Learning and Adaptation for Improving Handwritten Character Recognizers

Naveen Chandra Tewari and Anoop M. Namboodiri
International Institute of Information Technology, Hyderabad 500032, INDIA
{tewari@students., anoop@}iiit.ac.in

Abstract

Writer independent handwriting recognition systems are limited in their accuracy, primarily due the large variations in writing styles of most characters. Samples from a single character class can be thought of as emanating from multiple sources, corresponding to each writing style. This also makes the inter-class boundaries, complex and disconnected in the feature space. Multiple kernel methods have emerged as a potential framework to model such decision boundaries effectively, which can be coupled with maximal margin learning algorithms. We show that formulating the problem in the above framework improves the recognition accuracy. We also propose a mechanism to adapt the resulting classifier by modifying the weights of the support vectors as well as that of the individual kernels. Experimental results are presented on a data set of 16,000 alphabets collected from 470 writers using a digitizing tablet.

1. Introduction

Development of a reliable writer independent recognition system for handwritten characters, whether for online or offline data, is still an unsolved problem. The primary challenge for such a system is the extent of variability in writing styles of different users, coupled with the similarity between certain pairs of characters (see Fig. 1). Developing a classifier that performs well, and generalize well to an unseen handwriting is extremely difficult. For example, the characters *c* and *e* are very similar in appearance, specifically, when the character *c* is written with a leading upward stroke. In short, effective classification would require one to concentrate on specific and minute differences between two character classes, while accommodating for the large variations in their writing styles.

We note that the nature of the character classes leads to complex decision boundaries between pairs of characters. Due to this nature of the problem, discriminative models such as Support Vector Machines (SVMs) [5] that integrate a collection of pair-wise classifiers tend to be the more effective [1]. Table 1 shows the classification accuracies of four popular classifiers on the IRONOFF character dataset.

Table 1. Accuracies of various classifiers on the IRONOFF character dataset.

Nearest Neighbor	Decision Tree	Multilayer Perceptron	Support Vec. Machine
84.5%	73.8%	87.6%	91.03%

To design better performing classifiers, one needs to proceed in one of the following major directions: i) Enhance the models to deal with the larger variability of the different styles of writing, or ii) Adapt the classifier to work better for a specific person's handwriting.

The use of classifier combination, where one integrates the results of various classification algorithms have been tried to achieve the first goal [3]. However, as the complexity of the final classifier goes up, one is also faced with the problem of overfitting during the training process. A viable and more principled alternative to handle the variability is to use a combination of kernels along with a maximum margin learning algorithm. The implicit regularization of the classifier's complexity during the training process that uses structural risk minimization, avoids overfitting and lead to solutions that generalize better. Multiple Kernel Learning (MKL) can effectively accommodate classes that are generated from multiple sources. In our problem, we can assume that a character class is composed of multiple generative models, each corresponding to a different style.

In this work, we explore various kernels that are effective for the problem of online handwriting recognition, and show that the MKL paradigm can be used to improve the overall classification accuracy. We further propose a

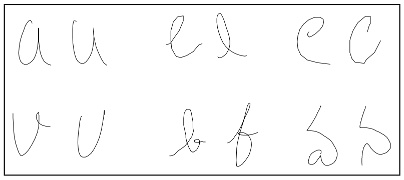


Figure 1. Confused character pairs.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.212

method to adapt the learned multiple kernel classifier to suit a specific person's handwriting. We assume that the characters are isolated (non-cursive), which is a valid assumption is many applications such as English form filling as well as many Asian and Indic scripts.

1.1 Previous Work

Development of writer independent recognition systems has been an area of active research in the past few decades due to its difficulty and application potential [9]. Generative classifiers such as HMMs [6] and template matching approaches such as DTW [8] are popular for online handwriting due to the sequential nature of the data. However, discriminative classifiers such as neural networks and support vector machines have been demonstrated to perform better to resolve similar looking characters. SVMs are discriminative classifiers with flexible decision boundaries based on a kernel function that maps the input data to a higher dimensional feature space [1]. However the use of appropriate kernel function is critical for the success of the approach, which has been studied in conjunction with the handwriting recognition problem. Bahlmann and Burkhardt [2] incorporated the DTW based distance measure, which is suitable for variable length feature vectors. However the accuracy remains low to be acceptable in general use.

