A Pattern Synthesis Technique with an Efficient Nearest Neighbor Classifier for Binary Pattern Recognition*

P. Viswanath, M. Narasimha Murty and Shalabh Bhatnagar
Department of Computer Science and Automation, Indian Institute of Science,
Bangalore - 560 012, India.
{viswanath,mnm,shalabh}@csa.iisc.ernet.in

Abstract

Important factors affecting the efficiency and performance of the nearest neighbor classifier (NNC) are space, classification time requirements and for high dimensional data, due to the curse of dimensionality, the training set size should be large. In this paper we propose novel techniques to improve the performance of NNC and at the same time to reduce its computational burden. A compact representation of the training set along with an efficient NNC which does implicit pattern synthesis is presented. A comparison of empirical results is made with relevant methods.

1. Introduction

Nearest neighbor classifier (NNC) is a very popular non-parametric classifier [3]. It is widely used because of its simplicity and good performance. It has no design phase and simply stores the training set. The space and classification time requirements are the two major shortcomings of the classifier. To add to this list, its performance depends on the training set size.

Cover and Hart [2] show that the error for NNC is bounded by twice the Bayes error when the available sample size is infinity. Duda *et al.,* [3] mention that for non-parametric techniques, the demand for a large number of samples grows exponentially with the dimensionality of the feature space. This limitation is called the *curse of dimensionality*. Re-sampling techniques like bootstrapping [1, 4] can be used to reduce the curse of dimensionality effect to some extent. However there is no unified solution for all of the shortcomings of NNC.

In this paper we propose a novel pattern synthesis technique called *overlap pattern synthesis (OLP-synthesis)*, a corresponding compact representation of the training set called *overlap pattern graph (OLP-graph)* and an efficient NNC called OLP-NNC. These techniques are suitable for domains where the features are binary or discrete valued.

The number of synthetic patterns generated by OLP-synthesis can be exponential in the number of original patterns(i.e., given training set). As a result, the synthetic patterns cannot be explicitly stored. To overcome this problem, OLP-NNC directly works with the OLP-graph and avoids explicit synthetic pattern generation.

Empirical studies show that the space and classification time requirements of OLP-NNC are smaller than those of NNC, k-NNC, and NNC based on bootstrap technique given by Hamamoto *et al.,* [4]. Further the classification accuracy of OLP-NNC is greater than that of other classifiers.

The rest of the paper is organized as follows: Section 2 describes overlap pattern synthesis. Section 3 describes OLP-graph. OLP-NNC is explained in Section 4. Experimental results are described in Section 5 and conclusions in Section 6.

2. Overlap Pattern Synthesis

Let $F = \{f_1, f_2, \cdots, f_d\}$ be an ordered set of features, with f_i $(1 \leq i \leq d)$ as the i^{th} feature. If $X = (x_1, x_2, \cdots, x_d)^T$ is a pattern in d-dimensional vector format then $X[f_i]$ is the feature-value of pattern X corresponding to feature f_i i.e., $X[f_i] = x_i$. Two patterns X and Y are said to have an overlap of length l if $X[f_i] = Y[f_i], X[f_{i+1}] = Y[f_{i+1}], \cdots, X[f_{i+l-1}] = Y[f_{i+1-l}]$ for some $i, 1 \leq i \leq (d - l + 1)$.

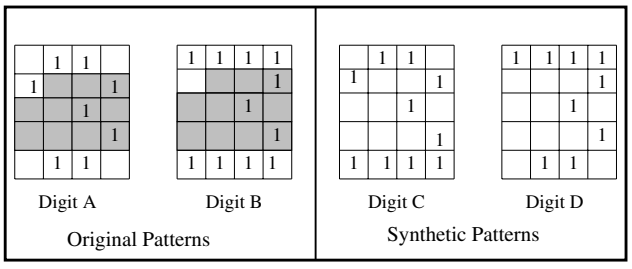
For a given overlap length l and two distinct patterns X and Y in a class, overlap pattern synthesis (OLP-synthesis) is done as follows: If X and Y have an overlap of length l or more, swap the respective non-overlapping portions of X and Y to generate two synthetic patterns.

OLP-synthesis is done for each class separately. For a given class of original patterns and for a given overlap length l, we find a pair of patterns having an overlap of length atleast l and add the two synthetic patterns generated to the set. All possible pairs are considered in this man-

* This work is partially supported by AOARD Contract #F62562-03-P-0318

ner until no more additions to the set are possible. The synthetic set generated thus is a superset of the original set.

