Online Recognition of Multi-Stroke Symbols with Orthogonal Series

Oleg Golubitsky Stephen M. Watt
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
{ogolubit,watt}@uwo.ca

Abstract

We propose an efficient method to recognize multi-stroke handwritten symbols. The method is based on computing the truncated Legendre-Sobolev expansions of the coordinate functions of the stroke curves and classifying them using linear support vector machines. Earlier work has demonstrated the efficiency and robustness of this approach in the case of single-stroke characters. Here we show that the method can be successfully applied to multi-stroke characters by joining the strokes and including the number of strokes in the feature vector or in the class labels. Our experiments yield an error rate of 11–20%*, and in* 99% *of cases the correct class is among the top 4. The recognition process causes virtually no delay, because computation of Legendre-Sobolev expansions and SVM classification proceed on-line, as the strokes are written.*

1. Introduction

The process of online handwriting recognition begins with sampling the digital ink by a digital pen or a similar device. The ink always includes the x and y coordinates of the points, and sometimes additional information. In order to remain device-independent, we will restrict our analysis to the x and y coordinates only. Since the ink data arrives in real time and there is a substantial interval between consecutive points, it is natural to use this time to recognize the symbols, rather than waiting until the entire trace is finished and causing a delay after pen-up. This delay is quite noticeable in modern recognition systems, particularly in the case of mathematical handwriting. Our goal is thus to develop online algorithms for online handwriting recognition, which would reduce the user's waiting time to a minimum.

In this paper, we address the problem of online recognition of individual multi-stroke handwritten symbols. Of course, in applications this problem is inseparable from the symbol segmentation, or stroke grouping, problem. On the other hand, one cannot expect a recognition system to be robust if it does not perform well on most individual symbols. One should note that there are a few symbols that are hard to recognize in isolation even for a human, for example small punctuation marks.

Our main target application is recognition of handwritten mathematical formulae. This differs from natural language recognition in several aspects: the number of symbols is larger, there is no fixed dictionary (although, certain subformulae certainly occur more frequently than others), the number of strokes in a symbol is not very large (compared, say, to Chinese), and strokes themselves are usually far from being straight.

Our earlier work has concentrated on recognition of single-stroke mathematical symbols. We have shown that coefficients of the truncated Legendre-Sobolev expansions of the coordinate functions $x(\lambda)$ and $y(\lambda)$, considered as functions of the Euclidean arc length λ, yield an accurate representation of the curves in a low (20–30) dimensional Euclidean vector space. This representation can be computed by an on-line algorithm as the curve is being written [3]. Moreover, the induced Euclidean distance is almost as accurate as the elastic matching (also known as dynamic time warping) distance between the original point sequences, yet 10–50 times faster to compute [5].

An important property of representing a curve by a vector of Legendre-Sobolev coefficients is that this vector depends linearly on the original coordinate functions $x(\lambda)$ and $y(\lambda)$. That is, linear combinations of coordinate functions correspond to linear combinations of their Legendre-Sobolev coefficient vectors. A particular linear combination that is of interest to us is a linear homotopy from one character curve to another, which is commonly referred to as morphing. Intuitively, if both curves belong to the same class (say, represent symbol α), then the intermediate curves that appear during the morphing process will usually belong to this class as well. In the space of Legendre-Sobolev coefficients this translates into the statement that, together with any two points that belong to the same class, the entire seg-

ment must be contained in that class. In other words, most classes are convex sets. If two convex sets in a vector space do not overlap, they can always be separated by a hyperplane. It is therefore natural to use binary linear classifiers order to describe classes of character curves. These classifiers can be efficiently learned using linear support vector machines [6]. SVM have been successfully used for recognition of handwritten symbols in [1, 7]. We extend these results by stating and experimentally verifying the linear separability hypothesis, choosing a more suitable functional basis, and proposing a fast on-line classification algorithm. Moreover, we use a much larger data set, including single- and multi-stroke symbols, in our experiments.

