PERGAMON

Pattern Recognition 35 (2002) 2311–2318

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Improved k-nearest neighbor classification

Yingquan Wu *, Krassimir Ianakiev, Venu Govindaraju

Center of Excellence for Pattern Analysis and Recognition, State University of New York at Buffalo, Buffalo, NY 14228, USA

Received 18 July 2000; received in revised form 26 June 2001; accepted 26 June 2001

Abstract

k-nearest neighbor (k-NN) classification is a well-known decision rule that is widely used in pattern classification. However, the traditional implementation of this method is computationally expensive. In this paper we develop two effective techniques, namely, template condensing and preprocessing, to significantly speed up k-NN classification while maintaining the level of accuracy. Our template condensing technique aims at "sparsifying" dense homogeneous clusters of prototypes of any single class. This is implemented by iteratively eliminating patterns which exhibit high attractive capacities. Our preprocessing technique filters a large portion of prototypes which are unlikely to match against the unknown pattern. This again accelerates the classification procedure considerably, especially in cases where the dimensionality of the feature space is high. One of our case studies shows that the incorporation of these two techniques to k-NN rule achieves a seven-fold speed-up without sacrificing accuracy.

Keywords: k-Nearest neighbor classification; Pattern classification; Classifier; Template condensing; Preprocessing

1. Introduction

The k-nearest neighbor (k-NN) rule [1–8] is a well-known decision rule widely used in pattern classification applications. The misclassification rate of the k-NN rule approaches the optimal Bayes error rate asymptotically as k increases [3] and is particularly effective when the probability distributions of the feature variables are not known, thereby rendering the Bayes decision rule [3] ineffective. The computational inefficiency of the k-NN rule stems from the following observation. To perform template [1] matching, the complexity of each matching is $O(n)$, where n is the dimension of the feature space. In order to achieve a high recognition rate, the feature dimension n and the template size M are chosen to be large. For example, consider the GSC recognizer which uses features based on gradient, structural, and concavity aspects of a character image [8] and uses the k-NN rule to achieve high classification accuracy. It has a feature dimension of 512 and template size of 32,000 [8] making it quite inefficient to match a test pattern against the entire set of prototypes. In this paper we propose two effective techniques to improve the efficiency: template condensing and preprocessing.

Template condensing is an important part of the nearest neighbor (1-NN) rule. The set of prototypes in the template are chosen so that classification obtained using any proper subset of the initial template leads to a gradual degradation in recognition accuracy. This greatly decreases the number of prototypes that an unknown pattern must be compared to with sacrifice of accuracy [9–13]. In this paper, we develop a novel method of selecting the subset of prototypes for general k-NN classification. The idea is motivated by the observation that, if

* Corresponding author. 139 Computer & System Research Laboratory, University of Illinois at Urbana-Champaign, 1308 West Main Street, Urbana, IL 61801, USA. Tel.: +1-217-333-3202; fax: +1-217-244-1653.

E-mail address: ywu4@uiuc.edu (Y. Wu).

[1] A set of labeled prototypes.

0031-3203/02/$22.00
PII: S0031-3203(01)00132-7

a large number of prototypes form a homogeneous cluster in feature space, then the number of prototypes in the neighborhood of the test pattern is usually larger than k (sufficient number according to the k-NN rule) when the test pattern is located in this area. This observation is further strengthened by the fact that k is usually quite small in real applications in order to keep the process of searching the nearest k prototypes efficient. Our idea is to "sparsify" dense homogeneous clusters by iteratively eliminating patterns which exhibit high "attractive capacities" (defined in Section 3). This not only reduces the template size significantly but also maintains the level of classification accuracy. In this sense the method presented in this paper differs from those described in Refs. [9–13].

We also describe a preprocessing operation wherein an unknown pattern is matched against a prototype in two sequential stages. In the first stage a quick assessment of the potential of match is made. The approach is motivated by an insightful observation that the norm of a pattern vector represents a characteristic of the pattern. In order for a full match to occur in the second stage, the difference of the norms of the prototype and the test pattern must be less than a predetermined threshold. The threshold is designed for each prototype individually. Prototypes that fail in the first stage of matching are not considered any further. A large portion of the prototypes are thus dynamically precluded. This preprocessing just takes one step, i.e., the complexity is $O(1)$ and is independent of the dimensionality of the feature space. Furthermore, such preprocessing does not sacrifice the accuracy for it only rejects prototypes which are not "close" to the test pattern in feature space, if properly applied.

