A Structure for Adaptive Handwriting Recognition

Vadim Mazalov and Stephen M. Watt
Department of Computer Science
University of Western Ontario
London, Canada
{*vmazalov, Stephen.Watt*}*@uwo.ca*

Abstract

We present an adaptive approach to the recognition of handwritten mathematical symbols, in which a recognition weight is associated with each training sample. The weight is computed from the distance to a test character in the space of coefficients of functional approximation of symbols. To determine the average size of the training set to achieve certain classification accuracy, we model the error drop as a function of the number of training samples in a class and compute the average parameters of the model with respect to all classes in the collection. The size is maintained by removing a training sample with the minimal average weight after each addition of a recognized symbol to the repository. Experiments show that the method allows rapid adaptation of a default training dataset to the handwriting of an author with efficient use of the storage space.

1 Introduction

Hardware support for digital handwriting has reached its maturity, while algorithms for handling and recognizing 2D input are still evolving. Our objective is to develop an online adaptive handwriting recognition algorithm, efficient in terms of the storage space, and to test the method on the dataset of handwritten mathematical characters. Mathematics is one of the most difficult forms of handwriting to be recognized due to non-trivial syntactic verification and the large set of classes (that include Latin and Greek alphabets, digits, operators and special characters) with many similar-looking symbols written in non-linear form. Nevertheless, some progress has been made by representing x and y coordinates of a sample as parameterized functions and approximating the functions with truncated series of orthogonal polynomials. The samples are classified with the distance to the convex hull of k nearest neighbors in the space of coefficients of approximation [2]. The method yields high accuracy, but has a significant drawback – it does not adapt to variations in writing style of trained classes. This is not acceptable in a production environment, since out of the box recognition applications are usually trained with a default dataset of samples. Such dataset relieves the user from an exhaustive training of a mathematical recognizer that may include several hundred classes. However, default training of some classes may differ from the writing style of the user. This concern is aggravated for online algorithms that typically depend on the direction and order of writing of strokes. Therefore, instances that appear identical visually, but written in different styles, will be represented by points, positioned in absolutely different locations in the coefficients space. Thus, some training samples may represent noise and have negative impact on efficiency and accuracy.

The exemplar-based learning in higher dimensions is challenging due to the increase of sparsity of samples of a class. Therefore, selection of training exemplars has been thoroughly studied in instance-based machine learning and related applications [6]. Some methods suggest to retain a subset of the original instances [3, 1], while other techniques propose to compute prototypes from the training data [4]. Due to the nature of our classification method, we investigate the former approach. It can be divided in *incremental* (start from an empty training set and add instances one by one), and *decremental* (start from the training set with all samples and remove instances that are redundant or decrease accuracy). A decremental procedure *DROP1* [6] suggests to remove a point if all of its neighbors can still be correctly classified without the point. This and many other techniques [3, 1] study the local relationship between samples without taking into account that the training dataset may change over time, moving the underlying points in various directions.

We develop an online algorithm for adaptive recognition of handwritten characters that is based on rein-

978-0-7695-4774-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICFHR.2012.169

forcement of samples that have positive impact on classification and removal of samples that cause error or are neutral. The method is suitable in both settings: When users train a recognizer from scratch or when they use the default dataset as the starting point. In the latter setting, to determine the average size of a training class, we model the error drop as a function of the number of samples and attempt to correlate parameters of the model with some spatial measurements of the class.

The proposed adaptive algorithm computes the participation weight of each of the k neighbors in a correct (incorrect) recognition and adds (subtracts) the value to (from) the total weight of the neighbor. In a sense, the method is similar to the *IB3* algorithm [1], in which removal or retaining of instances is based on counters. However, the *IB3* method is offline, meaning that it is run only once to select good classifiers out of the pool of training samples, while our algorithm is online and makes removal decisions with each new sample available from the input. The method presented has potential of asymptotic improvement in performance over the course of its use and is suitable for a variety of instance-based machine learning applications. Unlike some algorithms, based on neural networks or hidden Markov models, the proposed technique uses only gradual updates, making it suitable for real-time applications.

The main results of this paper are

- an experimental analysis of how error rate drops as a function of the class size;
- an empirical model for the error rate, fitting the experimental data well, to determine the average size of a class for desired accuracy;
- an adaptive algorithm for distance-based symbol recognition, using the functional approximation framework.

