Online Mathematical Symbol Recognition using SVMs with Features from Functional Approximation

Birendra Keshari and Stephen M. Watt
Ontario Research Centre for Computer Algebra
Department of Computer Science
University of Western Ontario
London Ontario, Canada N6A 5B7
{bkeshari,watt}@csd.uwo.ca

Abstract

We apply functional approximation techniques to obtain features from online data and use these features to train support vector machines (SVMs) for online mathematical symbol classification. We show experimental results and comparisons with another SVM-based system trained using features used in the literature. The experimental results show that the SVM trained using features from functional approximation produces results comparable to the other SVM based recognition system. This makes the functional approximation technique interesting and competitive since the features have certain computational advantages.

1. Introduction

Online mathematical symbol recognition will be an essential component of any pen-based mathematical interface. The main advantage of such pen-based mathematical interfaces over traditional keyboard-based input methods is that they can more naturally embody the usual two-dimensonal mathematical notations and wide range of mathematical symbols. With the increasing availability of pen-based devices such as PDAs, smart phones, Tablet PCs and smart boards, interest in this area has been growing. However, the existing sytems are still far from perfection because of the challenges that arise from the two-dimenstional nature of mathematical input and the large symbol set with many similar looking symbols. Here we address the problem of mathematical symbol recognition.

Support vector machines (SVMs) have been shown to be suitable for symbol recognition. Earlier work [4] has shown that SVMs can outperform other machine learning algorithms such as neural nets, HMM and nearest neighbour. A survey of different recognition algorithms in [10] shows that SVMs perform better than HMM on different UNIPEN data subsets. In this paper, we further explore the power of SVM with different features obtained using functional approximation of ink trace curves.

The movement of digital a pen on the surface of a digitizer produces online data which typically contains information including time, position, pressure and orientation of the pen tip. Usually, different features are obtained from these basic features (*eg*. change in direction) and these are used to train a machine learning based symbol recognizer. The accuracy of the recognizer is highly dependent on the nature of the features used for training. Different features have been used for online mathematical symbol recognition in the literature. In most previous work with SVMs, the original coordinate data sequences (X and Y) are interpolated and resampled and then other features, such as resampled X and Y coordinates, sines and cosines of turning angles, centre of gravity, relative length *etc.*, are obtained by straight-forward computation. Features are usually obtained by equidistant resampling as this produces better results than resampling in equal time. Some of the work that has used such features are [4] and [7]. Since the resampling rate is determined heuristically, information such as important corners can be missed easily if the resampling rate is not high enough.

An interesting approach to obtain features is by approximating the coordinate functions $X(s)$ and $Y(s)$ by truncated series in a functional basis. This technique does not require the original data to be resampled. We use such features to train SVMs and consider the experimental results. Earlier work [3] used K-means clustering to cluster the symbols using features obtained by function approximation. In this work, we explore using such features with SVMs and show our results on a larger data set and comparisons with other features.

The remainer of this article is organized as follows: Section 2 presents the main ideas behind SVMs. Section 3 describes the preprocessing steps and feature extraction techniques for two different feature sets. Features in the first set are obtained by employing functional approximation techniques and those in the second set are obtained after resampling. Section 4 presents the implementation details of the classifiers. We show experimental results and comparison between the two systems in Section 5 and conclude with future directions in Section 6.

2. Support Vector Machines

SVMs use a supervised learning algorithm based on the ideas of *VC dimension* and the *structural risk minimization principle* [2]. They have been widely and successfully used for pattern recognition in various fields. The method is also known as a "maximal margin classifier" since it determines a hyperplane that separates the two classes with the largest margin between the vectors of the two classes. Most of problems in real life are however linearly in-separable. SVM can nicely deal with such problems using a kernel that lifts the feature space into higher (possibly infinite) dimension. The linearly separable hyperplane in the higher dimensional space gives a non-linear decision boundary in the original feature space. The decision boundary of the SVM can then be expressed as follows:

$$f(x) = \sum_i (\alpha_i y_i K(x, x_i)) + b \qquad (1)$$

In equation (1), y_i is the label of training sample x_i and x is the sample to be classified. The parameters α and b are found by minimizing a quadratic function subject to some constraints [2]. $K(x, x_i) = \phi(x) \cdot \phi(x_i)$ is the kernel function, where ϕ maps the feature vector into a higher dimensional inner product space. The most commonly used kernels are:

- $K(a, b) = \exp(-\gamma ||a - b||^2), \ \gamma > 0$ (radial)
- $K(a, b) = (\gamma(a \cdot b) + r)^d, \ \gamma > 0$ (polynomial)
- $K(a, b) = \tanh(\gamma(a \cdot b) + r)$ (sigmoid)

An SVM is primarily a binary classifier. However, several SVM classifiers can be combined to do multi-class classification. One-against-all, one-against-one and DAG-SVM are some popular techniques used to combine binary classifiers to build multi-class classifiers. We decided to use one-against-one technique after studying the experimental results of different multi-class classification techniques applied to practical datasets presented in [15].