We present one simple example with hand written digits. Figure 1 illustrates two original digits (for digit '3') and two synthetic digits derived from the original digits. In this example the digits are drawn on two-dimensional grid of size 5×4. A cell with '1' indicates presence of ink. Empty cells are assumed to have '0'. Patterns are represented row-wise. So the digit A is $(0,1,1,0,1,0,0,1,0,0,1,0,0,0,0,1,0,1,1,0)^T$ and B is $(1,1,1,1,0,0,0,1,0,0,1,0,0,0,0,1,1,1,1,1)^T$. Let the overlap length required be 10. These patterns have an overlap of length 11 from feature f_6 to feature f_{16} (shown as shaded cells in the figure). Overlap pattern synthesis is done by swapping the non-overlapping portions. And by doing so we get two synthetic patterns as shown in Figure 1, *viz.,* the digits C and D, respectively.



Note: The empty cells are assumed to have 0's.
Shaded portion is the overlap.

Figure 1. Illustration of synthetic patterns.

In domains like optical character recognition (OCR), image recognition, etc., OLP-synthesis works well because in these domains, features have a natural ordering so that the set of features $F = \{f_1, f_2, \cdots, f_d\}$ becomes an ordered set with ordering $f_i \prec f_{i+1},\ (1 \leq i \leq d)$. Further F has the property that the correlation between f_i and f_j depends on $\mid i-j \mid$. As $\mid i-j \mid$ increases, the average correlation between f_i and f_j tends to zero. Empirical studies with hand written digit data show that this is true. OLP-synthesis cleverly chooses the values for nearby features. The following lemma is a property of OLP-synthesis which establishes a reason for its goodness.

Lemma 2.1 : Let X, Y be two patterns having an overlap of length l. Let P be an overlap synthetic pattern generated from X and Y. Then feature values for any consecutive $(l+1)$ features present in P are, respectively, present either in X or in Y.

It is easy to see that this lemma is true. By induction, it follows that feature values for any consecutive $(l+1)$ features present in a synthetic pattern are actually present in an original pattern.

3. Overlap Pattern Graph

Explicit generation and storage of synthetic patterns is not done because it increases space and time requirements.

For a given class of original patterns, overlap pattern graph (OLP-graph) is a compact data structure built by inserting each original pattern into it. But the patterns that can be extracted out of the OLP-graph form the entire synthetic set which can be synthesized by following the overlap pattern synthesis.

OLP-graph has two major parts, *viz.,* (i) directed graph and (ii) header table. Every node in the graph has (i) a feature value, (ii) an adjacency list of arcs and (iii) a node-link. A path (consisting of nodes and arcs) of OLP-graph from one end to the other represents a pattern. If two patterns have an overlap of specified length or more, then these two patterns share a common sub-path. A node of the graph represents a feature for a pattern.

Header table and node-links facilitate in finding the overlap. Header table consists of an entry for every feature. An entry in it points to a node in the graph which represents that feature for a pattern. The node-link of a node points to another node which represents the same feature but for some other pattern. That is, a Header Table entry for feature f_i is a pointer to the head of a linked list of nodes. Each node in this linked list represents feature f_i for some pattern. This linked list is called *feature-linked-list of* f_i.

A pattern is progressively inserted feature by feature. For every feature a possible overlap with the existing patterns in the OLP-graph is searched. If no overlap is possible then a new node is created for the feature.

3.1. Example

Let $\{(a,b,c,d,e)^T, (p,b,c,q,r)^T, (u,v,c,q,w)^T\}$ be the original set given for a class. Let the minimum overlap length required for the pattern synthesis be 2. The corresponding OLP-graph is shown in Figure 2. Node links which form *feature-linked-lists* are shown in dotted lines. A path consisting of nodes and arcs (shown in solid lines) from the left end to the right end represents a pattern that can be extracted from the OLP-graph. Thus the set of patterns that can be extracted is $\{(a,b,c,d,e)^T, (a,b,c,q,r)^T, (a,b,c,q,w)^T, (p,b,c,d,e)^T, (p,b,c,q,r)^T, (p,b,c,q,w)^T, (u,v,c,q,r)^T, (u,v,c,q,w)^T\}$.

