

# Tie-Breaking for Curve Multiclassifiers

Oleg Golubitsky                      Stephen M. Watt

Department of Computer Science

University of Western Ontario

London, Ontario, Canada N6A 5B7

ogolubit@uwo.ca

stephen.watt@uwo.ca

April 29, 2009

## Abstract

Many practical problems reduce to classifying curves among multiple classes, for example on-line recognition of handwritten mathematical symbols. By treating a curve as an algebraic object and computing truncated expansions of its parametric coordinate functions in an orthogonal functional basis, we obtain an accurate, compact, and geometrically intuitive representation of the curve as a point in a low-dimensional vector space. Previous work has shown that, with this representation, high top-k classification rates can be achieved using support vector machines. However, the gap between the top-1 and top-2 classification accuracies remained large.

We report on a variation of nearest neighbor classification using the distance to the convex hull of several nearest neighbors. This reduces the tie-breaking errors between the top two classes by about 20% and the overall error rates by about 10%. The technique is well adapted to classification among hundreds of classes, with the number of training samples ranging from ten to several thousands, and with strict requirements on the speed and memory use.

## 1 Introduction

Classification of parametric curves is a basic task in computer vision and pattern recognition, often arising as a sub-problem in a larger context. Examples of applications where parametric curves are sampled in real time

include: on-line recognition of hand-written text, formulae, and diagrams, astronomical time-series classification, medical diagnostic data evaluation, motion analysis and robot navigation. For performance reasons, it is often necessary to classify the curves in real time as well.

Our primary interest is in on-line recognition of hand-written mathematical formulae. This area of pattern recognition is characterized by the following particular features:

- Mathematical formulae are two-dimensional objects composed of symbols.
- A large number of symbol classes are used in mathematical formulae (about 300 of them).
- Individual symbols are each made up of a small number of strokes and are distinguished by how the strokes curve.
- Although there may be some acceptable variability, directional orientation is important both to formulae and the constituent symbols.
- There is no fixed dictionary beyond the alphabet of individual symbols. Certainly, some sub-formulae occur more often than others, but the frequencies of particular notations and sub-expressions strongly depend on the mathematical area.

These features make mathematical handwriting recognition a very different (and arguably more difficult) problem than other text recognition. On the other hand, symbols in mathematical formulae tend to be well segmented

and generally consist of few strokes. Therefore, recognition of isolated mathematical symbols is a well-posed and crucial sub-problem, which requires a special attention in the context of mathematical handwriting recognition. The growing demand for handwriting recognition applications for portable devices makes this problem even more challenging, given that the memory and speed parameters of such devices are usually orders of magnitude lower than those of general-purpose personal computers.

We consider the problem of on-line recognition of handwritten mathematical symbols as that of classification of parametric plane curves among multiple classes. By “on-line recognition” we mean that the recognition process commences as the curve is being traced out. For multi-stroke symbols, we join the consecutive strokes to form a single connected curve, remembering the original number of strokes for the classification purposes.

Earlier work [2, 4, 5] has shown that parametric plane curves can be accurately and succinctly represented by the vector of coefficients of the truncated Legendre-Sobolev series of the coordinate functions. It has been established that the dimension of the coefficient vector need not exceed 20–30, because at degree 10–15 the approximating curves already look like the originals. Moreover, the classes of handwritten characters, when mapped into this finite-dimensional vector space, enjoy nice properties of convexity and linear separability. Earlier work has shown that linear support vector machines combined with an appropriate majority voting scheme yield encouraging classification results [3]. Moreover, the classification can be performed while the curve is written, so that the results are ready immediately after the pen is lifted [6].

With appropriately chosen Legendre-Sobolev norm parameter and SVM margin/error trade-off constant, linear SVM with majority voting yield about 96.0% correct retrieval rate for 232 classes of handwritten mathematical symbols, trained with at least 9 samples per class. The correct class is ranked among the top 2 in about 98.3% of cases, and among the top 3 in about 99.0% of cases.

While these rates might already look encouraging, the noticeable gaps between the top-1 and top-2 correct classification rates and between them and the top-3 rate call for further investigation and improvement.

From the practical point of view, it is important to maximize the performance of the recognizer of individual characters, because the recognition rate of the entire

formula equals the product of the rates for the individual entries, save the potential benefits from using contextual information which, as mentioned above, must be used with care because of its high variability.

