Applying A Hybrid Method To Handwritten Character Recognition

Fu Chang[†]*, Chin-Chin Lin*[†‡] *and Chun-Jen Chen*[†]
[†]*Institute of Information Science, Academia Sinica, Taipei, Taiwan*
‡*Dept. of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan*
E-mail: fchang@iis.sinica.edu.tw, erikson@iis.sinica.edu.tw, dean@iis.sinica.edu.tw

Abstract

In this paper, we propose a new prototype learning/matching method that can be combined with support vector machines (SVM) in pattern recognition. This hybrid method has the following merits. One, the learning algorithm for constructing prototypes determines both the number and the location of prototypes. This algorithm terminates within a finite number of iterations and assures that each training sample matches in class types with the nearest prototype. Two, SVM can be used to process top-rank candidates obtained by the prototype learning/matching method so as to save time in both training and testing processes. We apply our method to recognizing handwritten numerals and handwritten Chinese/Hiragana characters. Experiment results show that the hybrid method saves great amount of training and testing time in large-scale tasks and achieves comparable accuracy rates to those achieved by using SVM solely. Our results also show that the hybrid method performs better than the nearest neighbour method.

1. Introduction

In pattern recognition, one deals with either *binary classification* in which each object is classified as one of two classes, or *multi-class classification* in which each object is classified as one of N classes, $N > 2$. SVM (Vapnik [11]) is very effective for binary classification and it can be used for multi-classification by decomposing the problem into binary classification sub-problems. Two useful methods for decomposing the problem (Hsu and Lin [1]) are one-against-one (Knerr et al. [5]) and DAGSVM (Platt et al. [8]). In the training phase, both methods require solving $N(N\text{-}1)/2$ binary classification problems. In the testing phase, the one-against-one technique conducts $N(N\text{-}1)/2$ classifications, while DAGSVM technique employs a directed acyclic graph that has $N(N\text{-}1)/2$ nodes and N leaves. The number of classifications for each object is reduced to N-1 in DAGSVM. The drawback of SVM is that, when the number of classes N becomes large, it incurs exhaustive amount of training time and produces an extremely large set of support vectors.

For large-scale pattern matching, a long-employed approach is the *nearest-neighbor* (NN) (Dasarathy [3]) classification method. The NN method, which matches each object with all training samples and finds the nearest sample or k-nearest samples as the basis for classification, takes no training time and is usually faster than SVM in large-scale pattern matching applications. In practice, the NN method is still too slow and does not perform as well as SVM.

We propose a method that exploits the advantages of both NN and SVM, and avoids their deficiencies. We conduct our method as follows. In the matching process, the set of all training samples is replaced by a much smaller set of *prototype*s. The SVM method is then used in a post-process that works on the top-rank candidates that have been obtained in the prototype-matching process. This paper presents all the ingredients in the training and testing processes associated with the hybrid method.

2. Prototype-Construction Problem and Its Solution

We assume that a set of training samples is given and that the samples' class types are also specified. Each sample is represented as a vector in n-dimensional Euclidean space. For two vectors $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ and $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ in this space, their distance is defined as

$$dist(\mathbf{v}, \mathbf{w}) = \sum_{i=1}^{n} | v_i - w_i |^2 .$$

A prototype can be any n-dimensional vector whose class type is also specified. Let $type(\mathbf{x})$ denote the class type of $\mathbf{x}$, when $\mathbf{x}$ is either a sample or a prototype. A set P of prototypes is said to be a *solution* to the problem of prototype construction if the following condition holds for every sample $\mathbf{s}$. There exists a prototype $\mathbf{p}$ in P such that $type(\mathbf{p}) = type(\mathbf{s})$ and $dist(\mathbf{s}, \mathbf{p}) < dist(\mathbf{s}, \mathbf{q})$ for all other $\mathbf{q}$ in P.

If P is a solution for prototype-construction, more than one prototype for the same class type may be found in P. If we examine the *attraction domain* of each prototype $\mathbf{p}$, defined as the set of all samples for which $\mathbf{p}$ is the nearest prototype, we find that it contains samples of the same class type. For this reason, each $\mathbf{p}$ in P can serve as the representative of its neighboring samples.

We now present a learning algorithm that solves the prototype-construction problem. The algorithm dynamically alters the number of prototypes as well as their locations, and is thus called *dynamic algorithm* (DA). It is given below.

