Segmenting handwritten mathematical expressions

Scott MacLean <smaclean@uwaterloo.ca>, George Labahn <glabahn@cs.uwaterloo.ca>

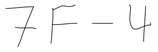
Symbolic Computation Group, University of Waterloo

Abstract

We are interested in recognizing handwritten mathematical expressions as part of the MathBrush project. A crucial component of any system for recognizing handwritten input is the segmentation of input into distinct symbols. We describe our experiments with segmentation and how our approaches are combined with symbol classification techniques.

Introduction

- A stroke is a time-ordered sequence of (x,y)-coordinates.
- Input to the segmentation algorithm is a sequence of strokes. Our recognition algorithm iterates through this sequence.
- Goal: find the number of strokes comprising the next symbol in the input.
- Segmentation is vital for a system which allows users to write naturally, rather than requiring them to use templates, pause between symbols, etc.
- Common approaches use symbol recognition with dictionary lookup instead of explicit segmentation [Nat95] or require the user to write strokes in a particular order [Smi99].
- With correct segmentation, symbol recognition is very accurate; with incorrect segmentation, recognition is almost always wrong. (The figure below shows a possible recognition given incorrect segmentation.)



"7-1=-L1"?

Approach

- Preprocessing: input strokes are joined together if one appears to be the continuation of another (eg. extension of fraction bars or root signs).
- Strokes are extracted into a list and sorted so that a stroke's successor is the stroke nearest to it not already extracted.
- Two components:
  - Heuristic method guesses how many strokes comprise the next symbol using stroke geometry.
  - Feature-correlation method extracts stroke features and resolves confusion between symbols to approximate a discrete probability distribution of segmentation possibilities.

Heuristic method

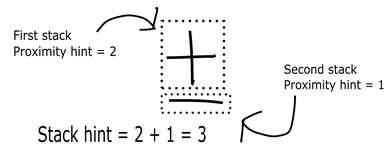
- Examines stroke geometry and produces a *proximity hint* and a *stack hint*, each with a confidence value of Low, Medium, or High.

Proximity hint

- Hint value is the number of consecutive strokes determined to be close to one another.
- Hint confidence is the average of local confidences computed at each such stroke.
- The first stroke is always included. A stroke S is included if at least one of the following applies:
  - The distance between S and a stroke already included is less than a small threshold; local confidence is High or Medium depending upon how close the strokes are.
  - The distance between S and a stroke already included is less than a larger threshold and the bounding boxes of those strokes obey certain constraints (two sides of the bounding boxes must be nearly overlapping, as in "k", or one side must be nearly overlapping and one stroke must be thin and placed close to the horizontal or vertical center of the other, as in "T"); local confidence is Low.
  - S is mostly contained within the bounding box of a stroke or group of strokes already included; local confidence is dependent on the ratio of bounding box overlap to total bounding box size.

Stack hint

- Simple distance thresholds fail for i, j, $=$, $\leq$, etc.
- Many math symbols consist of other symbols arranged in vertical stacks ($\leq$, $\equiv$, $\pm$, ...).
- Proximity hints are used to identify candidate stroke groups; these are considered to be stacked if:
  - their horizontal profiles overlap by at least 50% of the larger box's width, and
  - the vertical distance between the groups is sufficiently small.
- Hint value is the sum of proximity hints for all stacked groups.
- Hint confidence is the average of proximity hint confidences and confidences associated with each pair of stacked groups (typically Medium; High if horizontal profiles have very significant overlap).
- Dots are not required to exhibit significant overlap to be considered stacked but must be within a distance threshold.



