
Personalized Handwriting Recognition via Biased Regularization

Wolf Kienzle¹
Kumar Chellapilla

KIENZLE@TUEBINGEN.MPG.DE
KUMARC@MICROSOFT.COM

Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

Abstract

We present a new approach to personalized handwriting recognition. The problem, also known as writer adaptation, consists of converting a generic (user-independent) recognizer into a personalized (user-dependent) one, which has an improved recognition rate for a particular user. The adaptation step usually involves user-specific samples, which leads to the fundamental question of how to fuse this new information with that captured by the generic recognizer. We propose adapting the recognizer by minimizing a regularized risk functional (a modified SVM) where the prior knowledge from the generic recognizer enters through a modified regularization term. The result is a simple personalization framework with very good practical properties. Experiments on a 100 class real-world data set show that the number of errors can be reduced by over 40% with as few as five user samples per character.

1. Introduction

This paper addresses the personalization of a handwriting recognizer, as found in computer applications where the primary input device is not a keyboard but a digital pen. The basic idea is to increase the recognition rate for a specific user by adapting the recognizer to her personal writing style. The problem is also known as writer adaptation and has been extensively studied e.g. by Matić et al. (1993); Platt and Matić (1997); Brakensiek et al. (2001); Connell and Jain (2002). Writer adaptation is gaining new importance as devices such as Tablet PCs, PDAs (personal digital assistants), and SmartPhones (cell phones with

PDA capabilities) are becoming increasingly popular. The rationale behind personalizing a recognizer is simple: since handwriting is inherently unique, one expects that — given realistic design constraints — a personalized recognizer leads to better performance on the corresponding user’s data than a generic recognizer which has to work for a variety of users.

In practice, the recognizer has to adapt to a user’s writing style via sample data which requires explicitly prompting the user for training samples. In terms of usability, this is expensive and as a consequence the amount of personalization data will be very limited. Also, combining this new information with that already captured by the generic recognizer is a challenging problem. We propose solving the personalization problem by minimizing a modified regularized risk functional — in our case a modified support vector machine (SVM) — on the personalization data. We show that this technique, called biased regularization, provides a principled way for trading off generic and user-specific information and leads to state-of-the-art results in writer adaptation.

The paper is organized as follows. The model of our recognizer is described in Section 2. A brief introduction to SVMs is given in Section 3. This may be skipped by readers familiar with the subject. Biased regularization is introduced in Section 4, the resulting algorithm is discussed in Section 5. Sections 6 through 8 give experimental results, which are compared against previous approaches in Section 9. The paper concludes with a brief discussion in Section 10 and an appendix on the use of kernels.

2. The Model

Building a character recognizer requires solving a multi-class classification problem. In the case of our Latin character set (Section 6) the number of classes

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

¹Current address: Max-Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany.

is $n = 100$. The model that we use is a pairwise multi-class SVM classifier, i.e. it consists of an ensemble of all possible binary 1-vs-1 SVM classifiers. For a 100 class problem, there are $\binom{n}{2} = 4,950$ pairwise classifiers. The 4,950 outputs (all ± 1) are combined into a single output through majority voting, i.e. we simply count the number of wins and output the class label that won most often. There exist other, equally plausible SVM-based multi-class models such as 1-vs-rest, DAGSVM and monolithic methods (Hsu & Lin, 2002). We chose the 1-vs-1 architecture since it is easy to implement, scales to our 100 class problem, and is known to work reasonably well in practice. However, the personalization approach presented below is by no means restricted to this particular choice.

The model is as follows. For a given input \mathbf{x} , each pairwise classifier i -vs- j produces a binary output $h_{ij}(\mathbf{x}) \in \{1, -1\}$, indicating whether it will vote for class i or j , respectively. The h_{ij} are thresholded linear functions of the form

$$h_{ij}(\mathbf{x}) = \text{sgn}(\mathbf{w}_{ij}^\top \mathbf{x} + b_{ij}) \quad (1)$$

The output of the recognizer is an integer between one and n , computed via

$$f(\mathbf{x}) = \arg \max_i \left[\sum_k h_{ik}(\mathbf{x}) - \sum_k h_{ki}(\mathbf{x}) \right] \quad (2)$$

For notational convenience, we now drop the class indices ij , since all 1-vs-1 classifiers are treated independently and identically. This allows us to discuss the ideas below using a single binary classifier $h(\mathbf{x})$, two classes $\{-1, 1\}$, and one set of parameters \mathbf{w} and b .