This paper is organized as follows. Section 2 describes some basic preliminaries. Section 3 explains our approach to modelling the recognition error. The adaptive recognition algorithm is presented in Section 4. Section 5 gives the experimental results that show good approximation of the error function and rapid adaptation of the recognition algorithm to the writing style of a user. Finally, Section 6 concludes the paper.

2 Preliminaries

In online classification environment, a character is given as a sequence of points available from the digital pen. For a single-stroke sample, coordinates of points are represented as parameterized functions $X(\lambda)$ and $Y(\lambda)$, where parameter λ can be time, arclength, etc. These functions are approximated with truncated orthogonal polynomial basis. The 0-order coefficients of approximation regulate the initial position of the character and can be neglected to normalize location of the symbol. Dividing the rest of the vector by Euclidean norm can normalize the sample with respect to size. For a multi-stroke character, all strokes are joined consecutively and the resulting stroke is processed as it is described for a single-stroke symbol. Normalized coefficients of characters can be regarded as features describing objects to be classified. The recognition algorithm is based on the distance to convex hulls of k-nearest neighbors in the space of coefficients [2].

3 Modelling the Recognition Error

In our classification paradigm, the concept of personalized recognition can be reformulated as continues formation of the training set. A set of training characters of a class forms a cluster in the space. *A priori* knowledge of the average initial size of a training class to achieve a desired classification accuracy is important for compact storage. It has an additional usability-related benefit: When a new class is introduced to the dataset, the user can be informed about the expected error drop depending on the number of samples introduced to the class.

Here and below, we will use the following notation: n is the number of training samples that the class contains in a given moment and N is the maximal number of training samples available in the class. Based on our observation, convergence of the recognition error of samples of a class can be closely described by the models

$$\epsilon(n) = \frac{An + B}{n + C} \tag{1}$$

where A, B and C are parameters, or

$$\epsilon(n) = \alpha e^{-\beta f(n)} \tag{2}$$

where α and β are parameters, and $f(n)$ is a monotonically increasing function.

Our objective is to find values of the parameters for each class. We expect the parameters to be dependent on some inner properties of a class, as well as the positioning of the class relative to neighboring classes. Further, the mean parameters can be used to describe the average error drop.

4 Adaptive Recognition

Most commonly, misclassification of handwritten characters occurs when different samples are written

similarly, since writing styles of users can vary significantly. On the other hand, classes of characters provided by one user can usually be discriminated well. As discussed in Section 2, only k samples of a candidate class are used in classification of a test symbol. Each of these k exemplars should be awarded a weight, computed as a function of the distance to the test sample. If the training symbol is located relatively close to the test character, the weight should have large absolute value, otherwise the weight should be close to zero. If the training sample is of the same class as the test symbol, the weight should be positive and otherwise – negative.

In general, distances between training samples within a class do not follow any of the major univariate distributions, since a class may contain several styles that group the exemplars. Therefore, basing the weight on statistical properties of a class can be quite challenging. Instead, we take the weight as follows: For a given test sample t_s and a training exemplar t_i, the recognition weight has the form

$$w_{t_i} = \frac{1}{d(t_s, t_i) + 1}$$

where $d(t_s, t_i)$ is the distance between the points. This weight is added to the total weight of the sample t_i, if t_s and t_i belong to the same class, and subtracted otherwise.

When a new sample is recognized, it is added to the class, and simultaneously a sample with the minimal average weight is removed from the dataset to prevent its growth. Nevertheless, at any given moment, the size of a class should not be less than k (the number of nearest neighbours that form convex hull during classification). The outline of the method is presented in Algorithm 1.

Algorithm 1 Adaptive recognition algorithm

Input: t_s – a test sample to be recognized.
{Recognize the sample as explained in Section 2}
$Cl \leftarrow$ recognition class of t_s
{Recompute weights}
for $i = 1 \rightarrow T$ **do**
  {For each of the candidate classes}
  if $T_i = Cl$ **then**
    for $j = 1 \rightarrow k$ **do**
      {Increase the weight of each nearest neighbor t_{ij} in the correct class}
      $w_{t_{ij}} \leftarrow w_{t_{ij}} + \frac{1}{d(t_s, t_{ij})+1}$
    end for
  else
    for $j = 1 \rightarrow k$ **do**
      {Decrease the weight of each nearest neighbor t_{ij} in the incorrect class}
      $w_{t_{ij}} \leftarrow w_{t_{ij}} - \frac{1}{d(t_s, t_{ij})+1}$
    end for
  end if
  {Increase the counter}
  for $j = 1 \rightarrow k$ **do**
    $C_{t_{ij}} \leftarrow C_{t_{ij}} + 1$
  end for
end for
{Remove the exemplar with the minimal average weight among the classes with the number of samples $> k$}
Remove exemplar $t : \overline{w}_t = \min_{ij}\{\frac{w_{t_{ij}}}{C_{t_{ij}}}, |T_i| > k\}$
Assign an initial weight to t_s and add t_s to the recognized class.

