Personalization of an Online Handwriting Recognition System

Patrick Haluptzok, Michael Revow, Ahmad Abdulkader

Personalization of an Online Handwriting Recognition System

Patrick Haluptzok
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
patrickh@microsoft.com

Michael Revow
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
mrevow@microsoft.com

Ahmad Abdulkader
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
ahmadab@microsoft.com

Abstract

This paper proposes and compares some approaches for personalizing a handwriting recognizer to a specific user's handwriting style. A typical PocketPC or TabletPC is used by one person exclusively. The handwriting recognizer on such a device can customize its recognition to the specific writing style of the user. This paper presents the results of different personalization approaches for a neural network based classifier, showing how using data specific to the user can dramatically improve recognition accuracy.

Keywords: Personalization, Handwriting Recognition, Neural Network.

1. Introduction

A number of devices such as a TabletPC or PDA are used in a mode where the primary input is through a pen. Handwriting recognition can be used on such devices to allow the user a natural method for entering characters when a keyboard is not available. Typically the graphical UI provides an area for the user to write characters, and the ink written is converted into characters to provide input to the device as a keyboard would. An example of a typical UI is shown in Figure 1.

Figure 1. A typical input pad layout is shown. This UI is used on a keyboard-less device to provide a natural text input mechanism. The user writes in the input area with a pen and a handwriting recognizer converts the ink into text.

An ongoing challenge in using such a text input system on a device like the TabletPC is the error rate a user encounters from handwriting recognition. When a user first starts using a TabletPC they encounter a "walk-up" error rate. The handwriting recognizer has been trained to perform optimally over all writing styles that appear in the training data for a particular language. An individual typically shows much less variation in their writing style for each letter than the variation found over the entire training set of data across all users. In some cases a specific user may have a unique writing style that doesn't well match any of the writing styles the recognizer was trained for. Personalization of the handwriting recognizer to a specific user's writing style offers an ability to reduce the errors that user experiences which improves their overall satisfaction using the device. Personalization is defined in the context of this paper to mean taking some ink samples from a specific writer and then re-training or tuning the handwriting recognizer to that user's specific style.

This paper describes a method for personalizing a neural network based handwriting recognizer. First an overview of the architecture, features, and training method used in building the baseline recognition system is presented. Next a description of the personalization method is presented and the experimental results are shown.

2. Previous Work

In prior work some handwriting recognizers based on generative classifiers were built that supported personalization per user. For example in a nearest neighbor based prototype matching system new templates are added to the recognition database corresponding to the writer's style, and conflicting prototypes are removed. In a parametric model an updated estimate of the model parameters can be made to be better fit the user's data. For example when using a mixture of Gaussians to represent each character's distribution in feature space, an updated set of means and covariance matrices can be computed based on the user's personalization samples as described in [1]. Many of these previous approaches were focused on generative models where each character's distribution was represented in feature space. Our work focuses on personalizing a discriminative model. We have found in our work that discriminative models give the best accuracy on the handwriting recognition problem space when trained with large amounts of labeled data; other researchers have reported similar results [2].

3. Baseline Recognition System

3.1. Recognizer Structure

The character recognition system used for these experiments is based on a standard feed-forward neural network as shown in Figure 2.

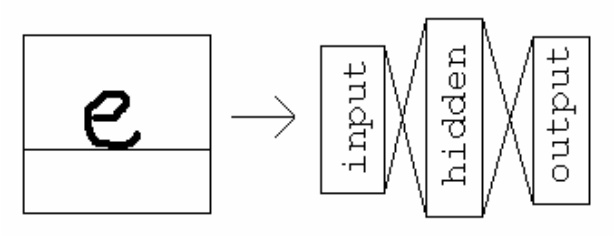

Figure 2. The baseline recognition architecture is a simple feed-forward neural network. The ink is featurized and normalized into 64 feature vector which is used as input to the neural network. The output layer of the neural network has 99 nodes, one for each character supported. The activation of each output node corresponds to the probability the ink is the character corresponding to that output node.

