# **PCA**

## **Visualization**

### 2-Dimensional PCA images

### 3-Dimensional PCA images

### Top 3 Eigenvectors

## Classification accuracies

The results are listed for accuracy with 3-NN

|  |  |
| --- | --- |
| Dimension | PCA with 3-NN |
| 40 | 97.45% |
| 80 | 97.38% |
| 200 | 97.09% |

### **Preserving 95% energy**

Deriving d

To get the 95% energy in **d**, a quick way would be to add up all the eigenvalues.

1. Sort the eigenvalues, eigenvector pairs in descending order
2. Sum all the eigenvalues
3. Initialize a sum of 0 and slowly add eigenvalues till you hit 95% of the sum
4. The k-eigenvalues added are the dimensions required to obtain the top 95% energy/variance in the data

# **LDA**

A Nearest neighbour with K = 3 window is applied onto the testing data.

## **Maximal dimensionality projection**

LDA can only project up to C-1 dimensions as it is designed to maximize the interclass means and minimize the intraclass means. Thus, as there are only 10 classes, It would only require projections of up to 9 dimensions for it to work effectively, which is shown in the table below

## **Classification accuracy**

|  |  |
| --- | --- |
| Dimension | LDA with 3-NN |
| 2 | 57.76% |
| 3 | 71.85% |
| 9 | 91.27% |

# **SVM**

**Linear**

|  |  |  |  |
| --- | --- | --- | --- |
| Cost (C) | Dimensions | | |
|  | **40** | **80** | **200** |
| 0.01 | 89.39% | 90.91% | 91.57% |
| 0.1 | 89.46% | 91.02% | 91.83% |
| 1 | 89.50% | 91.03% | 91.87% |
| 10 | 89.30% | 90.12% | 91.23% |

## **Analysis**

### **Effect of dimensions**

As the number of dimensions increase, the model seems to perform better. This is because the data is more separable at higher dimensions which allows the linear kernel to draw a line between the classes and separating the data

### **Effect of C/slack parameter**

As the C hyperparameter increases, it is a form of regularization resulting in a soft-margin SVM. This in turn leads to allowing more “errors” when training the model where some instances can be “wrongly classified” thus reducing overfitting. As observed in the data, there was less overfitting but when the allowance/slack became too great at 10, it resulted in underfitting and hence the model performed poorer.

The general trend is just that from 0.01 to 1, the model tended to generalize better but when the Cost was increased to 10, the model underfitted resulting in a poorer model

**RBF Kernels**

​All values are set with gamma = 0.1

|  |  |  |  |
| --- | --- | --- | --- |
| Cost (C) | Dimensions | | |
|  | **40** | **80** | **200** |
| 0.01 | 76.26% | 34.89% | 28.37% |
| 0.1 | 96.15% | 90.27% | 78.27% |
| 1 | 98.21% | 97.83% | 96.73% |
| 10 | 98.25% | 97.84% | 96.81% |

**Best SVM scores**

Using a RBF kernel with following parameters achieved an accuracy of 98.67%:

Gamma : 0.05

C: 5

Dimensions : 80

## **Comparison with Linear SVM**

### **Number of hyperparameters**

In RBF-kernels, we had more hyperparameters tuning which naturally allowed a more generalizable model

### **Kernel Difference**

As RBF kernels tends to project the initial data to higher dimensions, it appears that allowing a higher cost hyperparameter which in turn led to more leeway for the model to generalize seemed to do better in RBF cases, which could possibly be due to the separation of data in higher dimensional space

### **Best Model**

The best model expectedly turned out to be the RBF SVM kernel which achieved an accuracy of 98.67% which was the best.

# **CNN**

The performance