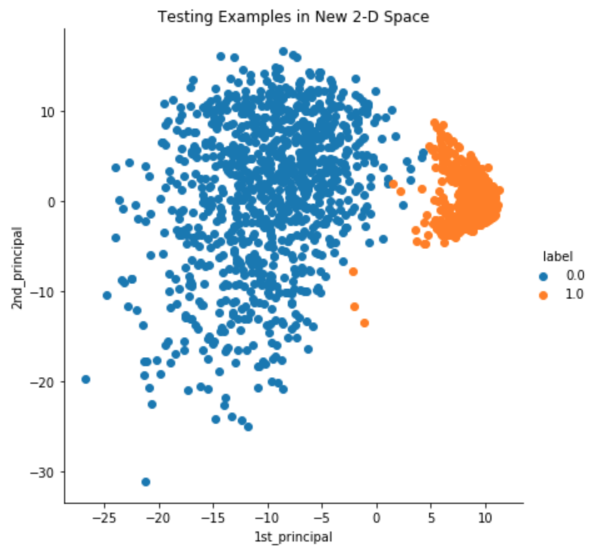
Step 4: eign analysis, since the values are in descending order, we can easily find the first and second principle component ratio from the index 0 and index 1 of values/1. The first principle component is around **0.13** and second principle component is around **0.05**, which means they are weight **eighteen percent** of total components. There is one thing very interested me when I use built in function to check the eignvalues and vectors, the manually result have inverse sign value compare to the built in function result, even my steps look right for PCA. After I did some research on this part, I know the eigenvectors calculated by my our program and the inbuilt PCA functions just differ in scaling i.e. when I multiply -1 to the eigenvectors I get, I will obtain the eigenvectors returned by the inbuilt PCA. The sign of the eigenvectors does not matter as it does not change the projection axis.

[1] Z. Dong with the School of Computer Sciences, Big Data System, Arizona State University `zdong27@asu.edu`

C. Task 3. Dimension reduction using PCA

Consider 2-D projections of the samples on the first and second principal components. I combine the 1st and 2nd principle component values in a 2-D array and use np.matual to fit this 2-D array with training.T to project the train/test data for 2-D projections on the 1st and 2nd principal component.These are the new 2-D representations of the samples. I visualize the training and testing samples in this 2-D space.





Even I use the training samples to get the eignvalues and eignvectors,I can see the distribution from second figure is similar the fist figure. The two classes clustered in this 2-D space, each class look like a normal distribution in some ways. However, as I mentioned before, the manually result have inverse sign value compare to the built in function result. Thus, I get the different direction on projection visualization than the built in function. But it will not influence the following task since the sign of the eigenvectors does not matter as it does not change the projection axis.

D. Task 4. Density estimation

Assume in the 2-D space defined above, samples from each class follow the Gaussian distribution, I need to estimate the parameters for the 2-D normal distribution for each class/digit, using the training data only. The multivariate Gaussian distribution is commonly expressed in terms of the parameters μ and cov, where μ is an $n \times 1$ vector and cov is an $n \times n$, symmetric matrix. (We will assume for now that cov is also positive definite, but later on we will have occasion to relax that constraint). Because samples from each class follow the 2-D normal distribution, the parameters for each class will be the mean and cov matrix. Firstly, I split the 0 and 1 class from the training projected data. Then I write a function to use np.mean and no.cov to estimate the parameters for class by MLE. After fitting these two class project data arrays to the function, I get
parameters for train0 normal distribution:mean: [-9.92384569 0.8514571] cov: [[25.32460567 15.90014464] [15.90014464 79.11311962]],
parameters for train1 normal distribution: mean: [8.71832365 -0.74802439] cov: [[2.06676736 -0.02148539] [-0.02148539 4.0841354]]

E. Task 5. Bayesian Decision Theory for optimal classification)

This task asks to use the estimated distributions for doing minimum-error-rate classification. Minimum-error-rate is the lowest possible error rate for any random result of the classifier. Bayesian error rate has important applications in pattern research and machine learning techniques. According to the minimum-error-rate classification rule:**we decide class 0 if P(class0)*P(x—class0) is larger than P(class1)*P(x—class1)**. Since the P(class0)=P(class1), we only need to compare P(x—class0) and P(x—class1). So I write a function to use multivariate-normal built in function to get two pdfs of normal distribution by fitting the training samples parameters and use var.pdf to get probability of each sample being class0 and class1 by fitting the projected testing samples and training samples. Finally, I compare the result from the ground truth label for testing set and training set, the training set accuracy is around **0.988**, the testing set accuracy is around **0.992**.

III. Conclusion

Based on this project, I use PCA,Density Estimation, and Bayesian Classification method to set up a experiment on MINIST dataset for handwritten digit"0" and "digit1". For more information, please take a look at jupyter notebook, I list down intermediate result for each task. Each step is connected to next step, it gives me a chance to review and summarize course content and the way to implement statistic method by python. I would like to read more paper to go deep on this statistic machine learning approach to explore more knowledge in the future.

References

[1] Sasan Karamizadeh, Shahidan M Abdullah, Azizah A Manaf, Mazdak Zamani, and Alireza Hooman. An overview of principal component analysis. Journal of Signal and Information Processing, 4(3B):173, 2013.