Pattern Classification Using Weighted Average Patterns of Categorical k-Nearest Neighbors

Yu Takigawa, Seiji Hotta, Senya Kiyasu, and Sueharu Miyahara
Department of Computer and Information Sciences, Nagasaki University, Japan
{hotta}@cis.nagasaki-u.ac.jp

Abstract

The recognition rate of the typical nonparametric method "k-nearest neighbor rule (kNN)" is degraded when the dimensionality of feature vectors is large. For reducing this difficulty, Mitani and Hamamoto have proposed a simple and strong classifier that outputs the class of a test sample by measuring the distance between the test sample and the average patterns, which are calculated using k-nearest neighbors belonging to each class. On the other hand, it is well known that distance-weighted kNN can improve its error rate due to robustness against outliers. Hence we propose a distance-weighted Mitani's classifier for improving error rates. In addition, we show how to apply kernel methods to our method. The performance of those methods is verified by experiments with handwritten digit patterns and binary classification problems.

1 Introduction

The nonparametric method of pattern recognition named k-nearest neighbor rule (kNN) is implemented on many pattern recognition systems because of its good performance and simple algorithm. The kNN rule determines the class of a test sample by voting of k-closest training samples. The main drawback in kNN is that recognition rates deteriorate when the dimensionality of feature vectors is large [1]. For overcoming this drawback, Mitani and Hamamoto have proposed the classifier that outputs the class of a test sample by measuring the distance between the test sample and the average patterns, which are calculated using k-nearest neighbors belonging to individual classes [2]. Hotta *et al.* have verified by experiments that this classifier in many cases outperformed other classifiers such as kNN and linear subspace methods [3]. In this paper, we term the Mitani's method a *CAP* classifier (classification using Categorical Average Patterns).

On the other hand, Dudani has presented the outline of distance-weighted kNN (WkNN). That is, training samples with smaller distance from a test sample are voted more heavily than ones with larger distance [4]. The WkNN rule can in some cases outperform an unweighted kNN rule when the size of training sets is finite [5]. According to this fact, it is expected that we can improve the recognition rates of CAP if we apply weighting-functions to CAP.

This paper presents a classifier that outputs the class of a test sample by measuring the distance between the test sample and the weighted average patterns, which are calculated using the categorical k-nearest neighbors and their distance values. In addition, we show how to apply kernel methods to the proposed classifier. The performance of those methods is verified by experiments with handwritten digit patterns and binary classification problems.

2 Classification Using Weighted Categorical Average Patterns

Before presenting the proposed method, we review a CAP classifier (i.e., Mitani's classifier) [2, 3] and WkNN [4].

2.1 CAP classifier

First, the procedure of CAP is explained intuitively. Figure 1 illustrates a test sample and its five nearest training samples of each class (only classes 3, 5 and 8 are shown). The CAP classifier computes the average patterns of the k-nearest neighbors for each class (see the rightmost in Figure 1). In this case, all weights of k-nearest neighbors are the same value $1/k = 1/5$. As shown in the rightmost in Figure 1, it seems that the

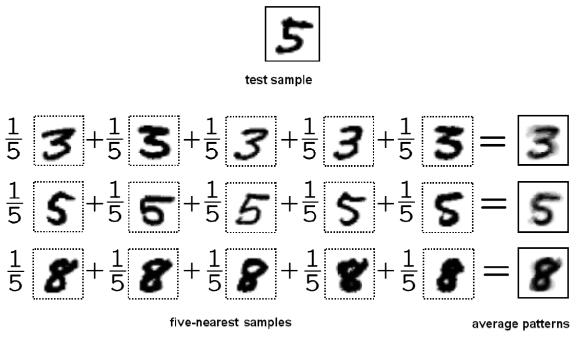

Figure 1. Outline of a CAP classifier: The top pattern is a test sample and its five nearest training samples are shown from the second row to bottom (only classes 3, 5 and 8 are shown). At the rightmost column are the average patterns of each class.

average pattern of class 5 is similar to the test sample, but other average patterns are not. In fact, if k was more than 1 the distance value between the test sample and the average pattern of class 5 was smallest, and the distance values of classes 3 and 8 were never less than that of class 5 [3]. Consequently, the CAP classifier exploits distance between a test sample and average patterns of each class for classification.