(1) Initiation: for each class type C, the initial C-prototype is the geometric center of all C-samples.
(2) Absorption: for each sample **s**, we find the nearest prototype **p**. If *type*(**s**) = *type*(**p**) and *dist*(**s**, **p**) < *dist*(**s**, **q**) for all other prototypes **q**, then **s** is *absorbed*. Otherwise, it is *unabsorbed*.
(3) New prototype construction: for each class type C, let the number of C-prototypes be *num*(C). If there are unabsorbed C-samples, we construct *num*(C)+1 C-prototypes. Otherwise, C-prototypes remain the same as in the previous iteration.
(4) Process cessation: if there are still unabsorbed samples, go to step 2. Otherwise, we stop the whole process.

In step 3, the construction of new C-prototypes is as follows. First, we select a sample from the unabsorbed C-samples. Then, we use the selected sample and existing C-prototypes as *seeds* and employ the K-means clustering method to form new C-prototypes. To select an unabsorbed C-sample, we focus on a set Ψ consisting of unabsorbed C-samples that are *not* themselves C-prototypes. We let each sample in Ψ cast a vote to the nearest sample in Ψ. We then select the sample in Ψ that gains the highest number of votes. To construct new C-prototypes, we apply the K-means method to group all C-samples according to the following procedure. The K-means method assigns each sample to the cluster whose seed is nearest, and reset the seed as the geometric center of this cluster. This procedure continues until all cluster seeds become unchanged. The final cluster seeds are then assigned as new prototypes.

It can be proved that DA terminates within a finite number of iterations, where the number of iterations is taken as the number of times step 3 is executed. This comes from the fact that the total sum of distances between samples and nearest prototypes of the same class types decreases by at least a constant number in each iteration. A sample thus remains unabsorbed for only a finite number of iterations.

3. Disambiguation Using SVM

The prototype-matching method achieves very high accuracy rates for k nearest prototypes when $k > 1$, but has a noticeable gap between top-k and top-1 accuracy rates. The disambiguation procedure bridges this gap. There are some requisites for the training and testing process. In the training process, we must determine which class types can be mistaken for another during the prototype-matching process. These types are always paired and are thus referred to as *confusing pairs*. For these pairs, we have to specify *reassessing schemes* using an SVM method. We use these schemes in the testing process to reassess the top-k candidates for each object.

Recall that, in the prototype construction process, we must determine the nearest prototype for each training sample **s**. At the end of the process, we find k_0 nearest prototypes for each **s**. The *class types* of these prototypes will be referred to as *candidates* of **s**. In the offline process, k_0 is a small integer but is not necessarily the same as k_1 in the testing process, in which k_1 candidates of test samples are reassessed. We collect the pairs (C_i, C_j), where C_i and C_j are rank-i and rank-j candidates of **s** for $1 \le i, j \le k_0$.

For each confusing pair (A, B) and its training samples, we use SVM to create a reassessing scheme. The purpose of the SVM is to provide *decision functions* for classifying objects into class A or B, where the parameters and support vectors that appear in the decision function are derived from an optimization problem using training samples of A and B as components. Details are given in *The Nature of Statistical Learning Theory* (Vapnik [11]). For handwritten character recognition, we adopt the dual formulation of the optimization problem using the polynomial kernel of degree 2. In *The nature of Statistical Learning Theory*, comparisons of SVM and other methods for classifying UPS handwritten numerals are given. SVM is shown to perform competitively.

After completing the offline process by determining the reassessing scheme for each confusing pair, we can address the online process. Suppose that an object O is given and its first k_1 candidates are already found. We apply reassessing schemes to all confusing pairs found within the top-k_1 candidates of O. When the confusing pair is (A, B) and the unknown object is classified as A, then A scores one unit. When all the confusing pairs in the candidate list are reassessed, we re-order the involved candidates. The candidate with the highest score is ranked first, the candidate with the second highest score is ranked second, and so on. If two candidates receive the same score, their relative positions remain the same as before. We then rearrange the involved candidates according to their assigned ranks.

4. Application to Handwritten Character Recognition

To test the effectiveness of our method, we apply it to the recognition of handwritten characters. Since we aim to test the hybrid method and to compare it with some alternatives, we want to minimize the effort of searching for the best possible features and of determining the best possible values for certain parameters. As far as recognition methods are concerned, feature selection plays a secondary role, since our results suggest that the *relative* standing of the compared methods changes little using different feature extraction techniques, although the *abso-*

lute standing of their performance is strongly affected by the selections.

There are seven applications we use in our experiments. They can be divided into three groups in terms of number of class types involved. The first group contains two *small-scale* classification tasks. The databases we employ are UPS [11] and CENPARMI [9] handwritten numerals. The second group consists of two *middle-scale* classification tasks. In each task, there are 350 class types of handwritten Chinese/Hiragana characters taken from ETL8B and ETL9B databases [4]. Each class type contains the same number of samples as the original database. The third group consists of two *large-scale* classification tasks, in which we use full ETL8B and ETL9B sets.