3.1.1. Output Layer

The output layer consists of 99 nodes; each node corresponds to a supported character. The output layer is computed via soft-max; each output node's activation is computed as shown in Equations 1 and 2, where j ranges over the M nodes in the previous layer, and w_{kj} corresponds to the weight connecting node j to node k.

$$a_k = \sum_{j=0}^{M} w_{kj} * y_j \tag{1}$$

$$y_k = \frac{\exp(a_k)}{\sum_{k'} \exp(a_{k'})} \tag{2}$$

3.1.2. Hidden Layer

The hidden layer was set to 150 for these experiments. A larger hidden layer gave generally better accuracy results at the expense of a larger and slower system. The hidden layer is made of sigmoid nodes; each hidden layer node's activation is computed by the sigmoid function as shown in Equation 3, with a_k computed as shown in Equation 1.

$$y_k = \frac{1}{1 + \exp(-a_k)} \tag{3}$$

3.1.3. Feature Extraction

The input layer consists of 64 features computed from the character ink. 56 of the features are the coefficients for Chebyshev polynomials that approximate the stroke shapes and contours of the ink, a featurization method described in [3]. The remaining 8 features are computed from the bounding box of the ink relative to the baseline, and other properties of the ink such as stroke count and overall curvature measures. Across the train set the mean and variance for each feature was computed and the features in the train and test sets were normalized by this so each feature input to the network would have zero mean and unit variance.

To perform character recognition the ink for a character is featurized and normalized into a 64 entry vector. This vector is used as the input to the neural network. The hidden layer and output layer activations are computed via standard feed forward propagation and the ink was classified as the character with the maximum output activation.

3.2. Training Methodology

The recognition system is trained optimizing a cross-entropy error function as shown in Equation 4. As discussed in [4] optimizing the cross entropy error function with a soft-max output layer will lead to the output nodes converging to probabilities, enabling the outputs to be combined in a principled way with a language model if one is available. Cross-Entropy optimization was found to consistently converge to an error rate that was 8% lower than Mean Square Error optimization for our baseline recognition system, similar to results reported by Simard et al [5]. In equation 4 *n* represents the number of training samples, *c* represents the number of outputs, t_{kp} is the target value for the k^{th} output when p^{th} patterns presented, and y_{kp} is actual output value from the neural network.

$$E = \sum_{p=1}^{n} \sum_{k=1}^{c} t_{kp} \ln \frac{y_{kp}}{t_{kp}} \tag{4}$$

The results of training the baseline recognition system are shown in Figure 3. The effects of over training can be seen, as the error rate on the test set initially falls to a minimum and then slowly starts to rise as the network is over-trained. This demonstrates the importance of using a method to prevent over-training such as using a validation set to stop training at an optimal generalization point.

This graph also shows the impact of using a larger hidden layer; the train and test error rates for the 600 node hidden unit layer are significantly more accurate than the 150 node hidden layer. This shows the trade-off that can be made between size and accuracy. The network weights were initialized randomly in the uniform range of [-0.01, 0.01]. Gradient descent over the weights with a learn rate of 0.001 and a momentum of 0.7 was used to minimize the error rate.

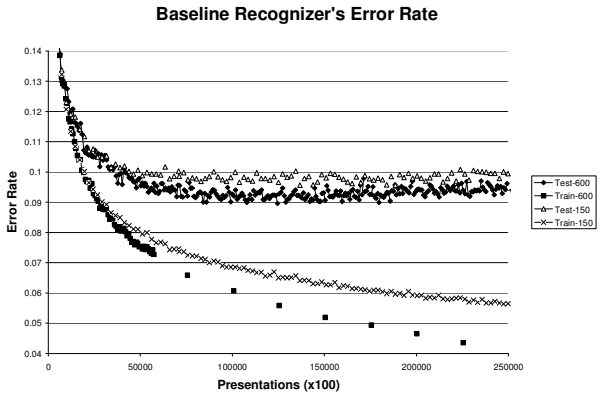

Figure 3. This graph shows the error rates of the neural network on the train and test sets during training of the baseline recognition system. The x-axis shows how many times samples had been presented during training. An epoch corresponds to 192,002 presentations, a complete pass through all the train data. All graphs are shown using presentations instead of epochs so that error rates for different sized training sets are comparable in terms of the amount of training time taken.