Next, the CAP classifier is formulated. Let $\boldsymbol{x}_i^j = [x_{i1}^j, ..., x_{id}^j]^T (i = 1, ..., N_j)$ be the d-dimensional training sample belonging to a class j, where N_j is the number of training samples belonging to a class j. When a test sample $\boldsymbol{q} = [q_1, ..., q_d]^T$ is given, the class of the test sample (denoted by ω) is determined by

$$\omega = \arg\min_j \left\{ \left\| \frac{1}{k} \sum_{i \in X_j} \boldsymbol{x}_i^j - \boldsymbol{q} \right\|^2 \right\}, \qquad (1)$$

where X_j is the set of the k-nearest training samples which belong to a class j. The following relationship is established between the individual samples of X_j:

$$\|\boldsymbol{x}_1^j - \boldsymbol{q}\| \leq \|\boldsymbol{x}_2^j - \boldsymbol{q}\| \leq ... \leq \|\boldsymbol{x}_k^j - \boldsymbol{q}\|. \qquad (2)$$

In short, the class that minimizes the distance between its average pattern ($\sum_{i \in X_j} \boldsymbol{x}_i^j / k$) and the test sample $\boldsymbol{q}$ is outputted as the class of the test sample. Note that CAP coincides with a nearest neighbor rule and a minimum distance method when $k = 1$ and $k = N_j$, respectively. The main shortcoming of CAP is that neighbors with large distance values deteriorate the average patterns when the value of k is large. This is attributed to the fact that all weights of k-nearest samples are equal to $1/k$ independently of their distance values. Actually, the error rates of CAP increase when the value of k is larger than the optimal one [3].

2.2 Distance-Weighted kNN rule

Next, we explain about the outline of the Distance-Weighted kNN rule (WkNN). Let w_i be the weight of the ith nearest samples. The WkNN rule determines the weight w_i by using a function of distance between the test sample and the ith nearest neighbor i.e., samples with smaller distance are weighted more heavily than ones with larger distance. Dudani has proposed the following simple function that scales weights linearly [4]:

$$w_i = \begin{cases} 1 & \text{if } d_k = d_1 \\ \dfrac{d_k - d_i}{d_k - d_1} & \text{if } d_k \neq d_1 \end{cases} \qquad (3)$$

where d_i is the distance to the test sample of the ith nearest neighbor, and d_1 and d_k indicate the distance of the nearest neighbor and the farthest (kth) neighbor respectively. Dudani has further proposed an *inverse distance weighting* function

$$w_i = \begin{cases} \dfrac{1}{d_i} & \text{if } d_i \neq 0 \end{cases} \qquad (4)$$

and a *rank weighting* function

$$w_i = k - i + 1. \qquad (5)$$

These weights are used as the value of one vote in the WkNN rule [4], but we use these functions for computing the weights of k-nearest samples belonging to individual classes.

3 Distance-Weighted CAP classifier

For overcoming the difficulty found on a CAP classifier, we employ the idea of WkNN to modify the weights of k-nearest samples. Let w_i^j be a weight of the ith nearest samples of a class j. First, the k-nearest samples of a test sample $\boldsymbol{q}$ are extracted from each class by using $d_i^j = \|\boldsymbol{x}_i^j - \boldsymbol{q}\|$. Next, the weight w_i^j is calculated by one of the above weighting-functions (e.g. $w_i^j = 1/d_i^j$). Then the weight is normalized by $w_i^j = w_i^j / \sum_{l=1}^k w_l^j$ for computing the weighted average pattern of a class j. Finally, the proposed method determines the class of a test sample by the following classification rule:

$$\omega = \arg\min_j \left\{ \left\| \sum_{i \in X_j} w_i^j \boldsymbol{x}_i^j - \boldsymbol{q} \right\|^2 \right\}. \tag{6}$$

In this paper, we term this method *WCAP* (classification using Weighted Categorical Average Patterns).