5 Experimental Results

This section presents experimental results of modelling the recognition error and the adaptive classification method. The experimental dataset is identical to the one described in [2].

5.1 Modelling the Recognition Error

We conducted a series of experiments to measure how the recognition rate changed as points were added to the classes. Each class was measured separately, in the following manner: All symbols from the class to be tested were removed from the training data set and the symbols from other classes were retained. Further, the samples from the test class were separated randomly into a test set P_i and a training set P_r. Then the symbols from P_r were added, initially one at a time and then in larger groups. After each addition of points, the recognition rate of the ensemble was measured using the test set. Thus, for each class, the recognition rate was tested first with 0 training points, then with 1 training point, then with 2, then after 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56, 64, ... until all the training points were used. The number of training points ranged from 10 to 2048, depending on the class. This whole process was repeated ten times, and the recognition rate recorded for a class after a particular number of points was reported as the average of these ten measurements. The testing sets were selected randomly, but disjoint. The set of classes is denoted as Ω. The outline of experiments is given in Algorithm 2.

Some of the samples have several class labels. Therefore, the recognition error can be less than 100%, even if the class has zero training samples in it. Results

Algorithm 2 Outline of the experimental setting

for Each class ω in the set of classes Ω **do**
  Split samples in the class for 10-fold cross-validation.
  for $i = 1$ to 10 **do**
    Take the i-th part P_i for testing and the rest P_r for training.
    {Introduce integer variables used in splitting the training set.}
    $s \leftarrow 0, k \leftarrow 3$
    while $s \leq |P_r|$ **do**
      Clear the training set for the class ω.
      Conduct training with the first s samples from P_r.
      Conduct testing with samples from P_i.
      if $s = 2^k$ **then**
        $k \leftarrow k + 1$
      end if
      {Increase the amount of training samples}
      $s \leftarrow s + 2^{k-3}$
      {where 2^{k-3} was selected heuristically, based on the observation that adding samples to a small training set has bigger impact than to a larger set}
    end while
  end for
end for

of recognition for all classes, depending on n, are given in Figure 1.

We make a few observations: First, we see that for all classes the recognition rate improves dramatically with each of the first few symbols added. Most of the functions have shape that can be modelled with (1). For approximation, we used the Nonlinearfit Maple [5] command to evaluate A, B and C. In classes with more than a few dozen samples, the error rate appeared to drop off similarly to a negative exponential function (2). In (2), $f(n) = \sqrt{n}$ was found to perform well. By taking the logarithm of both sides, the parameters can be evaluated as a linear regression between $\log(\epsilon(n))$ and $\sqrt{n}$. We used the LeastSquares Maple command to compute the least squares approximation.

We tested both models (1) and (2) and computed the average root mean square error (RMSE) among classes, obtaining respectively 0.03 and 0.87. Model (1) performed the better of the two, and so this is the one upon which we have concentrated. Examples of approximation with (1) for different values of N and the average model are given in Figure 2. We observed that classes of smaller size, with $N < 64$, are approximated not as good as larger classes, partially due to non-stable behaviour of the error function on the small testing set. Therefore, the mean parameters A, B and C were computed among classes with ≥ 64 training samples. The mean and the standard deviation of the variables are shown in Table 1. The small mean value of parameter A can be considered as an argument that the error model (1) can be simplified to $\epsilon(n) = \frac{B}{n+C}$. On the other hand, such simplification will make the model less robust and may have negative effect on the approximation accuracy. Therefore, we decided to keep the parameter.

