

Recognition of Hand-Printed Characters Using Induct Machine Learning

Adnan Amin, Aba Rajithan and Paul Compton

School of Computer Science and Engineering
University of New South Wales
2052 Sydney, Australia
(Email: amin@cse.unsw.edu.au)

Abstract. A goal of character recognition is to simplify and automate the development of character recognition algorithms. We describe here an approach based on applying preprocessing to data sets of characters and then applying machine learning to the data sets to build a knowledge base able to classify unseen preprocessed characters. The machine learning method, Induct/RDR, has a number of features which make it particularly suitable for character recognition. It also has the potential to integrate with a manual knowledge acquisition methodology if further refinement is required. Initial results on hand-printed Latin characters show an accuracy of 84% on unseen cases for the machine learning system alone.

1 Introduction

For the past three decades there has been increasing interest among researchers in problems related to machine simulation of the human reading process. Intensive research has been carried out in this area with a large number of technical papers and reports in the literature devoted to character recognition. This subject has attracted immense research interest not only because of the very challenging nature of the problem, but also because it provides, the means for automatic processing of large volumes of data in postal code reading [11, 16], office automation [6, 15], and other business and scientific applications [13, 1, 10].

The different approaches covered under the general term character recognition fall into either the on-line or off-line category, each having its own hardware and recognition algorithms. This paper presents a new technique for the recognition of hand-printed Latin characters using Induct/Ripple Down Rules (RDR) [9].

Conventional methods have relied on hand-constructed dictionaries which are tedious to construct and difficult to make tolerant to variation in writing styles. Induct/RDR uses learning by induction to build the knowledge base. An advantage of Induct over other machine learning methods is that it produces a very compact representation [2] and that it integrates with a knowledge acquisition methodology Ripple Down Rules which allows knowledge bases to be Error! Bookmark not defined.uilt and maintained by an expert without the support

of a knowledge engineer [3] Characters are scanned into the computer and pre-processing techniques transform the bit-map representation of the characters into set of primitives such as lines, curves and loops which can be represented in an attribute base form. A set of such representation for each character is then input to Induct/RDR which produces a decision tree for classifying each character..

2 Digitization and Preprocessing

2.1 Digitization

The first phase in our character recognition system is digitization. Documents to be processed are first scanned and digitized. A 300 dpi scanner is used to digitize the image. This generates a TIFF file which is then converted to 1-bit plane PBM file. The PBM format contains a small header which incorporates a file stamp followed by the dimensions of the image in pixels. The remainder of the file contains the image data.

2.2 Pre-thinning and Thinning

This step aims to reduce the noise that the binarization process yields. The pre-thinning algorithm used in this paper as shown in Figure 1.

Input: A digitized image I in PBM format.
Output: A pre-thinned image I' , also in PBM format.
Method:

1. For each pixel P in image I , let $P_0, P_1, P_2, P_3, P_4, P_5, P_6$ and P_7 be its 8 neighbors, starting from the east neighbor and counted in an anti-clockwise fashion.
2. Let $B(P) = P_0 + P_2 + P_4 + P_6$.
 Let P' be the corresponding pixel of P in.
3. If $B(P) < 2$ then set P' to white
 Else If $B(P) > 2$ then set P' to black
 Else set P' to the value of P ;

Fig. 1. Pre-thinning algorithm.

The thinning algorithm adopted in this paper is a parallel method based on Jang and Chin's algorithm [12].

3 Feature Extraction

3.1 Binary Tree Construction

An algorithm implementing a 3x3 window is used to trace along the path of the skeleton, recording the structural information of the trace path. A path is described as a tracing between junction or end point, where an end point has a single neighbor and a junction point has two neighbors as shown in Figure 2.

This path is stored in a node of the binary tree: where a choice of path to trace exists, a left and right node are formed beneath the current one and their respective paths traced out. A priority system is used which favors certain directions over others (without this, the window would trace the skeleton in random direction).

