A Fast Algorithm for a k-NN Classifier Based on Branch and Bound Method and Computational Quantity Estimation

Shin'ichiro OMACHI[†] and Hirotomo ASO[†]

[††] Graduate School of Engineering, Tohoku University, Sendai-shi, 980-8579 Japan

SUMMARY

Nearest neighbor rule or k-nearest neighbor rule is a technique of nonparametric pattern recognition. Its algorithm is simple and error is smaller than twice the Bayes error if there are enough training samples. However, it requires enormous computational quantities that is proportional to the number of samples and the number of dimensions of feature vector. In this paper, a fast algorithm for k-nearest neighbor rule based on branch and bound method is proposed. Moreover, a new training algorithm for constructing a search tree that can reduce the computational quantity is proposed. Experimental results show the effectiveness of the proposed algorithms.

Keywords: pattern recognition, nearest neighbor rule, k-nearest neighbor rule, branch and bound method, character recognition

1 Introduction

The nearest neighbor rule [1] is one of the simple and precise classification methods. Its algorithm is quite simple, and it is a nonparametric method that does not need knowledge about distribution of patterns. If there are enough training patterns, classification error will be smaller than twice the Bayes error. Because of its simplicity and precision, many researches are done, e.g., improving the algorithm itself [2, 3] or combining it with other methods [4]. Nearest neighbor rule can be expanded to k-nearest neighbor rule by using k neighbors of an unknown input pattern. Finding k neighbors is used not only for classification but also for estimation of sample distribution and Bayes error [5, 6]. However, these methods require enormous computation complexity that is proportional to the number of training samples and the number of dimensions of feature vector.

Some methods for finding k neighbors have been proposed. Fukunaga et al. have proposed a fast search method based on branch and bound method [7]. It realizes fast search by skipping search of subtrees those are unnecessary to be searched. However, although the efficiency of search strongly depends on the structure of search tree, the propriety of the construction method and the structure of search tree, e.g., height of the tree, number of children of one node, clustering algorithm, are not mentioned in [7]. Djouadi has proposed a method to decrease the number of training samples that are needed for distance calculation by dividing space. However, the effectiveness decreases as the number of dimensions of feature vector increases, and the method is not effective if the number of dimensions is greater than seven [8].

There are some methods for fast recognition using the nearest neighbor rule or the k-nearest neighbor rule on condition that they are only used for pattern recognition but not for estimation of sample distribution. These methods are mainly classified into two types. In some methods the number of samples for distance calculation is limited, and in the other methods the search space is limited. The former reduces computation time and space complexity, but the latter reduces only time complexity. To limit the number of samples for distance calculation, an effective subset are calculated from training data set [9, 10, 11], or a new set is reconstructed for classification [12]. The method proposed by Sethi [13] is the one that limits the search space. By using distances from unknown input to three points in the search space, the search space is limited. Consequently the number of samples for distance calculation becomes small, and the computation time is decreased. However, there is no guarantee that the results using subsets or using selected new sets are the same as the results using the original nearest neighbor rule or k-nearest neighbor rule. As described above, the nearest neighbor rule has desirable property such that the classification error is smaller than twice the Bayes error. It is meaningful to get the same results as the results using original methods.

In this paper, a fast algorithm for recognition using the k-nearest neighbor rule (or the nearest neighbor rule) is proposed. High dimensional vectors used for character recognition are the targets. This algorithm is based on branch and bound method. The method of Fukunaga et al. [7] focuses on finding k nearest neighbors in the strict sense. However, in pattern recognition by the k-nearest neighbor rule, candidate category is decided by a majority vote. It means that it is not necessary to find the exact k nearest neighbors. It is sufficient that the number of training samples of a certain category among the k nearest neighbors is more than $\lceil k/2 \rceil$. The proposed method focuses on this point and realizes an efficient search, and the same results as the results of the original method are obtained. Moreover, a new training method that constructs a search tree by estimating the computational complexity is proposed. Experimental results using feature vectors used for character recognition show the effectiveness of the proposed method.

2 The method of Fukunaga et al. [7]

In this section, the method that finds nearest neighbor fast proposed by Fukunaga et al. is briefly described.

First, training data set is divided into l subsets, moreover each subset is divided into l subsets again. By applying this procedure recursively, the search tree is constructed. An example of search tree ($l = 3$) is illustrated in Fig. 1. Each leaf of the search tree corresponds to one training sample. Consecutive numbers are assigned to the nodes except for the leaves. The root is node 0 and children of the root (nodes at level one) are node 1 through l. Node p has four parameters S_p, N_p, $\boldsymbol{\mu}_p$ and r_p. S_p is the set of samples associated with node p, N_p is the number of samples in S_p, $\boldsymbol{\mu}_p$ is the mean value of samples of S_p, and r_p is the distance from $\boldsymbol{\mu}$ to the farthest sample in S_p.