The number of strokes is an important parameter that can be used to distinguish between various characters. There are several ways of doing this. The first is to split classes into subclasses, so that each subclass contains samples with the same number of strokes. In other words, we simply include the number of strokes in the class label. Then, given a new character with a known number of strokes, we only need to match it against classes whose samples have the same number of strokes. This approach leads to good linear separability, but increases the number of classes. Another approach is to include the number of strokes in the feature vector. However the above convexity arguments do not apply to this new coordinate, unlike the Legendre-Sobolev coefficients. We confirm this intuition by linear separability tests and cross-validation analysis.

We also propose an algorithm that carries out the linear classification on-line, as the character is traced out. This should lead to a substantial improvement in real time performance, since the number of binary linear classifiers is quadratic in the number of classes. However, if the classification time needs to be reduced further, one could reduce the number of linear function evaluations from quadratic to linear, at a slight expense of the classification accuracy, by replacing the majority voting scheme with the one that looks for a class that gets all the votes.

This paper is organized as follows. In Section 2, we summarize the facts about Legendre-Sobolev expansions and an algorithm to compute them. In Section 3, we illustrate the properties of convexity and linear separability of character classes. In Sections 3.2 and 3.3, we describe the dataset and experimental results. In Section 4 we present an online algorithm for classifying a moving point with respect to an ensemble of binary linear classifiers. In conclusion we outline directions for improvement of the proposed methods.

2. Legendre-Sobolev series

Online handwriting recognition can be thought of as classification of plane curves. Let $x(\lambda)$ and $y(\lambda)$ be coordinate functions of a curve given by a sequence of points sampled by the digital pen, parameterized by Euclidean arc length λ. We recognize the character by attributing the curve to one of N classes.

First, we compute a representation of the curve in a finite-dimensional vector space. If the character consists of multiple strokes, we join consecutive strokes by straight line segments, which yields a single curve of total length L. Below is a summary of the algorithm proposed in [3, 5].

Step 1. As the curve is being written, accumulate the moment integrals

$$m_k(v) = \int_0^L v(\lambda)\lambda^k \, d\lambda,$$

for both $x(\lambda)$ and $y(\lambda)$ as $v(\lambda)$ and $k = 0, \ldots, d,$.

Step 2. When the curve is finished, apply a coordinate change to map the domain from $[0, L]$ to $[0, 1]$: $\bar{m}_k(v) = m_k(v)/L^{k+1}$.

Step 3. Consider the Legendre-Sobolev inner product:

$$\langle f, g\rangle = \int_0^1 f(\lambda)g(\lambda)d\lambda + \mu \int_0^1 f'(\lambda)g'(\lambda)d\lambda$$

Fix μ and the orthonormal polynomial basis with respect to this inner product: $\langle B_k(\lambda), B_l(\lambda)\rangle = \delta_{kl}, \ \deg B_k = k$. We perform this step in advance and store the coefficients of the resulting orthonormal Legendre-Sobolev polynomials.

Step 4. Compute the coefficients of the truncated Legendre-Sobolev series for the coordinate functions $x(\lambda/L)$ and $y(\lambda/L)$:

$$x_k = \langle x(\lambda/L), B_k(\lambda)\rangle, \ \ y_k = \langle y(\lambda/L), B_k(\lambda)\rangle.$$

These coefficients can be obtained by applying a linear transformation to the moment vectors $(\bar{m}_0(v), \ldots, \bar{m}_d(v))$. The transformation matrix can be easily derived from the coefficients of the basis polynomials.

Step 5. Center the curve by setting $x_0 = y_0 = 0$ and normalize the coefficient vector $(x_1, \ldots, x_d, y_1, \ldots, y_d)$ to length one.