For all databases except UPS and CENPARMI, we use the feature extraction method consisting of three basic techniques: non-linear normalization (Lee and Park [6], Yamada et al. [12]), directional feature extraction, and feature blurring (Liu et al. [7]). According to Umeda [10], these techniques are major breakthroughs in handwritten Chinese character recognition. For UPS data, we use the following extraction method. Because all images in UPS are 16×16 in scale, we simply take each number as a component to form a 256-dimensional vector. To CENPARMI data, two feature extraction methods are used. One is the same as we use for all other databases. It is referred to as 'direction' in Tables 1 through 3. The other method reduces a 64×64 original image into a 16×16 image. Each number in that image derives its value as the sum of 1's in a 4×4 block of the original image. This method is referred to as 'density' in Tables 1 through 3. We take two feature extraction methods to the same data set to show that the relative standings of classification methods are little affected by different feature extraction methods.

In Table 1, we list the number of class types (# CS), the number of training samples (# TrS) and the number of test samples (# TeS) in each application.

Table 1. Number of class types, training samples and test samples in the seven applications.

	# CS	# TrS	# TeS
UPS	10	4,000	2,000
CENPARMI (Density)	10	7,091	2,007
CENPARMI (Direction)	10	7,091	2,007
ETL8B (Subset)	350	28,000	28,000
ETL9B (Subset)	350	35,000	35,000
ETL8B (Full Set)	956	76,480	76,480
ETL9B (Full Set)	3,036	303,600	303,600

In all applications, we compare our hybrid method to 1-NN (which uses all training samples as prototypes and finds the nearest one to each test sample) and DA. The confusing pairs in the hybrid method are formed and used in the following way. In the training phase, we obtain confusing pairs (C_i, C_j), where C_i and C_j are rank-i and rank-j candidates of a sample for $1 \le i,\ j \le 5$. In the testing phase, we deal only with confusing pairs that appear in the top-3 candidates of each sample. We use the second-degree polynomial as kernel function for SVM. The software package, Torch [2], is used for conducting all SVM experiments.

In small-scale and middle-scale tasks, classification using solely SVM is feasible. We apply both one-against-one and DAGSVM methods to these data sets. In large-scale classifications, SVM takes an extremely long time for training and is thus not used. In Table 2, the first row lists all of the classification methods for comparison. When one-against-one or DAGSVM methods are used, the term 'confusing pairs' refers to the number of binary classification problems that have to solved in the training phase. From the results listed in Table 2, it is clear that the hybrid method achieves comparable accuracy rates to the two SVM methods and achieves better accuracy rates than 1-NN.

In Table 3, we list the total training and testing time (in seconds) of two SVM methods and the hybrid method. The number of support vectors (# SVs) employed in each method is also listed. Since the two SVM methods have the same training results, the results are only listed once. For the full ETL8B and ETL9B sets, SVM training times would be too long. Their times are extrapolated from those obtained from the subset data. The hybrid training consists of two parts: prototype and SVM training. To speed up prototype training, when we want to determine whether each sample **s** is absorbed or not, we match **s** only with the prototypes whose class types fall in $\Omega_k(\mathbf{s})$. For each **s**, we pre-determine $\Omega_k(\mathbf{s})$ as follows. We match **s** with all other samples and include in $\Omega_k(\mathbf{s})$ the k candidates of **s**. In all Chinese/Hiragana applications, k is set at 50. For a large value of k, the probability that the nearest prototype to **s** assumes a class type in $\Omega_k(\mathbf{s})$ is extremely high, thus justifying this speed-up method.

Table 3 shows that, compared to the SVM methods, the hybrid method requires a shorter testing time and a smaller or equal number of support vectors. The difference greatly increases when N is large, as shown in all Chinese/Hiragana results. The hybrid method also has a much shorter training time than the SVM methods when N is large. For the UPS and CENPARMI data, the hybrid method requires the same number of confusing pairs as SVM methods, thus incurring a slightly longer training time than SVM methods.

5. Conclusion

To remedy the costly computation time of SVM when N is large, we propose a hybrid solution that combines

SVM with a prototype learning/matching method. Applying this hybrid method to handwritten characters drastically cuts down training time, testing time, and the number of support vectors, as N increases. Our experiment results also show that the hybrid method maintains relatively the same accuracy rates as SVM. These results naturally prove that the hybrid method is a successful character recognition solution at all N scales.

6. References

[1] C.-W. Hsu and C.-J. Lin, A comparison of methods for multi-class support vector machines, *IEEE Transactions on Neural Networks*, vol. 13, no. 2, 2002.

[2] R. Collobert, S. Bengio, and J. Mariéthoz, Torch: a modular machine learning software library. *Technical Report IDIAP-RR 02-46*, IDIAP, 2002.