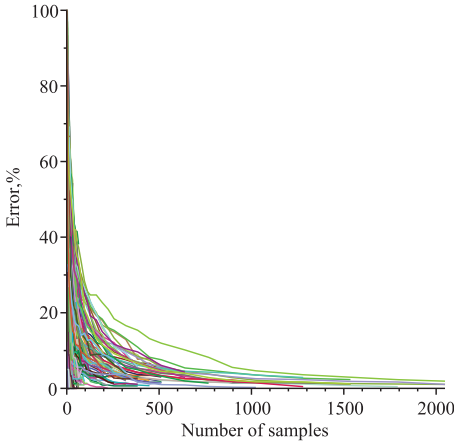

Figure 1. Recognition error for all classes, depending on n, the number of training samples in a class

	A	B	C
Mean	-0.007	11.718	23.398
σ	0.054	9.805	9.805

Table 1. The mean and the standard deviation of the parameters

The average RMSE between the modelled recognition rate and the actual recognition rate for classes of certain size is presented in Figure 3(a). Figure 3(b) shows the percentage of classes that are approximated with RMSE less or equal a given value.

5.2 Correlation between class measurements and A, B and C

We question whether parameters A, B and C are related to spatial characteristics of the class, such as positioning of points within the class and distance to neighboring classes. For each class i, the following measurements are considered (in Euclidean distance)

- R_1^i - the maximal distance from the class center to

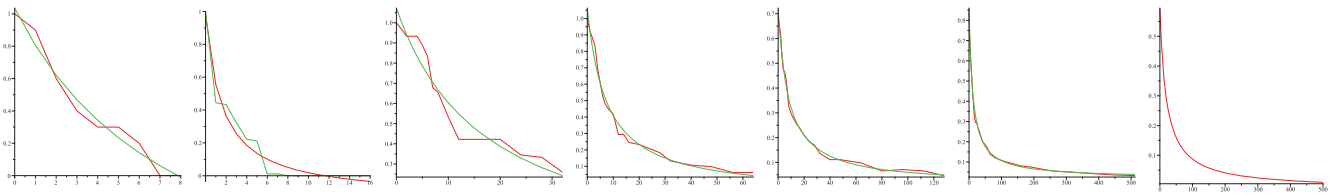

Figure 2. Examples of approximation of error for classes of different size N

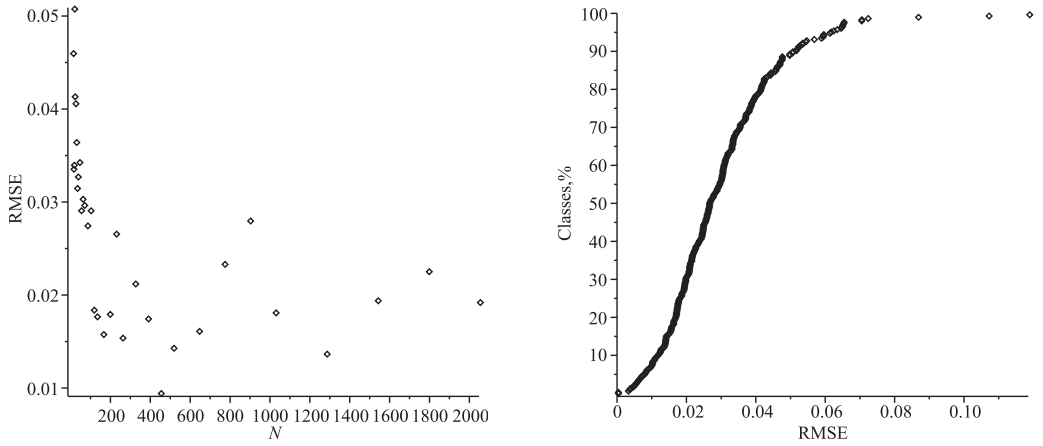

Figure 3. RMSE results: (a) Average RMSE for classes of different N, (b) Percentage of classes that are approximated with RMSE less or equal given RMSE

any point in the class.

- $R^i_{.75}$ - the minimum radius of a ball centered at the class center that encloses 75% of points in the class.
- R^i_a - the average of radii from all points in the class to the class center.
- $R^i_\sigma = R^i_a + \sigma_i$, where σ_i is the standard deviation of the radii from points in the class to the class center.
- D^i_F - the minimum distance between points of the class to the closest neighboring class.

In addition, we study the measurements

$$\check{D}^i_L = \min_{j \neq i}(d_{ij} - R^i_L - R^j_L),$$
$$\bar{D}^i_L = \operatorname{avg}_{j \neq i}(d_{ij} - R^i_L - R^j_L)$$

where L is any of the labels $1, .75, a, \sigma$ and d_{ij} is the distance between centers of classes i and j.