When an unknown input $\boldsymbol{x}$ is given, the nearest neighbor is found by searching this tree by depth first search. Among the nodes at the same level, the node which has smaller distance $d(\boldsymbol{x}, \boldsymbol{\mu}_p)$ is searched earlier. Using the following two rules, the number of nodes and samples are restricted, and fast search is realized.

Rule 1 Denote the distance from the unknown input to the current nearest sample as B. All the

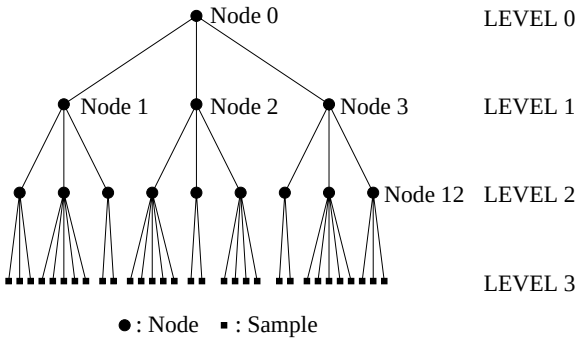


Figure 1: Search tree of [7].

nodes associated with node p does not need to be searched, if

$$B < d(\boldsymbol{x}, \boldsymbol{\mu}_p) - r_p.$$

Rule 2 Denote the distance from the unknown input to the current nearest sample as B. $\boldsymbol{x}_i \in S_p$ is not the nearest neighbor of $\boldsymbol{x}$, if

$$B < d(\boldsymbol{x}, \boldsymbol{\mu}_p) - d(\boldsymbol{x}_i, \boldsymbol{\mu}_p)$$

Here, $d(\boldsymbol{x}_i, \boldsymbol{\mu}_p)$ is calculated previously.

This method is easily expanded to the k-nearest neighbor rule by memorizing k nearest samples and by denoting B as the distance from the unknown input to kth nearest sample.

In [7], k-means algorithm is used as the clustering algorithm. The number of l is three, and the height of the search tree is four. That is, training data set is divided into 27 subsets. However, propriety of the clustering algorithm, number of l and height of the tree is not mentioned.

3 Proposed method

In this paper, a fast method is proposed on condition that the k-nearest neighbor rule is only used for pattern recognition. The proposed method is based on branch and bound method like the method proposed by Fukunaga et al.

In the case that the k-nearest neighbor rule is adopted to pattern recognition, it is not the most important thing that whether the exact k nearest neighbors are found or not. Denote the classification answer of traditional k-nearest neighbor rule is category c. If more than $\lceil k/2 \rceil$ samples among the selected k nearest neighbors are from category c, the proposed method can give the same classification answer as the traditional one.

For a fast search, it is important to reduce the number of calculation times of distance between an unknown input and training samples. We propose a method to construct a search tree that reduces the computational quantity by estimating the number of calculation times of distance.

In this section, first the search tree used in the proposed method is described, and the search algorithm is shown. Then the number of times of distance calculation is estimated and a training algorithm that reduces this number of times is proposed.

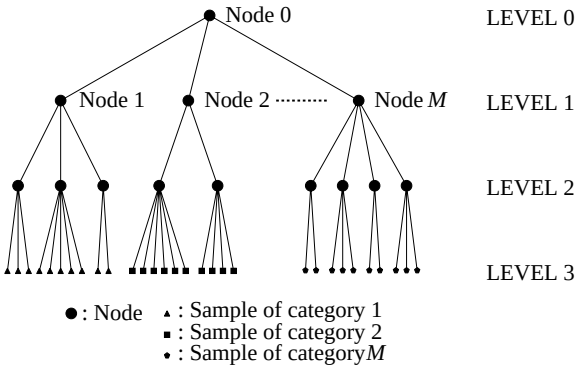


Figure 2: Search tree.

3.1 Search tree

Let M be the number of categories, D be the number of dimensions of feature vector, and N be the number of training samples of each category. In the search tree used in the proposed method, each node at level one whose parent is root node corresponds to one category. Accordingly the number of nodes at level one is M, and each leaf that is the descendant of the node p $(1 \leq p \leq M)$ corresponds to one training sample of category p. The subtree whose parent is node p is determined by hierarchical clustering. An example is shown in Fig. 2. The number of children of each node is not fixed.