In this paper we propose to use the distance to the convex hull of  $k$  nearest neighbors in order to improve the choice among the top  $t$  classes.

The paper is organized as follows. In Section 2, we summarize the theory of Legendre-Sobolev series of parametric plane curves and the algorithm for computing the finite dimensional feature vector using this series. In Section 3, we describe the basic classification procedure with linear support vector machines and majority voting and present the experimental cross-validation results, which justify the choice of the SVM trade-off parameter and show that linear SVM are not sensitive to the choice of the Legendre-Sobolev norm. In Section 4, we describe the algorithm for computing the distance to the convex hull of  $k$  nearest neighbors. In Section 5, we present the cross-validation results for the classification based on the distance to the convex hull of  $k$  nearest neighbors and choose the optimal values for the number of nearest neighbors, the number of top classes to reorder with this distance, and the Legendre-Sobolev norm (which does affect the distance-based classification). Section 6 concludes the paper.

## 2 Legendre-Sobolev series

Let a parametric curve in the plane be given by its coordinate functions  $x(t)$  and  $y(t)$ , where  $t$  is an increasing parameter that ranges over an interval  $[0, T]$ . In case of handwritten curves, this parameter could be time, Euclidean arc length, affine arc length, etc. Previous work [5] has shown experimentally that, among these three choices, the Euclidean arc length yields the most accurate distance. Therefore, we fix this parameterization.

Consider the first  $d + 1$  moment integrals of the coordinate function  $x(t)$ :

$$m_k^x = \int_0^T x(t)t^k dt, \quad k = 0, \dots, d$$

and, similarly, of  $y(t)$ . The appropriate value of  $d$  will be chosen below. Note that the moment integrals can be

accumulated as the points on the curve are received (see [4] for the computational details).

If  $p(t)$  is a polynomial of degree at most  $d$ , then the integral

$$\int_0^T x(t)p(t) dt$$

is a linear combination of the moments  $m_0^x, \dots, m_d^x$ , whose coefficients coincide with those of  $p(t)$ .

Consider the Legendre-Sobolev inner product in the space of differentiable functions on the interval  $[0, T]$ :

$$\langle f(t), g(t) \rangle = \int_0^T f(t)g(t) dt + \mu \int_0^T f'(t)g'(t) dt,$$

where  $\mu$  is a non-negative real parameter. When  $\mu = 0$ , the inner product is called the Legendre product. The inner product induces a norm and a distance metric in the space of parametric curves given by

$$\|(x(t), y(t))\| = \sqrt{\langle x(t), x(t) \rangle + \langle y(t), y(t) \rangle}$$

and

$$\|(x_1(t) - x_2(t), y_1(t) - y_2(t))\|,$$

respectively. For  $\mu = 0$ , this distance is the usual Euclidean distance between the curves, that is, the integral of the distances between their corresponding points. For a non-zero value of  $\mu$ , the differences between the *slopes* at the corresponding points are taken into account as well, and we obtain the distance between curves in the first jet space. Experimentally, it has been shown [5] that the Legendre-Sobolev distance yields better classification results than the Legendre distance.

The computation of the Legendre-Sobolev distance proceeds as follows [4, 5]. Assume that a value of  $\mu$  is fixed. Using the orthogonalization process, one can obtain the orthonormal basis of the subspace of polynomials of degree up to  $d$  with respect to the Legendre-Sobolev inner product on the interval  $[0, 1]$ . Let  $B_0(t), \dots, B_d(t)$  be the resulting polynomials, also called Legendre-Sobolev. Then the orthogonal projection of a given function  $f(t)$  onto the subspace of polynomials of degree up to  $d$  is given by the truncated series

$$\sum_{k=0}^d c_k^f B_k(t),$$

where the *Legendre-Sobolev* coefficients are given by the inner products of the function  $f(t)$  with the basis polynomials:

$$c_k^f = \langle f(t), B_k(t) \rangle.$$

As mentioned above, these inner products can be obtained as linear combinations of the moments. Note that, for the second term in the Legendre-Sobolev inner products, one can apply integration by parts to avoid computing the moments of  $f'(t)$ . Note also that, if the function  $f(t)$  is defined on  $[0, T]$ , rather than on  $[0, 1]$ , one can use the substitution  $t \mapsto t/T$ .

