Three Handwriting Adaptation Approaches for Digit Recognition

Dieter Lutz and **Martin Toepfer** and **Frank Puppe**
Department of Computer Science VI, University of Würzburg
Am Hubland, Würzburg, Germany
dieter.lutz@googlemail.com, {toepfer, puppe}@informatik.uni-wuerzburg.de

Abstract

Handwritten digit recognition in applications like automatic exam grading is challenging because handwritings inherently differ between development data sets and real application scenarios. To overcome this issue, we propose three handwriting adaptation methods and compare them on a data set of 2860 samples of 26 different users. We explain preprocessing and feature extraction steps, and suggest different adaptation approaches: two methods are similar to bootstrapping, and one method uses dimensionality reduction. Experimentally, we show that adaptive approaches yield significantly better results than the standard classifier. Adaptation improved the precision of an already good baseline by about one to four percent depending on the size of the training set.

1 Introduction

High accuracy handwritten digit recognition systems typically build upon thousands to tens of thousands of annotated training images [DeCoste and Schölkopf, 2002]. However, even then classifiers appear to be *brittle* when applied on slightly different data [Seewald, 2012]. One of the reasons for this is that the same digit can look differently when it was written by different persons as depicted in Figure 1. The samples on each side were written by one student respectively, and we can see clearly that each handwriting has distinct characteristics. As a result, samples of one class are consistent for a certain student, but there may be inconsistencies summing over samples of different handwritings. Therefore, it is difficult to categorize a sample without annotated samples from the same person for a normal classifier. The proposed adaptation methods deal with this problem by classifying a whole set of samples from one user at once instead of one by one. By this means, they can utilize the similarities of these samples to adapt the recognition process to new handwriting styles. The evaluation of the techniques shows that this adaptation of a classifier on the handwriting of one user is possible and can improve the precision of the classifier.

The foundation of the customization approaches in this paper is a Support Vector Machine classifier [Cortes and Vapnik, 1995] which is also used as a baseline in the evaluation. However the introduced techniques are not restrained to this type of classifier, but they can also be used with every other classifier which is able to specify the likelihood of its classification for every class. A possible use case of the proposed methods is automatic exam grading, like in [Mandel *et al.*, 2011]. Hence the considered classes are digits (0-9) as well as commas. On the exam sheets, every digit and symbol is written into a separate box. The exams get scanned and the boxes are cut apart. The resulting gray-scale images are grouped for every exam because they originate from the same user. These sets of images are the input for the classifier.

Our paper has the following structure. Firstly, we discuss related work and give an introduction to preprocessing and feature extraction of the images. In Section 5, we describe three different types of handwriting customization techniques. In the evaluation section, the used data set is described, and the results are presented and discussed. Finally, we suggest topics for future work and give a summary of our findings.

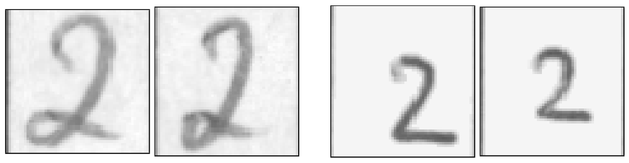
Figure 1: Two samples of digit “2”, written by 2 students with distinct handwritings.

2 Related Work

Handwritten digit recognition is a well examined field with many different approaches. One of the most successful classifiers in this domain are Support Vector Machines (SVM) [Cortes and Vapnik, 1995]. This is confirmed in [DeCoste and Schölkopf, 2002] where a Support Vector Machine approach with the lowest reported test error on the MNIST digit recognition task at that time was proposed.

Processing human and computer generated data has led researchers to adaptation techniques in different domains of artificial intelligence. For instance, context consistencies arise in text processing or speech recognition. Klügl et al. [2012], for example, collectively segment references of scientific papers using statistical graphical models. They exploit the homogeneity of formatting inside of sections originating from style guide usage. Intelligent speech recognition systems must also learn to map signals with speaker specific characteristics to general symbols. For instance, Leggetter and Woodland [1995] showed that linear transformations can be used to maximize the likelihood of the new data and thereby associate patterns across individual speakers. To adapt online[1] handwritten character recognition, Szummer and Bishop [2006] proposed to use a mixture of experts. They assume a supervised setting where some labeled examples of the new handwriting are always available. Classifiers are trained on clusters of similar handwriting styles, and combined to produce an adaptive model weighted by each classifiers' posterior probability on the labeled samples of a new handwriting. In this paper, however, we propose approaches for adaptation when no labeled data is present.

The subspace embedding technique in this paper is sustained by the assumption that identical digits are close to each other in a common subspace. This hypothesis is supported by [Chapelle *et al.*, 2002] where clustering of unlabeled instances in subspaces was utilized for kernel adaptation in Support Vector Machines, which improved their error rate.

