STAT 542: Final Project

Karthik Vasu (kvasu2), Yining Lu (yining13)

Due: 05/05/2022

Contents

Literature review **1**

Summary Statistics **2**

Pre processing **2**

PCA 2

Clustering **3**

Kmeans 3

Classification **4**

k-Nearest Neighbors 4

LDA 5

SVM 5

Conclusion 6

References **7**

Literature review

The Fashion-MNIST data set has been created by researchers at Zalando for the purposes of benchmarking ML algorithms. It consists of 70000 grayscale images of dimension 28*28. These are images of clothing articles like T-shirt, Trouser, Pullover etc with 60000 training samples and 10000 testing samples. The purpose of this data set is to provide a more challenging classifying compared to the original MNIST data. There are algorithms which 99% accuracy on this making it too easy for modern algorithms.

The best accuracy we found was by a GitHub user named *Andy Brock* who was able to achieve an accuracy of 96.7% using wide residual networks. A lot of people have implemented algorithms with high accuracy. They can be for on *Zalando Research's GitHub page*.

Xiao, Rasul, and Vollgraf (2017) test out a variety of classifiers including Decision Tree ,Gradient Boosting, K Neighbors, Linear SVC, Logistic Regression and many more. They achieve the best result using the SVC classifier with C=10 and the rbf kernel. The testing accuracy for this algorithm is 89.7% on the fashion data set and 97.3% on the original MNIST data. Gradient boosting performs well with testing accuracy at 88% and 96.9% respectively. This is achieved for n_estimators=100 and max_depth=10.

Meshkini, Platos, and Ghassemain (2019) perform classification on the Fashion-MNIST data set using convolutional neural networks. They compare the performance of several well-known deep learning frameworks, such as AlexNet, GoogleNet, VGG and ResNet, DenseNet and SqueezeNet. The authors also propose an additional step of batch normalization to enhance the training speed and accuracy of the model. The best results are achieved by ResNet44 and SqueezeNet with batch normalization with testing accuracy at 93.39% and 93.43% respectively.

Summary Statistics

Data table

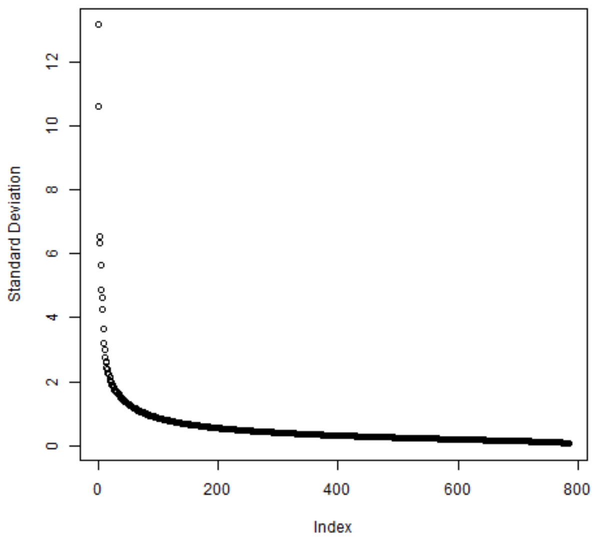
```
## Ytrain
##    0    1    2    3    4    5    6    7    8    9
## 6000 6000 6000 6000 6000 6000 6000 6000 6000 6000

## Ytest
##    0    1    2    3    4    5    6    7    8    9
## 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000
```

Pre processing

Before using PCA we have scaled and centered the data.

PCA

We perform PCA on the training data set and plot the standard deviation of each of the components in descending order. Using the elbow method the cutoff we choose is 1.75 and take all the components which have standard deviation above that value. It gives 28 components.



Clustering

Kmeans

To first determine which K to use we run the kmeans algorithm for k =10,20,. . . ,100 and measure for each cluster the ration # of votes the majority label got/ total number of elements in the clustes. We also sum over all the clusters by weighing the ratios according to the cluster size. The plot of this cluster confidence vs K is as follows.

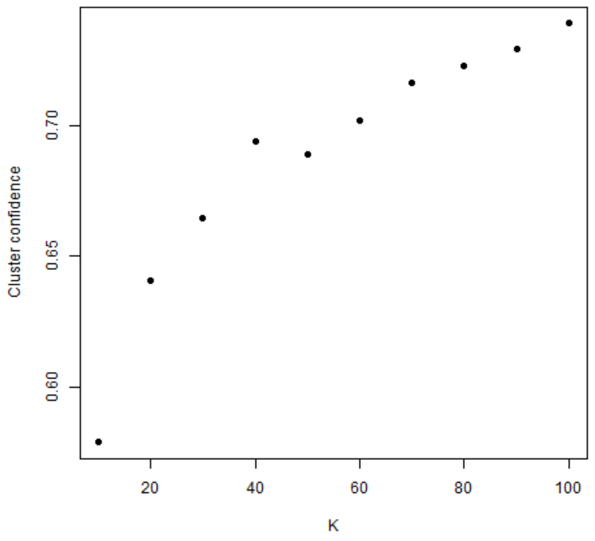


Figure 1: Cluster confidence vs K

From this plot we can see that as we increase K the confidence increases. We choose K=100 so that to get the best clustering while keeping computation time low.