It has been noted that a user's handwriting style is unique and by that account it can be used for identification of the writer [13]. It means that for a particular user's handwriting, the amount of variation present is relatively small, and is easier to model. Writer adaptation systems try to improve the performance of a generic classification system by modifying the model parameters to suit a particular person's handwriting.

Connell and Jain [4] extracted unique styles from general handwriting and train separate models on these styles. The writer adaptation is done by recognizing the styles present in a particular user's handwriting, and retraining the corresponding models using his handwriting. Vuori *et al.* [12] uses a generative model (character templates) for capturing the styles present in multi-writer data. They compare several adaptation strategies for template matching such as reshaping the prototype, adding the user's samples as prototype, and removing the confusing prototypes, for adaptation. Prototypes are generated using linear vector quantization (LVQ), which are adapted to the candidate samples by moving them towards or away from the user sample, based on their contribution to classification performance. Platt and Matic [10] exploited the fact that the output of a neural network is a function of its inputs even when the output is incorrect, and build an output adaptation module based on RBF network to adapt the recognizer.

2 Learning to be Writer Independent

The intra-class variability present among characters often result in decision boundaries that are too complex to be modeled by a single classifier. Combination of the classifiers [7] is commonly used to better model such cases. Each classifier can be based on a different model or they could be trained on different feature vectors. However, there are two important problems that limit such approaches. The individual classifiers are often trained independently, and hence do not guarantee optimal performance when combined. Moreover, as the number of classifiers are increased, the classifier training algorithms tend to overfit the training data. An ideal algorithm should be flexible to fit the complex distributions generated by the character classes, while avoiding overfitting.

2.1 Multiple Kernel Learning

Support vector machines (SVMs) provide a principled way of avoiding overfitting while learning from a set of training samples. The nature and complexity of the decision boundary of an SVM classifier depends on the kernel that is employed. The choice of kernel to optimize the performance depends on the problem at hand. Multiple kernel learning allows us to define a kernel as a weighted combination of multiple basic kernels, and learn the weights during the training phase, effectively selecting the most appropriate kernels and their relative importance. Given a set of kernels, $k_1, k_2, \ldots, k_m$, a combined kernel function may be defined as:

$$K(x_i, x_j) = \sum_{r=1}^{m} \beta_r k_r(x_i, x_j),$$

where $\beta_r \geq 0$ and $\sum_r \beta_r = 1$. The process of learning the suitable βs that can solve the problem optimally is known as multiple kernel learning. This combined kernel function can be incorporated into the framework of SVM objective function during training. Multiple kernel SVM learning solves the following convex problem [11]:

$$\min_{f,\beta,b,\xi} \frac{1}{2} \sum_{r=1}^{m} \frac{1}{\beta_r} \|f_r\|^2_{H_r} + C \sum_{i}^{N} \xi_i, \tag{1}$$

under the constraint: $y_i \left(\sum_{r=1}^{m} f_r(x_i) + b \right) \geq 1 - \xi_i$, with additional constraints on positivity and cumulative sums of β and ξ. Here, f_r is a function belonging to a reproducing kernel hilbert space (RKHS) H_r associated with kernel k_r, y_i is the label of sample x_i, b is the bias of SVM kernel, ξ_is are slack variables, and C is a regularization constant.

The challenge in applying MKL to a specific problem is to decide on the set of candidate kernels for combination by the learning algorithm. Too large a set could make the problem computationally infeasible and increase the chance of getting trapped in local minima while searching for the weights, β_i. In our experiments, we selected a set of potential kernels from a large set of possible ones using independent training, although this does not guarantee the optimal solution.

The optimization in SVM-based multiple kernel learning is done in two steps. In first step, all the weights are set to be equal and the SVM objective function is minimized. The second one consists of updating the weight vector d by an appropriate amount towards the minimum of the SVM objective function, while keeping the SVM coefficients constant. The steps in learning and classification of SVM-based multiple kernel learning are described in detail in Algorithm 1.