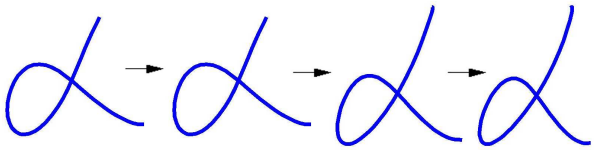
For our purposes, truncation order $d = 12$ turns out to be sufficient. Experiments have shown [2, 3, 5] that, for d between 10 and 15, the curves defined by the truncated Legendre-Sobolev series of the coordinate functions are visually indistinguishable from the original curves, for most characters in our dataset.

Since Step 1 is carried out while the curve is written, it causes no delay after pen up. Steps 2–5 amount to a linear transformation in a $(d+1)$-dimensional vector space. These steps are done after pen up, but their complexity depends only on the constant d (in fact, the complexity is quadratic in d). The entire computation after pen up is therefore almost instant, with its time complexity not depending neither on the parameters of the digital pen nor on the number of points in the strokes traced.

3. Learning and Classification

3.1. Convexity and linear separability

Out of a variety of available classification methods for classes of points in a vector space, we choose one that is particularly suitable for our problem. The choice is based on the observation that symbol classes are *convex*. Intuitively, if we gradually morph one curve into another curve from the same class, intermediate curves usually belong to this class as well.



The process of gradual morphing can be formally described as a linear homotopy. If $x_I(\lambda), y_I(\lambda)$ and $x_{II}(\lambda), y_{II}(\lambda)$ are coordinate function pairs for two curves, we can morph the first curve into the second by letting a homotopy parameter ξ vary from 0 to 1 and considering the intermediate curves

$$\begin{aligned} x_\xi(\lambda) &= (1-\xi)x_I(\lambda) + \xi x_{II}(\lambda) \\ y_\xi(\lambda) &= (1-\xi)y_I(\lambda) + \xi y_{II}(\lambda). \end{aligned}$$

Since the mapping from the coordinate functions to their Legendre-Sobolev coefficient vectors is linear, as the homotopy parameter varies from 0 to 1, the Legendre-Sobolev coefficient vector for the intermediate curve $x_\xi(\lambda), y_\xi(\lambda)$ traces the straight line segment, whose end points are the Legendre-Sobolev coefficient vectors for the original and target curves. Thus, if two points belong to the same class, the entire segment connecting them is contained in this class as well. In other words, character classes form convex sets in the finite-dimensional space of Legendre-Sobolev coefficient vectors. It is well-known that disjoint convex sets in a finite-dimensional vector space can always be separated by a hyperplane. It is thus natural to distinguish between the character classes using binary linear classifiers.

3.2. Verification of linear separability

In order to verify linear separability of classes experimentally, we trained an ensemble of linear SVM classifiers on a dataset of labeled curves. The dataset we are using consists of 56,069 samples of handwritten mathematical symbols from 280 classes provided by about 150 writers. The dataset incorporates samples collected at the Ontario Research Centre for Computer Algebra, the LaViola database of handwritten symbols, and the UNIPEN database of handwritten digits. All three datasets are merged together and stored in a single file in InkML format. A detailed description of our dataset can be found in [4].

Method	Ch.Typ	#sampls	#cls	#err	RetRate
I	All	56059	648	3229	94.2%
II	All	56069	1303	1951	96.5%
I	Multi	20425	403	868	95.8%
II	Multi	20425	838	374	98.1%
Any	Single	34634	465	1577	95.6%

Table 1. Results of linear separability test.

The original labels assigned to the samples during the collection process were not 100% accurate. Therefore, the entire dataset was manually inspected, and labels were corrected when necessary. A significant number of samples can be attributed to more than one class: some classes of mathematical symbols, for example `U`/`V`/`union`, `C`/`c`/`open_bracket`/`subset`, `4`/`y` are inherently ambiguous. However, these classes cannot be merged, because each of them contains samples that can be definitely attributed to their class, and do not belong to the other similar classes. In situations such as this, the samples are divided into three classes: those that are definitely a `4` (class `4`), those that are definitely a `y` (class `y`), and those that could legitimately be interpreted as either (class `4-y`). We labeled all ambiguous characters with such composite labels and obtained 648 classes. Note that some of these classes contain allomorphs that have not been labeled separately.