The cluster confidence is 73.58% and the majority label in each cluster are as follows

```
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## x    1    4    1    9    0    8    8    8    6     0     8     6     8     3
##   [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27]
## x     7     8     8     6     2     9     8     5     3     3     6     9     0
##   [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40]
## x     4     8     2     9     3     8     9     8     5     6     6     9     3
##   [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52] [,53]
## x     0     2     0     7     8     9     4     3     3     8     2     4     5
##   [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66]
## x     8     8     8     2     8     0     0     4     1     8     1     4     5
##   [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75] [,76] [,77] [,78] [,79]
## x     7     8     0     9     4     2     4     0     0     2     9     2     4
##   [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92]
## x     6     8     4     7     5     3     6     9     5     1     1     2     8
##   [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
## x     4     7     9     8     6     5     4      9
```

Classification

In this section, we implement multi-class classification models on the Fashion-MNIST dataset. We first use two algorithms, k-Nearest Neighbor (k-NN) and Linear Discriminant Analysis (LDA), which can be directly used for multiclass classification. Then, we extend Support Vector Machine (SVM), which is a binary classifier, to handle the multiclass case.

k-Nearest Neighbors

We first implement k-NN classification model to classify the Fashion-MNIST data. The Euclidean distance is used to measure the distance between variables. The raw data are scaled in the data pre-processing step, so the distance is scale-invariant.

To determine the parameter k, we use 5-fold cross-validation and calculate the overall mis-classification rate for each k. The figure shows the relation between mis-classification rate and k. We find that the best mis-classification rate is reached at $k = 10$.

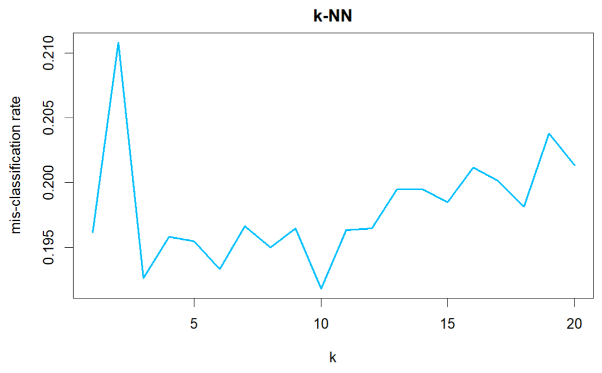
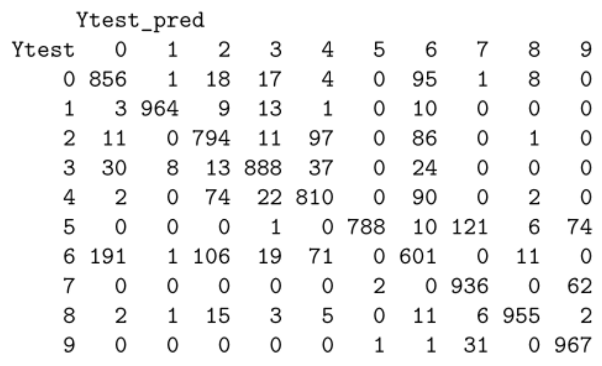


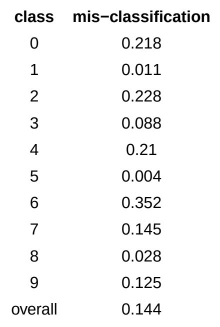
Figure 2: 5-fold CV Mis-classification Rate

Based on the result of cross-valiadation, we choose $k = 10$ to fit the k-NN model. The confusion matrix and mis-classification rate for each class are shown in the following two tables. For Fashion-MNIST data, the overall mis-classification rate of the 10-NN classifier is 0.144 and the accuracy is 0.856.



```
       Ytest_pred
Ytest    0   1   2   3   4   5   6   7   8   9
    0  856   1  18  17   4   0  95   1   8   0
    1    3 964   9  13   1   0  10   0   0   0
    2   11   0 794  11  97   0  86   0   1   0
    3   30   8  13 888  37   0  24   0   0   0
    4    2   0  74  22 810   0  90   0   2   0
    5    0   0   0   1   0 788  10 121   6  74
    6  191   1 106  19  71   0 601   0  11   0
    7    0   0   0   0   0   2   0 936   0  62
    8    2   1  15   3   5   0  11   6 955   2
    9    0   0   0   0   0   1   1  31   0 967
```