Algorithm 1 Multiple Kernel Learning

```
Set $\beta_r = \frac{1}{m}$ for $r = 1, \ldots, m$
for $t = 1, 2, \ldots$ do
    Minimize the SVM objective function keeping,
    $K = \sum_r \beta_r^t k_r$
    Compute the gradient of objective function w.r.t $\beta^t$
    Update the weights: $\beta_r^{t+1} = \beta_r^t + \gamma_t B_{t,r}$,
    where B is the gradient direction.
    if (change in $\beta$ is negligible), break.
end for
```

Once the kernel weights and the support vectors are learned, the classification of a new sample is straight forward as given in Algorithm 2.

Algorithm 2 Multiple Kernel Classification

```
Compute the combined Kernel matrix $K = \sum_r \beta_r^t k_r$
Solve the standard SVM classification problem with
    Kernel matrix K
```

Experiments in section 4 show that MKL can effectively model the variations in multi-writer datasets and hence improve the recognition performance. We now take a look at the second aspect of our problem, where we want to adapt a high performing generic handwriting recognizer to suit the writing style of a specific user.

3 Adapting Multiple Kernel Classifiers

The problem of writer adaptation is different from traditional classifier adaptation, where an existing generic classifier adapts to changes in distribution of data over time. In the latter case, our goal is to keep a generic classifier in tune with gradual variations in data, while for writer adaptation, we seek a new classifier that is specific to a writer, while utilizing the information gained from the generic classifier. In other words, we need to decide, what information from the original classifier is useful and what needs to be discarded to accommodate the adaptation samples.

The adaptation of MKL classifiers consists of two parts: i) adaptation of the support vectors to better model the decision boundary of a specific writer, and ii) adapt the parameters of the kernel combination (βs) to suit a specific writer's distribution. We primarily concentrate on the adaptation of the support vectors as they directly control the decision boundaries.

Multiple kernel classifiers are able to extract the complex boundary by mapping the samples into the higher dimensional feature space. However, the inter-class similarity among classes the decision boundary can be too complex, resulting in a large number of support vectors, while there variations present in a single user's handwriting is often limited. Hence, for a particular user, the decision boundary can be simple and be modeled with fewer support vectors. This can also help us to improve the recognition speed of the system, as the classification requires the computation of the kernel function for all support vectors. More importantly, certain support vectors that help model specific styles that are not present in a user's data could be negatively affecting the classification of that user's data. The removal of such support vectors can improve the recognition accuracy of the that particular user's handwriting.

To characterize the effect of support vectors, we define an influence function of the support vectors over the adaptation samples. For such a function, we would like to have the following properties: i) It should consider the amount of influence of a support vector on the discriminant function value of the adaptation samples, and ii) It should give higher weights to those samples that are near the decision boundary. The influence function of a support vector SV_i based on the user samples x_j is defined as:

$$Inf(SV_i) = \sum_j \frac{\alpha_i * label_i * label_j * K(x_j, SV_i)}{1 + |\sum_k \alpha_k * label_k * K(SV_k, x_j) - b|}, \tag{2}$$

where K is the kernel matrix. In the above definition, the numerator describes the effect of a support vector on the classification function. Moreover the sign, $label_i * label_j$, indicates whether the influence is positive or negative. The denominator is the absolute discriminant function value, which is related to the distance of the adaptation samples from the decision boundary. The resulting influence function gives higher weight to support vectors that contribute significantly to classification of a sample, relative to other SVs. The effect of a specific support vector around its location for an RBF kernel is shown in fig 2. For RBF kernels, we note that a support vector's influence depends on the distance from the sample, as well as the distance from the decision boundary.