We tested these classes for linear separability. For each pair of classes with non-overlapping composite labels, we trained a linear SVM that yields the least number of classification errors. Note that SVM classifiers depend on a numeric parameter that expresses the trade-off between the number of errors and the margin between the sample points and the separating hyperplane. For larger values of this parameter, we get fewer errors, but also a smaller margin. The complexity of SVM learning increases with the parameter value. We chose the maximal value, for which the learning procedure still succeeds within reasonable time.

It turns out that the number of strokes is an important parameter which cannot be ignored when classifying a character curve. We tested two ways of dealing with the number of strokes: Method I incorporates the number of strokes in the feature vector, and Method II adds the number of strokes to the class label. In the latter case, we get a larger number of classes (1303) whose degree of linear separability should be higher. The results are summarized in Table 1. The columns in the table correspond to: method; type of characters used in the experiment (single-stroke, multi-stroke, or all); total number of samples; total number of classes; number of errors (i.e., number of samples that fell on the wrong side of at least one separating hyperplane); and the correct retrieval rate (percentage of samples that fell on the correct side for all separating hyperplanes).

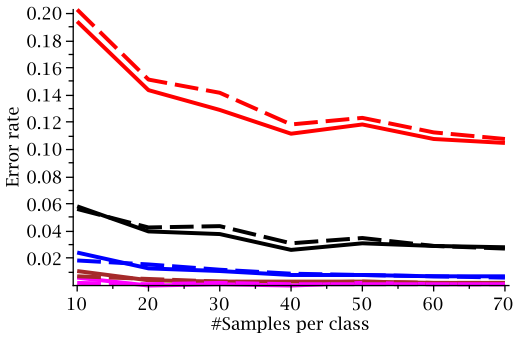


Figure 1. Top-K error rates for Methods I (dashed) and II (solid), $K = 1, 2, 3, 4, 5$.

3.3. Retrieval rates

We used 10-fold cross-validation to estimate the correct retrieval rates of the ensemble of linear SVM classifiers. The experiments were performed for classes with at least 70 samples; there are 111 such classes, out of the 1303 (for Method II). From these classes, we extracted random subsets of sizes 10,20,30,40,50,60,70 and partitioned each into 10 test sets randomly. For each test set, its complement (the union of the remaining 9 test sets) served as a training set. We trained a separating hyperplane for each pair of classes with the same number of strokes. Then, for each test sample, classes were ranked by the number of times the sample was classified as their member. The top-K error rates for $K = 1, 2, 3, 4, 5$ are shown in Figure 1 (solid lines). Method II yields an error rate of 19.5% with 10 samples per class (9 of which are used for training), 14% with 20 samples, and levels off at 11% for over 60 samples. The top-2, top-3, and top-1 error rates with 10 samples are 4.8%, 2.3%, and 1%, respectively. These rates are obtained for the Legendre-Sobolev parameter $\mu = 1/8$, truncation order $k = 12$, and SVM trade-off parameter $C = 10$, which have been determined to be optimal by cross-validation.

For comparison of Methods I and II, we used the same training and test sets, with the number of strokes removed from the class labels and added to the feature vector. As a result, classes corresponding to the same symbol written with different numbers of strokes merged, yielding 96 classes instead of 111. The number of samples per class has slightly increased, yet, for an easier comparison with Method II, we plot the error rates against the same numbers of samples (10–70), which can only be advantageous to Method I. Despite a smaller number of classes and a greater number of samples per class, Method I yields slightly higher error rates than Method II, shown in Figure 1 (dashed lines). We conclude that the number of strokes is an important feature for distinguishing between classes but not very suitable for SVM classifiers, likely because it violates the property of convexity discussed at the beginning of Section 3.

