Hybrid Mathematical Symbol Recognition using Support Vector Machines

Birendra Keshari and Stephen M. Watt
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
{bkeshari,watt}@orcca.on.ca

Abstract

Recognition of mathematical symbols is a challenging task, with a large set with many similar symbols. We present a support vector machine based hybrid recognition system that uses both online and offline information for classification. Probabilistic outputs from the two support vector machine based multi-class classifiers running in parallel are combined by taking a weighted sum. Results from the experiments show that giving slightly higher weight to the on-line information produces better results. The overall error rate of the hybrid system is lower than that of both the online and offline recognition systems when used in isolation.

1. Introduction

Providing mathematical input to computer applications in handwritten 2D-form promises to be much more convenient and natural than in a linear form form because of the 2D syntax of common mathematical notation. The major challenges in recognizing handwritten mathematical symbols are the large symbol set, many similar looking symbols, diagrams etc.

Experiments by various researchers have shown support vector machines (SVMs) to be well suited for symbol recognition. In [12], SVMs were shown to outperform other learning algorithms such as neural nets, HMM and nearest neighbor. According to the comparison results in [8], SVMs perform better than HMM on different UNIPEN data subsets. We further explore multi-class SVMs that produce posterior probabilities instead of a single label for the task of hybrid mathematical symbol recognition.

Offline symbol recognition operates on the image of the symbol. On the other hand, online systems operate on ink strokes obtained directly from pen devices such as graphics tablets, Tablet PCs and PDAs. The task of recognizing symbols from their images is usually more challenging than recognizing them from the traces due to the fact that the online information such as trace points, time, pen status and pressure, provide complete information about how the symbol was written. But treating the online strokes as an image also has advantages. For example, a writer can write the same symbol using different stroke orders, stroke directions and stroke numbers. This can make the online features ambiguous, and although the symbols look similar, such symbols easily get mis-classified, especially if the pattern is not present in the training set. Such problems can be tackled by treating the ink strokes as an image. Thus, offline and online information can be complementary to each other and combining them can make the recognition system more robust.

Previous work has been done on combining online and offline features to improve online and offline recognition [5, 6, 11]. Very closely related work is done in [14], which combines Hidden Markov Model based online and offline recognizers. The input data is first recognized by an online recognizer and then converted to an offline bitmap which is recognized by the offline system. Their experiments on digit recognition show that offline and online features are complementary to one another. In this paper, we use multi-class SVM classifiers that produce probability estimates according to the method described in [15] and show our results on a large set of mathematical symbols. Offline features are extracted in a different way than in [14] and different online features are used.

We discuss support vector machines in Section 2. Section 3 describes the system architecture. Online and offline recognizers are presented in Sections 4 and 5. We discuss the implementation in Section 6. Section 7 presents the experimental results and finally conclusions are drawn in Section 8.

2. Support Vector Machines

SVMs are supervised learning methods that have been widely and successfully used for pattern recognition in different areas. SVMs are based on the dual ideas of *VC* *dimension* and *structural risk minimization principle* [13]. The decision boundary in SVMs is a hyperplane that separates the two classes, leaving the largest margin between the vectors of the two classes. However, in real life, problems can be linearly in-separable. To deal with this problem, a non-linear decision surface is obtained by lifting the feature space into a higher dimensional space. A linear separating hyperplane is found in the higher dimensional space that gives a non-linear decision surface in the original feature space. The decision function of the SVM can be expressed as follows

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

where y_i is the label of training pattern x_i and x is the pattern to be classified. Parameters α_i and b are found by maximizing a quadratic function subject to some constraints [13]. $K(x, x_i) = \phi(x) \cdot \phi(x_i)$ is the kernel function, where ϕ maps the feature vectors into a higher dimension inner product space. The most commonly used kernels are:

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

Although a SVM is primarily a binary classifier, multi-class classifiers can be created by combining several binary classifiers. One-against-all, one-against-rest and DAG SVM are the popular techniques to combine binary classifiers to build multi-class SVMs. Comparisons between different methods in [3] show that DAG and one-against-one are more suitable for practical use than the other existing methods.

