Chapter 4
Representation Using Linear Combinations

Abstract This chapter deals with linear representations. Here, each new feature is a linear combination of the given input features. Even feature selection may be viewed as a special case of the linear representation schemes. We specifically consider, in this chapter, various types of feature selection schemes, principal components, random projections and non-negative matrix factorization.

4.1 Introduction

Linear representation is a very popular and relatively simpler scheme for representing patterns. Here, a feature g_j is extracted such that it is a linear combination of the given L features, $f_1, \cdots, f_L$. So,

$$g_j = \sum_{i=1}^{L} \alpha_{ij} f_i,$$

where α_{ij} is a real number and it indicates the contribution of feature f_i to the extracted feature g_j. Further, we typically extract l features $g_1, g_2, \cdots, g_l$ where $l < L$.

4.2 Feature Selection

Note that it is possible to view feature selection as a special case of such a linear combination. For example, g_j can be the same as f_p, if $\alpha_{pj} = 1$ and $\alpha_{ij} = 0$ if $i \neq p$. Further, we have seen in Chap. 2 that the feature selection schemes are categorized further into

1. *Filter Methods:* In filter methods, we rank the L features using a fitness measure that exploits the class labels but does not use any classifier. Based on the ranking,

M. N. Murty, M. Avinash, *Representation in Machine Learning*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-7908-8_4

the top l features are selected and used in building the ML model. There are different fitness measures to evaluate and rank the features. They are based on:

- *Distance:* We prefer values of a feature from the same class to be such that the intra-class distances are smaller and inter-class distances are larger.
- *Dependency:* Here, the correlation between a feature and a class is exploited in ranking. If the correlation is larger for a class and smaller for the other classes, then the feature in ranked better.
- *Mutual information* (MI:) If a feature and a class have a larger MI value, then the feature is good. It is seen that MI based feature selection works well on high-dimensional datasets.

2. *Wrapper Methods:* In this case, the selection of a subset of features is based on the performance of a classifier on the selected subset. The subset of features which gives the maximum classification performance is selected. This subset of features is used in designing the ML model. There are a variety of approaches including *genetic algorithm* based approaches that employ this method. These schemes can be more expensive compared to the filter based schemes because the number of subsets is much larger than the number of features.
3. *Embedded Methods:* Here, an ML model is built using the L features and the model directly selects/indicates the relevant subset of features. Several classifiers including ones based on decision trees, support vector machines, and Bayes classifier can be exploited to realize such an embedded scheme.

We will present some experimental results based on these schemes in this chapter.

1. Filter methods:
   Let us consider the patterns of classes labelled 7 and 9 from the *MNIST* data set. Each pattern is a 28×28 image, whose pixel values range from 0 to 255. Every pattern is converted into a binary row vector of dimension 1×784, such that a pixel value greater than 127 is replaced by 1 and pixel value less than or equal to 127 is substituted by 0. These row converted binary patterns are stacked to form the training set of size 12,214 and test set of size 2037 respectively. Each pattern is a binary vector of length 784.

   For each feature, the euclidean distance between the feature and its class label is calculated and shown in Fig. 4.1. Similarly *Mutual Information* and absolute value of *correlation coefficient* of each feature is calculated and shown in Figs. 4.2 and 4.3 respectively. The *correlation coefficient* of features whose value does not change throughout the training data set is set as $zero$ (since *MATLAB* returns **NAN** for those constant features).

   Features which represented pixel values in the top, bottom, left and right extremes of the image are mostly of same value (zeros) so they will not contribute to classification, which is observed from the *Mutual information* values in Fig. 4.2. The *Mutual information* becomes zero for feature number less than 100 and greater than 700 which correspond to the top and bottom pixels in the image. Similarly, *Mutual information* minima (touching 0) in Fig. 4.2, represents the left