	Measurement	Spearman	Kendal tau-a
A	$\check{D}_1$	-0.29	-0.19
B	$\check{D}_\sigma$	-0.55	-0.39
C	$\check{D}_\sigma$	-0.59	-0.42

Table 2. The measurements with the largest absolute values of the correlation coefficients for each approximation variable

Spearman and Kendall tau-a tests did not demonstrate sufficient correlation of these measures with the model parameters. The largest absolute values of statistically significant correlation coefficients for corresponding class measurements are presented in Table 2.

5.3 Adaptive Recognition

For this experiment, each character in the collection is assigned to the author who provided the symbol.

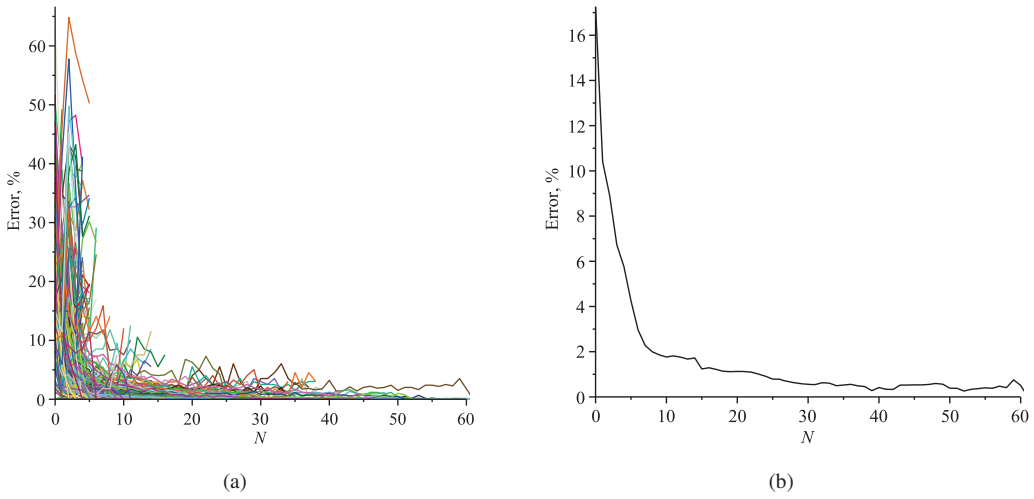

Figure 4. Adaptive recognition error of the $(N+1)$-th sample in a class: (a) For each author, (b) Average among the authors

Then for each author, the dataset is split in two parts: samples provided by the author (used in testing) and the rest of the dataset. During the training phase, for each class, we randomly select K samples and form the default training set. The value of K, the initial size of a training class, can be determined from the error modelling, and for this experiment we take $K = 30$. During the testing phase, a test sample is extracted from a randomly chosen class among those written by the test author and recognized. The recognition error of the N-th sample by the author is computed as the ratio of the number of misrecognitions of the N-th sample to the total number of N-th samples tested. This run is repeated 200 times and the average for each author is reported in Figure 4(a). Figure 4(b) shows the average error among all the writers. We observe that the adaptive algorithm on average results in a rapid decrease of error and converges to $\approx 99\%$ accuracy.

6 Conclusion

We have shown how handwriting recognition techniques based on functional approximation methods are well suited to adaptive setting. Rather than organizing the workflow as a training phase followed by a use phase, we see continuous improvement of recognition results taking advantage of correction history. In our setting, based on convex hulls of classes in the coefficient space, adaptation consists of weight-based evolvement of the shape of the class envelopes. The experiments have shown that the error rate drops approximately as $(An + B)/(n + C)$ as samples are seen, and that A, B and C slightly vary by class and correlate with class measurements to a minor degree. The average values of the parameters can be used to determine the size of each class in a default training dataset. The initial set assembled this way serves as an input to a weight-based adaptive classifier. The weight of an exemplar is computed from the distance to the test sample. With each recognition, the symbol with the minimal average weight gets deleted from the collection. Experiments show that the model allows rapid adjustment to the style of a particular writer and converges to approximately 99% accuracy.

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. In *Machine Learning*, pages 37–66, 1991.

[2] O. Golubitsky and S. M. Watt. Distance-based classification of handwritten symbols. *International J. Document Analysis and Recognition*, 13(2):113–146, 2010.

[3] P. Hart. The condensed nearest neighbor rule (corresp.). *Information Theory, IEEE Transactions on*, 14(3):515 – 516, may 1968.

[4] T. Kohonen. *Self-organization and associative memory*. Springer-Verlag New York, Inc. New York, NY, USA, 1989.

[5] Maplesoft. Maple 13 user manual, 2009.

[6] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. In *Machine Learning*, pages 257–286, 2000.