3. The Generic Recognizer

The generic recognizer consists of the above multi-class model with the binary classifiers $h(\mathbf{x})$ trained as linear soft margin SVMs. For $h(\mathbf{x})$ to be an SVM, its parameters \mathbf{w} and b have to minimize the regularized risk

$$R(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m [1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)]_+ \quad (3)$$

Here, the right term denotes the data fit on the m training points, measured by the hinge loss $L(\xi) = [1 - \xi]_+ = \max\{0, 1 - \xi\}$. The left term is the regularizer which ensures that the problem is well-posed by penalizing large components of \mathbf{w} and thus implements the maximum margin principle (Schölkopf & Smola, 2002). The tradeoff between a small $\|\mathbf{w}\|$, i.e. a large margin, and a good data fit has to be made a priori

by the choice of the regularization parameter C . The problem of how to choose this parameter will be addressed in Section 7.

The standard way of looking at an SVM however, is through the dual problem of minimizing (3), namely

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_{i=1}^m \alpha_i \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (4)$$

The derivation can be found in Schölkopf and Smola (2002); Chang and Lin (2001). Both problems (3) and (4) are equivalent. Given the solution to the dual, the optimal primal parameter \mathbf{w} corresponds to

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i. \quad (5)$$

Note that the form of the solution (5) implies that the normal of the decision hyperplane (1) lies in the span of the training points \mathbf{x}_i . The well-known representer theorem (Kimeldorf & Wahba, 1971; Schölkopf et al., 2001) states that this is indeed the case for *any* loss function, and for *any* regularizer that is a strictly monotonic increasing function of $\|\mathbf{w}\|$. We will return to this result in Section 4. Also, note that in the dual formulation (5), C bounds the magnitude of the coefficients $y_i \alpha_i$ and therefore limits the effect of the training points on the solution (5). This is in agreement with the role of C in the primal formulation (3), namely that of controlling the effect of the data fit term.

4. Biased Regularization

Let us suppose that we have the generic recognizer plus a small number of new user-specific training samples. Also, we assume that the original (user-independent) training samples used to build the generic recognizer are no longer available.

To motivate our approach, note that the regularization term $\|\mathbf{w}\|^2$ in (3) encodes the prior knowledge about small \mathbf{w} being preferable. This is based on various related ideas such as the Occam's Razor, smoothness of the decision boundary, or in the particular case of SVMs, the maximum margin principle. If we fully ignore the data fit term by setting $C = 0$, the solution becomes $\mathbf{w} = \mathbf{0}$. In a way, $\mathbf{w} = 0$ is our safest bet in case of unreliable or insufficient data.

The idea of this work is to exploit this regularization mechanism for personalization by exchanging the $\mathbf{w} = \mathbf{0}$ prior with the parameters of the generic recognizer. To this end, we propose retraining each pairwise

classifier by minimizing

$$R(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w} - \mathbf{w}_0\|^2 + C \sum_{i=1}^m [1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)]_+ \quad (6)$$

where \mathbf{w}_0 is the parameter vector of the corresponding pairwise classifier in the generic recognizer, and (\mathbf{x}_i, y_i) , $i = 1 \dots m$ are the new user-specific examples for the respective binary problem. Note that (6) is very similar to (3), with the only difference being the bias towards \mathbf{w}_0 rather than $\mathbf{0}$, hence the term *biased regularization*. In particular, setting $\mathbf{w}_0 = \mathbf{0}$ makes both problems identical, indicating that the generic recognizer is a special case of the personalized one.