Having thus computed the Legendre-Sobolev coefficient vector

$$(c_0^x, \dots, c_d^x, c_0^y, \dots, c_d^y)$$

for the coordinate functions  $x(t)$  and  $y(t)$ , we notice that the 0-order coefficients  $c_0^x$  and  $c_0^y$  correspond to the center of the curve. Therefore, setting them to zero normalizes the curve's position. Furthermore, the norm of the resulting coefficient vector is proportional to the size of the curve. Therefore, by normalizing the vector, we normalize the curve's size. We use the resulting normalized Legendre-Sobolev coefficient vector

$$(\bar{c}_1^x, \dots, \bar{c}_d^x, \bar{c}_1^y, \dots, \bar{c}_d^y)$$

to classify the curve among multiple handwritten symbol classes.

The dimension of the resulting vector space is equal to  $2d$ . The value of  $d$  between 10 and 15 has been shown to be sufficient to approximate handwritten symbol curves to the extent that the approximation is visually indistinguishable from the original [2, 4]. Therefore, it must be also sufficient for all classification purposes.

### 3 SVM classification

Handwritten symbol curves generally enjoy the following property: If one curve is gradually morphed into another curve representing the same symbol, the intermediate curves represent that symbol as well (see Figure 1).

This property holds only if the two curves are written in the same way, that is, if they do not represent different allomorphs of the symbol. If they do, the intermediate curves may look arbitrarily, and may even occasionally resemble very different symbols (see Figure 2).

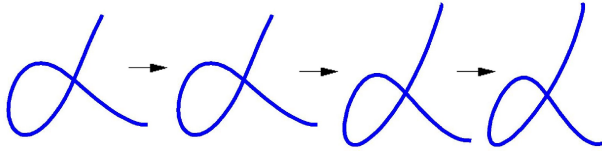


Figure 1: Linear homotopy between two samples from a class produces curves that belong to the same class.

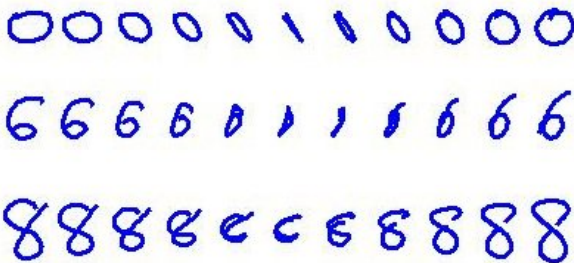


Figure 2: Linear homotopy between two samples representing different allomorphs produces arbitrary intermediate curves.

When one curve is gradually morphed into another using a linear homotopy, the corresponding vector of Legendre-Sobolev coefficients traces a straight line segment. Therefore, for a class of curves corresponding to a particular allomorph of a symbol, the corresponding Legendre-Sobolev coefficient vectors form a convex set in the feature space. Since disjoint convex sets are linearly separable, it is natural to use separating hyperplanes for classification. These hyperplanes can be efficiently learned by linear support vector machines [7]. It has been shown experimentally in [3] that allomorph classes are indeed linearly separable.

We combine the binary SVM classifiers for each pair of allomorph classes into a multi-class classifier using the majority voting scheme. While the performance of the resulting classifier on the training set is nearly perfect because of linear separability, this is not the case on the test set. In fact, this performance strongly depends on the SVM parameter  $C$ , which is the trade-off between the

training errors and the margin. For higher values of  $C$ , fewer errors are allowed; for lower values, the margin is wider. Naturally, for the linear separability test one would set the highest possible value of  $C$ , but it does not provide the best results on the test set.

Our experiments were performed on a dataset of 50,719 single- and multi-stroke mathematical symbols, which belong to 232 classes. These classes have been further subdivided into allomorph subclasses. Finally, the samples were visually inspected and each sample was labelled with *all* classes to which it can be attributed (usually, one, two, or three classes). For example, composite name *delta-five-s* labels the intersection of the three classes:  $\delta$ , 5, and *s*. For each such combination of classes with at least 12 samples (overall, 447 combinations), we assigned a unique composite label. The number of samples per class varied from 12 to 2750. Classification was performed among these 447 classes. A test sample is correctly attributed to a class, if the sample's composite label overlaps with the class's composite label.

