Fast Design of Reduced-Complexity Nearest-Neighbor Classifiers Using Triangular Inequality

Eel-Wan Lee and Soo-Ik Chae

Abstract—In this paper, we propose a method of designing a reduced complexity nearest-neighbor (RCNN) classifier with near-minimal computational complexity from a given nearest-neighbor classifier that has high input dimensionality and a large number of class vectors. We applied our method to the classification problem of handwritten numerals in the NIST database. If the complexity of the RCNN classifier is normalized to that of the given classifier, the complexity of the derived classifier is 62 percent, 2 percent higher than that of the optimal classifier. This was found using the exhaustive search.

Index Terms—Nearest-neighbor classifier, triangular inequality, computational complexity, NIST database, fast design.

1 Introduction

THE nearest-neighbor (NN) classifier with representation vectors $\{c_i\}$, for a given input $\boldsymbol{x}$, calculates the distance from c_i to $\boldsymbol{x}$ for all i and produces a representation vector, $\boldsymbol{c}_{min}$ such that its distance to $\boldsymbol{x}$ is the smallest [1], [2]. This NN classifier with multiple representation vectors per class, is simple and powerful, but it suffers from huge computational complexity if the number of classes is large. Therefore, much work on reducing its computational complexity as much as possible has been reported [2].

One method of doing this is to reorganize the search procedure by preprocessing the representation vectors. This was first explored in the paper of Fisher and Patrick [3] and in many other related papers [2]. These fast search algorithms reduce the computational complexity without reducing the number of representation vectors so as not to increase the misclassification rate of the NN classifier. Another method that can be incorporated before applying the first one is to reduce the number of representation vectors so long as the increase in the misclassification rate remains acceptable. This reduction can be obtained with a proper selection or training procedure, called editing [2].

The effectiveness of the fast search algorithms using these two methods is relatively reduced if the input space dimension D is increased or the number of representation vector is reduced. It is reported in [4] that the expected number of distance calculation in NN classifier is $O(N^{1-1/D})$. Most fast search algorithms [2] have outlined the asymptotic behavior of their performances for relatively low input space dimension (D = 2 ~ 10) and a large number of representation vectors (N = 1,000 ~ 10,000).

In the VQ of images, D is typically 16 and N is reduced to 256, or 512 after optimization of the representation vectors. There has also been much work on reducing large computational complexity in the encoding procedure of the VQ [5]. In particular, several fast search algorithms using triangular inequality elimination (TIE) [6], [7], [8] were recently reported to reduce the computational complexity of encoding procedure further in comparison with the algorithms using PDE (Partial Distance Elimination), K-d tree and other inequalities especially in the image coding applications. In these fast search algorithms, inequality (1) is used as a necessary condition for the calculation of the distance $d(\boldsymbol{x}, \boldsymbol{c}_i)$, i.e., the calculation of the distance between the current representation vector, $\boldsymbol{c}_i$, and an input vector, $\boldsymbol{x}$, is performed only if the condition is satisfied.

$$|d(x, a_j) - d(c_i, a_j)| < d_{\min}^{i-1}. \tag{1}$$

Here, $d_{\min}^{i-1}$ is the current minimum distance, which is

$$\underset{j=1,\cdots,i-1}{MIN}\, d(x, c_j),$$

and the vector $\boldsymbol{a}_j$ is called an anchor vector. It is reported that three or four anchor vectors are enough in the VQ coding of images [6]. Consequently, most previous work on the fast encoding procedure for the image VQ did not focus on a systematic method of determining the number of anchor vectors by fully exploiting the information within the training data, such as the input space dimension, the number of representation vectors, and its variance. The classification procedure in the NN classifier is similar to the encoding procedure of the VQ. Therefore, we focus on the design problem of a classifier with reduced computational complexity after proper selection of representation vectors. We will denote these properly edited representation vectors as class vectors.

We propose a reduced-complexity nearest-neighbor (RCNN) classifier using the TIE, in which a parameter to be optimized is the number of anchor vectors. In the RCNN classifier with class vector $\{\boldsymbol{c}_i\}$, an anchor vector $\boldsymbol{a}_j$ is defined as a class vector such that $\boldsymbol{a}_j \in \{\boldsymbol{c}_j\}$ and its distance to the input is always calculated. Note that this definition is slightly different from that in [6] because the anchor vectors defined here are selected only from the class vectors. We also propose a fast algorithm of selecting an anchor vector set that minimizes the computational complexity of the RCNN classifier.