The dual of (6) is given by

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_{i=1}^m \alpha_i (1 - \omega_i) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \quad (7)$$

where

$$\omega_i = y_i \mathbf{w}_0^\top \mathbf{x}_i. \quad (8)$$

The derivation is completely analogous to the standard (\mathbf{w}_0 -free) case and the problem in (4) becomes a special case of the problem in (7), namely for $\mathbf{w}_0 = \mathbf{0}$. Analogously to (5), the optimal solution satisfies

$$\mathbf{w} = \mathbf{w}_0 + \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i. \quad (9)$$

The form of (9) makes intuitive sense: the prior knowledge from the user-independent data is represented by \mathbf{w}_0 , whereas the user-specific data enter as a linear combination whose coefficients are bounded by C . In particular, if we set C to zero, we get back the generic recognizer \mathbf{w}_0 .

It should be mentioned that the possibility of generalizing the regularizer in this way has been pointed out in Schölkopf et al. (2001), Remark 1. There, the authors state that the representer theorem still holds in the presence of an additional regularization term $-\mathbf{w}_0^\top \mathbf{w}$, which adds an extra multiple of \mathbf{w}_0 to the solution. By expanding the square term in (6) and noting the multiple of \mathbf{w}_0 in (9), one can see that their generalization of the regularizer is equivalent to our method. As a consequence, the biased regularization approach to personalization is very general and can be extended to arbitrary loss functions and a wide range of regularizers.

Finally, biased regularization can also be motivated from a MAP (maximum a posteriori) interpretation of SVMs (Sollich, 2000), where the L_2 -regularizer comes

from a Gaussian process prior on \mathbf{w} . There, using \mathbf{w}_0 simply means that this Normal distribution has a nonzero — instead of the usual zero — mean.

5. Personalization Algorithm

This section describes how to implement SVMs with biased regularization (7) for personalization. Solving standard SVMs (4) efficiently is a well studied problem and several good implementations exist. Sequential Minimal Optimization (SMO) (Platt, 1998) is the perhaps most commonly used algorithm for solving (4), for example as in LIBSVM (Chang & Lin, 2001). SMO can be easily adapted to solve (7) instead of (4), since the only difference is a translation of the objective function in α -space. In SMO, there are basically two mechanisms: one that follows the gradient (or a related decent direction) downhill and one that makes sure that every iterate satisfies the constraints. With biased regularization, the gradient changes only by the constant ω_i , while the constraints stay the same. This simple change can be readily applied to any standard SMO implementation.

Before we give the actual pseudo code, a subtlety about the decision threshold b in (1) needs to be addressed. Suppose the generic model uses some value b_0 . One could argue that b_0 is just as much part of our prior knowledge as \mathbf{w}_0 , and should be taken into consideration accordingly. On the other hand, the non-penalized b makes the problem (6) translation invariant, and we might wish to retain this property during personalization. It is not clear which of the two approaches will give better results, and the decision will mostly be determined by the designer's personal preferences.

In the current study, we decided to put a penalty on deviations from b_0 . Mainly for the sake of simplicity, we use the same penalty as for the deviations from \mathbf{w}_0 . This is achieved by making the following changes to the method: we replace

$$\begin{aligned} \mathbf{x}_i^\top &\leftarrow (\mathbf{x}_i^\top, 1), \quad i = 1 \dots m \\ \mathbf{w}_0^\top &\leftarrow (\mathbf{w}_0^\top, b_0) \\ \mathbf{w}^\top &\leftarrow (\mathbf{w}^\top, b) \end{aligned} \quad (10)$$

and minimize

$$R(\mathbf{w}) = \frac{1}{2} \|\mathbf{w} - \mathbf{w}_0\|^2 + C \sum_{i=1}^m [1 - y_i(\mathbf{w}^\top \mathbf{x}_i)]_+ \quad (11)$$

with respect to \mathbf{w} . Note that this implicitly introduces the term $\frac{1}{2}(b - b_0)^2$, which penalizes deviations of the personalized threshold from b_0 . The problem now has the form of a linear SVM without a threshold, wherein

Algorithm 1

SVM training with biased regularization (BRSVM)

1. Initialization:

$$\alpha_i = 0, \forall i$$

$$\omega_i = y_i \mathbf{w}_0^\top \mathbf{x}_i, \forall i$$

2. Gradient:

$$s_i = y_i \sum_{j=1}^m y_j \alpha_j \mathbf{x}_j^\top \mathbf{x}_i - 1 + \omega_i, \forall i$$