An important point to observe is that first and second patterns in the original set have an overlap of 2 features *viz.,* f_2, f_3 and their values are b, c respectively. In the OLP-graph, the node corresponding to b is shared but that corresponding to c is not. The reason is that if node corresponding to c is also shared then $u \rightarrow v \rightarrow c \rightarrow d \rightarrow e$ becomes a valid path *i.e.,* $(u,v,c,d,e)^T$ becomes a synthetic pattern which can be extracted out of the graph. However,

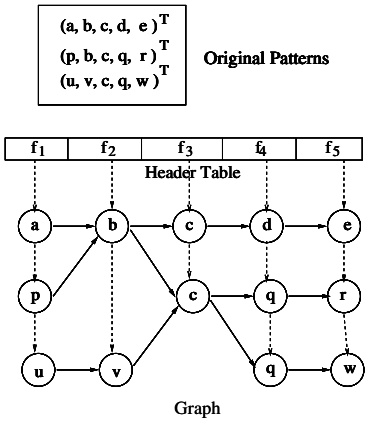


Figure 2. Illustration of OLP-graph.

this is not a valid synthetic pattern according to overlap pattern synthesis as it cannot be synthesized from the original set with overlap length 2. To make this point clearer, we define the property *node-sharability* along with *sharable-node* below.

Node-sharability and sharable-node: For a given OLP-graph G, pattern X, feature f_i and overlap length l, we say *node-sharability*(G, X, f_i, l) is true if there is a sub-path of nodes in G, $v_i \rightarrow v_{i+1} \rightarrow \cdots \rightarrow v_{i+l-1}$ such that feature value in v_j is equal to $X[f_j]$ for $(i \leq j \leq i + l - 1)$, and v_i is a node in the *feature-linked-list* of f_i. In this context *sharable-node*(G, X, f_i, l) is v_i.

Let G_0 be the initial empty OLP-graph and G_i be the OLP-graph after inserting the i^{th} pattern from the given original set. Then in the above example,
node-sharability$(G_1, (p, b, c, q, r)^T, f_2, 2)$ is true but
node-sharability$(G_1, (p, b, c, q, r)^T, f_3, 2)$ is false.

3.2. OLP-Graph Construction

An OLP-graph can be constructed for each class of patterns and for a given overlap length l as described in Algorithm 1.

The method iteratively adds patterns from the original set to the already constructed OLP-graph.

The functions *Add-to-feature-linked-list(v, f_i)* appends node v to the feature-linked-list of f_i, and *Add-arc(u, v)* adds an arc to the adjacency list of arcs of node u pointing to node v provided such an arc already does not exist in the adjacency list. The time complexity of the method is $O(n^2dl)$ where d is the dimensionality of patterns and l is the overlap length. Since d and l are constants the effective time complexity of the method is $O(n^2)$.

The space required by the method is largely due to the space occupied by the OLP-graph G. G consists of a Header Table and a graph. The space required for the Header Table is $O(d)$, and that for the graph is $O(nd)$. Since each original pattern (total of n patterns) occupies a path in G, a path is of size d. So the effective upper bound on space complexity is $O(n)$. However, empirical studies (Section 5) show that the actual space consumed by an OLP-graph is much smaller than that of the original patterns.

Algorithm 1 Build-OLP-graph($\mathcal{X}_j, l$)

```
{ Let G be the empty OLP-graph having Header Table with all
entries being empty. }
for each pattern X in X_j do
  Parent = NULL;
  for i = 1 to d do
    if node-sharability(G, X, f_i, l) is true then
      v = sharable-node(G, X, f_i, l);
      Add-arc(Parent, v);
    else
      v = Create a new node;
      v.feature-value = X[f_i];
      Add-to-feature-linked-list(v, f_i);
      Add-arc(Parent, v);
    end if
    Parent = v;
  end for
end for
Output(G);
```

4. NN Classification using Synthetic Patterns

Even though OLP-graph is a compact representation for the entire synthetic set that can be generated by overlap pattern synthesis, the conventional NN classifier with the entire synthetic set takes a large amount of time. The reason is that the synthetic set can be exponentially larger in size as compared to the original set which depends on the overlap length considered.