In our method, the set of training samples are divided into subsets hierarchically to construct a search tree. For this reason, hierarchical clustering method is needed. It is efficient for the search if Rule 1 described in Section 2 is satisfied at much more times. The complete linkage method is used here, because the maximum distance between samples in a cluster becomes small by this method.

Let $\boldsymbol{x}_i$ be a training sample, and C_i be a cluster. At the initial state,

$$C_i = \{\boldsymbol{x}_i\}, \quad 1 \leq i \leq N. \tag{1}$$

Then, i and j that minimize the following $h(i,j)$ are combined recursively.

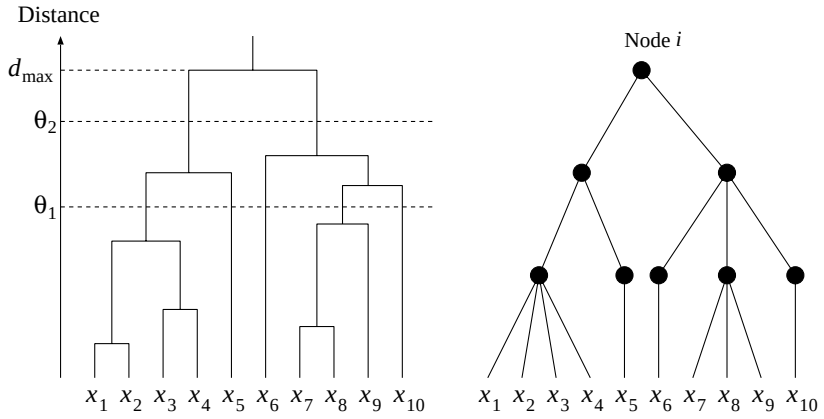
$$h(i,j) = \max_{\boldsymbol{x}_s \in C_i, \boldsymbol{x}_t \in C_j} d(\boldsymbol{x}_s, \boldsymbol{x}_t) \tag{2}$$

Therefore a clustering tree is constructed. An example is shown in Fig. 3(a). In the figure, the height of each node shows a distance determined by Eq. (2), and $d_{\max}$ is the maximum distance between two samples in that cluster.

In the proposed method, a threshold of distance is determined and the search tree is constructed by clustering tree constructed by this threshold. The search tree of Fig. 3(b) can be constructed from the clustering tree of Fig 3(a) if two thresholds θ_1, θ_2 are given.

3.2 Search algorithm and estimation of computational quantity

In the proposed method, k nearest neighbors of an unknown input are saved if any of them is changed with other neighbor during the search. The category that most samples among the final k nearest neighbors belong to is recognition result. The search algorithm consists of the following four steps.



(a) Clustering tree. (b) Search tree.

Figure 3: Clustering tree and search tree for one category.

1. Selection of standard category
2. Search of the standard category
3. Search of the category except for the standard category
4. Second search of the standard category

Here, standard category is defined as the category whose center of gravity is nearest to the unknown input. The standard category is considered to have the highest possibility that the unknown input pattern belongs to. By choosing appropriate k neighbors (not necessarily the exact k nearest ones) from the standard category, it is expected that the number of applying Rule 1 and Rule 2 described in Section 2 increases during the search of categories except for the standard category. When the search of categories except for the standard category finishes, if there are not more than half samples of the standard category among the k nearest neighbors, the standard category is searched again.

For fast search, computational quantity is estimated and the thresholds are decided according to the estimated quantity[1]. Here, we assume that high dimensional vector is used here like the one for character recognition. Therefore the value D is very large. Since the number of category M and the number of samples per category N are not so large, $D \gg \log M$ and $D \gg \log N$.

In the following, each process of search algorithm is described, and computational quantity of each process is estimated. Let H be the height of search tree, and α_L be the average value of the number of children of each node of level L $(1 \leq L \leq H-1)$. Denote the unknown input vector as $\boldsymbol{x}$. The distance between $\boldsymbol{x}$ and the center of gravity $\boldsymbol{\mu}_p$ of node p is simply called distance between $\boldsymbol{x}$ and node p.

1. Selection of standard category

For each node p at level one, the category whose distance between $\boldsymbol{x}$ and its center of gravity is the smallest is selected. That is, the category c is selected with the following expression.

$$c = \underset{1 \leq p \leq M}{\operatorname{argmin}}\, d(\boldsymbol{x}, \boldsymbol{\mu}_p) \tag{3}$$

[1]Computational quantity estimation here is done only for threshold decision.

This category c is called standard category.