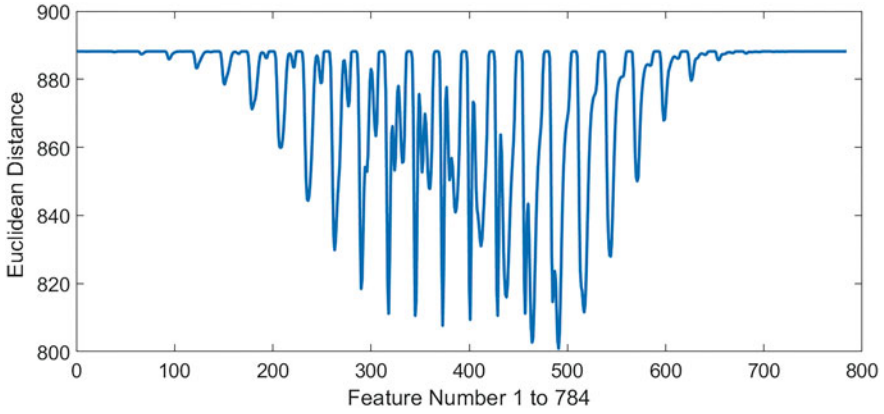

Fig. 4.1 Euclidean distance of each feature to the Class label

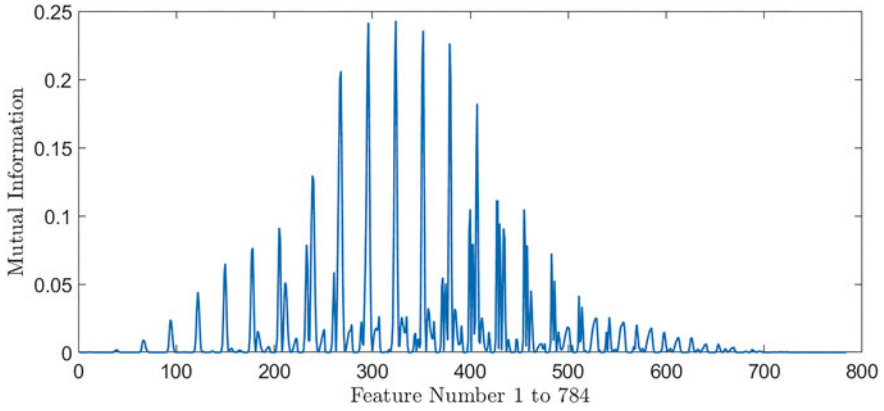

Fig. 4.2 Mutual information of each feature

and right extreme pixels of the image (whose values are constant throughout the training data set).

Based on the above mentioned *Filter methods*, 80 features are selected out of 784 features. These 80 features are used for classification by *Decision tree* classifier with Maximum number of splits, $N = 9$ and *k-NNC* classifier with $K = 10$. The results are shown in the bar (Fig. 4.4).

From Fig. 4.4, we observe that the accuracy of *Decision Tree* with 80 selected features and all 784 features, remains almost the same, since *Decision tree* automatically selects the important features and the *test accuracy* of *Decision tree* is low compared to that of *KNNC*, and further since the classes 7 and 9 are having highly non linear decision boundary, we need a deeper tree to classify correctly. Since, *KNNC* is a non linear classifier, it is able to perform well. Out of the *filter*

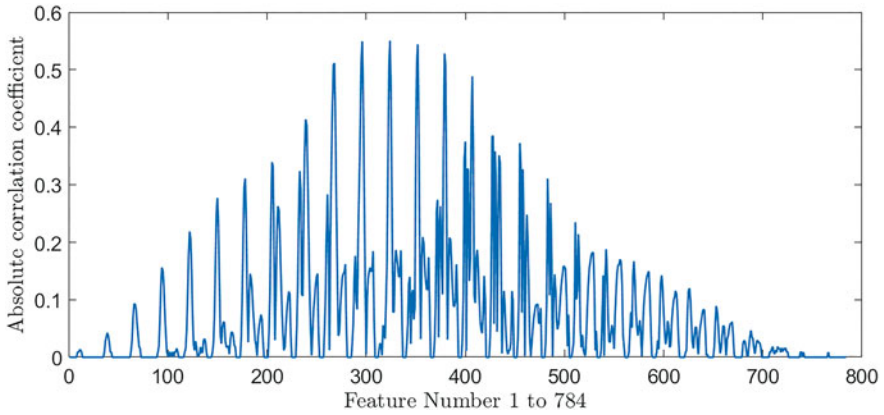

Fig. 4.3 Absolute value of correlation coefficient of each feature

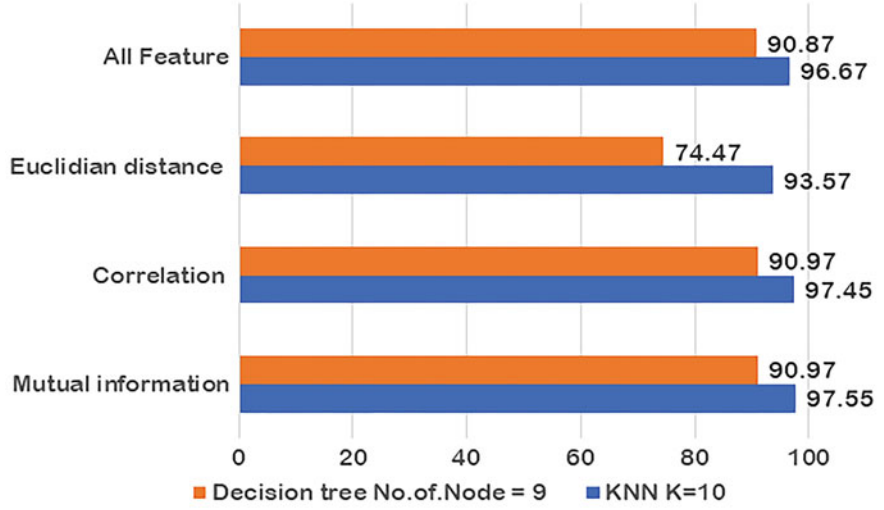

Fig. 4.4 Comparison of various filter methods

methods considered, *Mutual Information* based feature selection performed the best on this data set.