Adaptation of digit recognition models to user characteristics is strongly related to semi-supervision. A very popular technique in this domain is bootstrapping [Yarowsky, 1995] which successively populates the training set with the most confident predictions on the unlabeled data. Two of the methods (Best-First-One/Two) that we propose in Section 5 can be regarded as bootstrapping methods. Best-First-One directly populates the training set with certain unlabeled instances, whereas Best-First-Two creates a new classifier with them and balances this classifier against the initial one.

3 Preprocessing

The first stage of the classification process aims to reduce the impact of different scales, positions and intensities of the symbols. Furthermore it tries to remove noise from the images. In the following, we illustrate all preprocessing steps for the example shown in Figure 2.

Horizontal Cropping As the first step the digit is cropped horizontal. This is done to remove potential vertical lines originating from the frame of the boxes. The detection and removal of lines at the borders is achieved by calculating the average gray-scale value of the two most left and right pixel columns. If the average value for the first column is below 230 the column is removed and the second column is taken into consideration in the same way. Otherwise the image is not modified. Additionally a variance-based heuristic is applied to try and crop the digit horizontally in order to get rid of lines in the interior.

Binarization The next preprocessing step is the previously mentioned binarization. The main goal is to make the input invariant to different intensities of the handwritten symbol. To put this into effect Otsu's method is used to compute a threshold based on the histogram of gray-scale values. The threshold is then applied to categorize the image into black and white pixels.

Centralization The symbol is moved to the center of the image by its center of mass in order to compensate translations.

Symbol-Cropping After centralization, we remove irrelevant and noisy parts of the image to reduce the effect of differently sized symbols. We apply two methods. Firstly a variance-based heuristic, similar to the one used earlier, but this time in horizontal and vertical directions. Secondly, the left and right boundaries of the symbol are estimated as the 3rd and the 97th percentile of the x-coordinates of the black pixels respectively. The bottom and top boundaries are identified analogously.

Resizing Finally, we resize the image to the uniform size of 20x30 pixels by antialiasing. By this means, identical symbols should have nearly the same size, and, essentially, we achieve scale invariance to some degree.

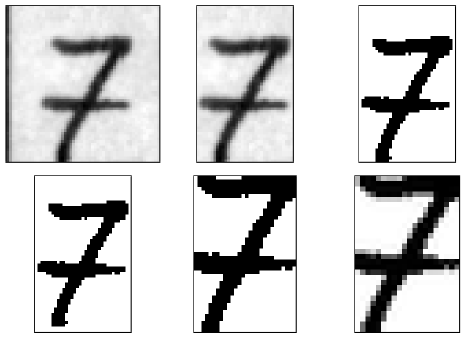
Figure 2: Preprocessing steps: Original, Horizontal Cropping, Binarization, Centralization, Symbol-Cropping, Resizing.

4 Feature Extraction

Feature extraction is the next stage of the classification process. The features used can be easily extracted due to the extensive preprocessing. There are also only two different types of features. These are explained in the following.

Pixel Gray-Scale Value The first type of features are the gray-scale values of the image resulting from the preprocessing. The images have 20x30 pixels, so there are 600 pixel features in total. These are intended to represent the general shape of the digits.

Zone Gray-Scale Value The second type of features are also gray-scale values. These are extracted during preprocessing after the symbol gets centralized. The basic idea is to lay a coarse grid over the image and use the average gray-scale value of the resulting zones as features. In the actual implementation, the image is resized to 14x14 pixels and their value is extracted. This amounts to 196 zones features. The purpose of this type of feature is to distinguish between similar shapes by properties removed through symbol cropping.

5 Approaches for Handwriting Adaptation

Handwriting adaptation is the final and central stage of our system. In contrast to static handwritten digit recognition systems, we utilize the similarities between samples of the handwriting of one user to improve the classifier's performance particularly on this user's set of samples.

One of the inherent challenges of adaptation is how to model and obtain handwriting specific information for a new person. The naive approach is clustering the samples and assigning a class to each cluster. However it is difficult to get an accurate clustering in a real-world application. So a similar method is used which places similar samples near each other in a low-dimensional space. This well-known method is called subspace embedding. Another way to utilize the similarities of a user's samples is bootstrapping. The idea of bootstrapping in this case is to classify only the certain samples at first and then use these samples as if they

[1]online handwriting recognition processes path trajectories rather than static scanned images

were annotated samples to improve the further classification. This method is utilized in both Best-First techniques.