The rest of the paper is organized as follows. In Section 2, we introduce the general k-NN classification. In Sections 3 and 4, we present template condensing and preprocessing respectively. We present experimental results in Section 5, and draw conclusions in Section 6.

2. Preliminary: k-NN classification

Let p be the number of classes, and $\mathscr{C} \triangleq \{c^{(i)},\ i=1,2,\ldots,p\}$ be the set of class labels. Let $\boldsymbol{\Phi}$ be a set of labeled patterns referred to as a *template*. A labeled pattern $\boldsymbol{y} \in \mathbb{R}^n$ in the template is referred to as a *prototype*, where n denotes the pattern dimension. $w(\boldsymbol{y})$ denotes the weight of a prototype $\boldsymbol{y}$, i.e., the number of prototypes $\boldsymbol{y}$ in the template. The class label of a prototype $\boldsymbol{y}$ is denoted by $c(\boldsymbol{y})$.

Let $\mathscr{H}(\boldsymbol{x},\boldsymbol{y})$ be the matching measure between pattern $\boldsymbol{x}$ and $\boldsymbol{y}$, where $\mathscr{H}$ is supposed to be a non-negative and symmetric function. The larger value of $\mathscr{H}(\boldsymbol{x},\boldsymbol{y})$ shows the greater degree of similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$. The reciprocal of the Hamming distance is a commonly used measure for matching binary patterns. It is defined as follows:

$$\mathscr{H}(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|\right)^{-1}. \tag{1}$$

Let $\boldsymbol{y}^{(1)},\boldsymbol{y}^{(2)},\ldots,\boldsymbol{y}^{(d)}$, be the d prototypes which are nearest to $\boldsymbol{x}$, in the sense of $\mathscr{H}$ among all the prototypes in the template $\boldsymbol{\Phi}$, and also satisfy $\sum_{j=1}^{d} w(\boldsymbol{y}^{(j)}) = k$. The unweighted voting power of each class is computed as follows:

$$v_i \triangleq \sum_{j=1}^{d} \delta(\boldsymbol{y}^{(j)}, c^{(i)}) \cdot w(\boldsymbol{y}^{(j)}), \quad i=1,2,\ldots,p, \tag{2}$$

where $\delta(\cdot,\cdot)$ satisfies

$$\delta(\boldsymbol{y}^{(j)}, c^{(i)}) \triangleq \begin{cases} 1 & \text{if } c^{(i)} = c(\boldsymbol{y}^{(j)}), \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The weighted voting power of each class is computed as follows:

$$\tilde{v}_i \triangleq \sum_{j=1}^{d} \mathscr{H}(\boldsymbol{x},\boldsymbol{y}^{(j)}) \cdot \delta(\boldsymbol{y}^{(j)}, c^{(i)}) \cdot w(\boldsymbol{y}^{(j)}), \quad i=1,2,\ldots,p. \tag{4}$$

The unknown pattern $\boldsymbol{x}$ is classified to the class $c^{(t)}$ with the highest voting power v_t (or $\tilde{v}_t$).

We observe that the computational complexity of the matching measure $\mathscr{H}$ is $O(n)$. To achieve high accuracy, both the feature dimension (n) and the template size (M) tend to get large thus making the k-NN method time consuming. In this paper we demonstrate two novel methods to improve efficiency, namely, template condensing and preprocessing.

3. Template condensing

In the k-NN classification the $(k+1),(k+2),\ldots,$ prototypes in the template nearest to an unknown $\boldsymbol{x}$ do not affect the classification of $\boldsymbol{x}$. In fact, k is usually chosen to be a small number, otherwise sorting k nearest patterns over a template of size M, after all matching measures $\mathscr{H}(\boldsymbol{x},\cdot)$ are calculated, will need computational complexity $O(kM/p)$ [14]. Often the number of prototypes (all of a single class) which are nearer to $\boldsymbol{x}$ than prototypes of other classes is much larger than k (which gives the sufficient number according to the k-NN rule). Specifically, we can imagine an area in feature space that is homogeneous in its class label and is also very dense with the prototypes. If the test pattern $\boldsymbol{x}$ is located in such an area, then the number of prototypes in this area which are closest to the unknown pattern could be much greater than k. Herein lies the motivation for our method of reducing the template size.