This paper is organized like the following: In Section 2, we introduce the RCNN classifier with a parameter M, the number of anchor vectors. We also propose a design procedure of the classifiers for each parameter M ∈ {1, ..., N} in Section 3. We then introduce the computational complexity curve that plots the computational complexity versus the parameter M in Section 4. With examples, we show that the optimal value of M that minimizes the computational complexity depends on the noise of the data as well as other properties of the data. We explain the fast design procedure of the near-optimal RCNN classifier in Section 5 and apply it to the classifier design for the handwritten numerals in the NIST data base HWDB-1 [9] in Section 6. Then we conclude the paper in Section 7.

2 Reduced Computational Complexity NN Classifier

For a given classifier with a class vector set $U = \{\boldsymbol{c}_i \mid i = 1, ..., N\}$, $A = \{\boldsymbol{a}_i \mid i = 1, ..., M\}$ and $B = \{\boldsymbol{b}_i \mid \boldsymbol{b}_i \in U, \boldsymbol{b}_i \notin A\}$ denote the anchor vector set and the nonanchor vector set of an RCNN classifier, respectively. The RCNN classifier outputs a class vector $\boldsymbol{c}_{min} \in U$ for an input $\boldsymbol{x}$ if its distance to the input is the smallest among the class vectors. We will call $\boldsymbol{c}_{min}$ the NN (nearest neighbor) vector. The classifier first calculates the distance $d(\boldsymbol{x}, \boldsymbol{a}_i)$ to each anchor vector $\boldsymbol{a}_i \in A$. The distance to a nonanchor vector $\boldsymbol{b}_i \in B$ is calculated only if its greatest lower bound defined in (2A) is less than the current minimum of the input distances to the class vectors that have been calculated so far.

$$d_{LOW}^{A}(\boldsymbol{x}, \boldsymbol{b}_i) = \underset{\boldsymbol{a}_j \in A}{MAX} |d(\boldsymbol{x}, \boldsymbol{a}_j) - d(\boldsymbol{b}_i, \boldsymbol{a}_j)|, \tag{2A}$$

• *The authors are with the School of Electrical Engineering, Seoul National University, San 56-1, Shinlim-dong, Kwanak-gu, 151-742 Seoul, Korea. E-mail: {wan; chae}@belle.snu.ac.kr.*

Manuscript received 15 Oct. 1996; revised 5 Mar. 1998. Recommended for acceptance by I. Sethi.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106544.

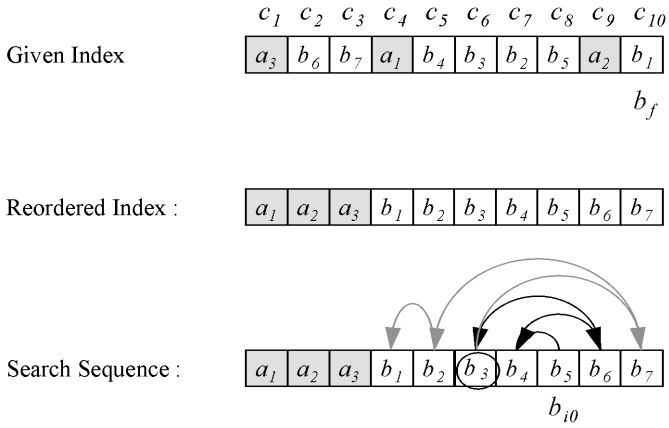


Fig. 1. Search phases in the reordered index for N = 10. In this example, the winner vector is $\boldsymbol{b}_3$ and the initial nonanchor search vector is $\boldsymbol{b}_5$. The search sequence of phase II is represented with solid lines while that of phase III is represented with dashed lines.

$$d^A_{LOW}(\boldsymbol{x}, \boldsymbol{b}_i) < d_{\min} \tag{2B}$$

Note that if $d(\boldsymbol{x}, \boldsymbol{b}_k) = \underset{\boldsymbol{c}_i \in U}{MIN}\, d(\boldsymbol{x}, \boldsymbol{c}_i)$,

$$d^A_{LOW}(\boldsymbol{x}, \boldsymbol{b}_k) \le d(\boldsymbol{x}, \boldsymbol{b}_k) < d(\boldsymbol{x}, \boldsymbol{b}_i) \text{ for all } \boldsymbol{b}_i \text{ in B}.$$