For selecting standard category, distance calculation of $\boldsymbol{x}$ between each node at $L = 1$ and selection of the category whose distance is the smallest are necessary. The time for distance calculation is MD and the time for sorting is $M \log M$. Since $D \gg \log M$, time for sorting can be ignored and the computational quantity is MD. In the following, time for distance calculation is only considered, while the time for sorting is also ignored. Distance calculation necessary for selecting the standard category is M.

2. Search of the standard category

The nodes which are the descendants of node c are searched. The search is done depth-first, and for the same level nodes, the node whose distance from $\boldsymbol{x}$ is smaller is searched earlier. In other words, starting with a node at $L = 1$, in the case of $L < H - 1$, calculate the distances between $\boldsymbol{x}$ and all of its children of the node and find the node whose distance is smallest. The same procedure is done for the found node. In the case of $L = H - 1$, calculate the distances between $\boldsymbol{x}$ and all of its children that correspond to samples and find k nodes whose distances are small. If the number of children of selected node at $L = H - 1$ is smaller than k, the same procedure is done for the node whose distance is the second smallest.

For this search, distance calculations of all the children of the nodes at level L $(1 \leq L < H - 1)$ and all the children of the node at $L = H - 1$ are done. The number of distance calculation n_1 is,

$$n_1 = \sum_{L=1}^{H-1} \alpha_L. \tag{4}$$

Strictly speaking, if the number of children of the selected node at $L = H - 1$ is smaller than k, another search is necessary. However, since another search is necessary in few cases when k is small, the computational quantity for another search can be ignored.

3. Search of the category except for the standard category

Starting from node p $(1 \leq p \leq M, p \neq c)$, the nodes which are the descendants of node p are searched. The search is done depth-first, and for the nodes at the same level, the node whose distance from $\boldsymbol{x}$ is smaller is searched earlier. In this case, the nodes for search is restricted by applying Rule 1 and Rule 2. Concretely, for all the nodes of $L = 1$ except node c, the following procedure is done.

In the case of $L < H - 1$, distances between the unknown input $\boldsymbol{x}$ and all of the children of the node are calculated and sorted. Then for all the nodes which do not satisfy Rule 1, the same procedure is done. The node whose distance is smaller is processed earlier. In the case of $L = H - 1$, distance between the unknown input $\boldsymbol{x}$ and each of the child of the node which does not satisfy Rule 2 is calculated. It is compared with currently selected k distances, and if the calculated distance is smaller, it is changed with the largest distance among the current k distances.

Here, denote the average number of nodes of level L which do not satisfy Rule 1 as β_L. $\beta_L \leq \alpha_L$ is satisfied. The estimated number of distance calculation times n_2 for one category is,

$$n_2 = \alpha_1 + \beta_1(\alpha_2 + \beta_2(\cdots(\alpha_{H-3} + \beta_{H-3}(\alpha_{H-2} + \beta_{H-2}\alpha_{H-1}))\cdots)). \tag{5}$$

4. Second search of the standard category

When the process **3.** finishes, the categories that the selected k training samples belong to are checked. If more than $\lceil k/2 \rceil$ samples of the standard category c are included in these selected k samples, then the standard category c is the classification result. If there are less than $\lceil k/2 \rceil$ samples of the standard category c, the second search will begin at node c. Those nodes that are node c's children but have not been searched before are the targets of the second search. The method of the second search is the same as the method used in **3.** The category that most finally selected training samples belong to is the category that the unknown pattern belongs to.

If the standard category needs to be searched twice, the number of distance calculation times of the second search is the difference between the average number of one category described in **3.** and the number described in **2.** However, since the standard category is the category whose center of gravity is closest to the unknown pattern $\boldsymbol{x}$, the possibility that the standard category is the classification result is extremely high. Therefore, for most cases, there is no necessity to make the second search of the standard category. The calculation times spent on this part can be ignored.

3.3 Training algorithm for constructing search tree

According to the result of Section **3.2**, the estimated total calculation times denoted as n is

$$n = M + n_1 + (M-1)n_2. \tag{6}$$

Obviously, how to design the structure of search tree to make n small is a decisive factor. To design the structure of search tree, a clustering tree of each category as shown in Fig. 3(a) is constructed. As described above, the search tree is constructed by adding several thresholds to the clustering tree. In the following section, how to choose the appropriate number of thresholds and the appropriate values of thresholds are explained.