2. Wrapper methods:
   Refer to Sect. 4.3, where we have used *Genetic Algorithm* to select the optimum subset of 100 Principal components from the total set of 10,304 Principal components for the **ORL** face data set, having the *test accuracy* of *KNNC* classifier as the fitness criterion for selecting features.
3. Embedded methods:
   Let us consider the *Iris* data set. It has four features: *Petal length*, *Petal Width*, *Sepal length* and *Sepal width*. We have used *Decision Tree* to classify the three

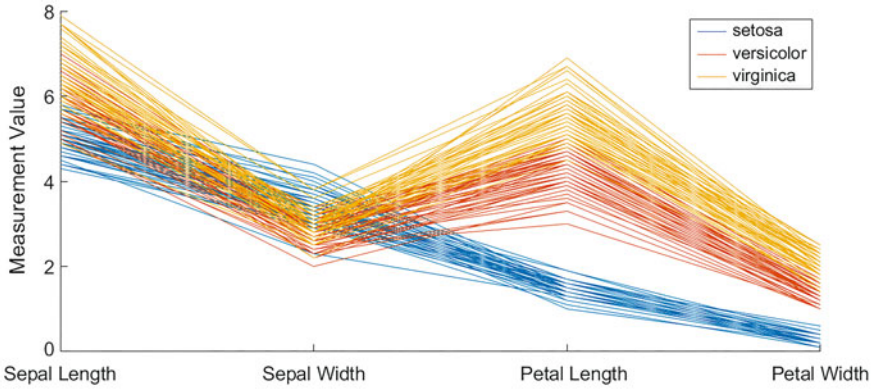

Fig. 4.5 Data visualization of Iris data set using parallel coordinate plot

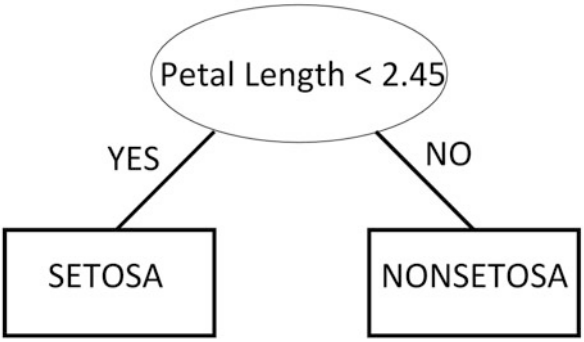

Fig. 4.6 Decision tree on Iris data set

flower species (*setosa*, *vericolor* and *virginica*) based on these four features. We used *Parallel coordinates plot* from **MATLAB**, to visualize the *Iris* data set in Fig. 4.5.

From Fig. 4.5, feature *Petal Length* clearly classifies class label *setosa*. So, the *Decision Tree* classifier has given the first preference to this feature and keep it at the *Root* of the *Decision Tree*, as shown in Fig. 4.6.

Let us calculate the *predictor importance*. It is calculated by summing changes in the mean squared error due to splits on every predictor and dividing the sum by the number of branch nodes. The estimates of *predictor importance* are shown in Fig. 4.7. From Fig. 4.7, we observe that predictor *Petal length* is the most importance feature.

Root of the *Decision Tree* is shown in Fig. 4.6 and the rule captured by the tree is as follows:

(1) if PetalLength $<$ 2.45 then setosa else (PetalLength $\geq$ 2.45) nonsetosa

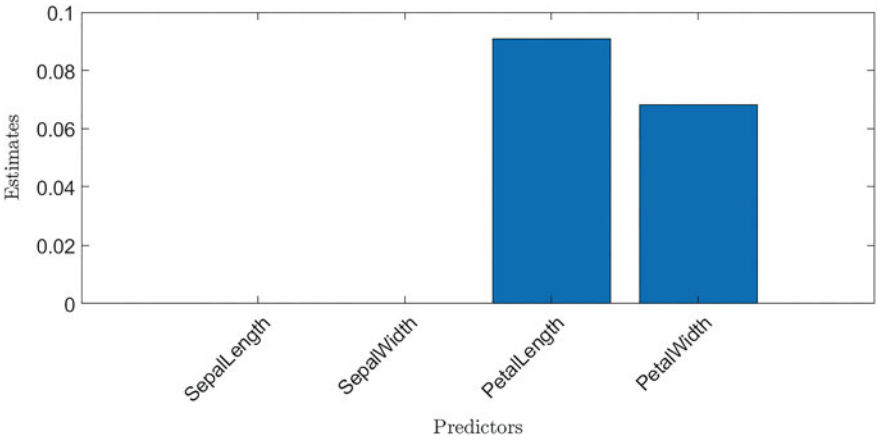

Fig. 4.7 Predictor importance of predictors-decision tree

4.3 Principal Component Analysis

The most popular linear representation scheme is based on principal components (PCs). PCs are the leading eigenvectors of the covariance matrix of the data. Eigenvectors are considered in the decreasing (non-increasing) order of their corresponding eigenvalues. The variance in different directions, that is present in the data, is characterized by the eigenvalues. The covariance matrix is symmetric; so, it is possible to select the eigenvectors to be orthonormal. The top l eigenvectors are used in the analysis where the value of l is chosen based on total variance explained by the top l eigenvalues.

Let us consider the patterns of MNIST data set from Classes labelled 7 and 9. The dimensionality of the data set is 784 (28×28 pixels in each image). From the covariance matrix of the Training data patterns, 784 eigenvalues and their eigenvectors are calculated. The eigenvectors are sorted in descending order based on their corresponding eigenvalues. The dimensionality of both the training and test sets is reduced based on the projections of the data points on the top l eigenvectors. The K-Nearest Neighbor classifier with K = 3, (three-nearest neighbor classifier) is used for classification. The results are plotted in Fig. 4.8. The X-axis depicts the value of l, the number of top eigenvectors used for classification and the Y-axis shows the test accuracy using the $KNNC$ with K = 3.