Therefore, the distance $d(\boldsymbol{x}, \boldsymbol{b}_k)$ is always calculated because its greatest lower bound is always less than the current minimum distance. Consequently, the performance of the RCNN classifier using inequality (2B) is equal to that of the given NN classifier. We need (N − M) • M memory locations to store the precalculated distance between each pair $(\boldsymbol{b}_i, \boldsymbol{a}_j)$ for $\boldsymbol{b}_i \in$ B and $\boldsymbol{a}_j \in$ A. To get the minimum distance as early as possible, in many fast search algorithms the search list is rearranged by using a proper distance estimate such as the distance from the origin [6], [7], [8]. Similarly, we rearrange the nonanchor vectors in the ascending order of their distances to an arbitrarily chosen vector $\boldsymbol{b}_f$ so that this ordering can be used in the zig-zag scan as shown in Fig. 1 [10].

After calculating the distances to M anchor classes, we find a minimum among them. Then, we calculate the greatest lower bound of all the input distances to each nonanchor vector, and determine a nonanchor vector $\boldsymbol{b}_{i0}$ whose greatest lower bound is the minimum. Starting from the vector $\boldsymbol{b}_{i0}$, we test the condition (2B) for nonanchor vectors in the zig-zag order as shown in Fig. 1. The flowchart of the search procedure of the RCNN classifier is illustrated in Fig. 2.

The distance calculations in the RCNN classifier can be decomposed into three phases. The distance calculations for M anchor vectors are in phase I. If the NN vector is one of nonanchor vectors, the distance calculations for the nonanchor vectors until the NN vector, are in phase II. If the NN vector is one of the anchor vectors, no calculation is undertaken in phase II. All the distance calculations undertaken after finding the NN vector, are in phase III. In Fig. 1, for example, the distance calculations for $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3\}$ are in phase I, those for $\{\boldsymbol{b}_3, \boldsymbol{b}_4, \boldsymbol{b}_5, \boldsymbol{b}_6\}$ are in phase II, and those for $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_7\}$ are in phase III. Each phase is different from the others in many respects, especially in the variation of the computational complexity which depends on the noise in the data. The number of distance calculations in phase I is constant and equal to M. The number of distance calculations in phase III depends only on the noise in the data. The number of distance calculations in phase II depends on both the tightness of the greatest lower bounds and the noise in the data. Note that the distance from the NN vector to the input is not zero because of the noise in data. Therefore, the computational complexity is dependent on the noise in data. The tightness of the greatest lower bound, which affects the computational complexity in phase II, is a function of the parameter M, as well as of the noise in the data.

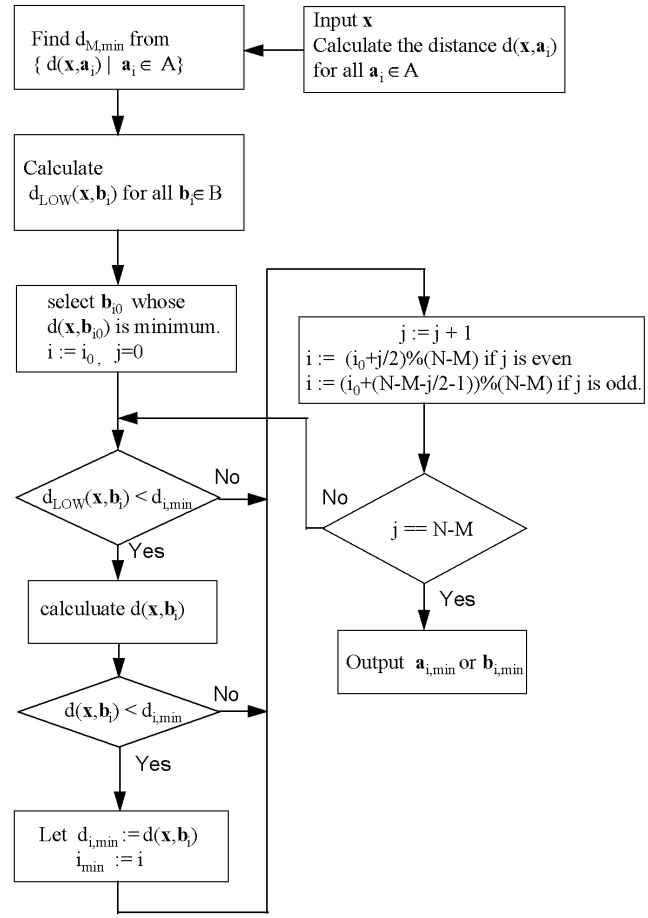


Fig. 2. Search procedure of the RCNN classifier using TIE.