3.1 Kernel WCAP

In recent years much research has been conducted on a kernel method (e.g. [6, 7]), to which WCAP described above can be applied. If we define appropriate kernel functions between structured data such as a tree or a graph, WCAP can be applied to such structured data. In addition, if we can use kernelized WCAP, recognition rates will be improved by using an appropriate kernel function for a specific problem. Therefore, kernelization is necessary for general use of classifiers. When a test sample $\boldsymbol{q}$ is given, the kernelized WCAP rule determines the class of the test sample by

$$\omega = \arg\min_j \left\{ \left\| \sum_{i \in X_j} w_i^j \Phi(\boldsymbol{x}_i^j) - \Phi(\boldsymbol{q}) \right\|^2 \right\}, \tag{7}$$

where $\Phi(\cdot)$ is a mapping function that maps samples from an input space to a high-dimensional space, and X_j is the set of the k-nearest training samples in high-dimensional space that belong to a class j. We can represent an inner product in the high-dimensional space $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle$ by an appropriate Mercer kernel $K(\boldsymbol{x}, \boldsymbol{y})$, so the squared Euclidean distance between the test sample $\boldsymbol{q}$ and the training sample $\boldsymbol{x}_i^j$ in the high-dimensional space is written as

$$\begin{aligned} d_i^j &= \|\Phi(\boldsymbol{x}_i^j) - \Phi(\boldsymbol{q})\| \\ &= \sqrt{\langle \Phi(\boldsymbol{x}_i^j), \Phi(\boldsymbol{x}_i^j) \rangle - 2\langle \Phi(\boldsymbol{x}_i^j), \Phi(\boldsymbol{q}) \rangle + \langle \Phi(\boldsymbol{q}), \Phi(\boldsymbol{q}) \rangle} \\ &= \sqrt{K(\boldsymbol{x}_i^j, \boldsymbol{x}_i^j) - 2K(\boldsymbol{x}_i^j, \boldsymbol{q}) + K(\boldsymbol{q}, \boldsymbol{q})}. \end{aligned} \tag{8}$$

In the same way, Equation (7) can be expanded as

$$\begin{aligned} &\left\| \sum_{i \in X_j} w_i^j \Phi(\boldsymbol{x}_i^j) - \Phi(\boldsymbol{q}) \right\|^2 \\ &= \sum_{l,m \in X_j} w_l^j w_m^j K(\boldsymbol{x}_l^j, \boldsymbol{x}_m^j) - 2 \sum_{i \in X_j} w_i^j K(\boldsymbol{x}_i^j, \boldsymbol{q}) \\ &+ K(\boldsymbol{q}, \boldsymbol{q}). \end{aligned} \tag{9}$$

Hence, the class that minimizes the above equation value is outputted as the class of the test sample. In this paper, we term this method *KWCAP* (Kernel WCAP).

4 Experiments

4.1 Experimental results on handwritten digit data

We first have done experiments with the handwritten digit datasets MNIST [8] and USPS [9]. The MNIST dataset consists of 60,000 training and 10,000 test images. The USPS dataset consists of 7,291 training and 2,007 test images. It is well known that the USPS dataset is more difficult to recognize than MNIST because USPS consists of little and mislabeled training samples. For feature extraction, we extracted *local stroke direction* feature [10]. The local stroke direction technique divides each digit pattern into a 8×8 grid and assigns each pixel the direction (vertical, horizontal, diagonal right, and diagonal left) of the vector that covers the maximum number of consecutive black pixels. The numbers of pixels in each grid cell that are assigned each direction are output. This feature set represents each digit pattern as a 256 dimensional vector. We show an example of local stroke feature description in Figure 2. In addition, we use the Gaussian kernel as a kernel function in experiments:

$$K(\boldsymbol{x}_i^j, \boldsymbol{q}) = e^{-\alpha \|\boldsymbol{x}_i^j - \boldsymbol{q}\|^2}. \tag{10}$$