Feature-correlation method

- Deficiencies of the heuristic method:
  - The heuristics are independent of the symbol database.
  - The hints are single values. If they are wrong, recognition will likely be incorrect.
  - It is not always correct to group strokes together if one is inside the other's bounding box (eg. square roots).
- The feature-correlation method extracts feature vectors from input and prototype symbols and performs rough classification.
  - Features include minimum x- and y-coordinate, width, height, first and last points, arclength.
  - Match score is the 1-norm of the difference between input and prototype vectors.
  - The smallest score for each number of strokes is recorded. These are normalized to sum to one to approximate a discrete probability distribution.
- Unfortunately, features of symbols with few strokes typically match database symbols more closely than those of symbols with many strokes, resulting in oversegmentation (as in the example in the introduction).
- But many multi-stroke symbols contain strokes resembling smaller symbols (eg. "E" contains "F"). We use this resemblance to compensate for oversegmentation.
  - The *resolution matrix* measures how likely one symbol is to be classified as another. (These values are precomputed using data extracted from the symbol database.)
  - Call the database symbols $M_1, M_2, ..., M_n$ and let α_{ij} be the entry in the resolution matrix measuring the likelihood of recognizing M_i as M_j.
  - Record all match scores less than some threshold, mapped to a confidence value in the range [0,1].
  - If the confidence that an input I is of class M_i (based on extracted feature comparison) is $C(I = M_i)$, then the segmentation score after bias adjustment is
    $$C(I = M_i) + \frac{1}{N}\sum_{j=1}^{n} \alpha_{ij} C(I = M_j)$$
    where N is the number of nonzero α's in the summation.
- Since this method uses features extracted from database symbols, it benefits from training.
  - A user may add new samples to the database for symbols she writes in a unique way (ie. differently from the database models), and the feature-correlation method will automatically include this new data in its predictions.

Integrating with classification

- Results of the two methods are combined: heuristic guesses are assigned scores based on confidence levels, and normalized vectors representing the distribution of segmentation probabilities are created for each of:
  - feature-correlation score,
  - sum of feature-correlation and stack hint scores, and
  - sum of feature-correlation and proximity hint scores.
- During recognition, classifiers assign match scores to database symbols. Lower scores indicate better matches.
- A candidate's match score is divided by the appropriate entry from one of the segmentation probability vectors to give the final score. Which vector to use depends on attributes of the database symbol, namely:
  - *container* symbols (eg. "$\sqrt{}$") use feature-correlation score
  - *stacked* symbols (eg. "i","=") use feature-correlation plus stack hint scores
  - all other symbols use feature-correlation plus proximity hint scores

Experimental results

Test set "A" consists of well-spaced drawings of individual symbols from a single writer. Set "B" consists of mathematical expression samples from multiple writers.

- The data indicates the number of symbols which each method correctly segmented.
- The heuristic method was considered correct if either hint was correct.
- The feature-correlation method was considered correct if the segmentation ranked most probable was correct.
- For the combination method, attributes of the (known) input symbol were used to select the probability vector. If the segmentation ranked most probable in that vector was correct, then the method was considered correct.

Test Set	No. Symbols	Heuristic method	Feature method	Combination
A	1240	1227 (99.0%)	1140 (91.9%)	1226 (98.9%)
B	699	674 (96.4%)	624 (89.3%)	693 (99.1%)

Current work: incorporating time information

- Shorter time between pen-up and pen-down $\Longrightarrow$ higher likelihood that consecutive strokes are part of the same symbol.
- We use an exponential distribution to guess whether the strokes should be part of the same symbol based on time measurements between pen-up and pen-down.
- Elapsed time between strokes is user-specific. A sampling period after the recognizer is initialized allows parameter estimates to settle to a steady state.
- Time information cannot always be used.
  - A user may go back to cross a "t" or extend a fraction bar after writing other symbols.
  - A user may pause at any time before writing the next stroke.
- May only increase or strengthen estimates produced by other methods; cannot reduce estimates.

References

[Nat95] K.S. Nathan, H.S.M. Beigi, Jayashree Subrahmonia, G.J. Clary, H. Maruyama. *Real-Time On-Line Unconstrained Handwriting Recognition Using Statistical Methods*. ICASSP 1995. Vol. 4, pp. 2619-2622, 1995.

[Smi99] S. Smithies, K. Novins, J. Arvo. *A Handwriting-Based Equation Editor*. In *Proc. Graphics Interface*, Kingston, 1999.