Algorithm 1 Online Classification w.r.t. Hyperplanes

Require: Set of linear functions $\mathcal{H}$;
Sequence of points sampled in real time.
Ensure: Signs of the linear functions at the last point.
PriorityQueue $\leftarrow \{(H, 0) \mid H \in \mathcal{H}\}$
$P \leftarrow$ firstPoint()
$l \leftarrow 0$
while $\exists$ more points **do**
$P' \leftarrow$ nextPoint()
$l \leftarrow l + ||P' - P||$
repeat
$(H, d) \leftarrow$ PriorityQueue
$\rho \leftarrow H(P')$
$s_H \leftarrow \operatorname{sgn}(\rho)$
PriorityQueue $\leftarrow (H, |\rho| + l)$
until $d > l$
$P \leftarrow P'$
end while
return $\{s_H \mid H \in \mathcal{H}\}$

4. An online classification algorithm

Not only the Legendre-Sobolev feature vector can be computed while the character curve is written, but also the linear classifiers need not wait until the entire curve is finished. As the character curve evolves, its Legendre-Sobolev feature vector also traces a curve in the feature space.

Let a sequence of points in $\mathbf{R}^n$ be arriving in real time. Let function `firstPoint:()`$\rightarrow \mathbf{R}^n$ initiate the sequence and return the first point. Let function `nextPoint:()`$\rightarrow \mathbf{R}^n$ wait, if necessary, for the next point to arrive and then return it. Then Algorithm 1

- **Inputs** a set $\mathcal{H}$ of linear functions $H : \mathbf{R}^n \rightarrow \mathbf{R}$ normalized so that, for any point $P \in \mathbf{R}^n$, $|H(P)|$ is the Euclidean distance from P to the hyperplane H.
- **Retrieves** a sequence of points $P_1, \ldots, P_N$ using the above functions `firstPoint()` and `nextPoint()` and computes the signs of the linear functions from $\mathcal{H}$ at the last point P_N.
- **Minimizes** the expected number of operations after the arrival of the last point.

The underlying assumption is that the sequence of points lies on a curve that is not completely random. This allows to effectively predict the sign of a linear function at the endpoint by looking at the signs of this function at certain intermediate points. The algorithm maintains a priority queue containing pairs (H, d), ordered increasingly by d.

Correctness of the algorithm is implied by the following invariant, which holds at the end of each iteration of the `while` loop: $s_H = \operatorname{sgn} H(P')$ for all $H \in \mathcal{H}$.

Our goal is to reduce the time complexity after the arrival of the last point. We will show that this complexity is

reduced by a factor that is linear in the number of points. In other words, the algorithm achieves an expected linear speedup, compared to the offline version that does all linear function evaluations at the end.

Assume that the points lie on a curve $\gamma(\lambda)$ parameterized by the Euclidean arc length λ, and let L be the total length of the curve. Then the number l computed in the algorithm approximates the length of the part of the curve seen so far. Fix a hyperplane $H \in \mathcal{H}$. If we know the value of $H(P)$ for $P = \gamma(\lambda)$ and set $\rho = |H(P)|$, then for all $\lambda' < \lambda + \rho$ and $P' = \gamma(\lambda')$ we have $\operatorname{sgn} H(P') = \operatorname{sgn} H(P)$. In other words, if we have evaluated the sign of a linear function H at a point P, and the distance from P to the hyperplane defined by H is ρ, then for all points that are within the arc length of ρ from P along γ, the sign will be the same, so we do not need to re-evaluate it (see Figure 2).

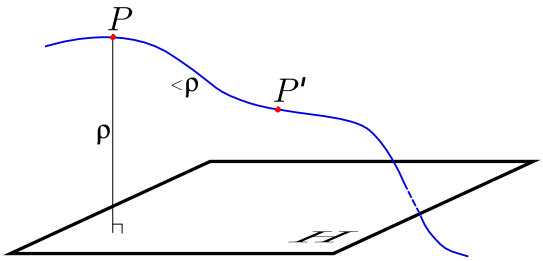


Figure 2. Sign does not change.

