Rapid and brief communication

Efficient pattern synthesis for nearest neighbour classifier

Monu Agrawal[a], Neha Gupta[b], R. Shreelekshmi[a], M. Narasimha Murty[a,*]

[a]*Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India*
[b]*Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India*

Abstract

Synthetic pattern generation is one of the strategies to overcome the curse of dimensionality, but it has its own drawbacks. Most of the synthetic pattern generation techniques take more time than simple classification. In this paper, we propose a new strategy to reduce the time and memory requirements by applying prototyping as an intermediate step in the synthetic pattern generation technique. Results show that through the proposed strategy, classification can be done much faster without compromising much in terms of classification accuracy, in fact for some cases it gives better accuracy in lesser time. The classification time and accuracy can be balanced according to available memory and computing power of a system to get the best possible results.

Keywords: Pattern synthesis; Prototyping; Classification; Clustering; Partitioning

1. Introduction

The most popular non-parametric classification method is the nearest neighbour classification method. It is widely used because of its simplicity and good performance. Cover and Hart [1] showed that the error for 1-NN classifier is bounded by twice the Bayes error when the available sample size is infinity. Practically we cannot have sample space of infinite size. Synthetic pattern generation is one of the strategies used to increase the sample size. Partition-based pattern synthesis [2] generates an artificial training set of $O(n^p)$ where p is the number of blocks and n is the number of patterns. But its main disadvantage is that it takes more time as compared to simple Nearest Neighbour classifier on the original data set. Our proposed methodology reduces classification time by taking prototypes for each block in each class.

1.1. Notations and definitions

d:	*Dimensionality of the data set.*
C:	*Number of classes.*
n:	*Number of training patterns.*
p:	*Number of blocks in the partition.*
l:	*Number of clusters.*
K:	*Number of nearest neighbours considered for KNNC.*
F:	*It represents set of features* $\{f_1, f_2, \ldots, f_d\}$
Partition:	*Partition of F is obtained as* $\{B_1, B_2, \ldots, B_p\}$ *such that* $B_i \subseteq F$, $1 \leqslant i \leqslant p$. $\cup_i B_i = F$, *and* $B_i \cap B_j = \phi$, if $i \neq j$, $1 \leqslant i, j \leqslant p$.

* Corresponding author. Tel.: +91 80 2293 2468; fax: +91 80 2360 2911.

E-mail addresses: monu@csa.iisc.ernet.in (M. Agrawal), neha@ee.iisc.ernet.in (N. Gupta), lekshmi@csa.iisc.ernet.in (R. Shreelekshmi), mnm@csa.iisc.ernet.in (M. Narasimha Murty).

where B_i is the ith block.

Merge Operation ($\diamond$) :

If P and Q are two sub-patterns in blocks B_i and B_j, respectively, where $i \neq j$, then merge of P and Q written as $P \diamond Q$ is a sub-pattern in $B_i \cup B_j$.

1.2. Partition-based pattern synthesis

Partition-based [2] pattern synthesis is a bootstrap method which generates an artificial training set of $O(n^p)$ so as to improve the accuracy of the classifier when not enough training data is available. This method is based on partitioning of the data set and the synthetic patterns are obtained as combinations of these partitions. The partition-based pattern synthesis using Greedy KNNC approach (*GDC-KNNC*) is explained in Algorithm 1.

Algorithm 1. Greedy KNNC for partition-based synthetic patterns

```
Let T be the test pattern to be classified.
Apply the partitioning on the training data set and T.
for each block $B_j$ do
    find the $k$-nearest neighbours in each class.
end for
Combine all $i$th nearest neighbour blocks in a class to obtain a single pattern and thus generate $k^*C$ synthetic patterns.
Find global $k$-nearest neighbours from the new synthetic patterns obtained.
Apply KNNC to classify the pattern.
```

1.3. Prototype selection

The intention of prototype selection [3] is to reduce the data set size and arrive at most promising data representatives by eliminating outliers. The main drawbacks of having large data set size are the following:

Time complexity: The efficiency of KNN Classifier is $O(n)$. So, as the size of data increases, the time needed by the algorithm also increases.

Resources: KNN classifier needs to have the complete data set stored in memory to carry out classification. If the size of the data set is very large, the computer would need to use the disk as swap memory. This loss of resources has an adverse effect on efficiency due to the increased access to the disk.

Sensitivity to noise: KNN classifier is sensitive to noise due to its operation on local neighbourhood only. Presence of outliers causes reduction in accuracy obtained.

In this paper, we have applied prototype selection by replacing the large data set by a subset of cluster representatives.