We propose a NN classifier called OLP-NNC which has classification time upper bound equal to $O(n)$ where n is the original training set size. OLP-NNC stores the partial distance computations in the nodes of the OLP-graph and avoids recomputing the same. This method is suitable for distance measures like Hamming distance, Squared Euclidean distance, etc., where the distance between two patterns can be found over its parts (called partial distance) and added up later to get the actual distance.

OLP-NNC first finds the distance between the test pattern and its nearest neighbor (NN) within a synthetic set of a given class (represented by an OLP-graph). The class label assigned to the test pattern then is that for which the NN has the least distance. Algorithms 2 and 3 describe finding distance of NN within a class. Distance measure used is Hamming distance, since binary valued features are assumed.

Algorithm 2 Find-Min-Dist(Graph G, Test Pattern T)

{Let *min-distance* be an integer initialized to maximum possible value. }
for (each node v in the feature-linked-list of f_1 in G) **do**
  d = Find-Dist$(v, T, 1)$;
  if ($d <$*min-distance*) **then**
    min-distance $= d$;
  end if
end for
return(*min-distance*);

Algorithm 3 Find-Dist(Node v, Test Pattern T, Integer i)

if (v is marked as visited) **then**
  return (v·*partial-distance*);
else
  $d = \mid T[f_i] -$ *v.feature-value* $\mid$;
  Find L = List of descendant nodes of v;
  if (L is not empty) **then**
    for (each node w in L) **do**
      $d = d +$ Find-Dist$(w, T, i+1)$;
      if ($d <$*min-distance*) **then**
        min-distance $= d$;
      end if
    end for
    v·*partial-distance* $=$ *min-distance*;
  else
    v·*partial-distance*$= d$;
  end if
  Mark v as visited;
  return(v·*partial-distance*);
end if

5. Experimental Results

We conducted experiments using handwritten digit data. The original training sample consists of 667 patterns for each class of digits labeled from 0 to 9, making it a total of 6670 patterns. The test data consists of 3333 patterns drawn independent to the training set. The dimensionality of each pattern is 192. We performed experiments with 2000(200 X 10), 4000(400 X 10) and 6670(667 X 10) training patterns respectively. Since the data is of the binary type we used Hamming distance as distance measure.

We compared our results (see Figure 3) with the conventional 1-NN classifier (NNC), k-NN classifier (k-NNC), and NNC based on the bootstrap technique given by Hamamoto *et al.,* [4] where each training pattern is replaced by weighted average of its r nearest neighbors within the same class. The parameters r, k (for k-NNC) and l (overlap length for OLP-NNC) are the best values based on 3-fold cross validation.

	Number of Training Patterns	Space Required (KB)	Design Time (Sec)	Classification Time (Sec)	Classification Accuracy (%)
NNC	2000	772	0	98	87.67
	4000	1544	0	175	90.16
	6670	2575	0	306	91.11
k–NNC	2000	772	0	106	87.79
	4000	1544	0	200	90.22
	6670	2575	0	329	92.68
NNC (Based on Hamamoto's Method)	2000	772	16	99	88.86
	4000	1544	34	172	90.84
	6670	2575	55	310	92.88
OLP–NNC	2000	311	6	91	92.44
	4000	473	10	145	92.89
	6670	629	22	205	93.85

Figure 3. Experimental Results

The results are in favor of OLP-NNC both in terms of space, classification time requirements and classification accuracy(CA). An important observation is that OLP-NNC with only 2000 training patterns is giving CA close to if not better than that of others with 6670 patterns. Moreover with identical number of training patterns it clearly outperforms the other methods.

6. Conclusion

Overlap pattern synthesis is a novel technique to generate artificial patterns, which can reduce the curse of dimensionality effect for high dimensional data. Overlap pattern graph (OLP-graph) is a very compact representation of the training set. OLP-NNC does implicit pattern synthesis to find the NN of a given test pattern. Empirical results with handwritten digit data establish its superiority. With suitable preprocessing of the data sets these techniques can be extended to other domains.

References

[1] B.Efron. Bootstrap methods: another look at the jackknife. *Annual Statistics*, 7:1–26, 1979.

[2] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[3] R. O. Duda, P. E.Hart, and D. G. Stork. *Pattern Classification*. A Wiley-interscience Publication, John Wiley & Sons, 2 edition, 2000.

[4] Y. Hamamoto, S. Uchimura, and S. Tomita. A bootstrap technique for nearest neighbor classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):73–79, 1997.