The outputs from standard SVMs are not calibrated probability estimates. More interesting tasks such as post-processing and combining classifiers can be done with SVMs when the outputs are probability estimates instead of labels, and obtaining posterior probabilities with SVMs is an important topic. A good measure can be the distance of the pattern from the hyperplane. Platt [7] developed a method to transform this distance into a posterior probability by applying a sigmoid function on the outputs of the SVM as follows:

$$p(y = 1|f(x)) = \frac{1}{1 + \exp(Af(x) + B)} \tag{2}$$

where $f(x)$ is the output from the SVM. Parameters A and B are obtained by minimizing the negative log-likelihood function on training data. An efficient way to do this has been described in [15] which has also been implemented in the `libsvm` [1] library.

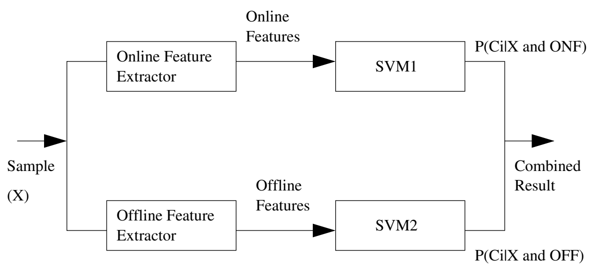

Figure 1. System architecture.

3. System Architecture

Our system can be viewed as the composition of four modules: *Offline* and *Online Feature Extractors* and *Offline* and *Online SVM based classifiers*, as shown in Figure 1. To recognize sample X, copies of its ink strokes are provided to both the online and offline feature extractors. These feature extractors extract the online and offline features after preprocessing the sample and provide the feature vectors to SVM1 (online classifier) and SVM2 (offline classifier) respectively. The online and offline components may be run in parallel. Online classifier outputs $P(C_i|X \text{ and } ONF)$, the probability that the sample belongs to class C_i given the sample's online features. Similarly, the offline classifier outputs $P(C_i|X \text{ and } OFF)$, the probability the sample belongs to class C_i given the sample's offline features. The probability estimates from both classifiers are combined to decide the final class of the sample.

4. Online Recognizer

Each sample is pre-processed and feature vectors are extracted to train the online SVM and to predict the sample as described below.

4.1. Preprocessing

The following steps are involved in preprocessing:

- **Smoothing:** The following central smoothing is used to remove noise:

$$\begin{aligned} x_i &= 0.25x_{i-1} + 0.5x_i + 0.25x_{i+1} \\ y_i &= 0.25y_{i-1} + 0.5y_i + 0.25y_{i+1} \end{aligned}$$

- **Filling intermediate points:** We fill the intermediate points between every pair of consecutive points on each stroke by linear interpolation. This removes the time information but we found that spatial alignment is more useful than temporal alignment of the points on

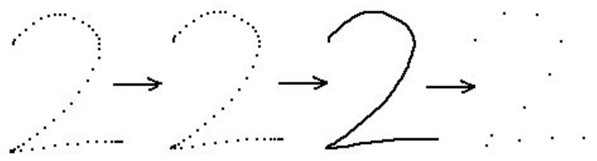

Figure 2. Processing input sample by smoothing, filling and resampling.

the strokes. Hence, we place the points on the strokes at equal distance during resampling.

- **Resampling:** We resample the points to reduce the total number of points on each stroke to 11. This is done by selecting every $N/11^{th}$ point, where N is the total number of points on the stroke.
- **Size Normalization:** Each stroke is scaled by $1/\max(h, w)$, where h and w are height and width of the bounding box of the symbol. Strokes are then translated so that the whole symbol fits inside a square of unit length and their centers coincide.