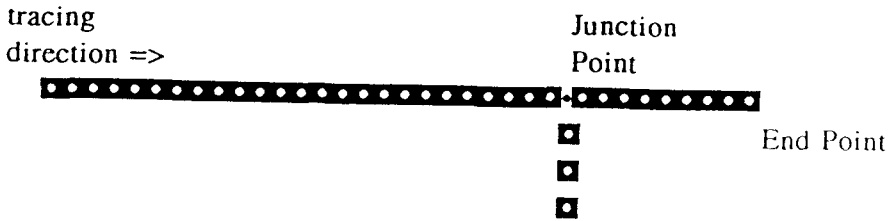


Fig. 2. Tracing path.

The starting point for tracing the skeleton is based on several criteria. The image is divided into three horizontal regions and the top and bottom region are searched for end points or junction points. This ensures that the starting point does not split a path into two subpaths. If no such points are found, as with the letter "O", the left most pixel of the image is used as starting point.

3.2 Structural information

The structural information for each path traced is saved as follows:

1. Freeman Code [7] chain: an 8-directional code describing the tracing of the path.
2. Frame: co-ordinates describing a minimal-size frame which contains the path used to calculate size of loops in the image (e.g. "a" = small, "D" = large) as well as approximate center.

3. Positional: co-ordinates describing the start and ending points of the path used to determine positional relationship between loops and between loops and touching path (e.g. "b" = left touching path, "d" = right touching path).
4. Loop: pointer indicating paths joining previously explored section (e.g. "e", "6")

The completed tracing results in the segmentation of the character into paths or strokes which will be formed into primitives.

3.3 Smoothing

Upon completion of the binary tree, a smoothing step allows redundancies and noise to be removed from the tree, a smoothing of the binary tree is designed to minimize the number of nodes in the tree and minimize the Freeman code chain. Loops whose paths contain multiple nodes are identified and then compressed to single node. Redundancies in the Freeman code are smoothed and noise is reduced. At points of change in the Freeman code, we use simple, but effective, smoothing algorithm, illustrated in Figure 3 for direction 0 only. The patterns and codes are rotated for other directions. Any pattern in the first column is replaced by the pattern in the second column. Thus the string 070010 is replaced by 000000.

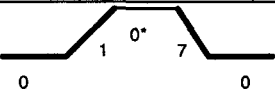
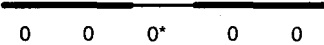
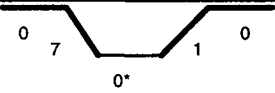
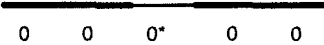
| Pattern | Replaced by |
|---|---|
|  |  |
|  |  |

Fig. 3. Smoothing algorithm.

3.4 Primitives

The structure information in the binary tree allows the formation of pattern primitives, or sub-pattern, which are used to describe the original image. There are two main primitives described in this system: straight lines and curves. A path may be described by a single primitive or by multiple primitives. The structure information in the tree is converted to these primitives using the following definition.

Breakpoint (Separator): divides a path into sub-paths more easily described by primitives. A breakpoint has at least one of two possible conditions:

- inflection point: a change in curvature, a positive (clockwise) curve followed by a negative (anti-clockwise) curve, or vice versa.
- cusp point: a sharp change in direction, two segments form an acute angle $\leq 90^\circ$.

Straight line: has its usual geometric definition as two points in sequence within a path. A point in a Freeman chain can be defined as a change in the Freeman code. Lines can be distinguished from curves in two ways: the length of a line segment is significant in comparison to the length of the path, or the path contains only two points.

Curve: these are formed by at least three segments of nearly equal length (usually small in relation to the length of the path) with no breakpoint. Two types of curves: Open and Closed.

- Open: there are four open curves useful in describing Latin characters. They face the four main points of the compass, eg. U, S, C.
- Closed (Loop): described by three sized (small, medium, and large) and also includes the double loop. e.g. a, R, D, 8.

Figure 4 shows the primitives used in this system.

Primitives

| Line | → | ↗ | ↑ | ↖ | ← | ↙ | ↓ | ↘ |
|----------------|----|----|----|----|---|----|---|----|
| | E | NE | N | NW | W | SW | S | SE |
| Curve (open) | ⌒ | ⌒ | ⌒ | ⌒ | | | | |
| | CN | CE | CS | CW | | | | |
| Curve (closed) | ° | o | O | OO | | | | |
| | LS | LM | LL | DL | | | | |

Fig. 4. Primitive features used in this system.