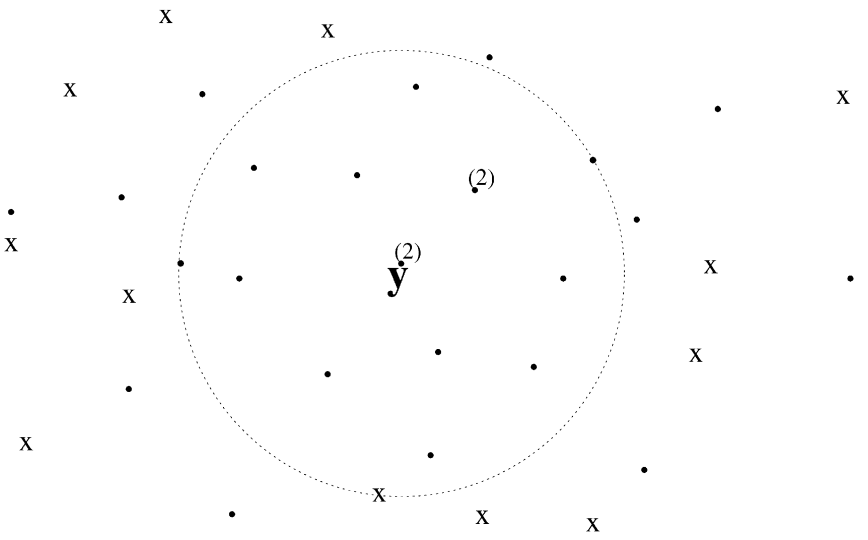

Fig. 1. Pictorial representation of the attractive capacity in 2D.

We will now describe the training process whereby we decide on the template $\boldsymbol{\Phi}$. We start with an initial (labeled) pattern set $\boldsymbol{\Omega}$ that is sufficiently large, and then iteratively refine this set using the algorithm described below.

We assume that all p classes are equally probable. The training set $\boldsymbol{\Omega}$ is initially created by extracting feature vectors from a very large set of images of interest wherein each class has equal representation. We assume the feature extraction algorithm is perfect so that any two images of different classes generate different features (pattern vectors). However, different images of the same class may have identical pattern vectors. We count the weight of each pattern and keep only one of the identical patterns. We then define the attractive capacity $s_{\boldsymbol{y}}$ of a labeled pattern $\boldsymbol{y}$ with respect to $\boldsymbol{\Omega}$ as the number of patterns from class $c(\boldsymbol{y})$ in $\boldsymbol{\Omega}$ which match better than any other patterns from other classes. Analytically, the attractive capacity of a labeled pattern $\boldsymbol{y}$ is defined as follows:

$$s_{\boldsymbol{y}} \triangleq |\{\boldsymbol{x} \in \boldsymbol{\Omega}\colon \mathscr{H}(\boldsymbol{y}, \boldsymbol{x}) > r_{\boldsymbol{y}}\}|, \tag{5}$$

where $r_{\boldsymbol{y}}$ is interpreted as attractive radius, such that

$$r_{\boldsymbol{y}} \triangleq \max\{\mathscr{H}(\boldsymbol{y}, \boldsymbol{x})\colon \forall \boldsymbol{x} \in \boldsymbol{\Omega},\ c(\boldsymbol{x}) \neq c(\boldsymbol{y})\}. \tag{6}$$

Note that the prototypes accounted for in $s_{\boldsymbol{y}}$ have the same class label as $\boldsymbol{y}$ while it is not necessary that $\boldsymbol{y} \in \boldsymbol{\Omega}$.

Fig. 1 illustrates the concept of attractive capacity. In the figure, "." denotes the prototypes of the same class as $\boldsymbol{y}$, and "x" denotes the prototypes of the class different from $\boldsymbol{y}$. In the example of Fig. 1, the attractive capacity of $\boldsymbol{y}$ is $s_{\boldsymbol{y}} = 13$.

We note that the patterns that have high capacities are at the center of areas dense in prototypes of the same class. Thus, a reasonable approach would be to remove some of the prototypes with high attractive capacity from the pattern set $\boldsymbol{\Omega}$. However, if we eliminate all the patterns whose attractive capacity is over a certain threshold, we might "over-sparsify" the dense area, and thus inevitably lower accuracy. Let us consider an example to illustrate the disadvantage of such a method. Suppose the patterns of one particular class are densely clustered in feature space but are far from the patterns of any other class, in $\boldsymbol{\Omega}$. In such cases, the capacity of all patterns in that class are equally high. Thus, we would end up removing all the training patterns of that class, according to the above method. Of course this is not reasonable. We must also keep in mind that the capacity of a prototype is determined not merely by itself but mainly by others. Furthermore, the attractive capacities change dynamically. When certain patterns are eliminated from the template, the attractive capacities of the remaining patterns may either increase or decrease depending on the distribution of the patterns in feature space.