Figure 2 illustrates these preprocessing steps.

4.2. Online Feature Vector

The online feature vector for each stroke is extracted after the symbol is preprocessed. The feature vector consists of coordinates of each point on the stroke, sines and cosines of the angles made by the line segments on the stroke, sines and cosines of the turning angles between line segments and the center of gravity of the symbol. Turning angle is calculated as shown in Figure 3. Center of gravity is $(\sum_i (x_i/N), \sum_i (y_i/N))$, where N is the total number of points and (x_i, y_i) are the coordinates of each point. Similar features have been used in [12]. It was found that coordinates and angle information are the most discriminating features. The decrease in error rate of the online system was very small when the size of the feature vector was increased by introducing new geometric features such as relative length. However, the number of support vectors was reduced.

5. Offline Recognizer

For the offline recognizer the online data for an isolated symbol has to be converted into an image. In [14] the points obtained from the online data are connected by straight lines and filter growing is applied to the foreground area three times to obtain the offline image before scaling down. In our case, we store the sample files in InkML [2] format. These files are loaded and rendered using black ink on a white background using Microsoft's Tablet PC SDK. Since pressure information is used to give stroke width while rendering, this information is available to the offline recognizer. The area under the bounding box of the symbol is captured and converted into a bitmap. We scale down the original bitmap by a factor of $15/\max(w, h)$, with w and h the width and height of the symbol bounding box, using a high-quality bilinear interpolation filter. We pad extra white pixels horizontally or vertically to make the bitmap 15×15 pixels in size. This method also preserves the aspect ratio. Figure 4 shows the rendered sample input and the zoomed version of its offline feature.

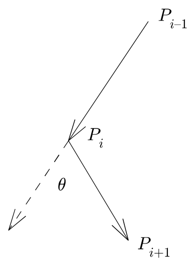

Figure 3. Turning Angle (θ).

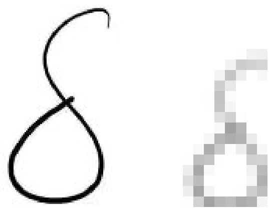

Figure 4. Original sample and its scaled down version (zoomed).

The final feature vector to be used by the SVM classifier is $(I_1, I_2, ..., I_N)$ where I_i is the intensity (gray level) of the pixel at point i and $N = w \times h$, the total number of pixels in the image.

6. Implementation

Our implementation employs the `libsvm` library [1] to train the classifiers and predict new test samples. Feature extractors generate the feature vector in a format usable by `libsvm` and features are scaled between 0 and 1. Radial basis functions (RBF) are used as the kernel for both of the classifiers. We use the one-against-one strategy for multi-class classification and posterior probabilities are obtained through an optimization performed on the pairwise class probabilities [15].

Best values of parameters γ and C are obtained by performing a grid search using cross-validation on the data by taking $\gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$ and $C = 2^{-5}, 2^{-3}, ..., 2^{15}$.

The outputs from the two classifiers are combined by taking the weighted sum as follows

$$P(C_i|X) = \alpha P(C_i|X, ONF) + \beta P(C_i|X, OFF) \quad (3)$$

where α and β are weights assigned to the online and offline classifiers respectively ($\alpha + \beta = 1$). There are many approaches to combining classifiers. We decided to take the weighted sum of the probabilities from the two classes after studying the results presented in [4].

7. Experiments and Results

The symbol set consisted of 137 unique mathematical symbols which can be categorized into Latin characters (upper and lower), digits, Greek characters, relational operators, arrows, basic operators, logical operators, delimiters, special characters and miscellaneous characters. Samples were collected from 5 different users using a Tablet PC having an ink resolution of 24570×18428 and pressure levels from 0 to 255. The size of the data was increased artificially 5 times using the tangent distance method explained in [9]. The data was collected and stored in InkML format for better portability. Separate experiments were carried out for writer–dependent and writer–independent tests to observe the effect of combining the recognizers on the overall performance of the system.