From Fig. 4.8, we observe that feature selection based on first few Principal components gave good test accuracy. It is to be noted that first few PCs will only be useful for discriminating between groups if within- and between-group variation have the same dominant directions.

We will show that first few PCs may not always give the best representation for classification; that is best test accuracy. In practice, it is possible that the selection of

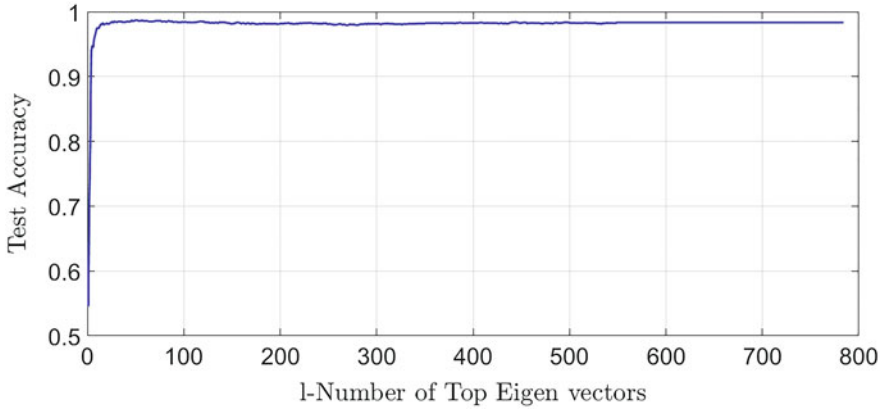

Fig. 4.8 Application of PCA on MNIST data set class label 7 and 9, with 'l' top principal components

last few PCs may provide a better discrimination. So, the feature selection should be based on subset of PCs from the entire spectrum of PCs. We will discuss this next.

For the experiments we use the ORL face Data set. Out of 400 images, 320 images are used for training and remaining 80 images for testing. The data set is divided, such that for each person, eight patterns are used for training and two patterns for testing. The dimension of the data set is 10,304 (92×112). From the covariance matrix of the training set, 10,304 Principal components and the corresponding eigenvalues are calculated, then these Principal components (PC) are sorted based on their eigenvalues (in descending order), such that the first PC corresponds to the highest eigenvalue and the last PC is based on the least eigenvalue.

Dimensionality reduction of both the training and test sets is based on using the first l PCs and KNNC, with K = 3 is used as the classifier. From Fig. 4.9 for the first 108 PCs, we got the maximum test accuracy of 0.95 and for the first 100 PCs, the test accuracy is 0.9375, When we use all the 10,304 PCs, the accuracy drops to 0.925.

To illustrate the point that first few PCs will not always give a better representation in the lower dimensional space, we randomly selected 100 PCs from different sections of the entire spectrum of PCs as shown in Table 4.1.

We present the results in Fig. 4.10.

From the figure, we observe that, for some random selection of 100 PCs, we got test accuracy of 0.9625, which is better than the test accuracy obtained from the first 108 PCs and first 100 PCs.

To find the optimum sub set selection of PCs from the entire spectrum of 10,304 PCs, we used Genetic Algorithm (a stochastic search technique), with population size 150 and maximum of 200 generations and show the results in Fig. 4.11. Here X axis represents the lth PC (as we have selected 100 PCs, it varies from 1 to 100) and

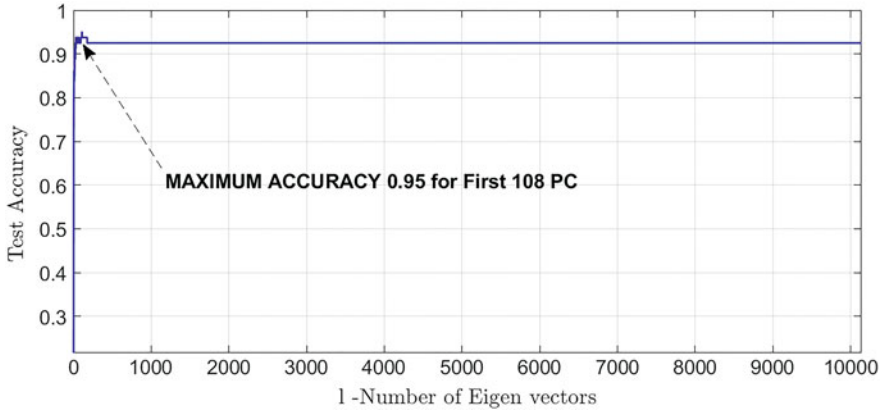

Fig. 4.9 Application of PCA on MNIST data set class label 7 and 9, with 'l' top principal components

Table 4.1 Random selection of PCs from the set of 10,304 PCs

Section Name	Number of PCs
First 100 [1–100]	60
[1000–3000]	10
[4000–6000]	10
[7000–9000]	10
Last [10,204–10,304]	10