An alternative approach is to remove the prototypes with high capacities gradually. Specifically, we can iteratively eliminate a small portion of patterns which correspond to the highest capacities. We introduce two parameters for this purpose: $\xi(t)$, $t = 1, 2, \ldots$—the maximum portion that can be eliminated in the tth iteration, and Γ—the threshold capacity below which a pattern must be reserved. $\xi(t)$ is supposed to be a decreasing function of time epoch t. The condensing procedure halts when all patterns have capacities no greater than Γ. We present the details of the procedure to condense $\boldsymbol{\Omega}$.

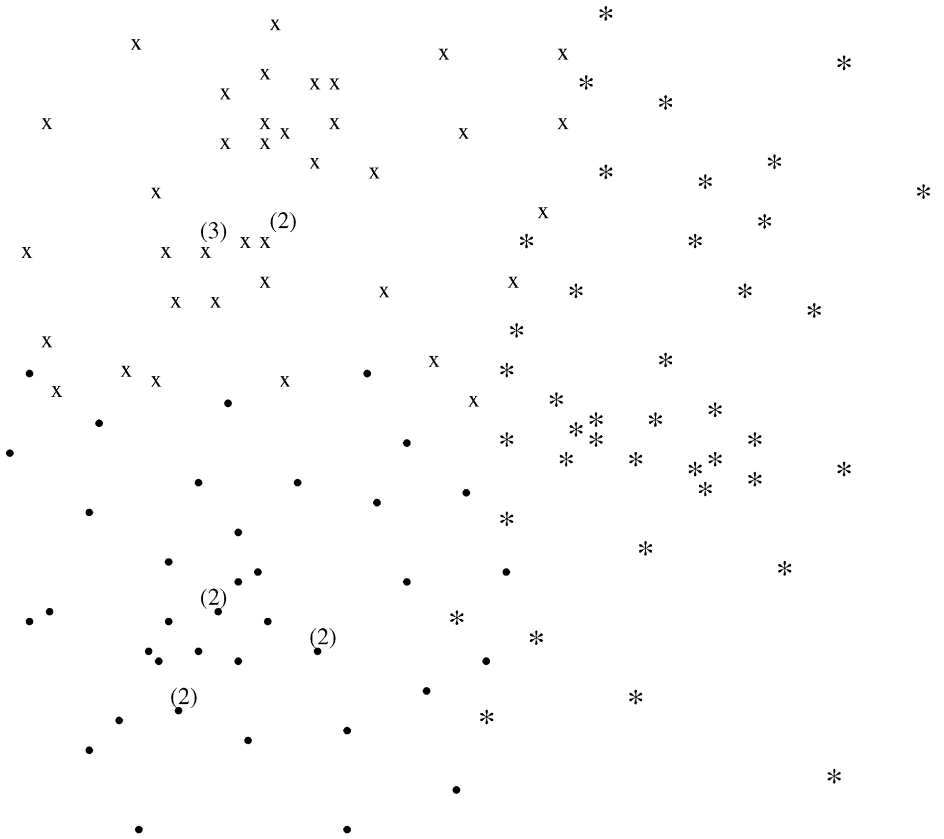

Fig. 2. Pictorial representation of original template.

3.1. Template reduction procedure

1. Calculate the weight of each pattern and keep only one out of all identical patterns. Set $t = 1$.
2. Calculate the attractive capacity for each pattern in the updated set $\boldsymbol{\Omega}$.
3. Check if all attractive capacities are no greater than $\boldsymbol{\Gamma}$. If true, set $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Omega}$ and stop.
4. Eliminate the patterns which exhibit capacities greater than Γ and are among the $\xi(t)$ portion of patterns corresponding to the highest capacities.
5. Set $t \leftarrow t + 1$; Go to Step 2.

As long as the values of $\xi(t)$ $(t = 1, 2, \ldots)$ are sufficiently small, the above approach enables a proper elimination of a large portion of redundant prototypes.

We present here an example to illustrate the proposed template reduction technique. Suppose we are given a randomly picked labeled pattern set $\boldsymbol{\Omega}$ of two dimensions as shown in Fig. 2, where each of the three classes contain 40 patterns, respectively. We also assume that the matching measure is the reciprocal of the Euclidean distance and 1-nearest neighbor rule is applied. After gradually eliminating patterns with high capacities, we finally obtain the template $\boldsymbol{\Phi}$ as shown in Fig. 3. The refined template retains 59 out of the 120 original patterns.