4 Classification using Induct/RDR

A data set of preprocessed Latin characters is passed to the Induct/RDR machine learning algorithm [9]. This produces a knowledge base which is then used with an interpreter to return a classification for an unseen character. The unseen characters go through the same preprocessing as the training data.

4.1 Background

The most common approach to machine learning for classification tasks, as exemplified in C4.5 [14], is to attempt to build a decision tree where each node represents an attribute and there is a branch for each value of the attribute. There are a variety of heuristics for deciding on which is the most useful attribute to add as a node at any point. The aim of the attribute selection is to produce a tree which is highly accurate on unseen cases and is normally fairly compact. C4.5 uses an entropy/information based algorithm which essentially attempts to find the most important attribute in separating the population into different classes at any given point in the decision tree. The information measure maximizes overall separation of the population into classes, irrespective of the separation between branches of members of individual classes

Induct also produces a decision tree but the tree has a different structure and is built quite differently [9, 8]. With Induct the most common classification is identified and then an attribute value pair is chosen as a selector for this classification and then further attribute value pairs conjoined to this to produce a rule for the classification. The attribute value pair is chosen by calculating the probability of the particular group of cases that it selects being chosen by chance. The attribute value pair which gives a result most different from a random chance is chosen. This process is repeated recursively for all cases selected by the rule and all cases not selected. The result is binary tree with a rule at each node providing a classification; if a further rule is satisfied the classification is replaced by the later classification. As Catlett has pointed out, the resulting tree is normally very unbalanced and is probably better described as a decision list with exceptions which are further decision lists [2]. Induct/RDR also provide a very compact representation [9, 2] and because of the way the probabilities are calculated is particularly suited for missing or noisy data [8]. Because the knowledge base is built by dealing with a single classification at a time, a character recognition knowledge base should tend to have good general rules for each character, but with exception rules where a general rule may encompass more than one character. These features of Induct/RDR would seem to suggest that it is a particularly suitable machine learning method for handwriting recognition.

A requirement in any induction is appropriate preparation of the training data. Appropriate selection of attributes for Latin characters and appropriate examples are a key part of the process. The rules produced should perform correctly for items in the training set and in addition should accurately classify unseen Latin characters. The suitability of the training set for inductive methods does not depend just on the number of training cases but on how well prepared the cases are. The cases have to be accurately classified and the appropriate features extracted. The best results are obtained on large training sets with thousands of cases.

4.2 Induction Using Induct/RDR

Induct/RDR takes as input a training set entered as a file of ordered sets of primitives values, each terminated by a comma, and uses induction techniques

to produce a set of rules in the form of a binary tree with IF TRUE and IF FALSE branches. Thirty three primitives were chosen to describe the character as shown in Figure 5.

LINES

- | | |
|--------------------------------|--------------------------------|
| 1. small vertical Line (svl) | 4. small horizontal line (shl) |
| 2. medium vertical Line (mvl) | 5. medium horizontal line(mhl) |
| 3. large vertical Line (lvl) | 6. large horizontal line (lhl) |
| 7. small backslash line (sbl) | 10. small slash line (ssl) |
| 8. medium backslash line (mb1) | 11. medium slash line (msl) |
| 9. large backslash line (lb1) | 12. large slash line (lsl) |

CURVES

- | | |
|------------------------------|------------------------------|
| 13. small north curve (snc) | 16. small south curve (ssc) |
| 14. medium north curve (mnc) | 17. medium south curve (msc) |
| 15. large north curve (lnc) | 18. large south curve (lsc) |
| 19. small east curve (sec) | 22. small west curve (swc) |
| 20. medium east curve (mec) | 23. medium west curve (mwc) |
| 21. large east curve (lec) | 24. large west curve (lwc) |

LOOPS

- | | |
|--------------------------------|-----------------------------|
| 25. small loop preceding (slp) | 28. small loop after (sla) |
| 26. medium loop preceding(mlp) | 29. medium loop after (mla) |
| 27. large loop preceding (llp) | 30. large loop after (lla) |
| 31. small loop (sl) | 32. medium loop (mp) |
| 33. large loop (lp) | |

Fig. 5. The primitives to describe the characters.