2. Proposed strategy

The original training data used for classification of a test pattern contains a large number of patterns that may include noise and outliers. Now if we apply any synthetic pattern generation technique on this data, many unnecessary patterns are formed. This reduces the accuracy and increases the testing time. In this paper, we have applied prototype selection by replacing the large data set by a smaller set of cluster representatives. The proposed method reduces the testing time by using prototypes of the training data set rather than the training data itself. The training data is partitioned into blocks. Then for every class, clusters of each block are formed. Clustering is done separately on each block and not on the data as a whole as the features in a particular partition are more correlated as compared to features in different partitions. The centroids of the clusters so obtained are used as data representatives. The synthetic patterns are generated from the compact representation of data. The method is explained in Algorithms 2 and 3.

Algorithm 2 Training

```
for each class $C_i$ do
    Obtain the partition $\pi_i$ of feature set F and apply this partitioning on training data to obtain blocks $B_1, B_2, \ldots, B_p$.
    for each block $B_j$ do
        Perform clustering on training data and get cluster representatives as $R_1, R_2, \ldots, R_l$.
    end for
end for
```

Algorithm 3 Testing

```
Apply the same partitioning on test pattern.
for each class $C_i$ do
    for each block $B_j$ do
        Find $k$ nearest neighbours of corresponding test sub-pattern as: $\{X^i_{j1}, X^i_{j2}, \ldots, X^i_{jk}\}$ from $R_1, R_2, \ldots, R_l$.
    end for
    Obtain $\{S^i_1, S^i_2, \ldots, S^i_k\}$ as:
    for $l = 1$ to $k$ do
        $S^i_l = X^i_{1l} \diamond X^i_{2l} \diamond \cdots X^i_{pl}$ where $\diamond$ is merge operation [2].
    end for
end for
Apply any nearest neighbour classification on the above obtained $k^*C$ synthetic patterns to label the test pattern.
```

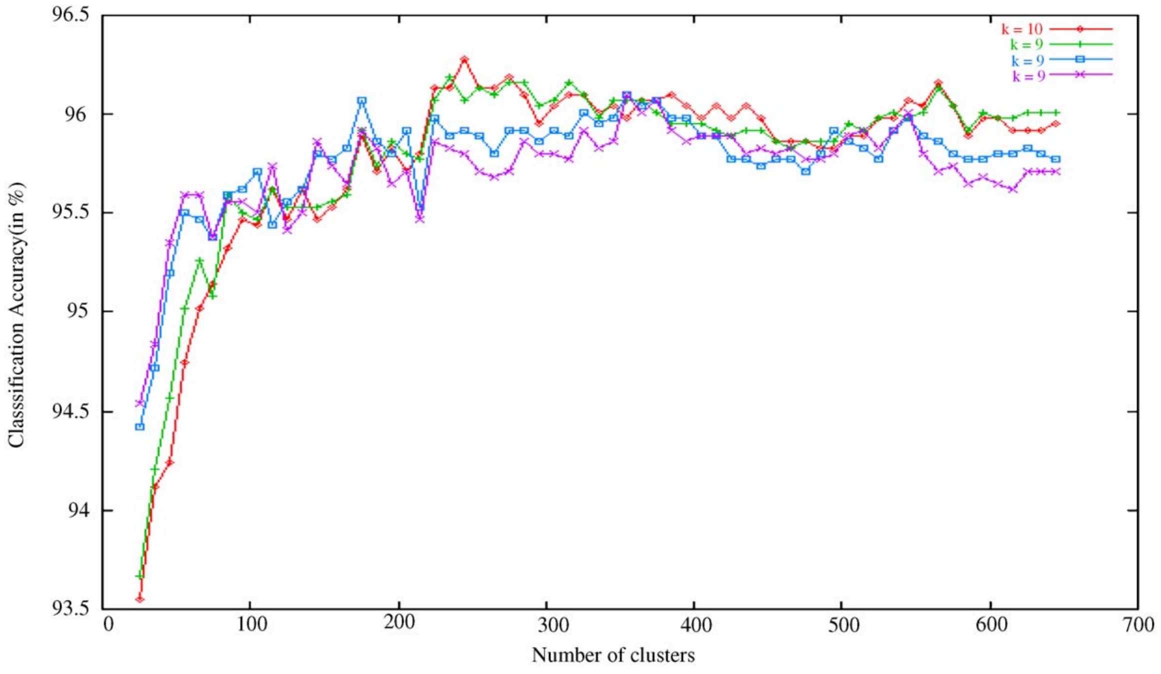

Fig. 1. Classification accuracy vs. no. of clusters.

