Syntactic Pattern Classification by Branch and Bound Search

Alan Y. Commike
Jonathan J. Hull

Department of Computer Science
State University of New York at Buffalo
Buffalo, NY, 14260

Abstract

A methodology for classifying syntactic patterns is proposed in this paper that performs a branch and bound search over a set of prototypes. The prototypes are first clustered hierarchically and the search is performed over the hierarchy. The proposed technique is applied to a pattern recognition system where images are described by the sequence of features extracted from the chain codes of their contours. A rotationally invariant string distance measure is defined that compares two feature strings. The methodology discussed in this paper is compared to a nearest neighbor classifier that uses 12,000 prototypes. The proposed technique decreases the time required to recognize a pattern by 93 percent and maintains a recognition rate of greater than 90 percent.

1 Introduction

Syntactic pattern recognition has been a topic of research interest for many years and has resulted in many successful applications such as character recognition [1] and digit recognition for postal codes [9]. A typical syntactic pattern recognition system first extracts a set of structural features from an input image. The features are then recognized by a set of rules. The rule that matches best provides the classification of the input image.

The advantages of this approach to pattern recognition are numerous. Most importantly, recognition can be fast as simple rules are often used. Sometimes a feature description is matched sequentially to rules and the first successful match provides the decision. This is a further advantage as the more frequently fired rules can be placed before the less frequently fired ones. Such speed considerations make this methodology ideal for real-time applications.

There are several disadvantages to implementing such a system. The most significant one is the development of the rule base. Typically, this is a time-consuming manual process in which a person writes a set of rules based on his or her experience and intuition. The rules are then used to recognize a set of training examples. The errors in recognition are inspected and the rule base is tuned to eliminate them. This process is repeated until a desired level of performance is achieved. Because of the complex interactions that can occur between rules, many person-months may be required. A further significant disadvantage of this method for rule base design is the difficulty of re-training on new data. The simple operation of adding a few new classes may require that the complete rule development process be repeated.

A non-parametric method that is relatively simple to train and often used to recognize statistical feature vectors is a nearest neighbor classifier. This technique uses a distance measure to compare an unknown pattern to a set of pre-classified training data. The decision of the classifier is the class of the training sample with the minimum distance to the unknown. A significant disadvantage of a nearest neighbor classifier is that the distance must be calculated between an unknown and every prototype each time a sample is recognized. This implies that the time requirement grows linearly with the number of training data. Thus, the overall recognition time can be impractical if thousands of prototypes are used.

The execution time of a statistical nearest neighbor classifier has been improved by first hierarchically clustering feature descriptions that are separated by small distances. A branch and bound search is then executed on the cluster hierarchy to recognize an unknown pattern [5]. This limits the number of comparisons that are performed. This methodology is successful when the clusters do not overlap one another and when there is a natural meaning to the radius of a

CH2983-5/91/0000/0432/$01.00 © 1991 IEEE

cluster. That is, the center and the radius (maximum distance from the center to any training sample) form a multi-dimensional sphere such that if the distance from the center to an unknown is greater than the radius, the unknown is outside the sphere. Thus, it is not necessary to compare the unknown to any of the samples within the cluster. This is true if the distance measure is symmetric, i.e., the distance from A to B is equal to the distance from B to A.

This paper proposes to adapt the hierarchical clustering and branch and bound search techniques to syntactic pattern classification. A distance measure between syntactic patterns is defined, based on the Levenshtein metric [7, 4]. A modification is introduced that allows for rotational differences between two patterns. This modified distance measure is used to construct a cluster hierarchy. The distance measure is also used in a branch and bound search over the hierarchy. Even though the distance measure is non-symmetric, it still allows for the construction of a hierarchy that adequately represents the data. It will be shown that a significant decrease in computation time is achieved at a minimal cost in recognition performance.

The rest of this paper discusses the proposed algorithm. The distance measure and the clustering algorithms are defined and experimental results are presented.

2 Algorithm Description

The algorithm contains two basic steps. A cluster hierarchy is first generated from the syntactic feature descriptions of a set of training data. An unknown pattern is classified by extracting its syntactic feature string and performing a branch and bound search over the cluster hierarchy.

2.1 Feature Extraction

A syntactic description of an image consists of structural feature primitives that describe different local variations. Figure 1a shows an example of a set of structural feature primitives that are defined by the curvature around the contour of a binary image.

