11 An Application: Handwritten Digit Recognition

In this chapter, we present an example of how pattern classification can be carried out in a digit recognition problem. There are ten classes corresponding to the handwritten digits ''0'' to ''9''. The data set consists of 6670 training patterns and 3333 test patterns. The nearest neighbour algorithm (NN), the kNN and the mkNN algorithms have been used on this data set. Classification is carried out for the test patterns and the classification accuracy reported.

To overcome the problem of using the nearest neighbour algorithm on such a large data set, it is first condensed. The k-means algorithm (KMA) and the fuzzy c-means algorithm (FCMA) are used on the data and the centroids of the clusters formed are used as prototypes representing all the patterns in the cluster. The reduced set so formed is used to obtain the classification accuracy of the test set. Condensation of the training patterns has been carried out using the MCNN algorithm and the CNN algorithm. This reduced set has been used on the test data to obtain classification accuracy. Even though the current application deals with a digit recognition problem, it is possible to extend the scheme to other pattern recognition applications dealing with large data sets.

11.1 Description of the Digit Data

Each original digit pattern is a binary image of size 32×24 pixels. So, the dimensionality of the original pattern is 768 (32×24). Keeping the computational resources we have in mind, the dimensionality of the data set is reduced as follows. Non-overlapping windows of size 2×2 are formed over the entire image and each window is replaced by one feature whose value corresponds to the number of one bits in that window. This results in 192 features, where the value of each feature varies from 0 to 4. There are 6670 (667×10) training patterns and 3333 test patterns. Figure 11.1 shows some of the patterns in the training data set. It can be seen that the patterns in the data vary in terms of orientation of the digit, width and height.

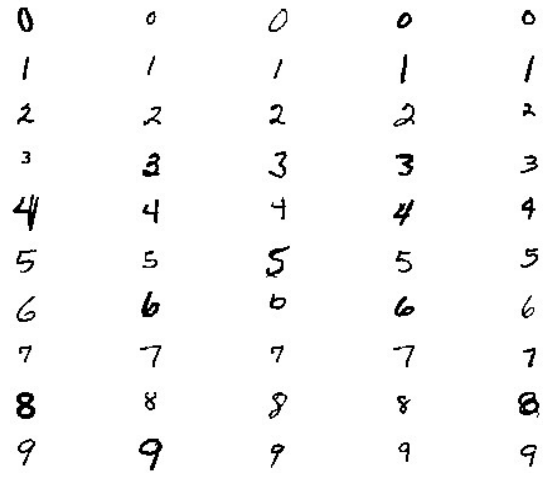

Figure 11.1 A sample collection of training patterns

To understand the data better, central tendency and dispersion of the patterns is considered. To do this, the non-zero features of the binary pattern is taken. Table 11.1 provides the results.

It can be seen that in the case of Class 0, the patterns consist of features ranging from 39 to 121. A high standard deviation reflects the large variation in the number of non-zero features of the patterns of the class.

Table 11.1 Statistics on non-zero features of the data set

Class Id.	Arithmetic mean of 1s per pattern	Standard deviation	Actual min. and max. of 1s across all patterns
0	66.8	12.2	(39, 121)
1	29.0	5.9	(16, 54)
2	63.8	11.5	(34, 105)
3	59.0	11.7	(27, 109)
4	52.2	9.4	(26, 96)
5	60.7	11.3	(31, 110)
6	57.7	9.4	(35, 94)
7	46.5	8.7	(26, 113)
8	66.7	12.9	(36, 117)
9	54.8	9.1	(31, 88)

The dimensionality of the data set is further reduced. Non-overlapping windows of size 2×2 is formed over the 16×12 digit data and each window is replaced by one value corresponding to the sum of the four values falling in the window. This reduces each digit to a 8×6 matrix of values in the range 0 to 16. These values are

linearly mapped to the range 0 to 4 and each pattern gets a total of 48 features. This reduction does not affect the shape of the character significantly and was necessitated to overcome the curse of the dimensionality problem and also to improve the speed of the nearest neighbour type of algorithms.

11.2 Pre-processing of Data

Pre-processing of data is to be carried out to bring about some uniformity in the data and make it amenable to classification.

The three most important operations to pre-process a character image are: (1) scaling, (2) translation, and (3) rotation. The first two operations are reasonably easy to perform and so they are performed frequently to normalise the image size. Pre-processing is done based on translation and scaling as described below.