4.1.1 Influence of weighting-functions on error rates

First, the influence of weighting-functions on error rates of WCAP was examined by using the MNIST dataset. Figure 3 and Figure 4 show the results of test and training error rates on each weighting-function, respectively. The results of KWCAP are not included in

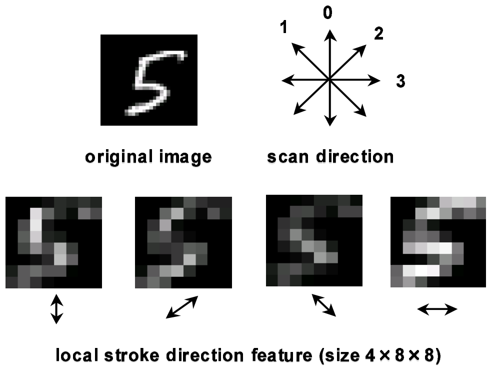

Figure 2. Example of local stroke direction feature.

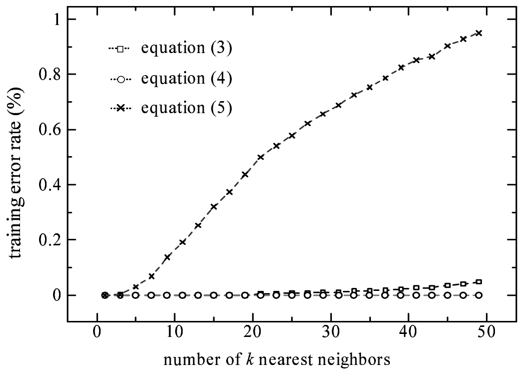

Figure 4. The training error rates of WCAP on each weighting-function.

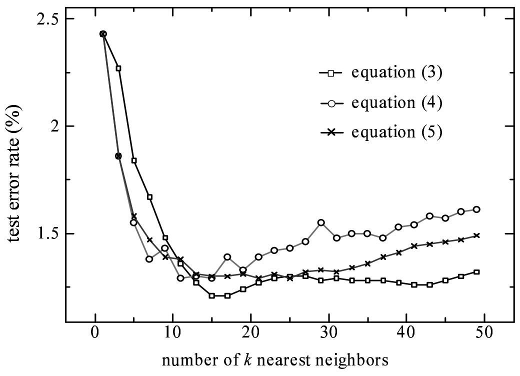

Figure 3. The test error rates of WCAP on each weighting-function.

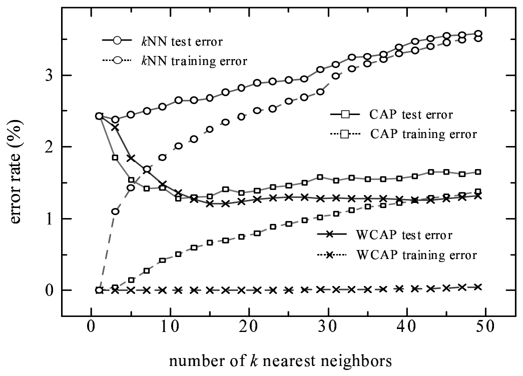

Figure 5. Relationship between k and error rates.

these figures, because they were almost same as those of WCAP. As shown in Figure 3, the test error rate obtained by Equation (3) was lower than those of Equation (4) and Equation (5) in most range of k. On the other hand, as shown in Figure 4, the training error rates obtained by Equation (3) and Equation (4) were equal to zero in most range of k. Hence we exploit Equation (3) for computing the weights of nearest samples in future experiments, which gives the lowest error rates on both test and training samples.