3. Selection:

 find k with either

- $\alpha_k = 0 \wedge s_k \leq 0$, or
- $0 < \alpha_k < C \wedge s_k \neq 0$, or
- $\alpha_k = C \wedge s_k \geq 0$.

 if no such k exists, **exit**
4. Update:

$$\alpha_k \leftarrow \min\{\max\{\alpha_k - \frac{s_k}{\mathbf{x}_k^\top \mathbf{x}_k}, 0\}, C\}$$

$$\text{goto } 2$$

the dual reads

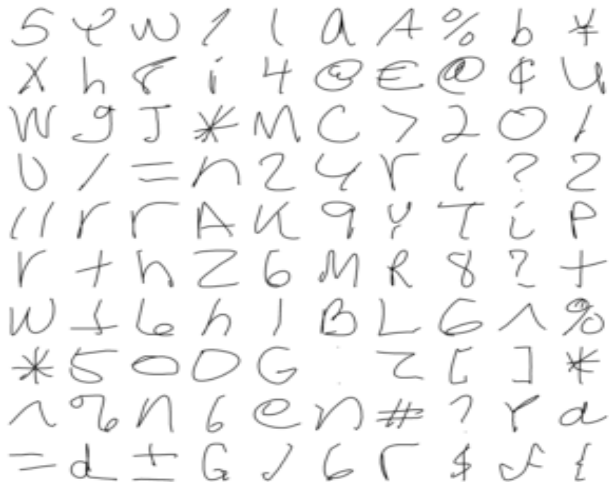
$$\min \quad \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_{i=1}^m \alpha_i (1 - \omega_i)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C \quad (12)$$

In comparison with (7), the equality constraint has vanished in (12). In the context of standard SVMs, the Kernel Adatron (KA) algorithm (Friess et al., 1998; Shawe-Taylor & Cristianini, 2004) may be viewed as a simplified version of SMO for cases with no threshold b , i.e. no equality constraint in the dual. As with SMO, KA is readily adapted to solve (12), the pseudo code is given as Algorithm 1. Line 1 initializes the coefficients to zero, and pre-computes the constant gradient offset ω_i . Lines 2 and 3 find a descent direction, and line 4 updates the coefficient α_k , subject to the constraint $0 \leq \alpha_k \leq C$. The *find* in Line 3 is deliberately loosely specified. In our implementation, we pick the k with the maximum $|s_k|$, which is a simple and popular choice (see Chang and Lin (2001)), but other heuristics are possible (Friess et al., 1998). Algorithm 1 reduces to KA (modulo the selection heuristic) for $\mathbf{w}_0 = \mathbf{0}$.

6. Data Sets

The performance of our method is tested on two real-world data sets, A and B . Data samples in both sets are uniformly distributed over 100 possible western handwritten character classes given by


 Figure 1. A random sample from data set A .

ABCDEFGHIJKLMNOPQRSTUVWXYZ
 abcdefghijklmnopqrstuvwxyz
 0123456789!"#\$%&'()*+,-./:;
 <=>?@[\\]^_`{|}~¢£¥\$°±€

The number of samples per user, however, varies between roughly 100 and 3,000. Each character sample consists of a class label $y \in \{1, \dots, 100\}$ and a feature vector $\mathbf{x} \in \mathbb{R}^{65}$. The features have been computed from ink stroke data, as in Rowley et al. (2002) and references (they are essentially coefficients of Tchebychev polynomials fitted to the ink strokes).

Data set A contains 200,000 handwritten characters from 215 users. We split set A into training (85%) and test (15%) sets. Data set B contains 84,000 samples from 21 users that did *not* contribute to set A . It has 40 samples per character and user. We also split set B into training and test set (30 and 10 samples of each character, respectively). A sample from set A is shown in Figure 1.