With the above idea in mind, let us estimate the expected number of times the sign of a fixed linear function H will be evaluated, as we move along the curve $\gamma(\lambda)$. Let r_H be the average distance from a point on $\gamma(\lambda)$ to H. Then, on average, we can travel the distance of r_H along $\gamma(\lambda)$ without having to recompute $\operatorname{sgn} H(\gamma(\lambda))$. Therefore, the expected number of times we compute $H(\gamma(\lambda))$ is L/r_H.

Let $\mathcal{H} = \{H_1, \ldots, H_M\}$ be a finite set of linear functions, and let $r_i = r_{H_i}$ be as above. Summing up over all $H \in \mathcal{H}$ yields the expected total number of evaluations:

$$\sum_{i=1}^{M} L/r_i = M \cdot L \cdot \left(\frac{1/r_1 + \ldots + 1/r_M}{M} \right).$$

If we assume that the length L and the mean inverse average distance to the hyperplanes are bounded by constants for all inputs $\mathcal{H}$ and $\gamma(\lambda)$, we obtain that the expected number of linear function evaluations is $O(M)$. These evaluations are distributed over N points, so on average we get $O(M/N)$ evaluations per point.

Maintaining the priority queue costs an additional factor of $O(\log M)$. However, after the last point arrives we do not need to maintain the priority queue anymore: we just need to extract a certain number of its top elements. This can be done without the logarithmic overhead, if the priority queue is implemented with a balanced binary tree, using the in-order traversal of the tree. Therefore, we obtain the expected time complexity of $O(M \log M/N)$ after each intermediate point, and $O(M/N)$ after the last point.

5. Conclusion

We have shown experimentally that truncated Legendre-Sobolev series provide a robust representation for handwritten symbols, for both single- and multi-stroke symbols alike. Using this representation one can classify symbols among a large number of classes with linear support vector machines. We have described an algorithm to compute both the Legendre-Sobolev feature vectors and the signs of linear classifiers as the curve is being written, so that the delay after pen-up is minimized. We obtained 86-87% correct retrieval rates with 20–30 training samples, and 99% correct retrieval rate for the top-4 classification. These are raw recognition results for individual characters. Further disambiguation can be achieved via dictionaries, for natural language text, or n-grams, for mathematics [8], or more expensive secondary classifiers.

One possible way to improve the correct retrieval rates would be to label different allomorphs of symbols and try to separate classes of allomorphs by hyperplanes. Preliminary observations suggest that the presence of different allomorphs in the same class may result in non-convex classes and diminish their linear separability.

References

[1] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line Handwriting Recognition with Support Vector Machines-A Kernel Approach. In *Proc. of the 8th IWFHR*, pages 49–54, 2002.

[2] B. W. Char and S. M. Watt. Representing and characterizing handwritten mathematical symbols through succinct functional approximation. In *Proc. Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pages 1198–1202, 2007.

[3] O. Golubitsky and S. M. Watt. Online stroke modeling for handwriting recognition. In *Proc. 18th Intl. Conf. on Comp. Sci. and Soft. Eng. (CASCON 2008)*, pages 72–80, 2008.

[4] O. Golubitsky and S. M. Watt. Distance-based classification of handwritten symbols. Technical Report TR-09-03, Ontario Research Center for Computer Algebra, 2009.

[5] O. Golubitsky and S. M. Watt. Online stroke modeling for handwriting recognition. In *Proc. Document Recognition and Retrieval XVI, San Jose, CA, USA*, pages C1–C10, 2009.

[6] T. Joachims. SVMLight: Support Vector Machine. *http://svmlight. joachims. org/, Univ. of Dortmund*, 1999.

[7] B. Keshari and S. M. Watt. Online mathematical symbol recognition using svms with features from functional approximation. In *Proc. Mathematical User-Interfaces Workshop (MathUI)*, 2008.

[8] E. Smirnova and S. M. Watt. Context Sensitive Mathematical Character Recognition. In *Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR)*, 2008.