3.3.1 Construction of search tree with height three

By connecting leaves which are training samples to node p $(1 \leq p \leq M)$, an initial search tree with height two is constructed. By inserting another level of nodes under level one, the height of the search tree changes to three (See Fig. 4). In this tree, the children of the nodes at level two are training samples. In this case, according to Eqs. (4) $\sim$ (6), the estimated calculation times of $n^{(3)}$ is as follows:

$$n^{(3)} = M + n_1^{(3)} + (M-1)n_2^{(3)}, \tag{7}$$

where

$$n_1^{(3)} = \alpha_1^{(3)} + \alpha_2^{(3)}, \tag{8}$$

$$n_2^{(3)} = \alpha_1^{(3)} + \beta_1^{(3)}\alpha_2^{(3)}. \tag{9}$$

By giving a threshold θ_1 to the clustering tree constructed by the complete linkage method described in Section **3.1**, a search tree with height three can be constructed (see Fig. 4). The threshold θ_1 is chosen to make the value of $n^{(3)}$ to be minimum. As shown in Fig. 3(a), if the

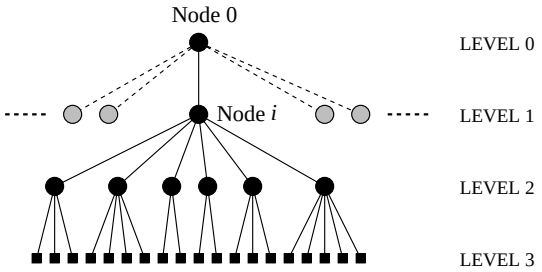


Figure 4: Tree with height three.

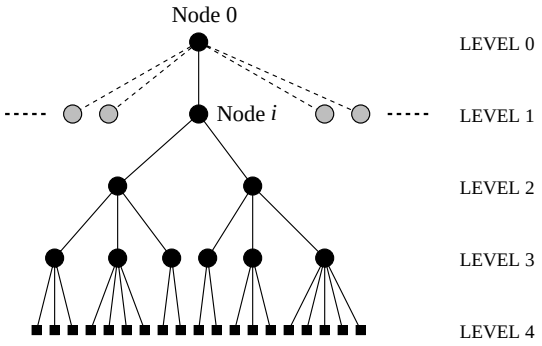


Figure 5: Tree with height four.

largest value among each category's $d_{\max}$ is $\hat{d}_{\max}$, θ_1 will be a value of $0 \leq \theta_1 \leq \hat{d}_{\max}$. Here, the interval of $[0, \hat{d}_{\max}]$ is divided into K equal parts, and the value of each part is

$$\theta_1 = \frac{\hat{d}_{\max}}{K} j, \quad 0 \leq j \leq K.$$

Using each of these values, $n^{(3)}$ is calculated, and the value that makes $n^{(3)}$ smallest is selected to be the threshold. Here, $K = 10$. Moreover, for each training sample, its k nearest neighbors are found. The number of nodes that do not satisfy Rule 1 are found, and the average value of the numbers is denoted as $\beta_1^{(3)}$.

3.3.2 Determination of the height of the search tree

As the search tree is heightened by inserting nodes level by level, reduction of calculation times is tried. If the calculation times is reduced when the search tree is heightened, then one more level is added. This procedure is repeated till the calculation times do not reduce anymore. In the meantime, the height of the search tree is decided. When nodes are inserted, they are inserted between level one and level two as shown in Fig. 4. Fig. 5 shows the tree with height four that is heightened from the tree shown in Fig. 4.

The estimated calculation times $n^{(4)}$ of the tree with height four shown in Fig. 5 is,

$$n^{(4)} = M + n_1^{(4)} + (M-1)n_2^{(4)}, \tag{10}$$

where

$$n_1^{(4)} = \alpha_1^{(4)} + \alpha_2^{(4)} + \alpha_3^{(4)}, \tag{11}$$

$$n_2^{(4)} = \alpha_1^{(4)} + \beta_1^{(4)}(\alpha_2^{(4)} + \beta_2^{(4)}\alpha_3^{(4)}). \tag{12}$$

In the above expressions, $\alpha_2^{(4)}$, $\beta_2^{(4)}$ and $\alpha_3^{(4)}$ correspond to $\alpha_1^{(3)}$, $\beta_1^{(3)}$ and $\alpha_2^{(3)}$ in Eqs. (8) and (9), respectively. In order to obtain the smallest value of $n^{(4)}$, two thresholds are necessary. For simplicity, θ_1 decided in the above section is used. Another threshold θ_2 is chosen as follows. If θ_1 is fixed, it means $\alpha_1^{(3)}$, $\beta_1^{(3)}$ and $\alpha_2^{(3)}$ are fixed. In other words, Eq. (11) and Eq. (12) can be rewritten as

$$n_1^{(4)} = \alpha_1^{(4)} + n_1^{(3)}, \tag{13}$$

$$n_2^{(4)} = \alpha_1^{(4)} + \beta_1^{(4)} n_2^{(3)}. \tag{14}$$

Here, $n_1^{(3)}$ and $n_2^{(3)}$ can be considered as constants. Threshold θ_2 is changed by the same way used for θ_1 in Section **3.3.1**, and $\alpha_1^{(4)}$ and $\beta_1^{(4)}$ are calculated. The domain interval is $[\theta_1, \hat{d}_{\max}]$. Therefore, the value that makes Eq. (10) smallest is selected. Moreover, if the result of Eq. (10) is smaller than the value of Eq. (7), the selected two thresholds are used to construct a tree with height four. Otherwise, the search tree is completed at height three.