Another challenge is how to integrate the similarity information into the classification process. One possibility is to model it as additional features. But it is not really clear how to make these features invariant to the user's specific differences and the weighting of the additional to the original features is difficult to handle. A simpler way is to directly modify the classifier's probabilities of class affiliation based on the similarity information. This is also the method utilized in all three adaptation techniques, which are proposed in the following.

A further common denominator is the use of an Support Vector Machine as an initial classifier derived from a training set of annotated samples. This classifier is mostly used to calculate the probabilities of class assignment for samples. So an additional challenge is to modify mainly the samples which the classifier is unsure about. We handle this by weighting the modifications of the following techniques by the certainty of the classifier. To determine this certainty a function based on the class probabilities is needed, which is the entropy function in our case.

5.1 Subspace Embedding

The first approach introduced is subspace embedding. The idea of subspace embedding is to reduce the dimensions of the feature vectors of the user's samples to a few concepts. This can be achieved through principal component analysis [Jolliffe, 1986]. Identical digits are likely to have the same concepts so they are also likely to be positioned near each other in the subspace. This is used to modify the sample's probabilities of class affiliation in order to shift the classification of unsure samples in the right direction. To be more concrete, for every sample a number of nearest neighbors are used for the modification. The influence of every neighbor decays based on their quadratic euclidean distance to the considered sample. The overall magnitude of the adjustments made to the probabilities also depend on the certainty of the sample's classification. To estimate how sure the classifier is about an assignment the normalized entropy over the probabilities of class affiliation is used. After these adjustments, the class with the highest probability is assigned to the sample.

5.2 Best-First-One

The Best-First-One method initially classifies all samples of the user set to gather the best samples, i.e. the ones which the classifier is most certain about. The certainty of a sample is identified by the entropy of it's probabilities of class assignment. A threshold is used to split the safe from the unsafe samples. However a specific fraction of the samples is always assumed as safe. These samples are treated as annotated examples and added to the training set. A classifier is created from the enhanced training set. The remaining samples are then classified with this new classifier.

5.3 Best-First-Two

The Best-First-Two technique is very similar to the Best-First-One method. The difference is that the safe samples are not added to the initial training set but used to create a new classifier only based on these samples. The classes for the remaining samples are then determined by the sum of weighted probabilities of both classifiers. The weights are appointed according to the entropies of the probabilities. This way the classifier, which is more sure about the class of a sample, is taken into account to a further extend. A direct advantage of this variation is that only the certain samples have to be learned and not the entire training set. This results in shorter runtimes, especially if the initial training set is large.

6 Evaluation

We evaluated the three different customization methods and the standard classifier, which does not use user specific information. Therefore, a user set was chosen as the test set, a training set was sampled from the remaining user sets, and the precision scores of the four techniques were measured on the test set. This process was repeated for each of the 26 user sets similar to a leave-one-out evaluation setting. The results were averaged over all repetitions for every method and training size. The implementation was done in Python. The libraries NumPy[2], SciPy[3], scikit-learn[4] and Pillow were utilized. In the following, the parameters used for each method are listed.

Support Vector Machine The Support Vector Machine classifier used a polynomial kernel with a degree of three. Probability calculations for class affiliation was enabled.

Subspace Embedding The feature vectors were reduced to three dimensions. The three nearest neighbors were used to adjust the sample's probabilities. The entropies were normalized with the factor $1/7$.

Best-First-One/Two The threshold chosen to separate the safe from the unsafe samples was 0.9. Ten percent of the samples of the user set were always assumed to be safe.

6.1 Data Set

A custom data set was used since most standard data sets do not provide user information associated with their samples. However this is necessary for the adaptation techniques to work. The data set was gathered using test forms with boxes for every symbol. It consists of 26 sets of different users. Every set contains 10 annotated examples for the digits 0-9 and the comma symbol. So there are 110 examples for each user in total and the data set amounts to 2860 samples overall.

6.2 Results

The results are plotted in Figure 3 with the size of the training set as the x-axis and the average precision as the y-axis. All adaptation methods exceed the standard classifier's curve regardless of the training set size. The distance between the standard classifier and the customization methods at a training set size of 100 amounts to about four percent. For the training set size 1000 the difference is still around one to two percent for all adaptation techniques.

Table 1 presents concrete numbers for these experiments. Values printed in bold type show the respective best approach for each training set size. Underlined values mark statistically significantly better precisions compared to the standard classifier (paired t-test, p-value <0.05).

Subspace embedding and Best-First-One provide very similar performance on all training set sizes. The precisions provided by Best-First-Two are also similar for the sizes 100, 200, 600 and 700. On the sizes 300 to 500 Best-First-Two was outperformed by the other two approaches by around 0.5%, but it surpasses them on the sizes 800 to 1000 by about the same amount.