(a) Confusion Matrix



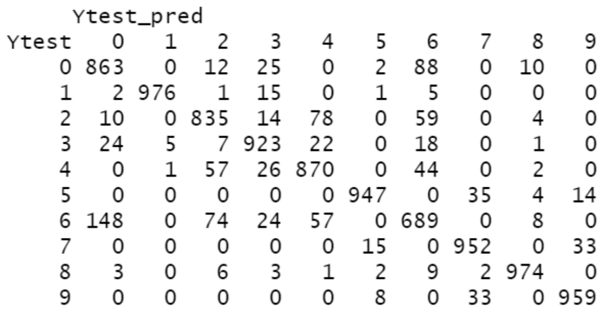
class	mis−classification
0	0.218
1	0.011
2	0.228
3	0.088
4	0.21
5	0.004
6	0.352
7	0.145
8	0.028
9	0.125
overall	0.144

(b) Mis-classification Rate

Figure 3: 10-NN

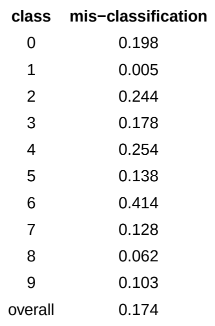
LDA

We then classify the Fashion-MNIST data using the LDA classifier. The confusion matrix and mis-classification rate for each class are shown in the following two tables. The overall mis-classification rate of the LDA classifier is 0.174 and the accuracy is 0.826. We find that the accuracy of LDA classifier is slightly lower than that of 10-kNN classifer.



```
      Ytest_pred
Ytest   0   1   2   3   4   5   6   7   8   9
    0 863   0  12  25   0   2  88   0  10   0
    1   2 976   1  15   0   1   5   0   0   0
    2  10   0 835  14  78   0  59   0   4   0
    3  24   5   7 923  22   0  18   0   1   0
    4   0   1  57  26 870   0  44   0   2   0
    5   0   0   0   0   0 947   0  35   4  14
    6 148   0  74  24  57   0 689   0   8   0
    7   0   0   0   0   0  15   0 952   0  33
    8   3   0   6   3   1   2   9   2 974   0
    9   0   0   0   0   0   8   0  33   0 959
```

(a) Confusion Matrix



class	mis−classification
0	0.198
1	0.005
2	0.244
3	0.178
4	0.254
5	0.138
6	0.414
7	0.128
8	0.062
9	0.103
overall	0.174

(b) Mis-classification Rate

Figure 4: LDA

SVM

We have applied two multiclass classifiers k-NN and LDA for the Fashion-MNIST classification. Now, we extend SVM to the multiclass form to solve the classification problem. SVM is a binary classifier and it does not support multiclass classification natively. However, we can break the multiclass classification problem into several binary ones. There are two common methods to extend SVM for multiclass classification, One-vs-One approach and One-vs-Rest approach.

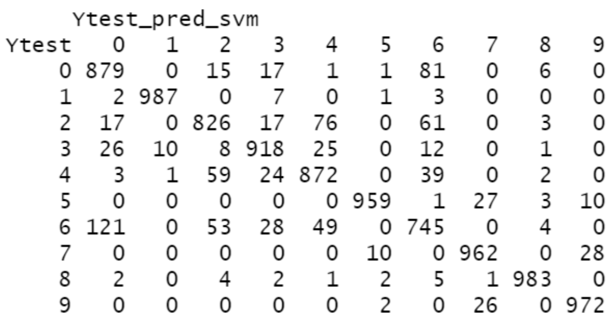
In One-vs-One approach, we fit SVM models for every two classes. Each classifier separates points of two different classes. Suppose we have k classes, we then fit $\frac{k(k-1)}{2}$ SVM models. In prediction stage, we input data into all binary classifiers. Each binary classifiers will decide a class that the input is belonged to. We let those binary classifiers vote for the class of input, and the prediction result is the class that most classifiers vote for.

In One-vs-Rest approach, we fit SVM models to distinguish points of one certain class from the other classes. Suppose we have k classes, we then fit k SVM models. In prediction stage, we input data into all One-vs-Rest classifiers. Each classifier will give a probability that the input is belonged to that class. The prediction result is the class with the largest probability.