5 Experimental Results and conclusion

The technique which has been adopted for this study is combines a purely structural method (based on structural primitives such as curves, lines,etc. in a similar manner to which human beings describe characters geometrically) and a classification test using Induct machine learning. In addition, this approach is efficient for feature extraction and recognition.

The system has been tested by using 15 alphabets in the learning stage to generate a decision tree. We then attempted to recognize characters from

seven unseen alphabets using the constructed decision tree and the rate of the recognition is 84% promising result and clearly shows that Induct machine learning technique is well suited to this type of application.

One of the advantages of using the Induct/RDR method is that not only can the performance be improved by adding further cases to the training data, but potentially rules can be added by hand for any cases that have been misclassified. The RDR knowledge acquisition methodology depends on the same type of knowledge base structure as produced by Induct/RDR to facilitate manual knowledge acquisition [3]. In fact using this technique a knowledge engineer is not required apart from initial pre-processing rules. In a medical domain an expert has used to RDR to build a 2000 rule expert system which is in routine use interpreting pathology reports, without support from a knowledge engineer [4]. This feature is of particular importance in the character recognition domain, because in fact in the cases misclassified it is generally clear to a non expert human what the actual character should be. In some cases however, it is not simply a matter of identifying features in the preprocessed characters, but of adding further features which are obvious to the human as needing to be included. In further work we intend to develop an appropriate interface for both rules and features to be added, not so much by a hand writing analysis expert, but by anybody who can read the writing. The final system would use the advantages of induction to rapidly build a fairly substantial knowledge base but then the advantages of individual refinements added by hand. The major challenge in this work will be in adding further features, however it has already been demonstrated in a medical domain that this can be achieved [5].

References

1. A. Amin and S. Al-Fedaghi. Machine recognition of printed Arabic text utilising a natural language morphology. *Int. Journal of Man-Machine Studies*, (6):769-788, 1991.
2. J. Catlett. Ripple down rules as a mediating representation in interactive induction. In *2nd Japanese knowledge acquisition for knowledge-based system workshop*, pages 155-170, 1992.
3. P. Compton and R. Janson. A philosophical basis for knowledge acquisition. *Knowledge acquisition*, (2):241-257, 1990.
4. G. Edwards et. al. Peirs: a pathologist maintained expert system for the interpretation of chemical pathology reports. *Pathology*, (25):27-34, 1993.
5. G. Edwards et. al. An expert system for time course data with expert-managed refinement in context. In *8th Australian Joint Conference on Artificial Intelligence*, pages 586-593, 1995.
6. L. Focht and A. Burger. A numeric script recognition processor for postal zip code application. In *Int. Conf. Cybernetics and Society*, pages 486-492, 1976.
7. H. Freeman. On the encoding of arbitrary geometric configurations. *IEEE Trans. Electronic Computer EC-10*, (10):260-268, 1968.
8. B. Graines. The trade-off between knowledge and data in knowledge acquisition. In *In G. Piatetsky-Shapiro and W. Frawley Knowledge Discovery in Databases Cambridge, MA MIT Press*, pages 491-505, 1991.

9. B. Graines and P. Compton. Induction of ripple down rules. In *5th Australian Conf. on Artificial Intelligence*, pages 349–354, 1992.
10. D. Guillevic and C. Suen. A fast reader scheme. In *2nd Int. Conf. on Document Analysis and Recognition*, pages 311–314, 1993.
11. L. Harmon. Automatic recognition of printed and script. *Proc. IEEE*, (60):1165–1177, 1972.
12. B. K. Jang and R. Janson. One pass parallel thinning: analysis, properties and quantitative evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, (11):1129–1140, 1993.
13. R. Plamondon and R. Baron. On-line recognition of handprint schematic pseudocode for automatic fortran code generator. In *8th Int. Conf. on Pattern Recognition*, pages 741–745, 1986.
14. J. Quinlan. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufman, 1993.
15. J. Schuermann. Reading machines. In *6th Int. Conf. on Pattern Recognition*, pages 1031–1044, 1982.
16. A. Spanjersberg. Experiments with automatic input of handwritten numerical data into a large administrative system. *IEEE Trans. Man and Cybernetics*, (4):286–288, 1978.