7.1. Writer Dependent Test

Symbols were read one by one from each user's sample file and separated uniformly into 5 files (index moduluo 5). One of these files was used for training and the rest for testing. Thus, a training size of 20% was taken and 5 experiments were carried out by training the classifier on one file and testing it on the balance. Parameters γ and C were determined by performing a grid search using 5-fold cross-validation on the training set with the help of the grid search tool available in [1] . The quantities Acc^i_{on} and Acc^i_{off}, denoting online and offline accuracies in the i^{th} experiment, were determined by evaluating online and offline classifiers separately on the test data. Different values of α and β were plugged into equation 3 to determine the combined probability. Based upon the combined probability, the overall combined accuracy Acc_{overall} was calculated. We define the change in overall combined accuracy in the i^{th} experiment, $\triangle Acc^i$, with respect to the maximum of the online and offline accuracies as follows

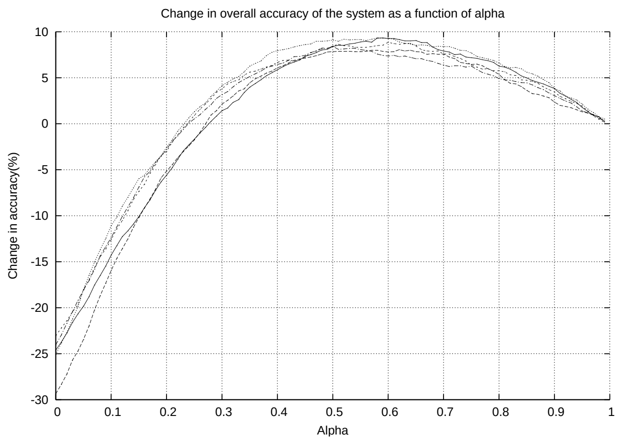

Figure 5. Effect of alpha on overall combined accuracy in 5 experiments (writer dependent).

$$\triangle Acc^i = Acc^i_{overall} - max(Acc^i_{on}, Acc^i_{off}). \quad (4)$$

Figure 5 shows the plot of α verses change in combined accuracy for each of the 5 experiments. An average change in combined accuracy has been shown in Figure 6.

It is observed that the maximum gain in accuracy is about 10% for online recognizer and about 35% for the offline recognizer with a training size of 20%. The best choice of α is difficult. However, as we can see from Figure 6, setting $0.5 < \alpha < 0.6$ seems to be most appropriate. In [14], the error rate of the online recognizer was decreased by 43%. But in our case, the online recognizer has a lower error rate than the offline, and therefore the offline recognizer benefited more from the combination. One of the main reasons is the differences in the features. These experiments show that online and offline features are complementary and their combination is fruitful.

7.2. Writer Independent Test

For the writer independent test, we used sample files from 4 users as training data and the one remaining file as test data. This was repeated 5 times by taking different combinations of sample files. Parameters γ and C were determined by performing cross-validation similar to that described in the previous section. The plot of the average change in combined accuracy as a function of α is shown in Figure 7. The maximum increment in the accuracy of the online recognizer is about 5% and that of the offline recognizer is about 22%. Values of α between 0.6 and 0.7 give better results.

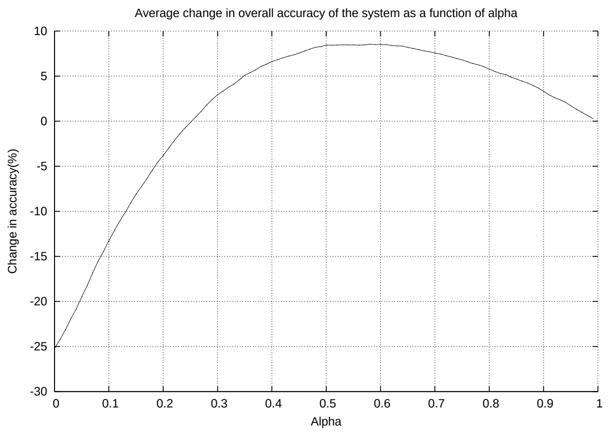

Figure 6. Average effect of alpha on overall combined accuracy (writer dependent).