One-vs-Rest approach is more computationally efficient than One-vs-One approach, since it only need to fit k SVM models while One-vs-One approach need to fit $\frac{k(k-1)}{2}$ models. However, in One-vs-Rest approach, the training data is unbalanced since the ratio of training data from each class is $1 : (k-1)$, which may cause biase.

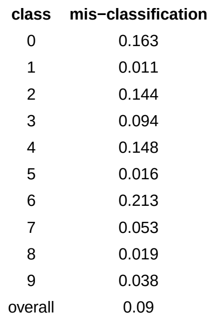
We fit the multiclass SVM model using One-vs-One approach, with using radial basis kernel. The overall mis-classification rate of kernel SVM is 0.09 and the accuracy is 0.91. The kernel SVM classifier outperforms k-NN and LDA.

In addition, a linear SVM model is constructed to compare with the RBF kernel. The overall mis-classification rate of linear SVM is 0.188 and the accuracy is 0.812.



```
     Ytest_pred_svm
Ytest   0   1   2   3   4   5   6   7   8   9
    0 879   0  15  17   1   1  81   0   6   0
    1   2 987   0   7   0   1   3   0   0   0
    2  17   0 826  17  76   0  61   0   3   0
    3  26  10   8 918  25   0  12   0   1   0
    4   3   1  59  24 872   0  39   0   2   0
    5   0   0   0   0   0 959   1  27   3  10
    6 121   0  53  28  49   0 745   0   4   0
    7   0   0   0   0   0  10   0 962   0  28
    8   2   0   4   2   1   2   5   1 983   0
    9   0   0   0   0   0   2   0  26   0 972
```

(a) Confusion Matrix

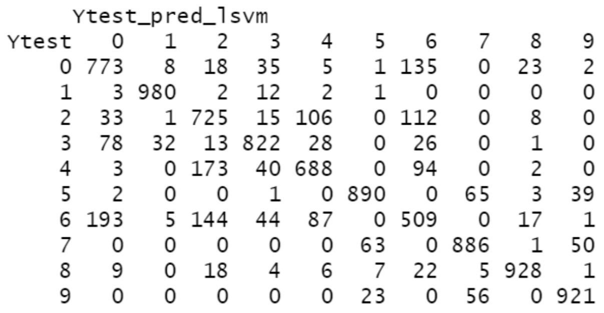


class	mis−classification
0	0.163
1	0.011
2	0.144
3	0.094
4	0.148
5	0.016
6	0.213
7	0.053
8	0.019
9	0.038
overall	0.09

(b) Mis-classification Rate

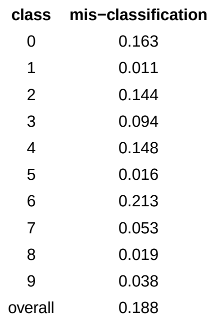
Figure 5: Kernel SVM (rbf)

Though linear SVM model does not perform better than k-NN and LDA, SVM with RBF kernel have a quite good performance on classification, which achieves an overall accuracy of 91%.



```
     Ytest_pred_lsvm
Ytest   0   1   2   3   4   5   6   7   8   9
    0 773   8  18  35   5   1 135   0  23   2
    1   3 980   2  12   2   1   0   0   0   0
    2  33   1 725  15 106   0 112   0   8   0
    3  78  32  13 822  28   0  26   0   1   0
    4   3   0 173  40 688   0  94   0   2   0
    5   2   0   0   1   0 890   0  65   3  39
    6 193   5 144  44  87   0 509   0  17   1
    7   0   0   0   0   0  63   0 886   1  50
    8   9   0  18   4   6   7  22   5 928   1
    9   0   0   0   0   0  23   0  56   0 921
```

(a) Confusion Matrix



class	mis−classification
0	0.163
1	0.011
2	0.144
3	0.094
4	0.148
5	0.016
6	0.213
7	0.053
8	0.019
9	0.038
overall	0.188

(b) Mis-classification Rate

Figure 6: Linear SVM

Conclusion

In this part, we build k-NN, LDA, linear SVM and kernel SVM to classify Fashion-MNIST data. The performace of different algorithms are shown in the table. The accuracy of four methods: kernel SVM > k-NN > LDA > linear SVM.

Table 1: Model Summary

	kNN	LDA	linear.SVM	kernel.SVM
Accuracy	0.8559	0.8256	0.8122	0.9103

References

Meshkini, Khatereh, Jan Platos, and Hassan Ghassemain. 2019. "An Analysis of Convolutional Neural Network for Fashion Images Classification (Fashion-MNIST)." In *International Conference on Intelligent Information Technologies for Industry*, 85–95. Springer.

Xiao, Han, Kashif Rasul, and Roland Vollgraf. 2017. "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms." August 28, 2017. https://arxiv.org/abs/cs.LG/1708.07747.