4. Preprocessing

In the previous section we have introduced a method to reduce the template size while maintaining nearly the original accuracy. In this section, we further enhance the efficiency of the k-NN algorithm. Our idea is to reject a large part of the template prototypes dynamically by carrying out computationally efficient preprocessing.

We observe that the norm of a prototype, $||\cdot||$, is a special characteristic of that prototype when appropriately defined (usually l_1 or l_2 norm). An unknown pattern $\boldsymbol{x}$ can be considered to be a distorted version of $\boldsymbol{y}$ if the difference of the norms, $|\,||\boldsymbol{x}|| - ||\boldsymbol{y}||\,|$, is within a certain threshold θ_y. It is worth noting that $||\boldsymbol{x}|| - ||\boldsymbol{y}|| = 0$ if $\boldsymbol{y} = \boldsymbol{x}$.

We observe that when the threshold is chosen smaller, larger numbers of prototypes are precluded from participating in the matching process. On the other hand, if the threshold is too small, the preprocessing procedure

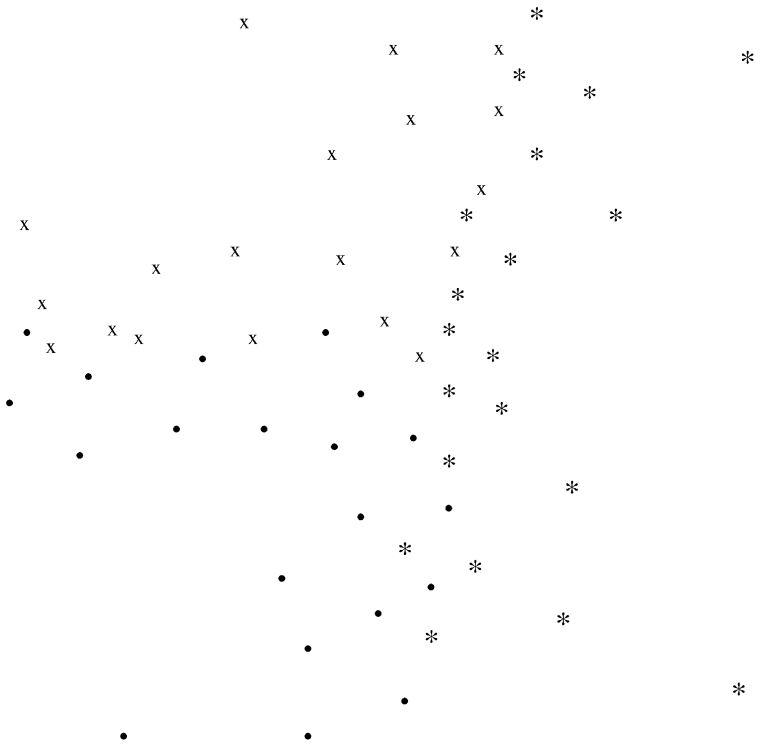

Fig. 3. Pictorial representation of reduced template.

even rejects the prototypes that are very close to the test pattern. Thus, a trade-off must be made to maintain accuracy while precluding unlikely prototypes from participating in the matching process. Keeping the competing issues in mind, we proceed to determine the appropriate threshold θ. For a given prototype $\boldsymbol{y}$, we consider its τ_y nearest neighbor prototypes $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(\tau_y)}$, from the class $c(\boldsymbol{y})$, as the distorted versions of $\boldsymbol{y}$ itself. Then, if an unknown pattern with feature $\boldsymbol{y}$ is tested (it is a coincidence that a testing pattern is identical to one of prototypes), $\boldsymbol{y}^{(i)}$, $i = 1, 2, \ldots, \tau_y$, are retained. We propose to determine the threshold as follows:

$$\theta_y \triangleq \max\{|\,||\boldsymbol{y}|| - ||\boldsymbol{y}^{(i)}||\,|\colon i = 1, 2, \ldots, \tau_y\}. \tag{7}$$