4.1.2 Influence of parameter k on error rates

Next, we investigated the relationship between parameter k and error rates by using the MNIST dataset. Figure 5 shows the results of kNN, CAP and WCAP. The result of KWCAP is not included in this figure, because it was almost same as that of WCAP. As shown in this figure, the error rates of kNN against test and training samples increased as k increased. In contrast, the test errors of CAP and WCAP decreased while k was less than or equal to about 15. Note that the test error rate of WCAP was almost flat while k was more than 15. In contrast, the test error rate of CAP slightly increased after $k = 15$. In addition, the training error rate of WCAP was equal to zero in most range of k, but that of CAP increased as k increased.

On the other hand, Figure 6 shows the cross-validation (leave-one-out method) curve of CAP and WCAP for the MNIST dataset. As shown in this figure, the estimated error rates of WCAP was almost flat while k was greater than 15. In contrast, the estimated

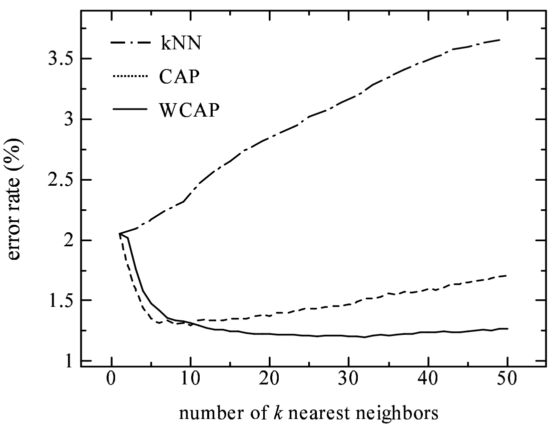

Figure 6. Cross-validation curves of CAP and WCAP on the MNIST dataset.

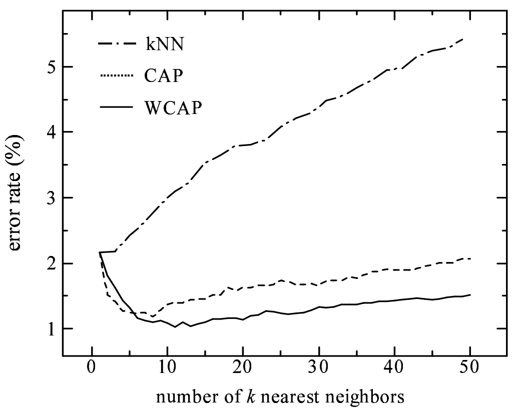

Figure 7. Cross-validation curves of CAP and WCAP on the USPS dataset.

Table 1. Error Rates on MNIST

method	test [%]	training [%]
kNN ($k = 5$)	2.39	1.43
WkNN ($k = 11$)	2.12	0.12
CLAFIC ($k = 30$)	3.68	3.87
SVM ($\alpha = 180, C = 10$)	0.92	0.01
CAP ($k = 11$)	1.28	0.50
KCAP ($k = 11, \alpha = 70$)	1.27	0.37
WCAP ($k = 15$)	1.21	0
KWCAP ($k = 15, \alpha = 70$)	1.21	0

Table 2. Error Rates on USPS

method	test [%]	training [%]
kNN ($k = 1$)	5.2	0
WkNN ($k = 5$)	4.73	0
CLAFIC ($k = 15$)	4.73	1.71
SVM ($\alpha = 260, C = 3$)	3.69	0.03
CAP ($k = 12$)	3.54	0.59
KCAP ($k = 12, \alpha = 130$)	3.44	0.43
WCAP ($k = 14$)	3.44	0.03
KWCAP ($k = 16, \alpha = 130$)	3.34	0.03

error rate of CAP increased after k=10. Similarly, the estimated error rate of kNN increased drastically after k=1. Hence selection of k on WCAP is easier than those on kNN and CAP. According to these results, it is thought that advantages of WCAP are obtained by using weighting-functions that are for robustness against outliers.