1. *Translation* The maximum size of a character is 192 (16×12) pixels where (1,1) corresponds to the first row and first column of the character. Similarly, (16,12) corresponds to the last row (16th row) and last column (12th column). So, the column index varies between 1 and 12 and the row index varies from 1 to 16. The character is shifted so that its centre in the horizontal direction falls on the 6th column which lies half way between columns 1 and 12 and its centre in the vertical direction falls on the eighth row which is half way between rows 1 and 16.
2. *Scaling* The pattern is stretched in the vertical direction. It is cut horizontally into two halves. The top half is shifted to start on the first row and the bottom half ends on the 16th row. Intermediate rows, if empty, are filled based on their nearest non-empty rows.

11.3 Classification Algorithms

A variety of neighbourhood classifiers have been used on the digit recognition data. The class label of a test pattern is decided based on the label of one or more neighbours here. The specific classifiers we have used are the nearest neighbour classifier (NN), the k-nearest neighbour classifier (kNN), and modified k-nearest classifier (MkNN). These algorithms have been explained in detail in Chapter 3.

11.4 Selection of Representative Patterns

Neighbourhood classifiers work well. For example, it was observed, based on a large collection of experiments, that the kNN is the most robust classifier among the

statistical pattern classifiers. However, it tends to use the entire training data set to obtain the neighbours of a test pattern. So, it has a time complexity of $O(\text{NMK})$, where N is the number of training patterns, M is the number of test patterns, and K is the number of neighbours obtained. There are several classifiers used earlier which use a representative set or a sub-set of the training data set. These include the minimal distance classifier (MDC), and the condensed nearest neighbour classifier (CNN). These methods have been explained in detail in Chapter 3.

Obtaining the condensed data set is a time-consuming process but once it is available, classification can be much faster as compared to using the entire training data set.

One method is to cluster the data and use the cluster centres as the prototypes in the condensed set. The popular algorithm for clustering is the k-means algorithm (KMA). Making use of the concept of fuzzy membership, we use the fuzzy c-means algorithm (FCMA).

CNN is typically good at reducing the size of the training data set. But this is achieved at the cost of some reduction in the classification accuracy. Also, CNN is *order-dependent*. The obtained *condensed* set varies in size and contents based on the order in which the training patterns are presented to the above algorithm.

To develop an order-independent algorithm which ideally gives the optimal set of patterns in the condensed set, an algorithm was developed which gives the same set of patterns even if the order of the data set is changed. In other words, it is an order-independent algorithm. This is the MCNN algorithm described in Chapter 3.

11.5 Results

Several experiments were conducted on the digit recognition data using the nearest neighbour based classifiers. Two training sets were with: (1) all the 6670 training patterns, and (2) a condensed set of 1715 training patterns obtained by using the CNNC on all the training patterns. Further, in each of these cases, we worked with two values for the dimensionality of the data: (1) using all the 192 features corresponding to the data set described earlier, and (2) using only 48 features obtained by further feature reduction. These results are presented in Table 11.2 and Table 11.3. For each case, the classification accuracy is calculated. This is a measure of how many patterns are classified correctly using the particular classification algorithm. If $N1$ patterns out of N patterns are classified correctly, then the classification accuracy, CA is

$$\text{CA} = \frac{N1}{N} \times 100$$

It is possible to observe from these tables that the mkNN is performing better than the other two classifiers and the best classification accuracy of 92.57% is obtained

by using mkNN on the entire training data set with 192 features as reported in Table 11.2.

Both the raw data and the pre-processed data have been used for the neighbourhood classifiers. The pre-processed data gives better results.

When centroids of classes are used as training patterns—a requirement in MDC—it is necessary to pre-process the data as described earlier. This data has been considered using all the 192 features and also using only 48 features. The results obtained using these data sets are reported in Table 11.4 and Table 11.5. It can be observed from these tables that MDC performs much better on the pre-processed data. This is expected because the centroid will represent processed data patterns better than the raw patterns. Due to this, only pre-processed data has been used to obtain representatives (centroids) using clustering algorithms KMA and FCMA.