However, this gives rise to another problem. How does one determine the number of neighbors, τ_y? As can be readily seen, τ_y is connected to the attractive capacity s_y. Moreover, higher attractive capacity should lead to higher numbers of attractive neighbors. Hence, we determine τ_y by

$$\tau_y \triangleq \min\left\{\tau\colon \sum_{i=0}^{\tau} w(\boldsymbol{y}^{(i)}) \geqslant \mathscr{F}(s_y)\right\}, \tag{8}$$

where $\mathscr{F}(\cdot)$ is a non-decreasing function. In order to be consistent with the k-NN classification rule, it is necessary that

$$\mathscr{F}(1) \geqslant k - 1. \tag{9}$$

The conformance to the k-NN rule can be demonstrated as follows. When a prototype $\boldsymbol{y}$ satisfying $s_y = 1$ is used as a pattern to be tested, then the closest prototype is itself; the second closest one is from another class; the $3, 4, \ldots, k$ closest ones are perhaps from the same class as $\boldsymbol{y}$. So it is necessary to keep the $k - 1$ closest neighbors of the same label (weights of prototypes are counted).

In summary, we first determine $\tau_y = \mathscr{F}(s_y)$. Then we determine τ_y prototypes, say $\boldsymbol{y}^{(i)}$, $i = 1, 2, \ldots, \tau_y$, which are closest to $\boldsymbol{y}$ and in the same class as $\boldsymbol{y}$. We finally determine the threshold θ_y such that $\theta_y \leftarrow \max_{1 \leqslant i \leqslant \tau_y}\{|\,||\boldsymbol{y}|| - ||\boldsymbol{y}^{(i)}||\,|\}$.

We may further deliberate preprocessing when $\mathscr{H}$ is symmetric in terms of $\boldsymbol{x}$ (thus $\boldsymbol{y}$ by symmetry to $\boldsymbol{x}$) in index subset (instead of whole index set $I \triangleq \{1, 2, \ldots, n\}$) $I_1, I_2, \ldots, I_l$, such that $\bigcup_{i=1}^{l} I_i = I$, $l \ll n$ (e.g., $l = 2$ vs. $n = 512$), and $|I_i| \gg 1$, $i = 1, 2, \ldots, l$. Then, it is appropriate to apply the above procedure to each index subset $I_1, I_2, \ldots, I_l$, respectively. Both the "or" and "and" relations can be used to combine the results of the sub-preprocessing procedures.

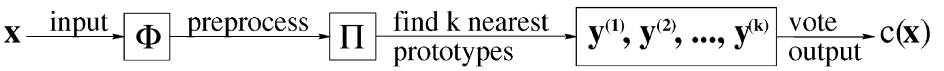

Fig. 4. Classification process.

Next we consider the issue of complexity. Note that norm and threshold (norms and thresholds of index subsets in latter) of each prototype can be pre-calculated in the training procedure. We just need to calculate the norm (norms of index subsets in latter) of unknown pattern once. A preprocessing procedure requires at most 3 ($4l-1$ in latter) operations. Most importantly, it is independent of the pattern dimension n. On the other hand, to obtain the matching measure $\mathscr{H}(\boldsymbol{y},\boldsymbol{x})$ the complexity is $O(n)$. Thus, the larger the pattern dimension n, the more powerful the preprocessing effect. A high dimension representation is usually chosen to provide high separability between different classes. The preprocessing method described can work effectively in such cases.

To classify an unknown pattern $\boldsymbol{x}$, we first check whether $|\,||\boldsymbol{x}|| - ||\boldsymbol{y}||\,| \leqslant \theta_y$. If the check fails, we simply do not consider $\boldsymbol{y}$ any further. Fig. 4 shows two stages of classification, where $\boldsymbol{\Pi}$ denotes the set of prototypes $\boldsymbol{y}$ which have qualified as the candidate prototypes of $\boldsymbol{x}$, i.e., $|\,||\boldsymbol{x}|| - ||\boldsymbol{y}||\,| \leqslant \theta_y$, and $c(\boldsymbol{x})$ denotes the label that $\boldsymbol{x}$ is classified to.

5. Experimental results

In this section we describe the application of the two techniques described in Sections 3 and 4 to handwritten numeral recognition where the number of classes is 10 ($p=10$). The training set $\boldsymbol{\Omega}$ of 126,000 patterns has an equal number of patterns in each class. The testing set has 25,300 patterns and again equal number in each class. The experimental platform is the SPARC 400 MHz computer.