4.2 Experimental results on benchmark datasets

4.2.1 Comparison with other classifiers

Table 1 shows the lowest error rates on MNIST with parameter values of *each classifier*: kNN, WkNN with Equation (3), the basic linear subspace method *CLAFIC* [11], Support Vector Machine (SVM) with the Gaussian kernel, CAP, Kernel CAP (KCAP) [3], WCAP and KWCAP. The parameter k in CLAFIC indicates the dimensionality of subspaces. The parameter α indicates the scale parameter of the Gaussian kernel (see Equation (10)). The parameter C in SVM indicates the soft margin constant. For SVM, we used the SVM package, *LIBSVM* [12]. As shown in Table 1, SVM outperformed all the other investigated techniques, and the test error rates of WCAP and KWCAP were lower than those of kNN, WkNN, CLAFIC, CAP and KCAP.

Table 2 shows the lowest error rates on USPS with parameter values of each classifier. As shown in this table, the proposed methods WCAP and KWCAP outperformed the other classifiers even if the number of training samples is small and training set contains mislabeled samples. Figure 7 shows the cross-validation (leave-one-out method) curve of CAP and WCAP for

Table 3. Confusion matrix of SVM on USPS.

	class 0	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	class 9
class 0	352	0	2	0	2	1	1	0	1	0
class 1	0	257	0	0	5	0	2	0	0	0
class 2	3	0	190	1	2	0	0	1	1	0
class 3	0	0	3	153	1	8	0	0	1	0
class 4	0	1	2	0	192	0	1	1	0	3
class 5	2	1	1	1	0	154	0	0	0	1
class 6	1	1	1	0	3	0	164	0	0	0
class 7	0	1	1	0	6	0	0	138	1	0
class 8	2	1	1	0	1	1	0	0	158	2
class 9	0	0	0	0	1	0	0	0	1	175

Table 4. Confusion matrix of WCAP on USPS.

	class 0	class 1	class 2	class 3	class 4	class 5	class 6	class 7	class 8	class 9
class 0	355	0	2	0	0	1	1	0	0	0
class 1	0	257	0	0	5	0	2	0	0	0
class 2	0	0	193	3	0	0	0	1	1	0
class 3	2	0	2	148	0	13	0	0	1	0
class 4	0	2	3	0	190	1	0	0	0	4
class 5	2	1	1	1	0	152	0	0	2	1
class 6	1	0	0	0	1	0	168	0	0	0
class 7	0	1	1	0	3	0	0	141	1	0
class 8	2	2	1	1	0	2	0	0	158	0
class 9	0	0	0	0	0	0	0	1	1	175

the USPS dataset. As shown in this figure, the parameter k that obtained minimum estimated error rates of WCAP was almost equal to the optimal value on test samples. In contrast, the parameter k that obtained the minimum estimated error rate of CAP was smaller than the optimal one. Hence selection of k on WCAP is easier than those on kNN and CAP even if the number of training samples is small.

4.2.2 Confusion matrix on USPS

Next, we show confusion matrices of SVM and WCAP for visualization performance of each classifier. Table 3 and Table 4 present confusion matrices of SVM and WCAP on USPS, respectively. Each column of matrices represents predicted classes, while each row represents actual classes. As shown these tables, the number of combination of mislabeling of SVM is 42, but that of WCAP is 35 even if the difference of the number of misclassified patterns is 4. In other words, WCAP is confusing two classes less than SVM. This property is desired one for improving accuracy of classifiers.