7. Building the Generic Recognizer

Prior to training the generic recognizer, we ran a model selection experiment to fix the regularization parameter C . For the sake of simplicity, we decided to use the same value of C for all 1-vs-1 classifiers. We computed cross-validation errors (8 folds) on training set A for various values of C (from 10^{-1} to 10^4), and chose the one that gave the lowest error. The 1-vs-1 SVMs were trained using a publicly available Matlab wrapper for LIBSVM (Chang & Lin, 2001). Figure 2 presents the

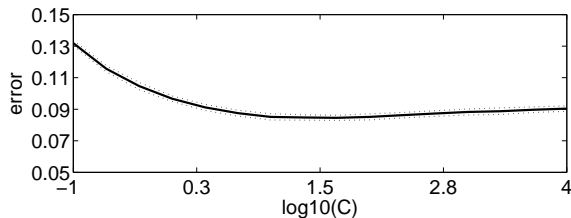


Figure 2. Fixing the regularization parameter C for the generic classifier. The solid line shows the mean cross-validation error for various choices of C , the dotted lines denote one standard deviation. The minimum is attained between $C = 10$ and $C = 100$.

cross-validation error as a function of C . As C is increased, the error first drops, reaches a minimum in the range of 10 to 100, and then increases slightly, as the model starts to overfit. As long as the value of C is above some minimum value, its effect on performance is not dramatic. Given that C varies over five orders of magnitude, the error performance seems rather insensitive to a poor choice of C . Based on these results, C was set to 20, producing an 8.4% error in cross-validation, and 8.2% error on test set A . In terms of computation time, a direct Matlab implementation yielded over 1,700 predictions per second on a 2GHz PC.

8. Personalization Experiments

At this point we discarded the generic training set A and focused on personalization performance on set B . The biased regularization approach was compared with from-scratch retraining, i.e. with simply training standard SVMs as before, albeit on personal data this time. To distinguish between the two methods, we will refer to the biased regularization method (Algorithm 1) as BRSVM, and to from-scratch retraining (standard SVM) simply as SVM.

8.1. Model Selection

Although $C = 20$ was found to work well for the generic recognizer (Section 7), we ran another model selection experiment to (re-) fix the regularization parameter C for personalization. We did this for two reasons. First, the optimal value for C may vary with the number of training samples (Schölkopf & Smola, 2002) and could change as the training set size for personalization (data set B) is two orders of magnitude smaller than for the generic recognizer (data set A). Second, C was chosen for a standard SVM, and might be inappropriate for BRSVM.

We computed cross validation errors (8 folds) on train-

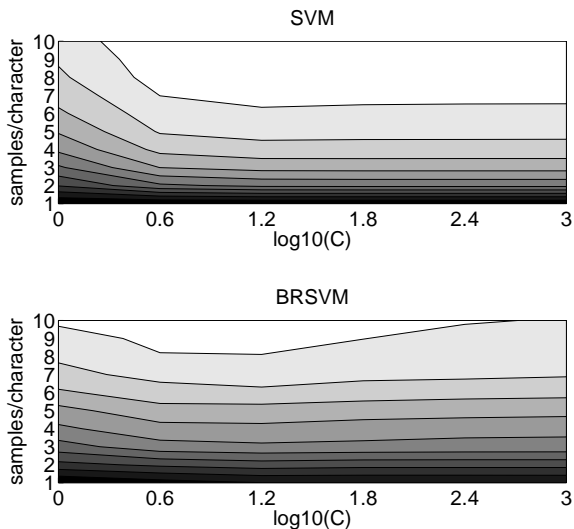


Figure 3. Fixing the regularization parameter C for the personalized classifier. Both panels show contour plots where lighter areas denote smaller error, and the black lines are level curves. The top panel illustrates how the cross validation error (averaged over all 21 users) depends on the number of samples and on C when personalizing via standard SVMs. The bottom panel shows the dependencies for the BRSVM method.

ing set B , while varying the number of training points per character from $k = 1 \dots 10$ and the regularization strength from $C = 1 \dots 1000$. As expected, for both methods accuracy improves with increasing amounts of training data. Along the C -axis, the SVM solution (Figure 3, top) does not change for values above $C = 10$, indicating that we are already in the hard margin domain there. BRSVM (Figure 3, bottom) seems to overfit slightly as C grows. A possible explanation for this is that the biased regularizer implements a tradeoff between large margin *and* closeness to \mathbf{w}_0 , and so the large margin effect decays faster as we increase C . On the other hand, BRSVM seems more robust to strong regularization than SVM. This is expected, since for $C = 0$ we still have the generic solution, $\mathbf{w} = \mathbf{w}_0$, as opposed to $\mathbf{w} = \mathbf{0}$ (SVM).