[3] B. V. Dasarathy, NN concepts and techniques, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, B.V. Dasarathy (Ed.), IEEE Computer Society Press, pp. 1-30, 1991.

[4] T. H. Hilderbrand and W. Liu, Optical recognition of Chinese characters: advance since 1980, *Pattern Recognition*, vol. 26, no. 2, pp. 205-225, 1993.

[5] S. Knerr, L. Personnaz, and G. Dreyfus, Single-layer learning revisited: A stepwise procedure for building and training a neural network, Neurocomputing: Algorithms, *Architectures and Applications*, J. Fogelman, Ed. New York: Springer-Verlag, 1990.

[6] S.-W. Lee and J.-S. Park, Nonlinear shape normalization methods for the recognition of large-set handwritten characters, *Pattern Recognition*, vol. 27, no. 7, pp. 895-902, 1994.

[7] C.-L. Liu, I.-J. Kim, and J. H. Kim, High accuracy handwritten Chinese character recognition by improved feature matching method, *4th Intern. Conf. Document Analysis and Recognition*, pp. 1033-1037, Ulm, Germany, 1997.

[8] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, Large margin DAG's for multiclass classification, *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, 2000, vol. 12, pp. 547–553.

[9] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, L. Lam, Computer recognition of unconstrained handwritten numerals, *Proceedings of the IEEE*, vol. 80, no. 7, pp.1162-1180, 1992.

[10] M. Umeda, Advances in recognition methods for handwritten Kanji characters, *IEICE Trans. Information and Systems*, vol. E79-D, no. 5, 1996.

[11] V. Vapnik, *The Nature of Statistical Learning Theory*, New York: Springer Verlag, 1995.

[12] H. Yamada, K. Yamamoto, and T. Saito, A nonlinear normalization method for handprinted Kanji character recognition – line density equalization, *Pattern Recognition*, vol. 23, no. 9, pp. 1023-1029, 1990.

Table 2. Training and testing results.

	1-NN		DA		1-against-1 SVM		DAGSVM		Hybrid	
	Acc	# PRs	Acc	# PRs	Acc	# CPs	Acc	# CPs	Acc	# CPs
UPS	94.47%	7,291	92.37%	393	95.12%	**45**	**95.47%**	**45**	95.22%	**45**
CENPARMI (Density)	92.90%	2,000	92.45%	335	95.60%	**45**	95.80%	**45**	**95.95%**	**45**
CENPARMI (Direction)	96.45%	2,000	95.15%	214	97.10%	**45**	97.35%	**45**	**97.60%**	**45**
ETL8B (Subset)	98.16%	28,000	97.81%	767	**99.47%**	61,075	**99.47%**	61,075	99.45%	**17,286**
ETL9B (Subset)	98.16%	35,000	97.81%	977	**99.05%**	61,075	**99.05%**	61,075	98.94%	**20,228**
ETL8B (Full Set)	97.00%	76,480	96.67%	3,558		456,490		456,490	**98.45%**	**64,884**
ETL9B (Full Set)	91.90%	303,600	92.71%	22,308		4,607,130		4,607,130	**96.07%**	**347,702**

Acc: Accuracy Rates; # PRs: Number of Prototypes; # CPs: Number of Confusing Pairs.

Table 3. Training time, testing time, and number of support vectors.

	1-against-1 SVM			DAGSVM	Hybrid		
	Training		Testing	Testing	Training		Testing
	Time	# SVs	Time	Time	Time (DA+SVM)	# SVs	Time (DA+SVM)
UPS	**139.0**	7,233	51.0	9.96	261 (122.0+139.0)	7,233	**3.9** (0.8+3.1)
CENPARMI (Density)	**46.9**	4,950	27.46	8.65	96.7 (49.8+46.9)	4,950	**4.1** (0.9+3.2)
CENPARMI (Direction)	**50.4**	2,753	19.56	5.96	72.6 (22.2+50.4)	2,753	**3.1** (0.7+2.4)
ETL8B (Subset)	32,046	1,666,126	233,310	1,111	**11,211** (2,184+9,027)	**565,874**	**13.1** (3.0+10.1)
ETL9B (Subset)	37,160	1,887,218	276,362	1,504	**16,029** (3,488+12,541)	**689,776**	**16.7** (3.4+13.3)
ETL8B (Full Set)	239,520	1.2×10^7	4.3×10^6	8,281	**44,155** (10,112+34,043)	**1,946,922**	**70.0** (42.0+28.0)
ETL9B (Full Set)	2,802,977	1.5×10^8	1.7×10^8	112,901	**245,421** (33,880+211,541)	**11,317,700**	**650.0** (526.0+124.0)