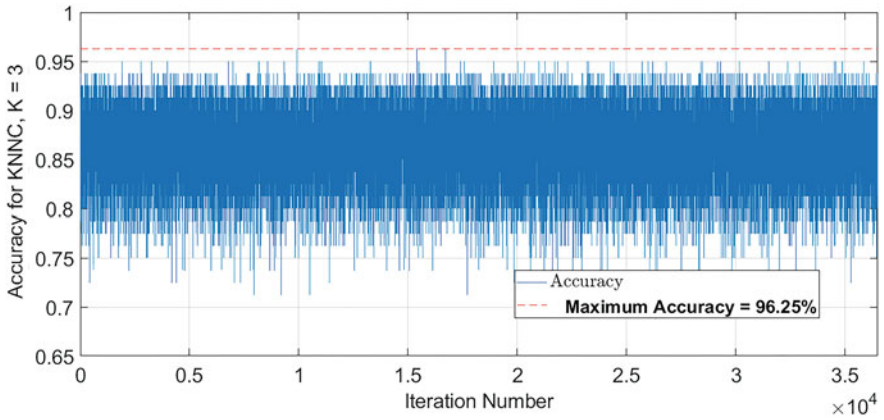

Fig. 4.10 Application of PCA on ORL data set, with random selection of 100 principal components

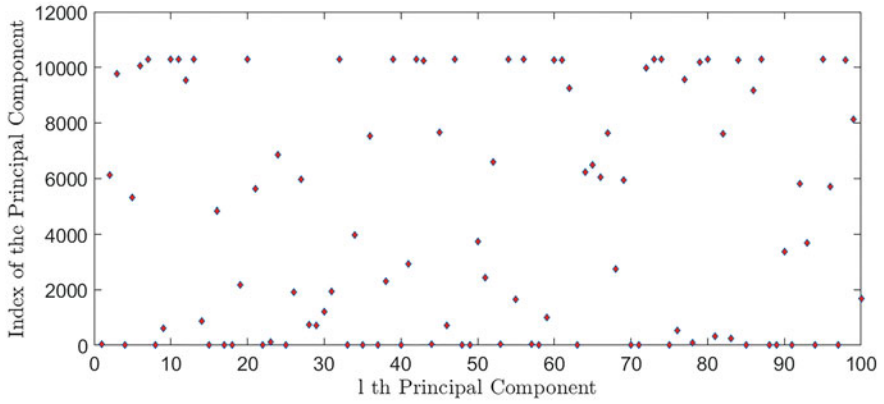

Fig. 4.11 Application of PCA on ORL data set, with 100 principal components given by GA

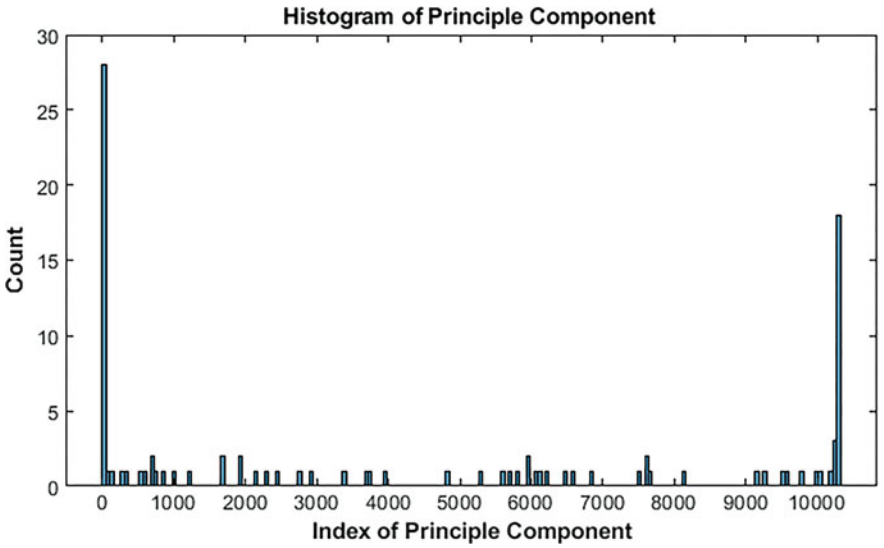

Fig. 4.12 Histogram of Index values of 100 Principal Components given by GA

Y-axis represents the index of the selected PC from the set of 10,304 PCs, where index 1 represents the first PC and index 10,304 represents the last PC.

From Fig. 4.11, we observe that the optimum sub set of 100 PCs, which gave a test accuracy of 0.9625 has indices in the entire spectrum (from 1 to 10,304). So, neither the first few PCs nor the last few PCs may always give better discrimination in the lower dimensional space. The optimum sub set of PCs fall in the entire spectrum, which is evident from the Histogram in Fig. 4.12.

From the Histogram in Fig. 4.12, we also observe that the Last few PCs also play a significant role in discrimination and the optimum choice of PCs selects from the entire spectrum.

4.4 Random Projections

We have discussed in Chap. 2 that in the case of random projections, we reduce the dimensionality of the data from the L-dimensional space to a lower-dimensional space of dimension l, where l can be much smaller than L. This is achieved by using a matrix R that has random entries.