From Figure 3 we concluded that $C = 20$, which was also used for the generic recognizer, is a reasonable value for personalization as well. We therefore kept $C = 20$ unchanged for both SVM and BRSVM.

8.2. Biased Regularization vs. From-Scratch Retraining

For the comparison experiments, we randomly drew $1 \dots 20$ samples per character ($n = 100$) from training set B , which yielded training sets of size

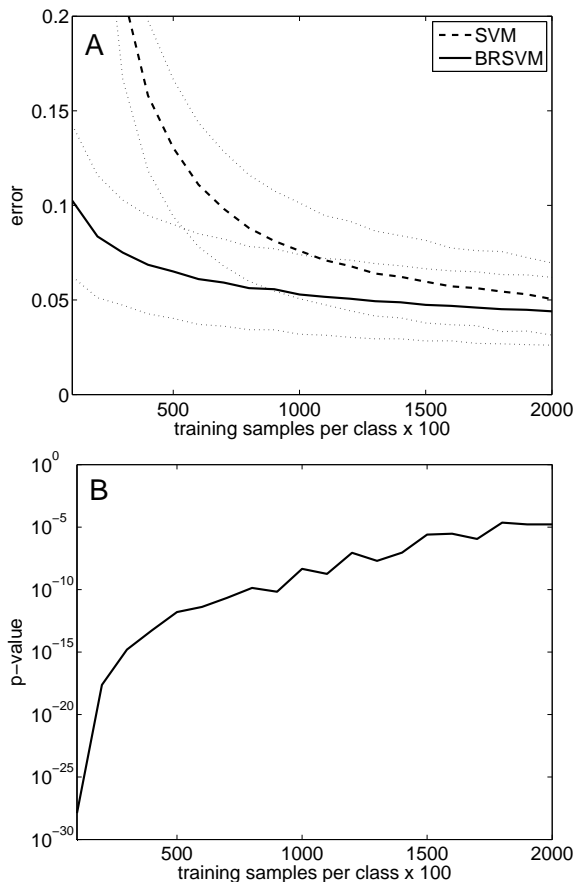


Figure 4. Top: Error rates for personalized models on user-dependent test data (test set *B*). The plot shows results for both methods (averaged over 21 users), SVM and BRSVM. The dotted lines denote one standard deviation. Bottom: p -values for a paired t -test of the improvement being insignificant (see text).

100, 200, . . . 2000. These data sets were used to sequentially personalize the generic recognizer. Experiments were repeated eight times for each user, each time with a different random seed for drawing the training subset. Figure 4 depicts the the mean error (average over all 21 users) on test set *B* as a function of the number of personalization samples.

8.2.1. PERFORMANCE GAIN

The error rates for the SVM and BRSVM versions of the personalized recognizers, are shown in Figure 4A. When no samples are available (far left), SVM operate at the baseline error rate corresponding to random guessing in a 100 class problem, which is at 99% (not shown). With BRSVM, this error rate is 10.2% and equals that of the generic recognizer on the test set *B*. The increase in this value from 8.2%, which is the

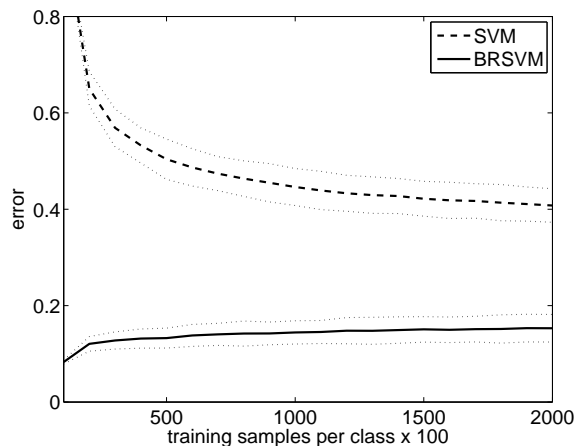


Figure 5. Error rates for the personalized model on user-independent test data (test set *A*). Analogously to Figure 4 (top), the plot shows results for both methods (averaged over 21 users), SVM and BRSVM, including error bars.

performance of the generic recognizer on the test set *A*, is probably due to the set of users in set *A* and *B* being disjoint.