In our first case study, the developed techniques are applied to the "Gradient" recognizer, which encodes local contour variation of the character image into a binary feature vector of 192-dimension (i.e., $n=192$) [7]. Unweighted k-NN classification is applied with the following matching measure $\mathscr{H}$:

$$\mathscr{H}(\boldsymbol{x},\boldsymbol{y}) = 5\cdot|\{1\leqslant i\leqslant n\colon x_i=y_i=1\}| + 2\cdot|\{1\leqslant i\leqslant n\colon x_i=y_i=0\}| \quad (10)$$

and $k=6$.

"Original" classification uses the training set of 126,000 patterns as template and the original k-NN classification rule. The experimental result is shown in Table 1 (Original). In the template reduction procedure, we set $\xi(t)=0.25(t+1)^{-0.5}$ and $\Gamma=300$. After seven iterations, the refined template retains 29,345 prototypes out of the 126,000. As shown in Table 1 in terms of Improved I, it reveals similar accuracy as the original k-NN algorithm. We also apply the proposed preprocessing to accommodate Improved I classifier. We limit the range of parametric function $\mathscr{F}(s)$ in Refs. [10, 30], and adopt the following definition $\mathscr{F}(s)=10+\lceil 20\frac{2}{\pi}\arctan(s/30)\rceil$. The experimental results are presented in Table 1 (Improved II). Preprocessing further reduces (average) classification time by roughly half. We note that the accuracy is about the same as the original. A combination of both techniques reduces the classification time by a factor of 6 at almost no loss in accuracy.

Table 1
Comparisons of performance of k-NN rule with or without developed techniques for "Gradient" recognizer

k-NN rule	Original	Improved I	Improved II
Mean time (ms/p)	134.5	32.1	21.2
Accuracy (%)	95.92	95.93	95.90

We then use the same labeled pattern set and conduct a similar study on the GSC recognizer [8]. The GSC classifier is based on a feature set that captures the gradient, structural, and concavity information from an image. Gradient features use the stroke shapes on a small scale (192 features). The structural features are based on the stroke trajectories at the intermediate scale (192 features). Finally, the concavity features use stroke relationships at a global level (128 features). In all, there are 512 binary "GSC" features, i.e., $n=512$ [8]. The classification procedure uses the weighted k-NN rule. The following matching measure $\mathscr{H}$ is used:

$$\mathscr{H}(\boldsymbol{x},\boldsymbol{y}) = 2\cdot|\{1\leqslant i\leqslant n\colon x_i=y_i=1\}| + |\{1\leqslant i\leqslant n\colon x_i=y_i=0\}| \quad (11)$$

and $k=6$.

We set $\xi(t)=0.25(t+1)^{-0.5}$ and $\Gamma=300$ (same as the first study). The final template of 32,312 prototypes are chosen after six iterations. The template size is reduced by 74.4%. The results are listed in Table 2 (Improved I). In the preprocessing stage, the parametric function

Table 2
Comparisons of performance of k-NN rule with or without developed techniques for "GSC" recognizer

k-NN rule	Original	Improved I	Improved II
Mean time (ms/p)	307.1	78.7	41.5
Accuracy (%)	97.62	97.59	97.64

$\mathscr{F}(s)$ is set as $\mathscr{F}(s) = 5 + \lceil 20\frac{2}{\pi} \arctan(s/30) \rceil$. As we can see from Table 2 (Improved II), preprocessing enables a significant savings in computation time. We note that Improved II classifier even shows a little improvement in accuracy. This is not unusual since preprocessing efficiently gets rid of unlikely matches, and can improve accuracy by blocking some of the original k nearest prototypes which would result in potential mismatches. It also verifies the claim that preprocessing is more powerful in cases where the features space has large dimensionality by comparing to the first case study. In summary, the combination of the two techniques reduces the (average) classification time by more than 7 times with *no* accuracy drop.

6. Conclusions and future studies

In this paper we have shown how to improve the efficiency of the k-NN classification by incorporating two novel ideas. The first idea is the reduction of the template size using the concept of attractive capacity. The second idea is a preprocessing method to preclude participation of a large portion of prototype patterns which are unlikely to match the test pattern. This work notably speeds up the classification without compromising accuracy.

The proposed template reduction technique is distinct from the methods in literature, which reduce the template size but result in classification accuracy degradation [9–13]. Our method attempts to sparsify the dense attractive areas and is efficiently implemented by gradually eliminating the prototypes with high attractive capacity. It allows us to discard a lot of redundant prototypes while keeping useful ones. Consequently, it is able to reduce the template size but maintain the same level of the accuracy. In the future, we would like to adopt a strict mathematical framework to derive optimal rules for the selection of prototypes.