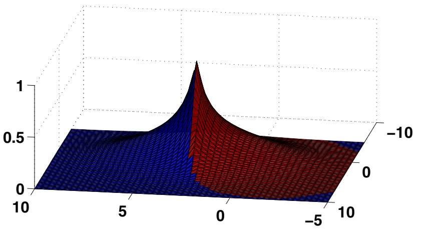
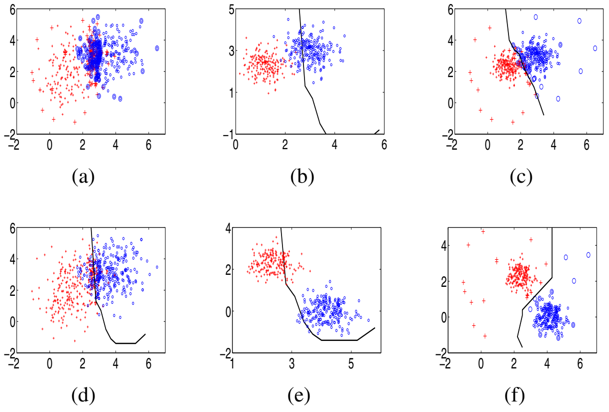


Figure 2. Influence of a support vector.

Note that a negative value for the total influence indicates that the specific support vector is doing more harm than good for classifying samples belonging to the user under consideration. Hence, we propose to remove such SVs from consideration during adaptation. It may so happen that even after removing support vectors with high negative influence, some of the misclassified sample may still be classified incorrectly. The adaptation process now re-runs the SVM optimizer with the remaining support vectors, along with the adaptation data. For re-training we keep only those samples that are near the decision boundary and may help in the recognition at later stage. If the amount of adaptation data is significant in size, one could rerun the complete MKL algorithm, thus adapting the kernel coefficients β also. Fig 3 shows the adaptation results on a synthetic dataset in two dimensions. The samples of the two classes are denoted by $+$ and o and the larger symbols denote the support vectors. The second column shows two different adaptation datasets along with the initial decision boundary, while the third column shows the results of adaptation, along with the modified support vectors.



(a) (b) (c)

(d) (e) (f)

Figure 3. (a,d): Training data with support vectors and decision boundary, (b,e): adaptation data, and (c,f): adapted classifiers.

4. Experimental Results and Analysis

To study the effect of using multiple kernels for writer independent handwriting, as well as its adaptation, we conducted our experiments on two datasets from three sources, two containing English lower-case alphabets, and the third containing 64 characters from Telugu, an Indian language.

The *Ironoff dataset* consists of $10,865$ lower case English alphabets collected from around 400 users. We augmented the dataset with another one containing $6,186$ samples collected from 70 users. We refer to this augmented dataset as the *Combined dataset* in our experiments. The *Telugu dataset* contains $16,346$ samples that covers the basic characters set (64 classes). All the datasets were collected using digitizing tablets that sample the pen position at around 100 hertz.

Each character sample is first normalized to a fixed size and then smoothened by a Gaussian filer. For RBF and polynomial kernels we spatially resampled each character to contain exactly 60 points. For each point we have extracted the following feature vectors: 1,2) normalized x and y coordinates, 3) curvature, and 4,5) sine and cosine of the tangent direction. Each character is hence mapped to a 300 dimensional feature vector.

4.1. Writer Independent Recognizer

For multiple kernel experiments, we consider Polynomial kernels of three orders and Radial Basis Function kernels with different γ coefficients. We have also considered a variable length feature vector kernel namely the Gaussian Dynamic Time Warping (GDTW). In the GDTW kernel space, the similarity between two sequences x_i and x_j is defined based on the DTW distance $(Dist(x_i, x_j))$ between the two sequences as: $K(x_i, x_j) = exp(-\gamma Dist(x_i, xj))$. Since DTW can deal with variability in the speed of writing, which was being lost in fixed length feature vectors, we consider the original variable length sample points to compute this kernel.

Table 2. Classification accuracies of single and multiple kernel classifiers.

Method	IRONOFF	COMBINED	TELUGU
RBF $(\frac{1}{2}k)$	88.36	89.51	91.67
RBF (k)	89.43	90.01	92.53
RBF $(2k)$	**91.03**	**91.04**	**92.60**
Poly $(1st)$	88.35	88.89	88.41
Poly $(2nd)$	89.20	89.56	89.34
Poly $(3rd)$	89.56	90.03	90.23
DTW	84.00	86.34	88.54
MKL	**92.71**	**93.53**	**95.42**

Table 2 shows the results of using an SVM classifier using each of the above kernels with different parameters, followed by that of the MKL classifier. The best accuracies obtained by the single kernel as well as the MKL are highlighted for comparison. The results on all three datasets consistently shows improvements over the best performing single kernel classifier, which in our problem, happens to be the RBF kernel with $\gamma = 2k$ (k is a scaling factor $= 2^{-7}$).