3.3. Baseline Data Set and Weight Set

For our experiments the net was trained using a training data set of 192,002 samples from 225 different users. A validation set of 37,984 samples from 34 different users was used during training of the baseline recognizer to determine when to stop training. The test set is 58,966 samples gathered from another 21 different users. There are approximately 28 samples per character from each user in the test set.

The neural network had the minimum error rate on the validation set after training for 10,621,500 presentations. That was the baseline weight set used for the incremental personalization experiments. This baseline weight set corresponded to a 10% error rate on the test set. As can be seen in Figure 3, the validation set minimum error rate was in the correct range where the minimum error rate on the test set occurred, and was close to the 9.78% minimum error rate achieved on the test set during training.

These experiments were done on US English with single character data. Training was done uniformly across the characters, where ink for each character is presented an equal number of times. Test error rates were computed for each user by taking the raw count of the characters incorrectly classified in the test set and dividing that by the total number of characters for that user. The average error rates weighted each user equally.

4. Personalization

4.1. Overview

The approaches presented here to personalize the handwriting recognizer are based on training the neural network with a user's ink data. The user can explicitly enter ink data for each character in an enrollment application or the system can implicitly collect ink from the user that has been written and corrected in their daily use of the device. Considerable work and care needs to be taken when using implicit data for training. In the personalization approach presented here one needs to balance the character counts of the data when training the neural network; possibly augmenting the implicit data with training data from other users for under-represented characters. Since labels aren't available for implicit data, the recognizer's own recognition result is used as the label for any ink data that isn't corrected. The system is designed so that when the user corrects any recognition errors the corrected text will be used as the label for the implicit ink data. But since not all misrecognition errors are corrected, using implicit data requires handling misrecognitions and the full details are beyond the scope of this paper. Our results presented here are based on explicit data where the user has written every character the same number of times for personalization, although similar results can be achieved using data collected implicitly.

4.2. Scratch Approach

The first approach we tried was training a neural network from scratch on just that user's ink data, following the same method used to train the baseline recognition system. We evaluated the accuracy achieved using 1, 2, 5, and 10 samples of ink per character when training the neural network based recognizer. This showed that good accuracy results for a user can be achieved with relatively few samples per character. For users in a language that a localized recognizer isn't available for their character set, this would offer the ability for a user to completely train the recognizer on their writing style from scratch and have recognition accuracy better than the "walk-up" accuracy achieved in the current English recognizer.

4.3. Incremental Approach

The second approach we tried was training the neural network on the user's ink data, but starting from the neural network fully trained on the train set data. This showed the user could achieve substantial accuracy improvements by providing additional ink data. We evaluated the accuracy using 1, 2, 5, and 10 samples of ink per character. The results showed that the neural network could find a substantially better local minimum starting from a fully trained neural network than starting from scratch. Clearly the user achieved much better accuracy results starting from a fully trained neural network.

4.4. Personalization Data Set

In these experiments each of the 21 users in the test set also provided a separate personalization training set. The training set had exactly 10 samples per character which was used to train on. For each of the 21 users some portion of the data (1, 2, 5, or 10 characters) from their personalization training data was used to personalize the recognizer. The impact of that change was then computed using the separate personalization test set of each user.

5. Experimental Results

For each of the 21 users in the test set the baseline recognizer was personalized using the "Scratch Approach" and the "Incremental Approach". The test set was broken into 21 different subsets corresponding to each of the users. After personalization on just one user's train data the error rate on that user's test data was measured. This was done independently for each of the 21 users in the test set.