4.2.3 Experimental results on binary classification problems

Finally we tested the proposed method on 13 benchmark datasets of binary classification problems [13, 14]. Each benchmark consists of 100 (or 20) random partitions of data for form test and training sets. Table 5 (see the last page) shows the lowest average test error rates and its standard deviations of *each classifier*: kNN, WkNN, CAP, KCAP, WCAP and KWCAP. Table 6 shows parameters of each classifier optimized on test error rates[1]. For comparison with other classifiers, see [13, 14]. The lowest average error rates are shown in boldface type, and the second ones are shown in italic type. As shown in this table, the results of CAP and WCAP were better than those of kNN and WkNN in some cases. In addition, the results of KCAP and KWCAP were better than those of CAP and WCAP in many cases. That is, the use of kernel method helped improve performance of CAP and WCAP.

5 Conclusions

This paper has presented the algorithm of Weighted-CAP (WCAP) that outputs the class of a test sample by measuring the distance between the test sample and the weighted average patterns, which are calculated using categorical k-nearest neighbors and their distance values. The weighting-functions that were proposed by Dudani were used for reducing the influence of samples with large distance on average patterns. In addition, we showed how to apply kernel methods to WCAP for improving the recognition performance. It was verified by experiments that WCAP was often superior to un-weighted CAP and kernel methods helped improve the recognition performance of WCAP.

In short, the proposed method includes the following advantages: 1) The proposed methods can achieve lower error rates than other nonparametric methods such as kNN, subspace methods and a un-weighted CAP classifier. 2) The proposed method can achieve low error rates even if the dimensionality of feature vectors is large. Hence, it is possible to improve recognition rates by employing kernel methods to WCAP. 3) We can implement CAP and KCAP easily because of its simple algorithms. 4) There is no need to reconstruct systems when samples are added. Future work will include speeding up and theoretical explanation of the proposed method.

Acknowledgments This research was supported by the Ministry of Education, Culture, Sports, Science and Technology in Japan under a Grant-in-Aid for Scientific Research C (No. 17500115).

[1]These parameters should be estimated by using a statistic estimator such as a cross-validation method. In this paper, however, their values were determined simply by using a rough searching.

References

[1] K. Fukunaga. Bias of nearest neighbor error estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(1):103–112, 1987.

[2] Y. Mitani and Y. Hamamoto. Classifier design based on the use of nearest neighbor samples. *Proc. of 15th Int. Conf. Patt. Recog.*, 2:773–776, 2000.

[3] S. Hotta, S. Kiyasu, and S. Miyahara. Pattern recognition using average patterns of categorical k-nearest neighbors. *Proc. of 17th Int. Conf. Patt. Recog.*, 4:412–415, 2004.

[4] S.A. Dudani. The distance-weighted k-nearest neighbor rule. *IEEE trans. on systems, man and cybernetics*, 6(4):325–327, 1976.

[5] J.E.S. Macleod, A.Luk, and D.M. Titterington. A re-examination of the distance-weighted k-nearest neighbor classification rule. *IEEE trans. on systems, man and cybernetics*, 17(4):689–696, 1987.

[6] V. Vapnik. *Statistical learning theory*. John Wiley and Sons, 1998.

[7] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Trans. on Neural Networks*, 12(2):181–201, Mar. 2001.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Intell. Signal Process.*, 306–351, 2001.

[9] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[10] T. Akiyama and N. Hagita. Automated entry system for printed documents. *Patt. Recog.*, 23(11):1141–1154, 1990.

[11] S. Watanabe, and N. Pakvasa. Subspace method in pattern recognition. *Proc. 1st Int. Joint Conf. on Patt. Recog.*, Washington DC, 2–32, 1973.

[12] C.C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. 2001, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[13] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing*, 36(10):41–48, 1999.