Using 4-fold cross-validation with 75/25 training/test split on this dataset, we have determined that  $C = 10$  (as in [7]) is the optimal value for the SVM trade-off constant. The summary of the results on various training/test splits is shown in Table 1 and Table 2.

The parameter  $\mu$  of the Legendre-Sobolev norm affects the SVM classification accuracy only slightly. It appears that  $\mu = 1/16$  yields the highest correct retrieval rates. The difference between retrieval rates at  $\mu = 1/16$  and  $\mu = 1/8$  is very small, about .05%. The 4-fold cross-validation averages for various values of  $\mu$  are shown in Table 3.

Our main goal in this paper is to reduce the gap of about 2.3% between the top-1 and top-2 correct retrieval rates. The idea is to use a more stable, but perhaps slower, classification method at the last stage of breaking the ties between the few top classes. One such method, which we explore in detail, is based on the distance to the convex hull of the nearest neighbors [1].

## 4 Distance to the convex hull

Nearest-neighbor-based classification is known to require a large number of training samples. However, the method can be refined so that even with a few training samples (9

	Set 1	Set 2	Set 3	Set 4	Average
Top-1	95.9	96.0	95.8	96.2	<b>96.0</b>
Top-2	98.3	98.4	98.2	98.5	<b>98.3</b>
Top-3	98.9	99.0	98.9	99.0	<b>99.0</b>
Top-5	99.3	99.5	99.3	99.4	<b>99.4</b>
Top-10	99.6	99.8	99.6	99.7	<b>99.7</b>

Table 1: Classification accuracies for  $C = 10$  ( $\mu = 1/8$ ).

	$C = 1$	$C = 10$	$C = 100$	$C = 1000$
Top-1	93.5	<b>96.0</b>	95.8	95.6
Top-2	97.3	<b>98.3</b>	98.3	98.3
Top-3	98.2	<b>99.0</b>	99.0	98.9
Top-5	99.0	<b>99.4</b>	99.4	99.4
Top-10	99.5	<b>99.7</b>	99.7	99.7

Table 2: Average classification accuracies for various  $C$ .

in our case), the accuracy gets higher than that of SVM-based classifiers with majority voting. We are going to use the idea proposed in [1], to replace the distance to the nearest neighbor(s) by the distance to the convex hull of several nearest neighbors.

Consider the top  $M$  classes, obtained as a result of SVM classification. Fix a number  $k$ . For a given test sample, find up to  $k$  nearest neighbors among the training samples from each of the top  $M$  classes. Compute the shortest distance between the test sample and the convex hulls of the nearest neighbors. Choose the class for which this distance is smallest.

The problem of computing the distance to the convex hull of points in a 20–30 dimensional vector space is non-trivial. In [1], it is suggested to use an approach similar to SVM learning. Indeed, for the maximal margin hyperplane separating a given point from a given convex object, the distance between the point and the object is equal to the double margin. This approach is efficient when the number of nearest neighbors is large. However, when it is less than the dimension of the vector space, one could take advantage of the fact that any set of points in generic position, of size not exceeding the dimension of the vector space plus one, forms a simplex. If the points are not

	0	1/64	1/32	1/16	1/8	1/4
Top-1	95.5	95.6	95.6	<b>95.7</b>	95.6	95.6
Top-2	98.1	98.3	98.3	<b>98.3</b>	98.3	98.3
Top-3	98.8	99.0	99.0	<b>99.0</b>	99.0	99.0
Top-5	99.3	99.4	99.4	<b>99.4</b>	99.4	99.3
Top-10	99.7	99.7	99.7	<b>99.7</b>	99.7	99.6

Table 3: Classification accuracies for various  $\mu$  ( $C = 1000$ ).

6	9	10	11	12	15	20
96.34	96.46	96.39	<b>96.51</b>	96.46	96.42	96.33

Table 4: Top-1 classification accuracy vs number of nearest neighbors (when top two classes are reordered).

in generic position, one can perturb them slightly; the distance will change only slightly as well.