Some of the properties of random projections are:

- $B_{n\times l} = A_{n\times L} R_{L\times l}$
- The value of l can be much smaller than L.
- If the entries in R are independently selected from a zero mean and unit variance distribution, then it is possible to show that

$$E[|| b_i ||^2] = || a_i ||^2,$$

  where b_i is the ith row of B and a_i is the ith row of A.
- Further, by selecting the value of $l = \mathcal{O}(logn/\epsilon^2)$ we can approximately preserve pairwise distances upto a factor of $(1 \pm \epsilon)$, where ϵ is a small real number.
- Similarly it is possible to preserve dot products. That is

$$E[b_i^t b_j] = a_i^t a_j$$

Let us consider the MNIST data set, the subset of patterns belonging to class labels $\{0, 1, 7, 9\}$ from both training set and test set are collected. Let us call this subset as '$\mathbf{X}$'. Then, $\mathbf{X}$ consist of 29,031(= 24,879 + 4152) patterns where first 24,879 patterns belong to the training set and the next 4152 belong to the test set. The dimension of $\mathbf{X}$ is $29{,}031 \times 784$, where each row represents a pattern.

The dimensionality of $\mathbf{X}$ is reduced to $\mathbf{l}$ using *Gaussian Random Projection* model from *python sklearn* package. After, the dimensionality is reduced, the training set patterns (24,879 patterns) and test set patterns (4152 patterns) are separated. The *KNNC* classifier with different $\mathbf{K}$ values is used for classification. The experiment is repeated for five different seed values $\{1, 2, 3, 4, 5\}$ and the average training set and test set accuracies obtained are shown in Table 4.2. The same experiment is repeated using *Sparse Random Projection* model which uses a Sparse Random matrix with density parameter as $1/3.0$, the results are tabulated in Tables 4.3 and 4.4 respectively.

Let us consider the ORL face data set. The Data set consist of 400 faces each with dimension 112×92. The original dimension of the data set is 10,304. The

Table 4.2 Gaussian Random Projection based dimensionality reduction on MNIST data set

l	eps	K	Mean train accuracy	Mean test accuracy
100	0.1	3	0.993432212	0.982996146
100	0.1	5	0.990779372	0.981358382
300	0.1	3	0.99462197	0.984537572
300	0.1	5	0.992555971	0.983815029
500	0.1	3	0.995023916	0.98473025
500	0.1	5	0.992756944	0.98371869

Table 4.3 Sparse random projection based dimensionality reduction: MNIST data set

l	eps	K	Mean train accuracy	Mean test accuracy
100	0.1	3	0.992869488	0.981888247
100	0.1	5	0.989959404	0.980684008
300	0.1	3	0.994372764	0.985356455
300	0.1	5	0.992089714	0.983140655
500	0.1	3	0.994597854	0.985308285
500	0.1	5	0.992379115	0.984104046

Table 4.4 Achlioptas random projection based dimensionality reduction: MNIST data set

l	eps	K	Mean train accuracy	Mean test accuracy
100	0.1	3	0.993335745	0.981743738
100	0.1	5	0.990956228	0.980828516
300	0.1	3	0.994469231	0.984007707
300	0.1	5	0.992113831	0.982755299
500	0.1	3	0.995023916	0.984104046
500	0.1	5	0.992580088	0.983188825

dimensionality of the data set is reduced to l components (selected for different values of *eps*) using the following Projection matrices.

- *Gauss Random Projection matrix*
- *Sparse Random Projection Matrix*
- *Achlioptas Random Projection matrix*

The minimum size of the reduced dimension l for a given *eps* is selected based on *Johnson-Lindenstrauss lemma*.

The experiment is performed for different values of *eps* ranging from 0.1 to 0.95 in the steps of 0.05. For each value *eps*, the minimum value of N is selected based on *Johnson-Lindenstrauss lemma*. Once the dimension of the data set is reduced, it is divided into training and test set such that for each person, eight patterns are selected for training and the remaining two patterns for testing .*KNNC* is used as the classifier for different values of K which range from 1 to 8. The results of the experiments are plotted in Figs. 4.13, 4.14, and 4.15 respectively.

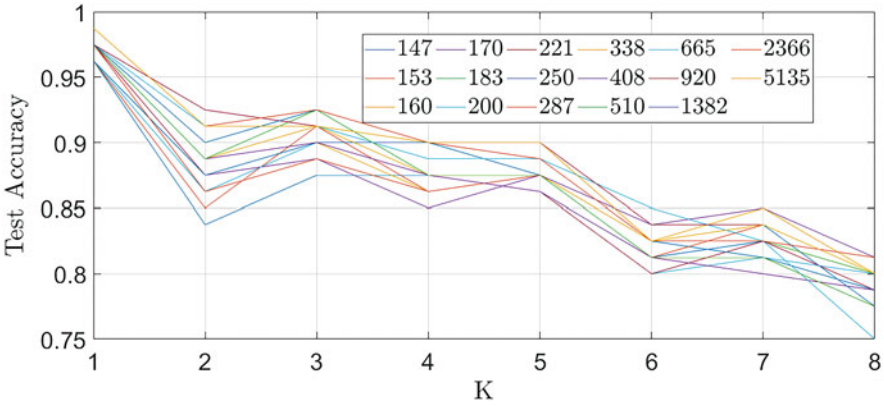

Fig. 4.13 Dimensionality reduction based on gauss Random Projection matrix: ORL Face Data set