5.1. Scratch Results per User

In Figure 4 the error rate for each user on their test set is shown as a function of the number of presentations the neural network has been trained on using the scratch training approach. The number of presentations is plotted in logarithmic scale to better show the rate at which the error rate drops. Starting from scratch the error rate during training starts at 99%, as expected when the weights are randomly initialized.

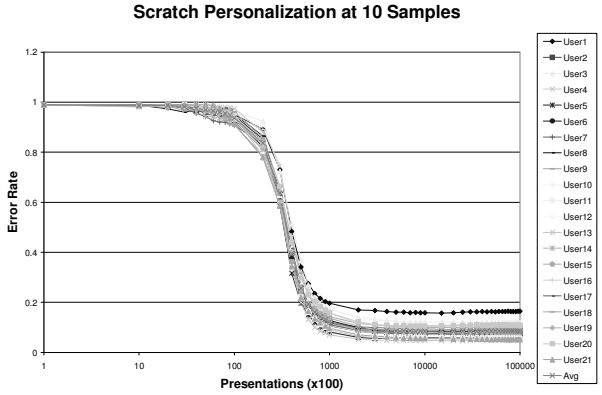

Figure 4. Scratch error rate per user when trained on 10 samples per character, as a function of presentations made during training.

5.2. Average Scratch Results

In Figure 5 the average error rate measured on the test set is shown using scratch personalization. The average error rate is computed by averaging the personalized error rate from each of the 21 users in the test set.

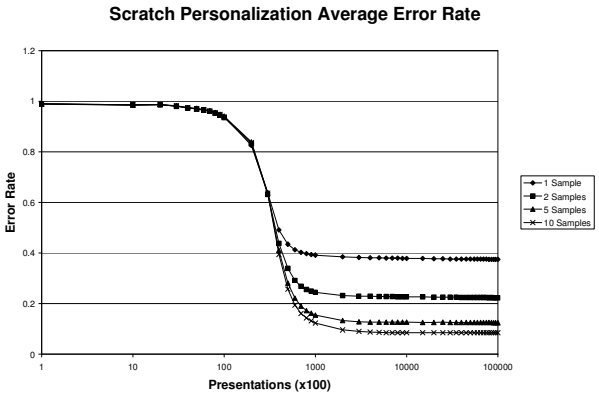

Figure 5. The average error rate plotted as a function of training time. Each line shows the average error rate when trained from scratch with 1, 2, 5, or 10 samples per character.

5.3. Incremental Results per User

In Figure 6 the error rate for each user on their test set is shown as a function of the number of presentations the neural network has been trained on using the incremental training approach. As expected when starting from a weight set that was trained on a large set of users the per user error rate starts around 10% and reduces further as more samples from the user's train data are seen.

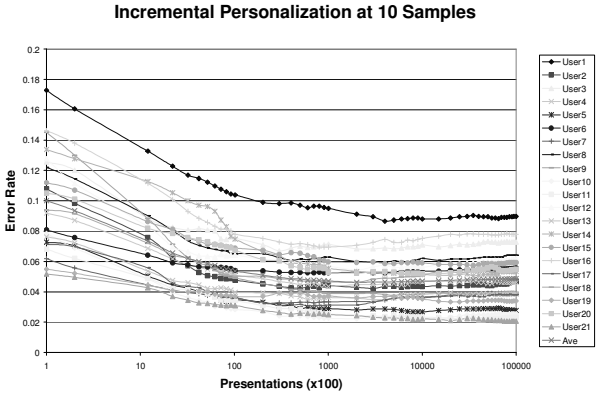

Figure 6. Incremental error rate per user when trained on 10 samples per character, as a function of presentations made during training.

5.4. Average Incremental Results

In Figure 7 the average error rate measured on the test set is shown using incremental personalization. Just as in the original training of the baseline recognizer overtraining is clearly seen if the training runs to long. Using cross-validation the overall optimal number of times to present the samples for different training set sample counts was determined. Table 2 presents those results.