The distance between a point and a simplex can be computed using the algorithm proposed in [8]. Informally, it can be described as follows. Given a point and a simplex, first find the projection of the point on the smallest affine subspace containing the simplex. If the projection is inside the simplex, the distance is found. If not, express the projection as a linear combination of the vertices of the simplex. Consider the vertices whose corresponding coefficients are non-negative, and find recursively the distance to the simplex defined by these vertices.

The pseudocode for the algorithm is given in Figure 3.

The depth of recursion is at most the dimension of the vector space, and each recursive call amounts to solving a linear system. Therefore, the complexity of this algorithm is  $O(N^4)$ , where  $N$  is the dimension. In practice it can perform much faster, because the dimension often drops by more than 1 at a recursive call.

We have implemented the algorithm and determined that the optimal value of the number of nearest neighbors in the convex hull is  $k = 11$  (see Table 4).

Furthermore, the optimal number of top classes to re-rank with the distance to the convex hull of  $k$  nearest neighbors is  $M = 10$  (see Table 5).

**Algorithm 1.**

INPUT:

*Point  $x$  and vertices  $\{s_1, \dots, s_k\}$  of a simplex.*

OUTPUT:

*Distance from  $x$  to the simplex.*

METHOD:

1. **if**  $k = 1$  **then return**  $\|x - s_1\|$
  2. Find  $v = \sum_{i=1}^k \alpha_i s_i$  such that  
 $\langle v - x, v - s_i \rangle = 0, i = 1, \dots, k.$
  3. **if**  $\alpha_i \geq 0$  for all  $i = 1, \dots, k$   
**then return**  $\|v - x\|$   
**else**  
 $S_+ = \{s_i \mid \alpha_i \geq 0, i = 1, \dots, k\}$   
**return**  $\text{DistToSimplex}(x, S_+)$
- end if**

Figure 3: Algorithm for distance to convex hull of  $k$  points

Finally, the optimal value of the Legendre-Sobolev parameter  $\mu$  is  $1/16$  (this parameter does have a significant effect on nearest-neighbor classification).

The best average correct retrieval rate obtained with this choice of parameters is 96.4%. For top-2 and top-3 classes, the rates can also be improved using the distance to the convex hull of nearest neighbors, to 98.6% and 99.1%, respectively. The overall reduction in the error rate is therefore 11.5%, and the gaps between the top-1 / top-2 and top-2 / top-3 rates have been reduced by 20%.

## 5 Implementation details

As our handwriting recognition algorithms are specifically targeted at applications on mobile devices, special care must be taken of the use of memory. We propose the following compact way of storing the data, so that it can be instantly retrieved by both SVM and nearest neighbor classifiers.

For the nearest neighbor classification, we simply store

$M$	2	5	8	10	12
Top-1	96.51	96.56	96.59	<b>96.61</b>	96.60
Top-2	98.32	98.63	98.70	<b>98.73</b>	98.70
Top-3	98.91	99.01	99.09	<b>99.11</b>	99.11

Table 5: Classification accuracy vs number of top classes to reorder (for 11 nearest neighbors).

the Legendre-Sobolev coefficient vectors of all available training samples. It has been established that 16 bits per coefficient give sufficient precision, and 24 coefficients are sufficient to accurately represent mathematical symbol curves. Therefore, we need 48 bytes per sample. Assuming that we will store about 64,000 samples (slightly more than we have now), all this data will fit in 3Mb.

For SVM classification, we need to store a separating hyperplane for each pair of classes. Assuming that the number of classes is 500 (slightly more than we have now), we obtain about 125,000 hyperplanes. Instead of storing the coefficients of the corresponding linear functions, we propose to store their support vectors. The coefficients can be instantly recovered from the support vectors at the classification time. More specifically, assume that, from the entire pool of training data, we can select at most 128 samples per class, which will serve as support vectors of the hyperplanes separating this class from the other classes. This is a realistic assumption, because SVM do not require large training sets. Then, for a particular hyperplane separating a given pair of classes, we can use 256 bits to indicate which of the selected training samples support this hyperplane. Overall, we will need at most 4Mb to store all hyperplanes.

Thus, the entire database for the symbol recognizer can be stored using 7Mb of memory, which most portable devices can easily provide.