An example of each feature is also shown. Eight contour primitives are given. Each of these is further described by one of 16 locations and one of eight directions. Figure 1b shows an example of the digit "2". Specific features detected around its contour are indicated by the letters "a" through "k". The full structural description of these features is shown in Figure 1c.

2.2 Clustering

A clustering process is applied over the training set to locate the subclasses that occur in each class. Each sample in the training set is associated with a truth value or *class*. Within each class, a large variation between images is exhibited. Choosing one image as a representative for an entire class is difficult to do without over-generalizing. A solution is to further subdivide each class into its subclasses. A representative for each subclass can then be chosen more easily. A class can then be represented by prototypes from the subclasses.

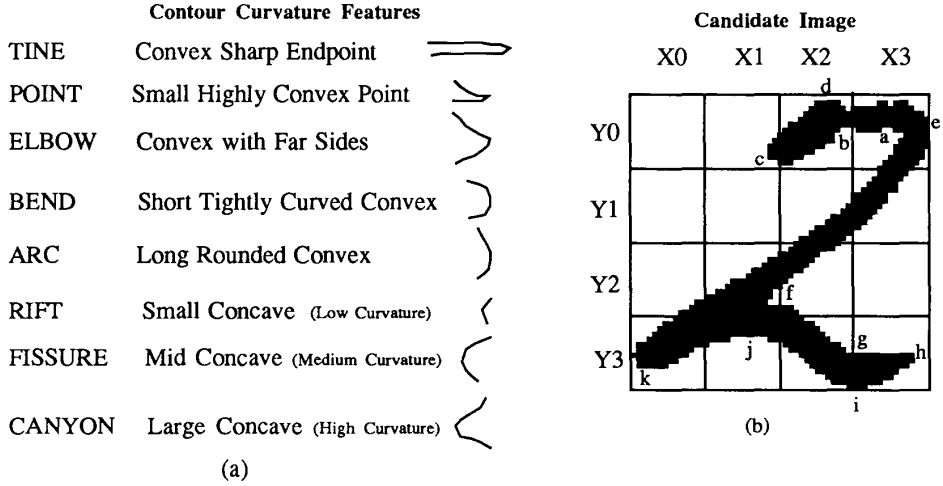
An agglomerative hierarchical clustering algorithm finds the subclasses in each class [2, 3]. All the data are separated into a small number of broad classes at the top of the hierarchy. These broad classes are further divided into smaller groups, and so on, until the terminal level of the hierarchy is reached. Individual training samples are stored here. Agglomerative techniques construct the hierarchy by successively merging samples and groups of samples together in a bottom-up fashion. Different levels of the hierarchy thus represent different amounts of blurring of the data.

The merging that occurs during clustering is a method of generalization. The effect of successive merges is to discard variations due to noise introduced by digitization, pen fluctuations, and writer inconsistencies, leaving only the essential descriptive features of a group of images. As the top of the cluster hierarchy is reached, more of the inconsequential features are removed.

The clustering algorithm first computes a similarity measure between each sample and every other sample. Each sample is then merged with the sample it is closest to, i.e., the sample that minimized the similarity measure. The centers of the groups are then computed and the algorithm is re-applied to the centers. This process is repeated until the number of centers stabilizes. The resulting hierarchy and the samples it contains are stored for later use.

2.2.1 Similarity Measure

The similarity measure is based on a modified weighted Levenshtein distance [7]. The Levenshtein distance is defined as the minimum number of edit operations needed to transform one syntactic string into another. The edit operations are: insertion, deletion, and substitution of one feature for another. An additional edit operation is added to make feature strings invariant to rotation. The algorithm for computing the similarity measure is stated below. It is based on a dynamic pro-



Features Extracted

a = Canyon @ West (x3, y0)
b = Rift @ South (x2, y0)
c = Tine @ West (x1, y0)
d = Arc @ North West (x2, y0)
e = Bend @ North East (x3, y0)
f = Canyon @ East (x1, y2)
g = Rift @ North (x3, y3)
h = Tine @ East (x3, y3)
i = Arc @ South (x2, y3)
j = Fissure, @ South (x1, y3)
k = Tine @ South West (x0, y3)

(c)

Figure 1: Feature Primitives

gramming technique and not only returns a numerical value for the distance between two feature strings, but also a trace of the edit operations needed to convert one string into the other with minimal cost [8, 10].