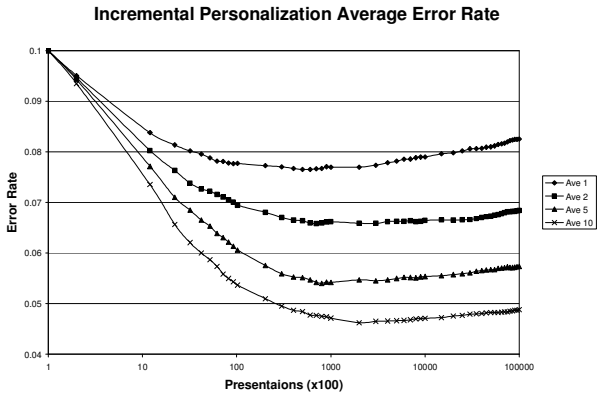

Figure 7. The average error rate plotted as a function of samples presented during training. Each line shows the average error rate when trained incrementally with 1, 2, 5 or 10 samples per character.

6. Discussion

The experiments show that a significant reduction in the average user error rate is possible by training on additional samples from the user.

In Table 1 it can be seen that the scratch personalization method reduces the error rate by 15% when 10 samples per character are provided. However with just 1, 2 or 5 samples per character the neural network converges to a higher error rate. Additional experiments showed that at 8 samples per character the scratch personalization method yielded an accuracy equivalent to the baseline recognizer.

Table 1. This shows the maximum relative improvement seen from scratch personalization compared to the baseline recognition error rate 10%.

Number of samples per character	Minimum Error Rate	Relative Improvement
1	37.51%	-275.1%
2	22.35%	-123.5%
5	12.48%	-24.8%
10	8.50%	15.0%

In Table 2 the incremental personalization method shows clear wins over the baseline recognizer at all sample counts. Interestingly for best results the number of presentations made to personalize the neural network increases sub-linearly, as shown by the "Samples Presented" column in Table 2. Or looking at in terms of epochs, the more data per character we have reduces the total number of complete passes over the data to obtain the best results.

Table 2. This shows the relative improvement seen from incremental personalization compared to the baseline recognition error rate of 10%. The "Samples Presented" column shows the optimal length of time to train the recognizer on the additional data for that number of samples per character.

Number of samples per character	Minimum Error Rate	Relative Improvement	Samples Presented (x100)	Epochs
1	7.65%	23.47%	600	600
2	6.58%	34.17%	700	350
5	5.40%	45.99%	800	160
10	4.62%	53.76%	2000	200

7. Conclusions

Personalization can provide a substantial improvement in recognition accuracy for a writer. For users in languages where baseline recognizers are not produced we have shown that with approximately 8 samples the handwriting recognition system can be trained on the user's data to produce a recognizer with better accuracy than the "walk-up" accuracy of a fully trained but non-personalized recognizer. In addition we have shown that the fully trained recognizer can have its accuracy improved dramatically, and that starting with a fully trained neural network on a large set of users results in superior accuracy for the personalized recognizer over starting from scratch. Personalization clearly provides a dramatic improvement in the accuracy for a user and should be deployed in commercial systems to improve user satisfaction.

References

[1] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification (2nd Edition)*, John Wiley & Sons, (2001).

[2] Andrew Y. Ng, Michael I. Jordan. "On Discriminative vs. Generative classifiers: A comparison of logistic regression and Naive Bayes". NIPS 2002

[3] Adcock, James L. "Method and system for modeling handwriting using polynomials as a function of time", US Patent 5,764,797, granted June 9, 1998.

[4] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, (1995).

[5] P. Simard, D. Steinkraus, and J. Platt. "Best practice for convolutional neural networks applied to visual document analysis", *International Conference on Document Analysis and Recognition (IDCRAR)*, pages 958-962, 2003.