We note that the accuracy of GDTW kernel is low compared to kernels based on the fixed length features. However, it is interesting to note that MKL selects the RBF ($\gamma = 2k$) and GDTW kernels to achieve the best performance. We also note that addition of other kernels do not provide significant improvements to the results and the corresponding weights (βs) turn out to be close to zero. It is well known that presence of complementary information leads to better results when combined, and in our case, the complementarity comes from the comparison algorithm. The purpose of our experiments is to examine the relative merit of the multiple kernel classifier. The results presented are not claimed to be optimal, but they are definitely representative. One could further improve the overall accuracies by including additional kernels as well as features.

4.2 Writer Adaptation

For the purpose of writer adaptation, we need multiple instances of data for each character class, preferably more than ten. Hence we limit our experiments to a dataset of ten writers, each contributing at least 20 samples per character. Out of this data, 8 were used for adaptation, and the remaining were used for testing. We start with the multiple kernel classifier that was obtained from the previous experiments, and adapt them to the each user, separately. The comparison between writer independent and the adapted systems is shown in Table 3. We note that there is a consistent improvement in the accuracy due to adaptation (average 5%). However, the extent of improvement varies considerably across the writers, depending on their consistency of writing. We note that most of the errors after adaptation are due to badly written samples. As the adaptation algorithm is designed to give high importance to the new user's sample, this can significantly affect the classification performance. Our method has the ability to cope with such data, as the number of adaptation samples increases.

Table 3. Recognition accuracies before and after adaptation for 10 different writers.

Wr.ID	Initial	Adapted	Wr.ID	Initial	Adapted
1	95.64	98.76	6	90.54	94.59
2	87.96	92.43	7	94.44	98.76
3	82.67	90.12	8	74.85	80.97
4	91.87	98.46	9	92.97	93.67
5	93.15	95.21	10	63.71	78.12

5 Conclusions and Future Work

We have presented an effective and efficient method for training a writer independent handwritten character recognition system using multiple kernel learning. Experiments show up to around 30% reduction in error rate over the state of the art, when tested on multiple datasets with English and Telugu alphabets. The adaptation process shows further improvement in accuracy, especially for those writers for which the generic classifier is not performing well. We are currently exploring ways to mitigate the problem of improving the adaptation performance in presence of incorrectly written or labeled data.

References

[1] A. R. Ahmad, M. Khalia, C. Viard-Gaudin, and E. Poisson. Online handwriting recognition using support vector machine. In *Proc. TENCON'04*, pages 311–314, 2004.

[2] C. Bahlmann and H. Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:299–310, 2004.

[3] K. Chellapilla, P. Simard, and A. Abdulkader. Allograph based writer adaptation for handwritten character recognition. In *Proc. 10th IWFHR*, pages 423–428, 2006.

[4] S. D. Connell and A. K. Jain. Writer adaptation of online handwriting models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24, 2002.

[5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[6] J. Hu, S. Lim, and M. Brown. Writer independent online handwriting recognition using an hmm approach. *PR*, 33:133–147, January 2000.

[7] J. Kittler, I. C. Society, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:226–239, 1998.

[8] L. Niels, Ralph Vuurpijl. Using dynamic time warping for intuitive handwriting recognition. In *Proc. IGS2005*, pages 217–221, 2005.

[9] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22, 2000.

[10] J. C. Platt and N. P. Mati'c. A constructive RBF network for writer adaptation. In *Advances in Neural Information Processing Systems*. MIT Press, 1997.

[11] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Proc. of ICML'07*, pages 775–782. ACM, 2007.

[12] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. On-line adaptation in recognition of handwritten alphanumeric characters. In *Proc. of ICDAR*, pages 792–795, 1999.

[13] B. Zhang, S. N. Srihari, and S. Lee. Individuality of handwritten characters. In *Proc. 7th ICDAR*, pages 1086–1090, 2003.