This procedure is repeated until the total calculation times do not reduce any more. Therefore, the height of search tree H is decided.

3.3.3 Reestimation of the threshold

The search tree constructed by the method described in Section **3.3.1** and Section **3.3.2** is obtained by finding the minimum calculation times of the current level when the calculation times of the levels larger than the current level is fixed. Continually, the thresholds are estimated again to obtain the smallest calculation times in total.

As an example, a search tree with height four is explained. Because the total calculation times can be estimated according to Eq. (10), the thresholds of θ_1 and θ_2 are chosen to make the result of Eq. (10) to be minimum. For this purpose, the following two steps are carried out.

- $\alpha_1^{(4)}$ and $\beta_1^{(4)}$ in Eq. (13) and (14) are fixed. If the value of θ_1 varies, the values of $n_1^{(3)}$ and $n_2^{(3)}$ change. Consequently the values of $\alpha_1^{(3)}$, $\beta_1^{(3)}$ and $\alpha_2^{(3)}$ also change. Find the value of θ_1 that minimizes Eq. (10).

- $n_1^{(3)}$ and $n_2^{(3)}$ are fixed. If the value of θ_2 varies, the values of $\alpha_1^{(4)}$ and $\beta_1^{(4)}$ change. Find the value of θ_2 that minimizes Eq. (10).

The above processes are repeated till convergent or certain times of repetition[2]. Here, θ_1 with the range of $[0, \theta_2]$ and θ_2 with the range of $[\theta_1, \hat{d}_{\max}]$ are changed by the same way of changing θ_1 described in Section **3.3.1**. In the case of the search tree with height five, to minimize the calculation times of a certain level nodes can be done bottom up with the parameters of other levels are fixed.

4 Experiments

In order to confirm the effectiveness of the proposed method, experiments are carried out. The Directional Element Feature [14] whose effect has been shown by character recognition is used in the experiments. This is a 196-dimensional vector ($D = 196$). The feature vectors are

[2] In the experiments done in this paper, processes converge within four times in any cases.

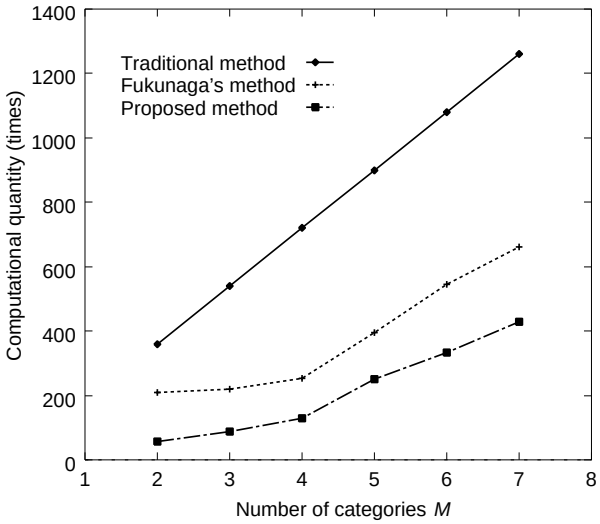


Figure 6: Results for nearest neighbor rule.

extracted from character images in the handwritten character database ETL9B [15]. Among the 200 data sets included in this database, 180 sets are used as training data, the other 20 sets are used to be test data. It means the number of training samples in each category is 180 ($N = 180$). Two kinds of experiments are done. One is done by using the nearest neighbor rule with changing the number of categories. Another is done by using the k-nearest neighbor rule with the number of categories is fixed to two. In these experiments, one category corresponds to one kind of character. In the experiments that uses M categories, the first M categories among the ETL9B are used.