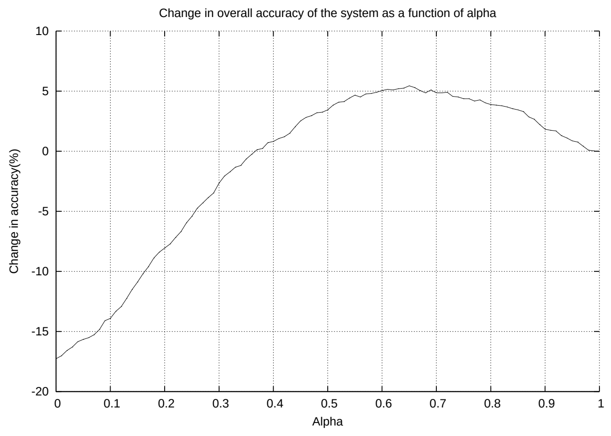

Figure 7. Average effect of alpha on overall combined accuracy (writer independent).

8. Conclusions and Future Work

We have presented a hybrid support vector machine based mathematical symbol recognizer that performs better than online and offline recognizers alone. Using probability estimates from a SVM to combine the classifiers has been shown to be very useful for the problem of mathematical symbol recognition.

In particular, for mathematical symbol recognition, we also believe that mathematical 2D context information can play an important role. In the future, we plan to integrate context information with this system. A 2D bigram/trigram model can be automatically extracted from large mathematical corpora and can be used to predict symbols given context information. We believe that the combination of such a predictor with the offline and online recognizers can make the system more robust.

References

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

[2] Y.-M. Chee, K. Franke, M. Froumentin, S. Madhvanath, J.-A. Magaa, G. Russell, G. Seni, C. Tremblay, S. M. Watt, and L. Yaeger. Ink markup language (inkml), October 2006. `http://www.w3.org/TR/InkML/`.

[3] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(1):415–425, August 2002.

[4] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[5] P. Lallican, C. Viard-Gaudin, and S. Knerr. From off-line to on-line handwriting recognition. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, pages 303–312, 2000.

[6] S. Manke, M. Finke, and A. Waibel. Combining bitmaps with dynamic writing information for on-line handwriting recognition. In *Proceedings of International Conference on Pattern Recognition*, pages 596–598, 1994.

[7] J. C. Platt. *Probabilistic outputs for support vector machines and comparison to regularized methods*, pages 61–74. MIT Press, 1999.

[8] E. H. Ratzlaff. Methods,report and survey for the comparison of diverse isolated character recognition results on the unipen database. In *Proceedings of ICDAR*, 2003.

[9] H. Schwenk and M. Milgram. Contraint tangent distance for on-line character recognition. In *International Conference on Pattern Recognition*, pages 515–519, August 1996.

[10] E. Smirnova and S. Watt. Combining prediction and recognition to improveon-line mathematical character recognition. Technical report, University of Western Ontario, 2006. `http:\\www.orcca.on.ca\TechReports\TR-06-06`.

[11] H. Tanaka, K. Nakajima, K. Ishigaki, K. Akiyama, and M. Nakagawa. Hybrid pen-input character recognition system based on integration of online-offline recognition. In *Proceedings of ICDAR*, pages 209–212, 1999.

[12] E. Tapia and R. Rojas. Recognition of on-line handwritten mathematical expressions in the e-chalk system-an extension. In *Proceedings of ICDAR*, 2005.

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

[14] A. Vinciarelli and M. Perrone. Combining online and offline handwriting recognition. In *Proceedings of ICDAR*, pages 844–848, 2003.

[15] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 99(5):975–1005, August 2004.