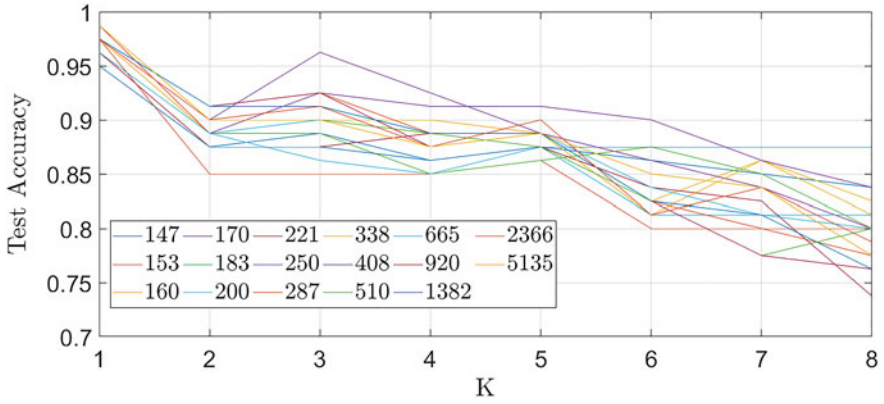

Fig. 4.14 Dimensionality reduction based on sparse Random Projection matrix: ORL Face Data set

4.5 Non-negative Matrix Factorization

Let us consider the NMIST Data set, the training and test patterns belonging to class labels 0, 1, 7, 9 are used for the experiments. The *NMF* module from *SKLearn* is used for dimensionality reduction. The tuning parameters for the modules are *alpha*, which is the constant that multiples the regularization term and *L1 ratio*, which controls the contribution of $L1$-$term$ errors in the regularization term. After reducing the dimensionality, *KNNC* classifier with $K = 5$ is used as the classifier. The NMF may be viewed as factorizing the data matrix A into W and H matrices. So, if A is of size $n \times l$, then we have

$$A_{n \times l} = W_{n \times K} * H_{K \times l}.$$

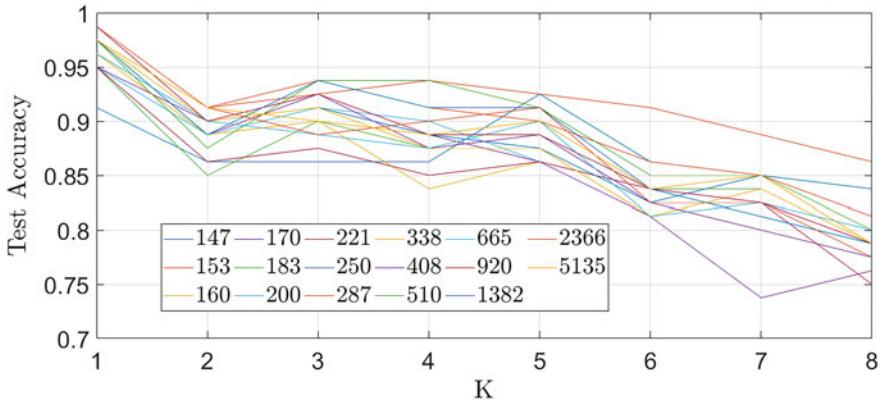

Fig. 4.15 Dimensionality reduction based on Achlioptas Random Projection matrix: ORL Face Data set

Table 4.5 NMF on MNIST Data set for class label 0, 1, 7, 9

l	alpha	L1	Train accuracy	Test accuracy
350	10	0	0.986414245	0.945086705
350	5	0	0.98548977	0.933526012
350	50	0	0.990112143	0.971820809
350	15	0	0.988705334	0.957129094
350	25	0	0.990152337	0.967003854
350	75	0	0.990272921	0.972302505
350	100	0	0.991076812	0.9727842

The experiment is repeated for various values of *alpha*, *L1 ratio* and l (the number of reduced components after dimensionality reduction). By Trail and error, the following results are obtained and tabulated in Table 4.5. Let us consider the smaller version of ORL-Data set, which consists of 4096 features instead of 10,304 features. The total data set consists of 400 faces. The Data matrix is of size 400×4096, where each row represents a face pattern.

NMF function from **MATLAB** is used for dimensionality reduction experimentation. After dimensionality reduction, the data set is divided into 320 Training patterns and 80 test patterns. *KNNC* classifier, with K values range from 1 to 8 is used for classification. The initialization value for W(*W0*) matrix plays a crucial role in determining the best representation (W) and test accuracy (making all the other parameters to take their default values and Maximum iteration step is taken as 100).

For the Random initialization of W (*W0*) and H, the results are shown in Fig. 4.16. Now, for the 400 features, Mutual information is calculated. Based on the Mutual information values, features are sorted and W is initialized with top l features (where, l is the number of reduced components). The *Multiplicative Update* algorithm is modified such that H is updated based on the initialization of W as the first step.

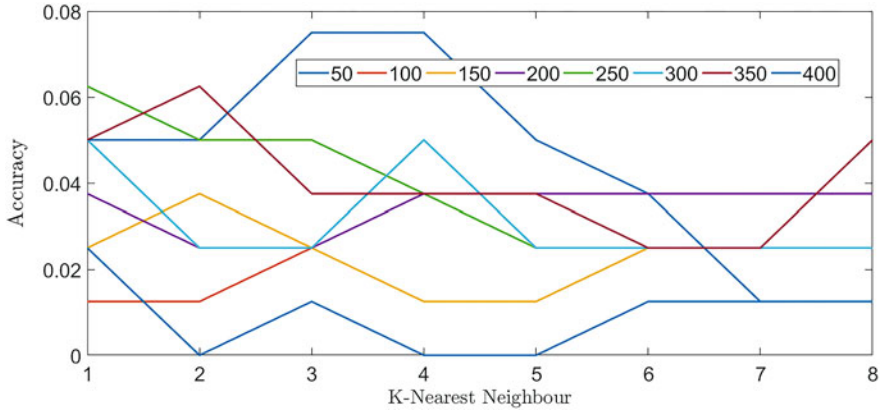

Fig. 4.16 Dimensionality reduction based on NMF on reduced ORL Face Data set with random initialization of W