The preprocessing described is based on the observation that the norm represents the intrinsic characteristic of a pattern vector. We have established that a test pattern is likely a distorted version of a prototype when the norm difference is below a threshold associated with that prototype. Each preprocessing operation takes $O(1)$ steps compared to the full matching complexity of $O(n)$. When the pattern dimension n is large, it is able to preclude a large portion of prototypes which are unlikely to match with the unknown pattern, and thus significantly promote classification efficiency while maintaining accuracy. We make two remarks here. Firstly, our work is built on the intuitive observation that "the norm is an intrinsic characteristic". Secondly, we have not proven theoretically that the proposed preprocessing is guaranteed to filter out only irrelevant prototypes and maintain accuracy.

Acknowledgements

The authors would like to thank the anonymous referees for their numerous comments which improved and clarified the presentation a lot.

References

[1] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inform. Theory 13 (1967) 21–27.
[2] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[3] K. Fukunaga, L.D. Hostetler, k-nearest-neighbor Bayes-risk estimation, IEEE Trans. Inform. Theory 21 (1975) 285–293.
[4] J.M. Keller, M.R. Gray, J.A. Givens Jr., A fuzzy k-nearest neighbor algorithm, IEEE Trans. Systems Man Cybernet. 15 (1985) 580–585.
[5] S.A. Dudani, The Distance-Weighted k-Nearest Neighbor Rule, Neighbor Neighbor Norms: NN Pattern Classification Techniques, IEEE Computer Society Press, Los Alamitos, CA, 1991, pp. 92–94.
[6] K. Hattori, M. Takahashi, A new nearest-neighbor rule in the pattern classification problem, Pattern Recognition 32 (3) (1999) 425–432.
[7] G. Srikantan, Image sampling rate and image pattern recognition, Dissertation, State University of New York at Buffalo, 1994.
[8] S.W. Lam, G. Srikantan, S.N. Srihari, Gradient-based contour encoding for character recognition, Pattern Recognition 29 (7) (1996) 1147–1160.
[9] P.E. Hart, The condensed nearest neighbor rule, IEEE Trans. Inform. Theory 14 (1968) 515–516.
[10] G.W. Gates, The reduced nearest neighbor rule, IEEE Trans. Inform. Theory 18 (1972) 431–433.
[11] C.L. Chang, Finding prototypes for nearest neighbor classifiers, IEEE Trans. Comput. 23 (11) (1974) 1179–1184.
[12] G.L. Ritter, H.B. Woodruff, S.R. Lowry, T.L. Isenhour, An algorithm for a selective nearest neighbor decision rule, IEEE Trans. Inform. Theory 21 (1975) 665–669.
[13] R.L. Brown, Accelerated template matching using template trees grown by condensation, IEEE Trans. Systems Man Cybernet. 25 (3) (1995) 523–528.
[14] S.O. Belkasim, M. Shridhar, M. Ahmadi, Pattern classification using an efficient KNNR, Pattern Recognition 25 (1992) 1269–1274.

About the Author—YINGQUAN WU received the B.S. and M.S. degrees in Mathematics from the Harbin Institute of Technology, Harbin, P. R. China, in 1995 and 1997, respectively. He received the M.S. degree in the Department of Electrical Engineering, State University of New York at Buffalo, USA, in 2000. Since 2000, he has been pursuing a Ph.D. in the Department of Electrical & Computer Engineering at the University of Illinois at Urbana-Champaign, USA.

About the Author—KRASSIMIR IANAKIEV received a Master (Hons.) degree from Sofia University in 1989 and a Ph.D. degree in Mathematics and Computer Science and Engineering in 1998 and 2000, respectively. His research interests include pattern recognition and fuzzy systems.

About the Author—VENU GOVINDARAJU received his Ph.D. in Computer Science from the State University of New York at Buffalo in 1992 and Bachelors of Technology from the Indian Institute of Technology, Kharagpur, in 1986. Venu has co-authored a total of over 115 technical papers (26 in journals) and has one US patent on cursive script recognition. His main areas of interest are human computer interaction and pattern recognition. He is currently the associate director of the Center of Excellence for Document Analysis and Recognition (CEDAR) and concurrently holds the associate professorship in the Department of Computer Science and Engineering, University at Buffalo. He is the associate editor of the Journal of Pattern Recognition and the IEEE Transaction on Pattern Analysis and Machine Intelligence. Venu Govindaraju is the Program Co-chair of the upcoming Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR) in 2002. He is a senior member of the IEEE.