UNIVERSITY OF
MARYLAND

ENEE633/CMSC828C

PROJECT 1 Report

SUBMITTED BY,

SAKET SESHADRI GUDIMETLA HANUMATH
UID: 116332293
EMAIL: saketsgh@umd.edu

CONTENTS

Overview

The objective of this project is to utilize the following machine techniques for classifying the fashion MNIST dataset, study how they work, compare their accuracies and discuss in detail how to program these in languages such as Python -

1. Bayes Classifier with Gaussian assumption
2. K Nearest Neighbor Classifier
3. Bayes Classifier with PCA and LDA.
4. K Nearest Neighbor Classifier with PCA and LDA

Dataset Description

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. It can be through this link https://github.com/zalandoresearch/fashion-mnist

The labels are as follows –

Label	**Description**
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Machine Learning Techniques

Bayes Classifier using Gaussian assumption –

The Bayes classifier compares the value of posterior probabilities $p(\omega|x)$ for each data point x and selects the class label which gives the maximum posterior probability. In the given case since priors' values are the same thus, the comparison is done between the likelihood values $p(x|\omega)$ and the max value is selected.

The steps taken to write the program are as follows –

1. Calculate mean(μ) and covariance(Σ) of each class 0-9
    a. The maximum likelihood estimates for mean and covariance using the gaussian assumption are nothing, but the sample mean and sample covariance.
    b. I utilized the functions np.mean and np.cov present in the numpy module to compute the ML estimates for each class.
    c. Stored all the values in a list corresponding to their class index.
2. Compute the discriminant functions for all the test samples using the formula below –
    a. $g_i(x) = \frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i)$
    The remaining terms contain det(Σ), a constant term, and natural log of prior probability which do not contribute anything for comparison thus have not been used to reduce unnecessary calculations.
    b. For eliminating the singular matrix issue, I used ***pseudo inverse*** or ***pinv*** function of numpy to calculate inverse bypassing the singular matrix issue.
    c. All the g values (for each sample and class) were computed and stored in a multidimensional array of size (10k, 10).
3. Assign test data to class ω_i if $g_i(x)$ gives the maximum value out of the data stored for 10 classes.

K Nearest Neighbor Classifier –

The idea behind KNN is very simple. It calculates the Euclidean distance $d(x, x_i)$ between the test data and all other data elements in the training set. The task of assigning test data to class ω_i is accomplished using the highest vote from the k selected nearest neighbors on the lowest value of d.

Scikitlearn's ***KNN.fit*** function has been used to do the classification.

Experimentation and Results

1. Bayes Classifier –

Classifier type	Accuracy (in %)	Number of Dimensions to work with	Time Taken (in sec)
Bayes Classifier	72.39	784	3.62
Bayes Classifier with PCA	**80.97**	**49**	**0.093**
Bayes Classifier with LDA	80.46	9	0.05

*Experimentation with **variance** parameter of PCA*

Variance (%)	Accuracy (in %) of Bayes Classifier after PCA	Number of Dimensions (original = 784)
95	79.08	187
90	80.69	84
86	**80.977**	**49**
80	79.49	24

Implementing Bayes classifier on the fashion MNIST dataset gave an accuracy of around 72.39% without reducing the original dimension size which was 784. The process of classification took around 3.6 seconds to complete.

PCA reduced the dimension size considerably when I used higher percentage of covered variance for e.g. 95. Further experimenting with different values of variances gives changes in accuracy (although minute). From the observations it was found that when the dimensions were reduced to 49 from 784 (with a % of covered variance of 86) the accuracy was at peak with a value of 80.977.

LDA with Bayes yields a very similar result to PCA with even lower dimension size of about 9 and with faster computation as expected.

Overall it was observed that using Bayes with PCA yielded the best result with decent accuracy of around 80.97%, less computation time of 0.093 seconds and total dimensions 49(covered variance of 86%). Although using 95% covered variance would yield almost the same result. Intuitively speaking it is better to have more variables that cover maximum variance of the dataset. PCA helps us de-convolute the data, set aside variables with low variance, and take care of collinearity. It is an optimal method in this case as it gives a decent tradeoff between number accuracy, speed and optimality.

2. *K Nearest Neighbors -*

Parameter (K)	Accuracy (KNN)	Accuracy (KNN + PCA)	Accuracy (KNN + LDA)
1	0.8497	0.8521	0.7911
3	0.8541	0.8592	0.814
5	0.8554	0.8623	0.8221
7	0.854	0.8619	0.8279
9	0.8519	0.8602	0.8314
11	0.8495	0.8602	0.8325
13	0.8468	0.8558	0.8316

15	0.8462	0.8561	0.8322
17	0.8441	0.8537	0.831
19	0.8427	0.8523	0.8309

Summarizing the above results

Classifier	Optimal K	Accuracy (in %)
KNN	5	85.54
KNN + PCA	**5**	**86.23**
KNN + LDA	11	83.25

Classifier	Time taken to compute KNN algorithm (for k = 1-19)
KNN	Around 200 mins
KNN + PCA	408.95 seconds
KNN + LDA	14.72 seconds

Initially KNN was implemented alone without reducing dimensionality (thus using original dimension size of 784) on odd values of k ranging from 1 to 19. It was observed that when the k value increased accuracy also started to increase as well and attained a maximum value of 85.54% at k = 5. From this point onwards the accuracy took a very slight dip in value. Thus, the optimal k observed was at 5. KNN when implemented alone takes lot of time, around 15-20 minutes for a single value of k depending upon how powerful the workstation is.

When PCA was used for dimensionality reduction the results were quite similar with this implementation scoring an accuracy of 86.23%. A covered variance of 0.95 or 95% was used for reducing the dimensions. The observations of KNN with PCA behave almost identical to standalone KNN with the exception that the computation is much faster (around 96% faster). For this implementation the optimal k was found to behave in a similar way as before with a slight increase of accuracy at first till k = 5 and slight dip from there onwards (although not strictly). In other words it actually fluctuates around the same value.

KNN with LDA proved to the best of the bunch in terms of computational efficiency by taking only 14.72 seconds to compute. There was a decrease in accuracy compared to the previous two classifiers with it getting around 83.25% accuracy. The optimal k in this case was 11 with accuracy of 83.25%. The behavior of k was similar with initial increase, reaching a maximum value and taking a slight dip in value from there on. This tells that PCA does a better job than LDA as it attempts to find a lower dimensional feature space that maximizes the separability of the datapoints.

Thus, for the given dataset of fashion MNIST, KNN + PCA gives the highest accuracy compared to the best classifier in the previous case viz. Bayes + PCA.

Conclusion

The two Machine Learning techniques- Bayes Classifier with Gaussian assumption and K-Nearest Neighbors were implemented on the fashion MNIST data sets. Along with these techniques, dimension reduction techniques, PCA and LDA, were also implemented for the above two classifiers. Based on the experiments and observations, the best classifier for the given problem was chosen.