Figure 1 shows an example of support vector machines. The decision boundary (solid line) separates the two classes (represented by square and circle) way such that the margin between the two classes is maximized.

3. Feature Sets

We compared two sets of different features. Feature Set I consisted of features obtained by functional approximation. Feature Set II consisted of features from resampled data as used in the literature. Details of the preprocessing and extraction of the features in these sets is described next.

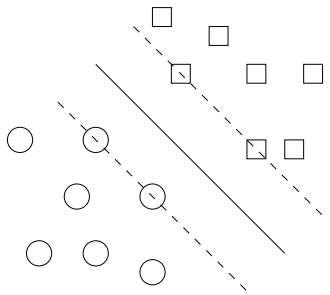

Figure 1. Example of SVMs: the decision boundary maximizes the margin between classes.

3.1. Feature Set I (Chebyshev Coefficients)

Preprocessing

In a preprocessing step, we removed the duplicate points and then applied gaussian smoothing to remove noise. After that, following transformations were applied to the samples:

$$\begin{aligned} t_{new} &= 2\frac{t - t_{min}}{t_{max} - t_{min}} - 1 \\ v_{new} &= 2\frac{v - v_{min}}{v_{max} - v_{min}} \end{aligned}$$

where v is x or y. This transformation places the time (t) values in the range $[-1, 1]$ which is the required domain for the functional approximation. In order to preserve the aspect ratio, x and y are scaled by the same factor and lie in the range $[0, 2]$.

Feature Extraction

We consider the situation where a function $f : \Re \to \Re$ can be approximated by a linear combination of a set of basis functions $B = \{b_i : \Re \to \Re, i \in \mathbb{N}_0\}$ as:

$$f(t) = \sum_{i=0}^{\infty} c_i b_i(t), \quad c_i \in \Re, b_i \in B. \qquad (2)$$

We can define an inner product on this linear space of functions as

$$\langle f, g \rangle = \int_a^b f(t) g(t) w(t) \mathrm{d}t$$

for suitable choices of a, b and "weight function" $w(t)$. If the basis functions are chosen to be orthogonal with respect to the inner product, then the coefficients c_i can be obtained easily by computing the integrals

$$c_i = \langle f, b_i \rangle / \langle b_i, b_i \rangle, \quad i \in \mathbb{N}_0$$

The series (2) is then unique for a given function.

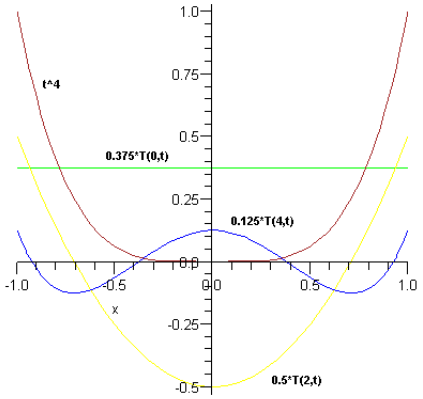

Figure 2. Approximating $f(t) = t^4$ using truncated Chebyshev series.

The inner product equips the space with a norm, $\langle f, f \rangle$, and certain bases have the property that a truncation of the series (2) by setting $c_i = 0$ for $i > N$ gives closer approximations to $f(t)$ as N increases.

The coefficients c_i of a truncated series can provide very compact information about a function f. In this work, we use Chebyshev polynomials of first kind, defined by $T_n(t) = \cos(n \arccos t)$, as basis functions. These are orthogonal but not orthonormal on the interval $[-1, 1]$ with $w(t) = 1/\sqrt{1-t^2}$. The coefficients are obtained using the relations

$$c_i = \frac{2 - \delta_{i0}}{\pi} \int_{-1}^{1} \frac{f(t)\, T_i(t)}{\sqrt{1-t^2}} \mathrm{d}t, \quad i = \in \mathbb{N}_0. \qquad (3)$$

Here $\delta_{i0} = 1$ if $i = 0$ and 0 otherwise. An example showing approximations to the function $f(t) = t^4$ using truncated Chebyshev series is shown in Figure 2.

The X and Y values of the sample data can be considered as two discrete functions of time t or of arc length. We fit natural splines of degree 3 to $X(t)$ and $Y(t)$ to obtain numerical functions $S_x(t)$ and $S_y(t)$. These two functions were then approximated by truncated Chebyshev series and the coefficients ere obtained by numberical integration of (3). As these coefficients are unique and provide useful characterizaion of the handwritten samples, we used these coefficients as features to train the SVMs.