Table 11.2 NN results using 192 features

Number of training patterns	NN(%) accuracy	kNN(%) accuracy	mkNNC(%) accuracy
6670 (all)	91.50	91.60	**92.57**
6670 (pre-proc)	92.17	92.50	**93.10**
1715 (condensed)	86.20	87.19	90.60

Table 11.3 NNC results using 48 features

Number of training patterns	NN(%) accuracy	kNN(%) accuracy	mkNN(%) accuracy
6670 (all)	85.00	86.20	88.18
6670 (pre-proc)	92.71	92.62	**93.49**
1715 (condensed)	76.33	81.55	82.84

Table 11.4 MDC results using 192 features

Number of training patterns	Raw data accuracy(%)	Pre-processed data accuracy(%)
6670 (all)	68.5	79.48
1715 (condensed)	68.08	80.89

Table 11.5 MDC results using 48 features

Number of training patterns	Raw data accuracy (%)	Pre-processed data accuracy (%)
6670 (all)	66.20	80.26
1715 (condensed)	65.92	81.55

KMA and FCMA were used to find out the centroids that represent various classes. Two sets of experiments were conducted corresponding to different number of centroids. In the first case, each class of 667 training patterns is grouped into 50 clusters and each of the resulting clusters is represented by the centroid of the data in the cluster. As a result, the entire collection of 6670 training patterns is represented by 500 centroids, 50 centroids from each class. In the second case, each class of training patterns is represented by 150 centroids after grouping the patterns into 150 clusters. This approach results in a total of 1500 centroids from the ten classes corresponding to the digits ''0'' to ''9''. In each case, clustering is done using KMA and FCMA. The classification accuracy is computed using NN, kNN, and mkNN on all the test patterns with the set of centroids forming the training data. We present the results obtained in Table 11.6 and Table 11.7.

It may be observed from both Table 11.6 and Table 11.7 that using KMA to obtain the centroids and having a small number of centroids gives results almost as good as using the entire training data. The FCMA takes a lot of time as compared to the KMA.

Besides, NNC, kNN and mkNN, there is the fuzzy kNN algorithm (FKNN). This has been used for classifying the test data using the centroids obtained by KMA and FCMA. These results are given in Table 11.8.

Table 11.6 Results using 500 centroids as training data

Clustering algorithm used	NN accuracy (%)	kNN ($k = 5$) accuracy (%)	mkNN ($k = 5$) accuracy (%)
KMA	92.41	90.28	**93.10**
FCMA	85.39	85.15	85.72

Table 11.7 Results using 1500 centroids as training data

Clustering algorithm used	NN accuracy (%)	kNN ($k = 5$) accuracy (%)	mkNN ($k = 5$) accuracy (%)
KMA	93.00	92.83	**93.64**
FCMA	88.69	88.12	88.87

Table 11.8 Results using fuzzy kNN classifier

Clustering algorithm used	No. of centroids	Accuracy (%)
None	6670	93.13
KMA	500	92.71
KMA	1500	93.94
FCMA	500	85.06
FCMA	1500	88.75

The results show that the accuracy obtained is of the same magnitude as that obtained using NN, kNN and mkNN.

The MCNN is an order-independent algorithm which finds a condensed set of prototypes to represent the training set. A set of patterns were found using MCNN and these patterns were used to carry out classification of the test data using NN algorithm. The results are presented in Table 11.9 and are compared to the CNN algorithm.

Table 11.9 Classification accuracy using CNN and MCNN

Algorithm used	No. of prototypes	NN algorithm accuracy (%)
CNN	1580	87.04 %
MCNN	1527	88 %
All patterns	6670	92 %

Another method of clustering is the leader algorithm explained in Chapter 8. This is an incremental algorithm which goes through the training data set only once and is therefore suitable for large data sets which are stored in secondary storage devices. In this method, the threshold is fixed and the resulting number of clusters depend on this threshold. Table 11.10 gives the number of prototypes and the classification accuracy obtained by using different threshold values. As the threshold is increased, the number of clusters reduces. This results in fewer prototypes. The classification accuracy reduces as the threshold is increased (i.e., when the number of prototypes decrease).

Table 11.10 Results using leader algorithm for clustering

Distance threshold	No.of prototypes	C.A. %
5	6149	91.24
10	5581	91.24
15	4399	90.40
18	3564	90.20
20	3057	88.03
22	2542	87.04
25	1892	84.88
27	1526	81.70