The minimal cost D of transforming string $A = a_1, a_2, \ldots, a_n$ into string $B = b_1, b_2, \ldots, b_m$ can be computed be the following recurrence equation for $D(n, m)$.

$$\min \begin{cases} D(n, m-1) + C(\text{“ ”}, b_m) & \text{insert } b_m \\ D(n-1, m) + C(a_n, \text{“ ”}) & \text{delete } a_n \\ D(n-1, m-1) + C(a_n, b_m) & \text{subst } a_n \text{ for } b_m \end{cases}$$

where $D(i, 0) = C(a_i, \text{“ ”})$ for all i, $0 \leq i \leq n$, and $D(0, j) = C(\text{“ ”}, b_j)$ for all j, $0 \leq j \leq m$.

The cost matrix C is used to weight each of the edit operations. Insertion, deletions, and substitutions are given by $C(\text{“ ”}, a_i)$, $C(a_i, \text{“ ”})$, and $C(a_i, b_j)$, respectively. Where a_i and b_j denote the primitives defined in the feature extraction step.

Rotation invariance is accomplished by a rotation of one of the feature strings. Therefore the overall similarity measure is given by the minimum of $D(n_k, m)$ for $k = 1, ..., n$, where n_k is the k^{th} circular rotation of string A.

2.2.2 Definition of Cost Matrix

The cost matrix adjusts the similarity measure to individual feature sets and initiates the appropriate merging during clustering. The objective of clustering is to generalize images into subclasses. The essential features for each subclass need to be inferred from the training samples. When two similar feature strings are compared, the excess information due to writer variation is removed. This leaves only those features that describe the particular image subclasses. This is accomplished by adjusting the values in the cost matrix. An example of this is making the cost of deleting a feature less than the cost of inserting it. This guarantees that clusters are formed by discarding excess features.

It must be noted that by using a weighted cost matrix where the cost of inserting a symbol may be different than the cost of deleting it, the similarity measure may not satisfy the basic axioms for a metric. Symmetry, where $D(i, q) = D(q, i)$, and the triangular inequality, where $D(i, q) \leq D(i, p) + D(p, q)$, do not necessarily hold. The implementation of the clustering algorithm has been designed to compensate for this.

2.2.3 Cluster Agglomeration

Cluster agglomeration is the merging of sample data into clusters. To merge two feature strings, the edit operations that were done in the calculation of the distance are performed on the strings. First the indicated rotation is performed to align the feature strings. The combination of insertions, deletions, and substitutions are then done, creating a composite string that represents the two original strings. Substitutions are performed by combining the two features in a fashion similar to a logical *or* operation.

After multiple merges of similar features strings, the agglomerative feature string may contain all the information associated with the strings that have been involved with the merging. It also can represent other images that were not merged. Shown in Figure 2a are three images with their individual and agglomerative feature strings. Also shown in Figure 2b is an image that was not involved in the merging operation but is represented by the agglomerative string.

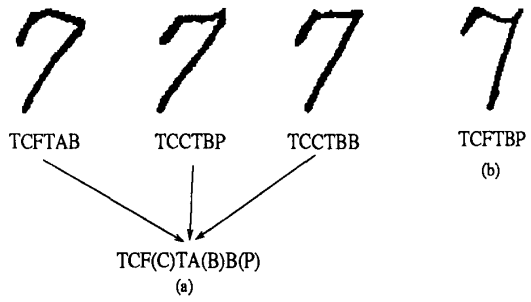


Figure 2: Agglomeration of Feature Strings

Shown in Figure 3b is a section of a cluster hierarchy. The images superimposed on clusters are representative of the corresponding subclasses. Figure 3a is a candidate image that is to be classified by searching the cluster hierarchy. As can be seen, the input image is very similar to two of the classes in the rule base, namely the twos and the sevens. In this case, a traversal of the cluster hierarchy is needed to correctly classify the candidate image as a seven.

3 Implementation

This section describes details of the implementation of the techniques discussed earlier. The system has been trained and tested on unconstrained handwritten digits, but can be extended to other image sets. All training and testing data are taken from handwritten digits that occurred in ZIP Codes taken from a sample of images of live mail [6]. Thus, the people that prepared the data did not know it would be used for this purpose.