4.1 Nearest neighbor rule

The experimental results of using the nearest neighbor rule with various numbers of categories are shown in Fig. 6. In this figure, besides the calculation times of the proposed method, the calculation times of the traditional method which computes the distances from unknown pattern to all of the training samples, and the calculation times of the method proposed by Fukunaga are also expressed. Because the number of training samples is 180, the calculation times of the traditional method is 180. As shown in the figure, for each value of M, the calculation times of Fukugana's method is about $1/2$ of that of the traditional method. Moreover, the calculation times of the method proposed in this paper is less than the result of Fukunaga's method in every case. Especially in the case that the number of categories is small, the effectiveness of our method is extremely large. For example, in the case of two categories, the computation time of our method is reduced to about one sixth of the traditional method and about one fourth of Fukunaga's method.

The structure of the constructed search tree is considered. For all the search tree constructed by the proposed method, the heights are four. It means the search trees are completed after two levels of nodes are inserted to the initial trees. In order to give the outline of the search trees, average number of nodes that have the same parent at the same level is shown in Table 1. The number of nodes at level one is M that is the number of categories. As shown in the table, compared with the numbers of nodes at level three and level four, the number of nodes at level

Table 1: The number of nodes of the constructed search tree (nearest neighbor rule).

Number of categories	2	3	4	5	6	7
Level 1	2.0	3.0	4.0	5.0	6.0	7.0
Level 2	10.0	11.3	10.0	11.8	11.7	12.3
Level 3	3.3	3.5	4.3	4.3	4.5	4.6
Level 4	5.5	4.6	4.2	3.6	3.4	3.2

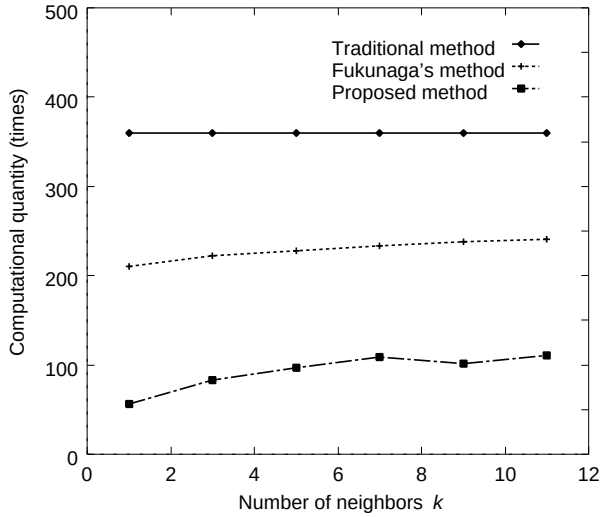


Figure 7: Results for k-nearest neighbor rule.

two is quite large. It means one category is divided into many clusters. As a result, the clusters considered unnecessary to be searched are found and dropped out earlier.

4.2 k-nearest neighbor rule

The experimental results of using k-nearest neighbor rule with various number k while the number of categories is two are shown in Fig. 7. In this figure, besides the calculation times of the proposed method, the calculation times of the traditional method, and the calculation times of the Fukunaga's method are also shown. As shown in the figure, the calculation times of Fukugana's method increases with k becoming larger. The experimental results have shown the calculation times of the method proposed in this paper is less than the results of Fukunaga's method in any cases. For example, in the case that $k = 11$, the computation time of our method is reduced to about one third of the traditional method and one second of Fukunaga's method.

The height of all the search trees constructed by the proposed method is four. In order to give the outline of the search trees, each average number of nodes of the same parent at the same level is shown in Table 2. The number of nodes at level one is equal to the number of categories that is two. As shown in the table, the numbers of nodes at level two are large in the cases of $k \leq 7$, while in the cases of $k \geq 9$, the numbers of nodes at level three are large. With k becomes larger, the distance between the unknown pattern and the k nearest neighbors consequently becomes larger. As a result, the number of nodes satisfied with Rule 1 at level two is small. For this case, it is effective to increase the number of nodes at level three.

Table 2: The number of nodes of the constructed search tree (k-nearest neighbor rule).

Number of neighbors k	1	3	5	7	9	11
Level 1	2.0	2.0	2.0	2.0	2.0	2.0
Level 2	10.0	17.0	17.0	17.0	4.0	4.0
Level 3	3.3	4.8	4.8	4.8	13.4	13.4
Level 4	5.5	2.2	2.2	2.2	3.4	3.4

By analyzing the structure of the search trees constructed in Section **4.1** and Section **4.2**, it is clarified that the efficiency of classification depends on the structure of the search tree. The method proposed by Fukunaga et al. has not emphasized the importance of the structure of search tree, however, it is an extremely important factor of fast algorithm.