[14] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

Table 5. Error rates [%] and its standard deviations

dataset	# of dimension	kNN	WkNN	CAP	WCAP	KCAP	KWCAP
Banana	2	11.3 ± 0.6	11.0 ± 0.5	11.8 ± 0.5	11.4 ± 0.6	$\mathit{10.7 \pm 0.5}$	$\mathbf{10.6 \pm 0.4}$
B. Cancer	9	$\mathit{25.3 \pm 4.0}$	$\mathbf{25.0 \pm 3.9}$	26.5 ± 4.5	26.0 ± 4.5	25.9 ± 4.4	25.9 ± 4.5
Diabetes	8	25.1 ± 1.7	25.1 ± 1.8	24.5 ± 1.8	24.5 ± 1.7	$\mathbf{23.7 \pm 1.9}$	$\mathit{24.3 \pm 1.6}$
German	20	25.2 ± 2.3	24.8 ± 2.5	24.6 ± 2.3	$\mathbf{24.3 \pm 2.1}$	24.4 ± 2.5	$\mathbf{24.2 \pm 2.0}$
Heart	13	$\mathbf{15.7 \pm 3.3}$	16.1 ± 3.4	$\mathit{15.9 \pm 3.4}$	17.3 ± 3.3	16.1 ± 3.5	17.3 ± 3.3
Image	18	3.4 ± 0.5	3.4 ± 0.5	3.3 ± 0.6	$\mathbf{3.1 \pm 0.5}$	3.3 ± 0.6	$\mathbf{3.1 \pm 0.5}$
Ringnorm	20	35.0 ± 1.4	35.0 ± 1.4	12.0 ± 0.8	12.4 ± 1.0	$\mathbf{1.4 \pm 0.1}$	$\mathit{1.6 \pm 0.1}$
F. Sonar	9	34.8 ± 1.9	34.8 ± 1.8	$\mathbf{34.4 \pm 1.7}$	34.8 ± 1.6	$\mathbf{34.4 \pm 1.7}$	34.8 ± 1.6
Splice	60	26.2 ± 2.1	24.9 ± 2.3	13.5 ± 0.8	$\mathit{12.6 \pm 0.8}$	12.9 ± 0.7	$\mathbf{12.5 \pm 0.9}$
Thyroid	5	4.4 ± 2.2	4.4 ± 2.2	4.4 ± 2.2	4.4 ± 2.2	$\mathit{4.2 \pm 2.1}$	$\mathbf{4.0 \pm 2.3}$
Titanic	3	22.8 ± 1.1	22.8 ± 0.7	23.1 ± 1.9	$\mathit{22.7 \pm 1.3}$	22.8 ± 1.5	$\mathbf{22.5 \pm 1.7}$
Twonorm	20	2.5 ± 0.2	2.5 ± 0.2	$\mathbf{2.4 \pm 0.1}$	$\mathbf{2.4 \pm 0.1}$	$\mathbf{2.4 \pm 0.1}$	$\mathbf{2.4 \pm 0.1}$
Waveform	21	10.7 ± 1.0	10.4 ± 1.1	$\mathit{10.2 \pm 0.5}$	10.3 ± 0.5	$\mathbf{9.9 \pm 0.6}$	10.3 ± 0.4
# of boldface		1	1	2	3	5	7
# of Italics		1	0	2	3	2	2

Table 6. Parameter values on each classifier.

	kNN	WkNN	CAP	WCAP	KCAP		KWCAP	
dataset	k	k	k	k	k	α	k	α
Banana	11	23	8	14	15	1.4	16	0.8
B. Cancer	17	39	27	50	27	0.09	50	0.007
Diabetes	35	40	102	62	102	0.5	104	0.05
German	13	33	44	48	42	0.05	59	0.005
Heart	35	47	68	60	67	10^{-4}	60	10^{-4}
Image	1	1	2	3	2	10^{-4}	3	10^{-4}
Ringnorm	1	1	4	7	93	0.1	15	1.4
F. Sonar	37	269	254	205	246	10^{-3}	205	10^{-3}
Splice	9	20	92	193	92	0.01	202	10^{-3}
Thyroid	1	1	1	1	9	1	4	0.25
Titanic	25	24	22	29	22	0.5	29	0.05
Twonorm	95	167	114	128	177	0.1	177	0.1
Waveform	41	83	36	50	40	0.05	41	0.05