BRSVM produces large improvements during personalization, even with few training points. With five samples per character, it reduces the error rate to 6.1%, which is 40% lower than without personalization (10.2%) and 45% lower than from-scratch training (11.1%).

8.2.2. SIGNIFICANCE OF THE IMPROVEMENT

As the sample size increases, the error rates for the two methods become comparable. With more than 10 or 15 samples, the differences in Figure 4A do not appear significant (one standard deviation error bars are shown as dotted lines). The relatively high variance however, comes from differences between users rather than within users. To illustrate this, the bottom plot in Figure 4 shows the p -values of a paired t -test of the hypothesis that two matched samples, i.e. the results of both methods for a fixed user, come from distributions with equal means. Examining Figure 4B, we note that the p -values are always well below 10^{-4} suggesting that even though the error bars overlap, BRSVM produces statistically significantly lower errors than from-scratch training.

Eventually, with lots of samples per class, the two methods will become indistinguishable in terms of performance. However, in a real-world setting, expecting even 10 samples per character from a user is probably already at the limit of than what we can afford in terms of usability.

8.2.3. INTER-USER GENERALIZATION

Next, we tested how well the personalized recognizers generalize to users not seen during personalization. Since the personalized recognizer is designed for use only with one specific user, inter-user generalization is not as important. However, given the improvements on the personalization set observed in Section 8.2.2 we were curious to know how much the generic recognizer had changed during BRSVM personalization, and how well the SVM personalized recognizer could generalize to other users. We took the recognizers from Figure 4 (two methods, eight trials, 21 users) and measured their performance on test set *A*. This corresponds to the scenario wherein a generic user (one of the 215 users in data set *A*) uses one of the personalized systems. Figure 5 shows the results.

The SVM recognizer starts at 99% error on the far left and learns user-independent information to some extent, but the error remains above 40%. It is obvious that samples from a single user, no matter how many there are, are not sufficient to learn a general classifier. As a result, the SVM personalized recognizer will do poorly on users other than the one it was personalized for.

On the contrary, using BRSVM, the error, starting at 10.2%, does grow, but saturates around 15.5% as we reach 20 samples per class. Even though asymptotically both methods achieve less than 5% error on the personalization data, the SVM error on the generic dataset is five times higher (48.7% vs 15.5%).

8.2.4. EXPERIMENT SUMMARY

The two main insights from the comparison experiments are

1. Personalized recognizers yield higher accuracies than a generic recognizer, even with very few training examples.
2. BRSVM yields significantly lower errors than SVM training, on both user-specific and user-independent data.

9. Related Work

The absence of a standardized data set for writer adaptation makes it difficult to compare the results obtained here to related work. Error measures also vary from handwritten characters to handwritten words, both with and without dictionaries and frequency priors. Nevertheless, we outline the results of some existing approaches on writer adaptation. Two neural

network based methods were presented by Matic' et al. (1993) and Platt and Matic' (1997), although with limited experimental results. Both methods personalize only the last layer of a neural network, which is an SVM in the former case and a constructive RBF network in the latter. The first approach uses 450 retraining samples (handwritten character) from seven users (40 classes) and yields a 2.5% character error after personalization. The performance of the generic system is not given. The second approach was tested on five users (72 classes) using 50-100 retraining samples (handwritten words) per user, and yielded a 45% improvement on the word error (using a dictionary). A larger experiment was conducted by Brakensiek et al. (2001), who used MAP estimates and EM (expectation maximization) to adapt an HMM (hidden markov model) for recognizing handwritten words from a dictionary. They report a 39% reduction in word error using 100 samples and 21 users.