5 Conclusion

In this paper, in order to realize a fast classification based on the nearest neighbor rule or the k-nearest neighbor rule, a new algorithm based on branch and bound method is proposed. Furthermore, an algorithm of constructing search tree is proposed. The proposed algorithm minimizes the total computational cost by estimating calculation times. Finally, the proposed method is applied to a classification problem using feature vectors extracted from character images. Compared with the traditional method, our method can reduce the computation time in any cases. Especially, in the case that the number of considered categories and the number of k are small, the effectiveness of our method is remarkable. For example, for the case of using the nearest neighbor rule ($k = 1$) and the number of categories is two, the proposed method is about four times faster than the traditional method.

To apply the proposed method in practical character recognition is a future work of this study.

References

[1] T.M. Cover and P.E. Hart, "Nearest neighbor pattern classification," IEEE Trans. Information Theory, vol.IT-13, no.1, pp.21–27, Jan. 1967.

[2] Y. Hamamoto, S. Uchimura, and S. Tomita, "A bootstrap technique for nearest neighbor classifier design," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.19, no.1, pp.73–79, Jan. 1997.

[3] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.18, no.6, pp.607–615, June 1996.

[4] W.P. Kegelmeyer Jr. and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.19, no.4, pp.405–410, April 1997.

[5] K. Fukunaga and D.M. Hummels, "Bayes error estimation using Parzen and k-NN procedures," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.PAMI-9, no.5, pp.634–643, Sept. 1987.

[6] L.J. Buturović, "Toward Bayes-optimal linear dimension reduction," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.16, no.4, pp.420–424, April 1994.

[7] K. Fukunaga and P.M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors," IEEE Trans. Computers, vol.C-24, no.7, pp.750–753, July 1975.

[8] A. Djouadi and E. Bouktache, "A fast algorithm for the nearest-neighbor classifier," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.19, no.3, pp.277–282, March 1997.

[9] P.E. Hart, "The condensed nearest neighbor rule," IEEE Trans. Information Theory, vol.IT-14, no.5, pp.515–516, May 1968.

[10] G.W. Gates, "The reduced nearest neighbor rule," IEEE Trans. Information Theory, vol.IT-18, no.5, pp.431–433, May 1972.

[11] G.L. Ritter, H.B. Woodruff, S.R. Lowry, and T.L. Isenhour, "An algorithm for a selective nearest neighbor decision rule," IEEE Trans. Information Theory, vol.IT-21, no.11, pp.665–669, Nov. 1975.

[12] C.L. Chang, "Finding prototypes for nearest neighbor classifiers," IEEE Trans. Computers, vol.C-23, no.11, pp.1179–1184, Nov. 1974.

[13] I.K. Sethi, "A fast algorithm for recognizing nearest neighbors," IEEE Trans. Systems, Man, and Cybernetics, vol.SMC-11, no.3, pp.245–248, March 1981.

[14] N. Sun, M. Abe, and Y. Nemoto, "A handwritten character recognition system by using Improved Directional Element Feature and subspace method," Trans. IEICE, vol.J78-D-II, no.6, pp.922–930, June 1995 (in Japanese).

[15] T. Saito, H. Yamada, and K. Yamamoto, "On the data base ETL9 of handprinted Characters in JIS Chinese characters and its analysis," Trans. IEICE, vol.J68-D, no.4, pp.757–764, April 1985 (in Japanese).

Authors

Shin'ichiro Omachi received his B.E., M.E. and Doctor of Engineering degrees in Information Engineering from Tohoku University, Japan, in 1988, 1990 and 1993, respectively. He has worked as a research associate at the Education Center for Information Processing at Tohoku University from 1993 to 1996. He is now an associate professor at Graduate School of Engineering, Tohoku University. His research interests include pattern recognition and parallel processing of images. Dr. Omachi is a member of the IEEE, the Institute of Electronics, Information and Communication Engineers, the Information Processing Society of Japan, and the Japanese Society of Artificial Intelligence.

Hirotomo Aso received his B.E., M.E., and Doctor of Engineering degrees in Electrical Engineering from Tohoku University, Japan, in 1968, 1970, and 1974, respectively. He was with the Department of Information Engineering, Tohoku University in 1973, and later joined the Faculty of Engineering, Nagoya University from 1979 to 1986. He is now a professor at Tohoku University. He is presently engaged in research of character pattern recognition, cellular automata, concurrent program schema, and parallel processing. He has received the Young Engineer Award of IEICEJ in 1978 and Achievement Award of IEICEJ in 1992. Dr. Aso is a member of the Institute of Electronics, Information and Communication Engineers, the Information Processing Society of Japan, the Japanese Society of Artificial Intelligence, Japanese Cognitive Science Society, IEEE, EATCS, and ACM.