Since portable devices are considerably slower than PCs, special care must be taken of the speed of the algorithms as well. For on-line handwriting recognition (as well as in many other real-time applications), computational delays can be significantly reduced by performing most of the computations on-line, as the curve is written. It has been shown previously that the Legendre-Sobolev coefficient vector can be computed mostly on-line [4], and

	1	2	3	4	5	6	7	8	Average
SVM Top-1	96.0	96.0	96.1	95.8	95.9	95.8	95.7	95.8	95.9
kNN Top-1	96.5	96.5	96.5	96.3	96.1	96.3	96.3	96.3	96.4
SVM Top-2	98.4	98.4	98.4	98.2	98.3	98.4	98.2	98.3	98.3
kNN Top-2	98.6	98.7	98.6	98.6	98.5	98.6	98.5	98.6	98.6
SVM Top-3	98.9	99.1	99.0	98.9	98.9	98.9	98.8	99.0	98.9
kNN Top-3	99.0	99.1	99.1	99.1	99.0	99.0	98.9	99.2	99.1

Table 6: 8-fold cross-validation results for SVM and distance to convex hull of nearest neighbors.

that linear SVM classification can be done on-line as well [6]. The method proposed in this paper requires, in addition, to compute the distance to the convex hull of  $k$  nearest neighbors, for the few top classes produced by SVM. The computation of the actual distance is fast and, since the number of the top classes is only about 10, can be neglected. It is an interesting and challenging problem to design a data structure that would allow to keep track of the nearest neighbors as the curve evolves, so that they are instantly available when the pen is lifted. We leave this problem for future research, noting that the technique developed in [6] might apply.

## 6 Conclusions

Truncated Legendre-Sobolev series provide a compact representation of handwritten curves, which can be computed on-line, as the curve is written. Moreover, fast and robust classification methods such as linear support vector machines, naturally apply to this representation, because it preserves the linear separability of curve classes. However, with only a few training samples per class, the SVM-based classifiers become somewhat unstable, showing a significant gap between the top-1 and top-2 correct retrieval rates.

We have demonstrated that nearest-neighbor classification can effectively stabilize the results of the SVM-based classification, when applied to only a few top classes. In order for the improvement to take place, several nearest neighbors must be taken into account. This may be achieved by computing the distance to their convex hull.

When the number of nearest neighbors is smaller than the dimension (as is our case), an efficient algorithm for computing the distance between a point and a simplex in an  $n$ -dimensional affine space can be applied. Our method relies, at least locally, on the convexity of the classes in the parameter space. This ensures that all points within (and on) the convex hull of a set of nearest neighbors in a class will also be in that class. We have found that the distance to the convex hull is more reliable than  $k$ -NN classification or the distance to the mean of  $k$  nearest neighbors.

## References

- [1] Vincent, P., Bengio, Y.: K-local hyperplane and convex distance nearest neighbor algorithms. *Advances in Neural Information Processing Systems*, The MIT Press, 2002. 985–992.
- [2] Char, B., Watt, S.M.: Representing and Characterizing Handwritten Mathematical Symbols through Succinct Functional Approximation. *Proc. Intl. Conf. on Docum. Anal. and Rec. (ICDAR) (2007)* 1198–1202. 4, 5  
2, 3
- [3] Golubitsky, O., Watt, S.M.: Improved Character Recognition through Subclassing and Runoff Elections. Submitted to ICIAP (2009). 2, 4
- [4] Golubitsky, O., Watt, S.M.: Online Stroke Modeling for Handwriting Recognition. *Proc. 18th Intl. Conf.*

- on Comp. Sci. and Soft. Eng. (CASCON) (2008) 72–80. [2](#), [3](#), [6](#)
- [5] Golubitsky, O., Watt, S.M.: Online Computation of Similarity between Handwritten Characters. Proc. Docum. Rec and Retrieval (DRR XVI) (2009) C1–C10. [2](#), [3](#)
- [6] Golubitsky, O., Watt, S.M.: Online Recognition of Multi-Stroke Symbols with Orthogonal Series. Submitted to ICDAR (2009). [2](#), [7](#)
- [7] Joachims, T.: Making Large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning. B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press (1999). [4](#)
- [8] Michelot, C.: A Finite Algorithm for Finding the Projection of a Point onto the Canonical Simplex of  $\mathbb{R}^n$ . J. Optimization Theory and Applications. Vol. 50, No. 1, July 1986. 195–200. [5](#)