3.2. Feature Set II (Standard Features)

Preprocessing

During preprocessing, all the duplicate points were removed from each sample and gaussian smoothing was applied to remove the noise. Each pair of consecutive points p_i and p_{i+1} were linearly interpolated and the resulting piece-wise linear curve was resampled at equal distance. Although this approach removes the time information, the result produced by such resampling is usually better than the result obtained by resampling in time. Each sample was normalized by preserving the aspect ratio so that it fit a unit square. Figure 3 illustrates the preprocessing steps applied to Feature Set II.

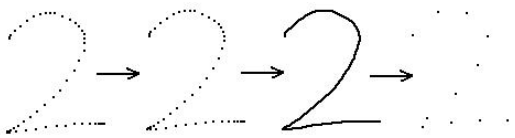

Figure 3. Processing steps applied to Feature Set II: smoothing, filling missing points and resampling.

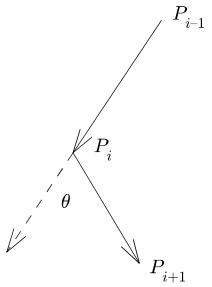

Figure 4. Turning Angle(θ).

Feature Extraction

Feature Set II consisted of the following features: coordinates of the resampled points, sines and cosines of the angle made by the line segments joining the points in the stroke, and the sines and cosines of the turning angle between the line segments and centre of gravity of the symbol. The turning angle is illustrated in Figure 4. The centre of gravity of the symbol is $(\sum_i x_i/N, \sum_i y_i/N)$, where N is the total number of resampled points and x_i and y_i are the coordinates of the points in the stroke.

Since all these features are derived from the resampled points they are directly dependent on the resampling rate. Ideally a good resampling rate should not miss any shape information such as curvatures, loops and so on. However, in practice, this rate is determined heuristically and therefore there is no such guarantee.

4. Implementation

We stored the handwritten samples in InkML, a digital ink format [8]. In both cases, these samples were parsed and the online features were calculated. To train the classifiers and predict new samples we use the `libsvm` [1] library. Two systems were separately trained on each feature set. All feature values were scaled between 0 and 1 before training and testing. A few kernels were tried and the radial basis functions (RBF) were found to be the most appropriate. These were therefore chosen as kernels for both the classifiers. A one-verses-one algorithm was used for multi-class classification.

Best values of γ (the RBF parameter) and C (the SVM regularization parameter) were found by performing grid search with γ in the range $2^{-15}, 2^{-13}, ..., 2^{3}$ and C in the range $2^{-5}, 2^{-3}, ..., 2^{15}$. The features in Feature Set I were computed using Maple [9], a computer algebra system.

5. Experimental Results

Publicly available datasets [12] were used for training and testing. The training set consisted of handwritten samples from 11 writers (seven male and four female) collected using a Hewlett-Packard Laboratories (HP) Compaq tc1100 Tablet PC. The symbol set consisted of 48 different symbols including *a*-*z*, 0-9, $\sum$, (,), $-$, $\sqrt{}$, $\int$, $\{$, $<$, $>$, $+$, $\neq$ and *else*. The training dataset consisted of samples written 10 times by each writer and testing dataset consisted of samples from 11 writers written 12 times each. The data was converted to InkML format [8] to conform to our system [7] and for future portability.

Two SVM classifiers were trained separately using the two sets of features mentioned in Sections 3.1 and 3.2.

- **Feature Set I**: The orders of the Chebyshev series (maximum value of i in (2)) were set to 11. The best of values γ and C were found by a 5-fold cross validation technique with grid search on the training set. The error rate of the system after testing was found to be 7.13%.
- **Feature Set II**: All the samples were resampled to a total of 30 points (Feature Set IIa). The best values of γ and C were found by 5-fold cross validation technique on the training set. The system had an error rate of 5.43% on the test dataset. In order to better compare the two systems, the samples were resampled to 11 points (Feature Set IIb). The new error rate on the test data was 6.51%. Another experiment was performed using resampled X and Y coordinates only (11 points) as feature vectors (Feature Set IIc). The experiment showed an error rate of 7.28% on the training data, which is a little worse than for Feature Set I. It was observed that removing other features from Feature Set IIb increased the number of support vectors.

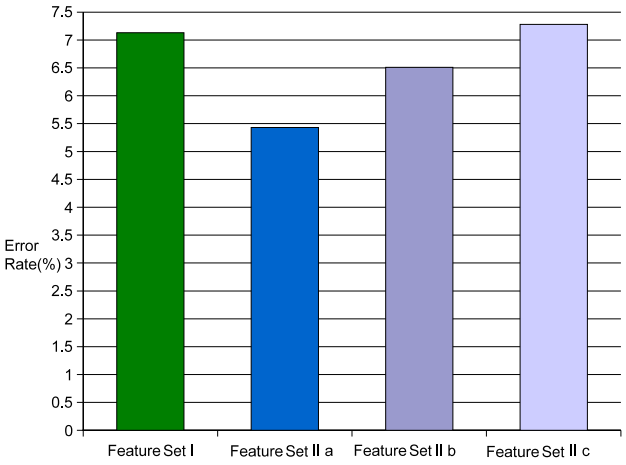

Figure 5. Comparison of error rates of different feature sets.