The most relevant comparison can be made with the work by Connell and Jain (2002). They propose an HMM based writer adaptation scheme that uses a whole set of models (called *lexemes*) for adaptation. They report a 58% improvement in character error from 23.8% to 10% on data from 11 users over 26 character classes. Each user provided 1,100 training examples during personalization, which corresponds to roughly 42 samples per character. In our experiments, the maximum number of samples per character is much lower (at most 20) and our method produced a 57% improvement in character error rate (from 10.2% to 4.4% with 20 samples) through personalization. While the two results seem similar, please note that besides having less than half the number of training points, our data set has four times as many classes, and that our accuracies are in a higher absolute range where improvements are usually harder to achieve. Still, a direct comparison cannot be made since the data sets are different.

10. Discussion

We have presented a new personalization method for handwriting recognition. The main contribution of this paper is the use of biased regularization for personalization as a principled way of trading off user-dependent versus user-independent information. Since the proposed method is a modification of standard SVMs, it inherits desirable properties such as convexity of the risk functional or the possibility to use kernels (see Appendix A).

A comprehensive evaluation on real-world data shows that our approach performs well in practice. First,

it significantly reduces the error rate, even with very few user samples. Second, the low computation time of both evaluation ($> 1,700$ detections per second) and retraining (< 10 seconds for 200 user samples) makes it well-suited for real-time applications, also on platforms with lower computation power.

Appendix A: The Kernelized Case

An important feature of SVMs and related algorithms is that they can be turned into more general nonlinear methods by merely using nonlinear kernel functions $k(\mathbf{x}_i, \mathbf{x}_j)$ instead of dot products $\mathbf{x}_i^\top \mathbf{x}_j$. To see that the use of biased regularization preserves this property, let the generic solution be

$$\mathbf{w}_0 = \sum_{i=1}^{m_0} \beta_{0i} k(\mathbf{x}_{0i}, \cdot). \quad (13)$$

Then, replacing all dot products in (12) with kernels yields

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^m \alpha_i (1 - \omega_i) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \end{aligned} \quad (14)$$

where

$$\omega_i = y_i \sum_{j=1}^{m_0} \beta_{0j} k(\mathbf{x}_{0j}, \mathbf{x}_i). \quad (15)$$

Implementing the kernelized version of Algorithm 1 is therefore straightforward: the only required change is to replace the offset initialization in Line 1 with (15) and the dot products in Line 2 and 4 with $k(\cdot, \cdot)$.

Acknowledgments

The authors would like to thank John Platt, Chris Burges, and Patrice Simard for useful comments.

References

- Brakensiek, A., Kosmala, A., & Rigoll, G. (2001). Comparing adaptation techniques for on-line handwriting recognition. *Sixth International Conference on Document Analysis and Recognition* (pp. 486–490).
- Chang, C. C., & Lin, C. J. (2001). *LIBSVM: a library for support vector machines*. Available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Connell, S. D., & Jain, A. K. (2002). Writer adaptation for online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 329–346.
- Friess, T., Cristianini, N., & Campbell, C. (1998). The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufman.
- Hsu, C. W., & Lin, C. J. (2002). A comparison on methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, *13*, 415–425.
- Kimeldorf, G. S., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, *33*, 82–95.
- Matić, N., Guyon, I., Denker, J., & Vapnik, V. (1993). Writer adaptation for on-line handwritten character recognition. *International Conference on Document Analysis and Recognition*. IEEE Computer Society Press.
- Platt, J. C., & Matić, N. P. (1997). A constructive RBF network for writer adaptation. *Advances in Neural Information Processing Systems 9*. MIT Press.
- Platt, J. (1998). *Sequential minimal optimization: A fast algorithm for training support vector machines* (Technical Report 98-14). Microsoft Research, Redmond, Washington.
- Rowley, H. A., Goyal, M., & Bennett, J. (2002). The effect of large training set sizes on online japanese kanji and english cursive recognizers. *International Workshop on Frontiers in Handwriting Recognition*.
- Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. *Proceedings of the 14th Annual Conference on Computational Learning Theory* (pp. 416–426). Springer Verlag.
- Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. MIT Press.
- Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.
- Sollich, P. (2000). Probabilistic methods for support vector machines. *Advances in Neural Information Processing Systems 12*. MIT Press.