3.1 Feature Extraction

The conversion from pixel images to syntactic feature strings is accomplished in three steps: chain code extraction, curvature calculation, and feature extraction. The chain code is extracted from both the inner and outer contour traces of each image. Directly from the chain code, a curvature value is calculated at each point along the contours. Noise introduced by the digitization process, pen fluctuations, and slight writer inconsistencies are removed by a smoothing function incorporated in the curvature calculation.

The eight contour features shown in Figure 1a are defined by the degree of curvature at each point. Concave regions are assigned negative curvature values and convex regions are assigned positive curvature values. The feature set is split into three concave and five convex features. Each contour feature also has a direction and a location. Direction is quantized to eight compass points and location is defined by superimposing a 4 x 4 cartesian grid with the origin in the upper left corner, onto the bounding box of the image [1].

3.2 Clustering

The clustering process is done by first creating an $N \times N$ similarity matrix for all distinct pairings of the N samples in the training data. Each row r_i, $1 \leq i \leq N$ of the matrix, is scanned, one at a time. The minimum distance $d_m = D(r_{i1}, r_{ij})$, $2 \leq j \leq N$, $j \neq i$ is calculated for row r_i. The list of clusters that have been formed are also scanned, the list is initially empty. The minimum distance $d_c = D(r_{i1}, c_k)$, $1 \leq k \leq C$, where C is the number of clusters created so far, are calculated. If $d_m < d_c$ then a new cluster is formed with the two children being d_{i1} and r_{ij}, otherwise d_{i1} can be added to cluster c_k.

Before a new cluster is formed, checks are made to insure that the image chosen to merge into the candidate image does not fit better elsewhere. In every data set, there are always outliers present. Merging the best fit images into the outliers prevents those images from being clustered into a more appropriate cluster later in the process. Once r_{ij} is chosen, $d_{ck} = \min D(r_{ij}, r_{il})$, where $1 \leq l \leq N$, $i \neq N$, $l \neq j$, the minimum distance feature string to r_{ij}, is calculated. It would be expected that $r_{i1} = r_{il}$. In the case where $r_{i1} \neq r_{il}$, r_{i1}

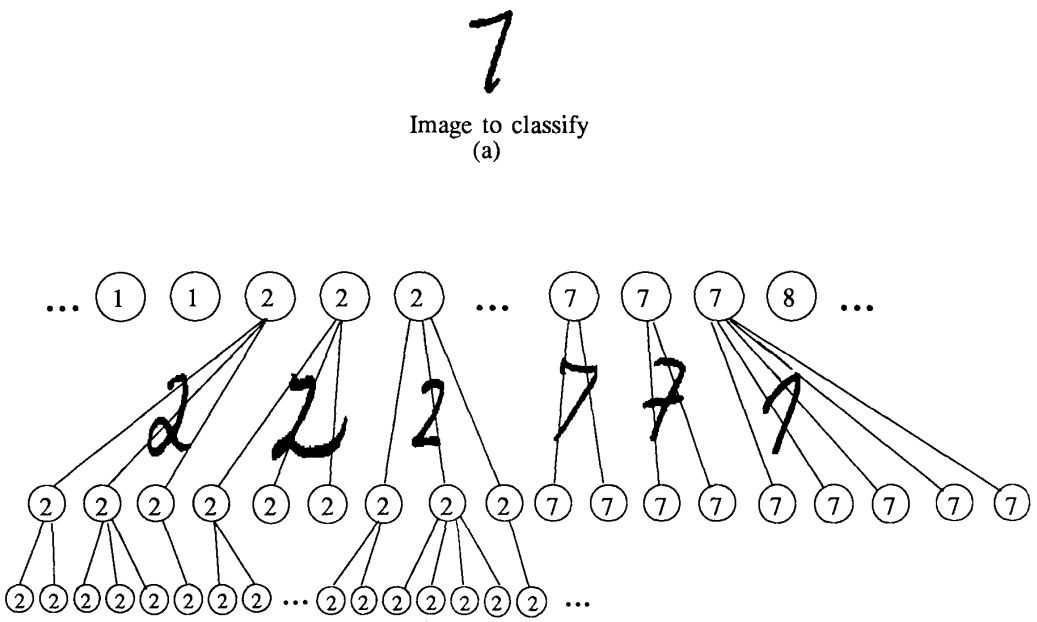


Figure 3: A portion of a cluster hierarchy

is probably an outlier and should not be part of this cluster.

