Ömer Alperen Katı
Bogazici University, Electric Electronic Eng., 09.12.2022

EE 573 Pattern Recognition - Project 4 Report

Introduction

In this project, the old dataset is used. PCA is applied to the dataset since the number of features (400) is large compared to the total number of learning samples (500). Since it is sufficient for explaining 99% of the variance in the dataset, dimensionality is reduced to 61. Also, MDA is applied to the dataset to enhance the separation between classes.

PART 1: Parzen window estimation with PNN

Since PNN algorithm also employs Parzen window estimation, it is used for classifying the test samples. The following sigma values are used for Gaussian kernel widths:

```
>> sigma_pnn
sigma_pnn =
    [0.0001    0.0010    0.0100    0.1000    1.0000]'
```

Fig 1: Simulated sigma values for Parzen window estimation with PNN.

Recall, precision, and accuracy rates for each class (columns) and for each sigma (rows) are shown as follows:

```
>> recall.pnn
ans =
    0.9600    0.7600    0.8800    0.6800    0.8800
    1.0000    0.8800    0.8800    0.8400    0.8800
    1.0000    0.8400    0.8400    0.8800    0.8800
    1.0000    0.8400    0.8400    0.8800    0.8800
    1.0000    0.8400    0.8400    0.8800    0.8800
>> precision.pnn
ans =
    0.9231    0.6333    0.9565    0.8095    0.8800
    0.9259    0.7586    0.9565    0.9545    0.9167
    0.8621    0.7778    0.9545    0.9565    0.9167
    0.8621    0.7778    0.9545    0.9565    0.9167
    0.8621    0.7778    0.9545    0.9565    0.9167
>> accuracy.pnn
ans =
    [0.8320    0.8960    0.8880    0.8880    0.8880]'
```

Fig 2: Corresponding recall, precision, and accuracy rates for each simulated sigma value for Parzen window estimation with PNN.

In terms of every metric, $\sigma = 0.0010$ gives the best performance.

PART 2: Parzen window estimation with PNN and voting scheme

In this part, again, PNN is employed for classification. But instead of classifying each test sample one time, each feature of the test samples is considered as one-dimensional sample and classified with

PNN. After letting each feature to vote for classification by itself, the mode of the vote set is taken as the final classification decision. The same sigma values with Part 1 are given as input to the PNN:

```
>> sigma_pnn
sigma_pnn =
    [0.0001    0.0010    0.0100    0.1000    1.0000]'
```

Fig 3: Simulated sigma values for Parzen window estimation with PNN and voting scheme.

Recall, precision, and accuracy rates for each class (columns) and for each sigma (rows) are shown as follows:

```
>> recall.pnn_vote
ans =
    0.9200    0.6800    0.6400    0.4800    0.6400
    0.9600    0.4400    0.6400    0.7200    0.4000
    0.9600    0.4000    0.6000    0.6800    0.5200
    0.9600    0.4000    0.6000    0.6800    0.5200
    0.9600    0.4000    0.6000    0.6800    0.5200
>> precision.pnn_vote
ans =
    0.5610    0.4857    0.9412    0.8571    0.8889
    0.5455    0.4783    0.8889    0.6000    1.0000
    0.5217    0.4545    0.8824    0.6296    1.0000
    0.5217    0.4545    0.8824    0.6296    1.0000
    0.5217    0.4545    0.8824    0.6296    1.0000
>> accuracy.pnn_vote
ans =
    [0.6720    0.6320    0.6320    0.6320    0.6320]'
```

Fig 4: Corresponding recall, precision, and accuracy rates for each simulated sigma value for Parzen window estimation with PNN and voting scheme.

In terms of almost every metric, $\sigma = 0.0001$ gives the best performance.

PART 3: K- Nearest Neighbor estimation

In this part, k-nearest neighbor estimation for $K = 1$ and $K = 5$ is used to classify the test samples. Recall, precision, and accuracy rates for each K value are given consecutively as follows:

```
>> recall.knn
ans =
    0.9600    0.7600    0.8800    0.6800    0.8800
    0.9600    0.8800    0.8800    0.6800    0.9200
>> precision.knn
ans =
    0.9231    0.6333    0.9565    0.8095    0.8800
    0.9231    0.6667    1.0000    0.9444    0.8846
>> accuracy.knn
ans =
    [0.8320    0.8640]'
```

Comparison of the three approaches

The accuracy rates of both KNN and PNN varies between 0.8 and 0.9. Thus, it can be said that their performances are almost matched. Depending on the sigma value, one can get a lower or higher performance using PNN other than KNN.

For PNN with voting scheme, the performance is clearly the worst.

References

PCA.m, MultipleDiscriminantAnalysis.m, PNN.m and Nearest_Neighbor.m functions in DHS toolbox are used in this project.