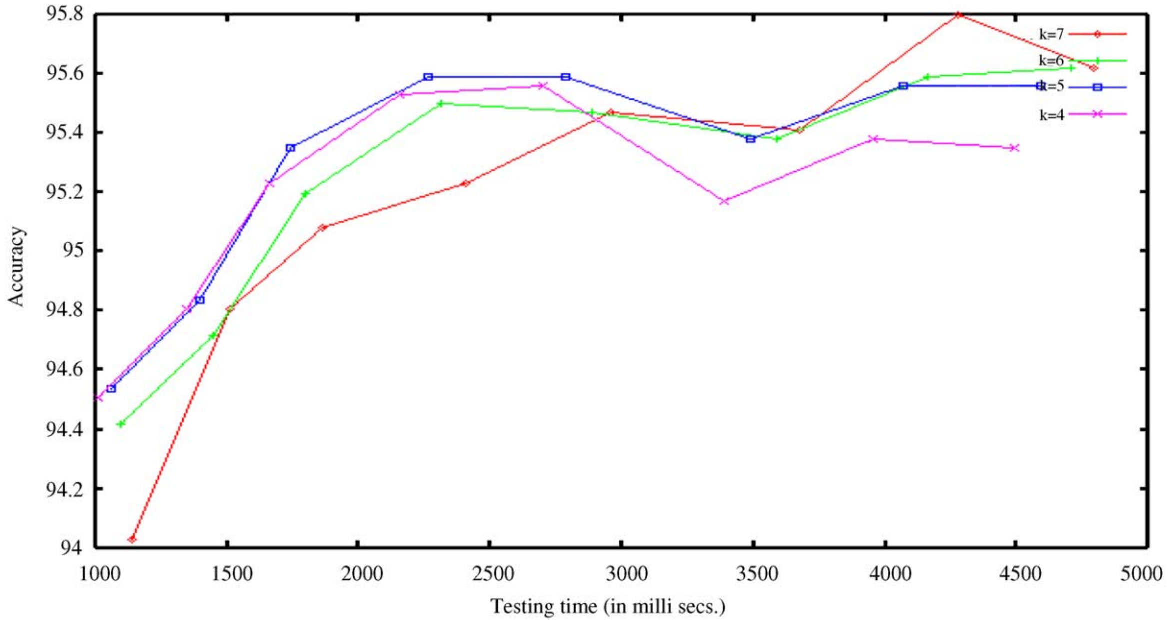

Fig. 2. Classification accuracy vs. testing time.

3. Experimental results

We have applied the proposed strategy on OCR data [2] having 667 patterns per class. We measured clustering time, testing time and accuracy (%) varying K from 4 to 10 and number of clusters from 25 to 645. The results are shown in the graphs. We have obtained the maximum accuracy of 96.28% at $K = 10$ and number of clusters $= 245$. Fig. 1

Table 1
Accuracy vs. K

K	Accuracy(%)
2	95.68
3	**96.28**
4	95.41
6	92.14
12	92.5
16	89.92

Table 2
Comparison with partition-based pattern synthesis

	K	Clusters	Accuracy(%)	Time(s)
KNNC	7	—	93.63	10.03
GDC-KNNC	8	—	96.04	19.08
With prototyping	10	245	**96.28**	**14.55**
With prototyping	5	55	**95.59**	**2.26**
With prototyping	4	25	**94.51**	**1.001**

shows accuracy (%) vs. number of clusters for different values of K. Fig. 2 plots accuracy vs. testing time for different values of K. It depicts that at low values of parameters (such as $K = 4$ and number of clusters $= 25$) we are able to attain a high accuracy of 94.51% in 1.009 s. With testing time of 2.26 s (with $K = 5$, number of clusters $= 55$), we are able to achieve an accuracy of 95.59%. We can observe that the increase in accuracy with increase in number of clusters or increase in value of K is not much.

Table 1 shows classification accuracy(%) measured against different partitions. We have obtained highest accuracy for partition with 3 blocks. Table 2 compares the proposed strategy with partition-based pattern synthesis [2] and KNNC. It can be seen that with little memory (such as 25 clusters per class) and less classification time the strategy is able to classify the data with a good accuracy. As compared with KNNC proposed strategy is approximately 10 times faster and also yields better classification accuracy.

4. Conclusion

In this paper, we have proposed a more efficient method for pattern classification using partition-based pattern synthesis. Our experimental results show that when compared with partition-based pattern synthesis, the proposed method takes less classification time without much reduction in classification accuracy and in fact in some cases it gives better classification accuracy. This strategy is useful in applications with memory and time constraints such as in handheld devices.

References

[1] T.M. Cover, P.E. Hart, Nearest neighbour pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[2] P. Viswanath, M.N. Murty, S. Bhatnagar, Fusion of multiple approximate nearest neighbor classifiers for fast and efficient classification, Inf. Fusion 5 (2004) 239–250.
[3] T.R. Babu, M.N. Murty, Comparison of genetic algorithm based prototype selection scheme, Pattern Recognition 34 (2) (2001) 523–525.