Once the first level of clusters is created, a similar process is followed to create the rest of the hierarchy. A similarity matrix for level n of the cluster hierarchy is created. All the cluster centers are computed and level $n+1$ of the hierarchy is created. Each cluster is now an agglomeration of more specific cluster centers. No information is lost, so each cluster can be expanded into its constituent parts, all the way to the original feature strings.

4 Recognition Results

4.1 Nearest Neighbor

A nearest neighbor classifier was constructed to test the performance of the similarity measure. A training set of 12,000 handwritten digits and a testing set of 2711 handwritten digits were used. The nearest neighbor classifier achieved a correct rate of 94% with a substitution rate of 6%. No provision was made for rejects. The execution time on a SUN-4 workstation was about 300 CPU seconds per digit because every unknown required 12,000 comparisons each time it was recognized.

This performance is consistent with other classifiers trained and tested on the same data sets. Error cases were often due to segmentation deficiencies and true ambiguities. For example, in some cases, different people asked to identify the same digit would classify it differently. There is an arbitrary judgment as to when, for example, a zero becomes a six, a six a nine, or when a seven becomes a two.

4.2 Branch and Bound of Cluster Hierarchy

To perform recognition over all classes, a branch and bound search is initiated over the cluster hierarchies

for each class. The clusters at the top of the hierarchy are compared to the unknown, using the similarity measure. This is done by computing the distance between the unknown and the agglomerative string that represents the cluster. The cluster that yields the minimum distance is expanded and the descendants are placed on a queue. At each iteration, the queue is sorted by the distance to the unknown and the cluster center with the minimum distance is expanded. This is continued until one of the original training samples is reached that has the lowest cost.

The performance of this method is comparable to the nearest neighbor algorithm. A 90 percent recognition rate is achieved. Classification of an unknown is done in 168 comparisons on the average with an average running time of 20 seconds per digit. This is a reduction of 93 percent over the nearest neighbor classifier.

5 Conclusions and Remarks

A methodology for recognizing syntactic patterns that retains nearly the complete power of a nearest neighbor classifier but requires significantly less computation time was presented. A hierarchical clustering algorithm was used to generate a tree of clusters and a branch and bound search was executed on the tree. Experimental results showed that over a 93 percent reduction in computation time was achieved and a correct recognition rate of greater than 90 percent was maintained.

Future work includes an investigation of the generation of a rule base from the cluster hierarchy. The rule base would retain the information of the hierarchy. However, it would be searched by rule matching operations that are much more computationally efficient than a string distance measure. Preliminary work currently under way has demonstrated the viability of this approach.

References

[1] D. D'Amato, L. Pintsov, H. Koay, D. Stone, J. Tan, K. Tuttle, and D. Buck. High speed pattern recognition system for alphanumeric handprinted characters. In *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pages 165–170, Las Vegas, Nevada, June 1982.

[2] E. Diday and J.C. Simon. Cluster analysis. In K.S Fu, editor, *Digital Pattern Recognition*. Springer-Velag, second edition, 1980.

[3] B. Everitt. *Cluster analysis*. Halsted Press, second edition, 1981.

[4] K.S. Fu and S.Y. Lu. A clustering procedure for syntactic patterns. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7(10):734–742, October 1977.

[5] K. Fukunaga and P.M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, pages 750–753, July 1975.

[6] J.J. Hull, A.Y. Commike, and T.K. Ho. Multiple algorithms for handwritten character recognition. In *International Workshop on Frontiers in Handwritting Recognition*, pages 117–130, Montreal, Canada, April 1990.

[7] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966.

[8] D. Sankoff and J.B. Kruskal, editors. *Time warps, string edits and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley Publishing Company, 1983.

[9] L. Stringa. Lcd: A formal language for constraint-free hand-printed character recognition. In *Proceedings of the Fourth International Conference on Pattern Recognition*, pages 354–358, Kyoto, Japan, November 1978.

[10] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, January 1974.

Acknowledgment

This work was supported by the Office of Advanced Technology of the United States Postal Service.