6. Conclusions and Future Work

We have presented a SVM-based mathematical symbol recognizer which used features obtained by the functional approximation of the digital ink trace. We compared the discriminating capability of such features with other features. The experimental results showed that with the same number of coordinate features, the functional approximation technique achieved accuracy slightly better than the coordinate features obtained after resampling.

One of the main advantages of basing an SVM on character features derived from numerical functional analysis techniques is device and scale invariance. Once the degree of the functional approximation is given, no heuristic analysis is required to determine suitable resampling and ensuring resampling doesn't obscure features. The features (series coefficients) can be computed without resampling, and trace curves recognized directly. Indeed, characters can be completely represented by these feature vectors and traces recovered from them if needed. Representing characters in this way allows many high-level geometric computations to be computed by analytic formulae rather than by further numerical approximation.

We have determined the degree of functional approximation necessary to obtain the same accuracy as standard methods for a given problem. The question remains what degree of functional approximation *should* be used in general? To see this, we think geometrically about the symbol set. What is the number of cusps and turns that must be represented to obtain a recognizable glyph? That number determines the maximum degree of the functional approximation. Alternatively, one can start with a notion of

maximum desired error and determine the degree of the Chebyshev approximation required to bound this error.

The functional approach to feature extraction for mathematical symbol recognition appears to be an interesting and useful direction. We believe that there are many promising extensions to further improve the classifier we have described. In the future, we plan to further explore the system by transforming $X(t)$ and $Y(t)$ to a geometric invariant frame. Such features would help to incorporate the local change in direction information into the feature set which might improve the accuracy of the system. We anticipate that such features when combined with the current approach could make the system robust against orientation differences of the samples. While further experiments are needed to determine whether our approach is as competitive with other character sets, there is nothing in our method that is specific to the symbol set of the experiment. (All we have relied upon is that characters are isolated.) It is therefore a rather appealing prospect that we might have a general approach that does not need to be re-coded and re-tuned for each new character set.

References

[1] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines*, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[2] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, 1998.

[3] Bruce W. Char and Stephen M. Watt, "Representing and Characterizing Handwritten Mathematical Symbols through Succinct Functional Approximation", *Proc. ICDAR*, 2007.

[4] Tapia, Ernesto and Rojas, Raul, "Recognition of On-Line Handwritten Mathematical Expressions in the E-Chalk System-An Extension", *Proc. ICDAR*, 2005.

[5] Stephen M. Watt and Xiafang Xie, "Recognition for Large Sets of Handwritten Mathematical Symbols", *Proc. ICDAR*, 2005.

[6] Elena Smirnova and Stephen M. Watt, "Combining Prediction and Recognition to Improve On-Line Mathematical Character Recognition", *Proc. ICFHWR*, 2008.

[7] Birendra Keshari and Stephen M. Watt, "Hybrid Approach to Mathematical Symbol Recognition Using Support Vector Machines", *Proc. ICDAR*, 2007.

[8] Yi-Min Chee and Max Froumentin and Stephen M. Watt (editors), *Ink Markup Language (InkML)*. `http://www.w3.org/TR/InkML/`, 2006.

[9] Maplesoft. *Maple User Manual*. Maplesoft, 2007.

[10] Eugene H. Ratzlaff, "Methods, Report and Survey for the Comparison of Diverse Isolated Character Recognition Results on the UNIPEN Database", *Proc. ICDAR*, 2005.

[11] Claus Bahlmann, Bernard Haasdonk and Hans Burkhardt, "On-line Handwriting Recognition with Support Vector Machines - A Kernel Approach", *Proc. IWFHR*, 2002.

[12] `http://www.graphics.cs.brown.edu/research/pcc/`.

[13] C. S. Sundaresan and S. S. Keerthi, "A study of representations for pen based handwriting recognition of tamil characters", *Proc. ICDAR*, 1999.

[14] J.J. LaViola and R.C. Zeleznik, "A Practical Approach for Writer-Dependent Symbol Recognition Using a Writer-Independent Symbol Recognizer", *IEEE Transations on Pattern Analysis and Machine Intelligence*, Vol 29-1, 2007, pp 1917-1926.

[15] Chih-Wei Hsu and Chih-Jen Lin, "A Comparison of Methods for Multi-class Support Vector Machines", *IEEE Transactions on Neural Networks*. Vol 1-13, 2002, pp 415-425.