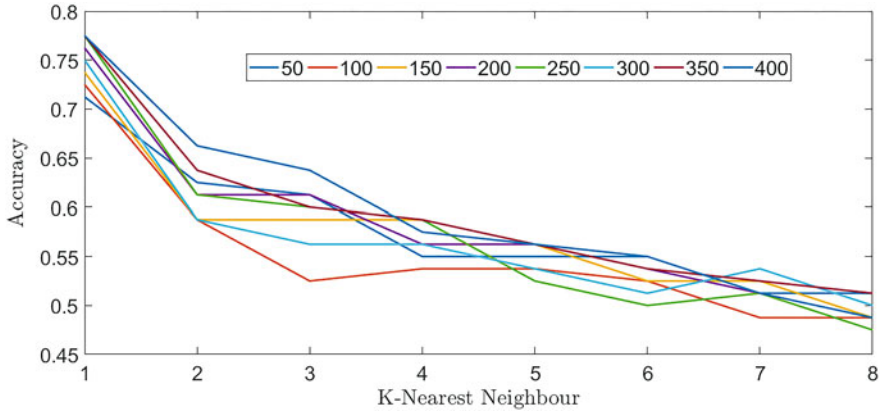

Fig. 4.17 Dimensionality Reduction based on NMF on reduced ORL Face Data set with Mutual information based initialization of W

The results for the *Mutual information* based feature initialization W are shown in Fig. 4.17. The *Root Mean Square of Residual Error* between data matrix and its approximation $W \times H$ for various methods of W initialization are shown in Fig. 4.18. From Fig. 4.18, we observe that improper selection of initial values of W results in reaching local optima.

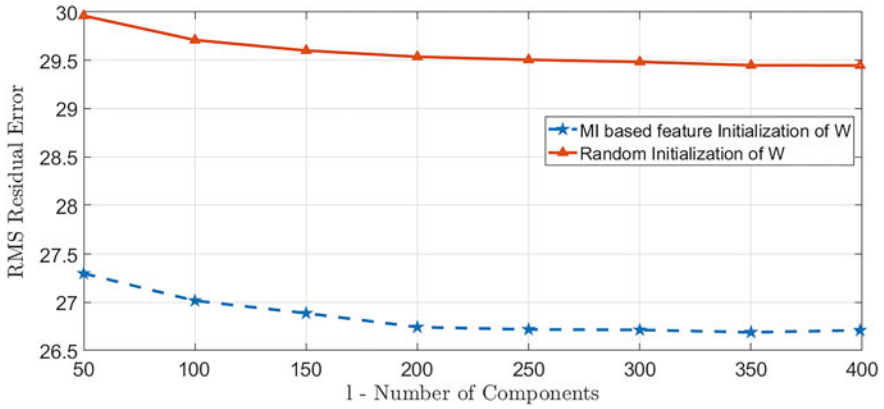

Fig. 4.18 RMS Error Comparison for various methods of initialization of W

4.6 Summary

In this chapter, we have examined various feature selection and linear feature extraction techniques in terms of their applicability. The feature selection schemes include:

1. **Filter methods:** We have considered distance, correlation and mutual information (MI) based methods. We have observed that the MI based scheme is good to reduce the dimensionality of large dimensional data sets.
2. **Wrapper methods:** We have used subsets of PCs to represent the data and $KNNC$ is used in classification in the process of selecting features using a GA.
3. **Embedded methods:** It is shown using the Iris data set that Decision Tree classifier can automatically select a subset of the features as a part of classification.

Subsequently, we have considered linear feature extraction schemes. These include:

1. PC based,
2. Random Projections based, and
3. NMF based:

It is observed that the top PCs may not be the best for representing the data used in classification. So, it is not correct to use the top few PCs for discrimination or classification. Random projections provide a good alternative to deal with high-dimensional data sets. NMF based reduction needs a better initialization scheme to avoid local-minima problems.

References

1. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining Knowl. Discov. **2**(2), 121–167 (1998)
2. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press (2001)
3. Rifkin, R.M.: Multiclass Classification. Lecture Notes, Spring08. MIT, USA (2008)
4. Witten, I.H., Frank, E., Hall, M.A.: Data Mining, 3rd edn. Morgan Kauffmann (2011)
5. Prakash, M., Murty, M.N.: A genetic approach for selection of (near-) optimal subsets of principal components for discrimination. Pattern Recogn. Lett. **16**, 781–787. Elsevier (1995)
6. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS'00: Proceedings of the 13^{th} International Conference on Neural Information Processing Systems, pp. 535–541 (2000)
7. MNIST Dats Set: https://www.tensorflow.org/datasets/catalog/mnist
8. ORL Face Dats Set: https://www.kaggle.com/datasets/tavarez/the-orl-database-for-training-and-testing
9. Scikit-Machine Learning in Python: https://scikit-learn.org/stable/