[2]http://www.numpy.org/

[3]http://www.scipy.org/

[4]http://scikit-learn.org/

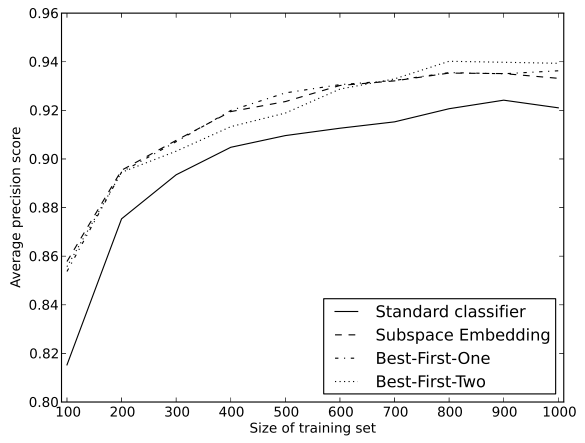

Figure 3: Average precision of the different methods for several training set sizes

Table 1: Table of average precisions. Bold values show the best approach for each training set size. Underlined values are significantly better compared to the standard classifier.

Size of training set	Standard classifier	Subspace Embedding	Best-First-One	Best-First-Two
100	0.815	**0.858**	0.854	0.856
200	0.875	**0.895**	0.895	0.894
300	0.893	**0.908**	0.907	0.903
400	0.905	0.919	**0.920**	0.913
500	0.910	0.924	**0.927**	0.919
600	0.913	0.930	**0.931**	0.929
700	0.915	0.932	0.932	**0.933**
800	0.921	0.935	0.936	**0.940**
900	0.924	0.935	0.935	**0.940**
1000	0.921	0.933	0.936	**0.939**

6.3 Discussion

Figure 3 shows that the adaptation methods provided consistently superior precision in comparison to the standard approach. The improvements were statistically significant in 20 out of 30 cases (see Table 1 for details). Hence it can be assumed that the proposed methods are able to adapt to handwritten digits and commas of a new handwriting in order to improve their performance. In summary, all techniques provide considerable improvements to the standard classifier by about the same extend and are recommended for further investigation.

7 Future Work

In this work, we applied a special subspace embedding technique, different methods of dimensionality reduction like Isomap projections could be considered for a dispersion of different symbols in the subspace. Future work can further investigate other approaches to weight the certainties of the initial recognition, or the distance functions used. For the Best-First methods one important parameter is the optimal threshold to separate certain samples from uncertain ones. The Best-First-Two approach could possibly be further enhanced by more precise weighting between the two classifiers, and incorporating more complex features, confer, for example, [Leibfried, 2012].

8 Summary

We proposed three approaches for handwriting adaptation of digit recognition: one approach that applies subspace embedding and two approaches that are similar to bootstrapping. Our experiments showed that adaptive techniques can enhance the precision of the standard classifier significantly which emphasizes the importance of adaptation in this domain. The overall precision scores of the three approaches were comparable. Their notable improvements over an already good baseline ranged from about one to four percent.

Acknowledgments

We thank Alexander Hörnlein and all participants for their help with the data set.

References

[Chapelle *et al.*, 2002] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 585–592. MIT Press, 2002.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[DeCoste and Schölkopf, 2002] Dennis DeCoste and Bernhard Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1-3):161–190, 2002.

[Jolliffe, 1986] Ian T. Jolliffe. *Principal component analysis*, volume 487. Springer-Verlag New York, 1986.

[Klügl *et al.*, 2012] Peter Klügl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Collective information extraction with context-specific consistencies. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *ECML/PKDD (1)*, volume 7523 of *Lecture Notes in Computer Science*, pages 728–743. Springer, 2012.

[Leggetter and Woodland, 1995] Chris J. Leggetter and Philip C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech and language*, 9(2):171, 1995.

[Leibfried, 2012] Felix Leibfried. Recognition of handwritten digits. Diplomarbeit, University of Würzburg, Würzburg, September 2012.

[Mandel *et al.*, 2011] Alexander Mandel, Alexander Hörnlein, Marianus Ifland, Edeltraud Lüneburg, Jürgen Deckert, and Frank Puppe. Cost analysis for computer supported multiple-choice paper examinations. *GMS Z Med Ausbild*, 28(4):Doc55, 2011.

[Seewald, 2012] Alexander K. Seewald. On the brittleness of handwritten digit recognition models. *ISRN Machine Vision*, 2012:10, 2012.

[Szummer and Bishop, 2006] Martin Szummer and Christopher M. Bishop. Discriminative writer adaptation. In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2006.

[Yarowsky, 1995] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In Hans Uszkoreit, editor, *ACL*, pages 189–196. Morgan Kaufmann Publishers / ACL, 1995.