3 Selection of the Anchor Vectors

We should select an anchor vector set that minimizes the computational complexity of the RCNN classifier from ${}_N C_M$ candidates for a given number M by exploiting the mutual distance relations among the class vectors. First, we construct a sequence of anchor vector sets A_i for i = 0, ..., N starting with the empty set $A_0 = \phi$ by applying a greedy algorithm. Similarly, we construct a sequence of nonanchor vector set B_i for i = 0, ..., N. A discriminative measure, $P_e(\,)$, is required to determine which class vector in B_i outperforms all the other nonanchor vectors in distinguishing all the remaining nonanchor vectors. We select a class vector $\boldsymbol{a}_i \in B_i$ as an anchor vector and add it to A_i to form A_{i+1} if

$$a_i = \arg\max_{r_k \in B_i} P_e(r_k) \tag{3}$$

To define the discriminative measure, we define the nearest vector $\boldsymbol{n}_j$ of $\boldsymbol{b}_j$, for each $\boldsymbol{b}_j \in B_i$, such that $\boldsymbol{n}_j \in B_i - \{\boldsymbol{b}_j\}$, $\boldsymbol{n}_j \neq \boldsymbol{b}_j$ and $d(\boldsymbol{n}_j, \boldsymbol{b}_j) \le d(\boldsymbol{b}_l, \boldsymbol{b}_j)$ for all $\boldsymbol{b}_l \in B_i - \{\boldsymbol{b}_j\}$. To simplify the error model, we neglect the probability of class $\boldsymbol{b}_j$ being misclassified to classes which are not $\boldsymbol{n}_j$ because their distances to the input are larger than that of the class $\boldsymbol{n}_j$. Therefore, the discriminative measure increased after the addition of $\boldsymbol{r}_k \in B_i$ for the construction of A_{i+1} from A_i is defined as follows:

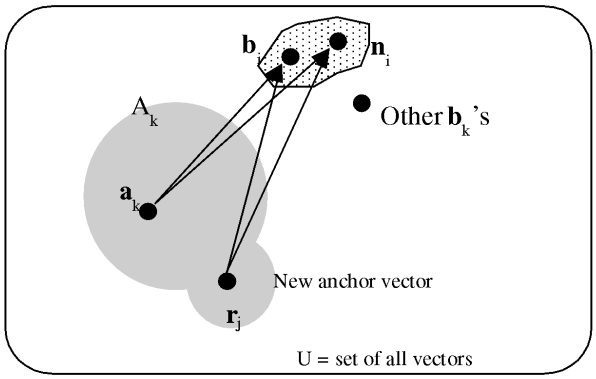


Fig. 3. Computation of the proposed discriminative measure.

$$P_e(\boldsymbol{r}_k) = \sum_{\boldsymbol{b}_j \in B_i - \{\boldsymbol{r}_k\}} \left\{ d_{LOW}^{A_i \cup \{\boldsymbol{r}_k\}}(\boldsymbol{b}_j, \boldsymbol{n}_j) - d_{LOW}^{A_i}(\boldsymbol{b}_j, \boldsymbol{n}_j) \right\}. \quad (4)$$

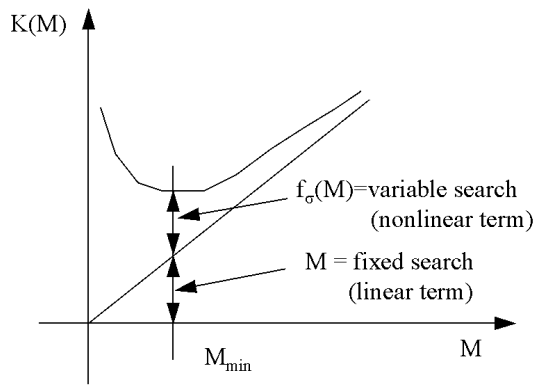
Each term in the summation corresponds to the nonnegative change of the greatest lower bound for a nonanchor vector $\boldsymbol{b}_j$ resulting from the addition of an anchor vector, $\boldsymbol{r}_k$. By applying the proposed algorithm to each M ∈ {1, ..., N} sequentially, we obtain a sequence of the class vectors $(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N)$ before determining the optimal number M_{min} of anchor vectors that minimizes the computational complexity of the RCNN classifier. A set that contains first M elements of the sequence is defined as the anchor vector set of the RCNN classifier with a specified M.

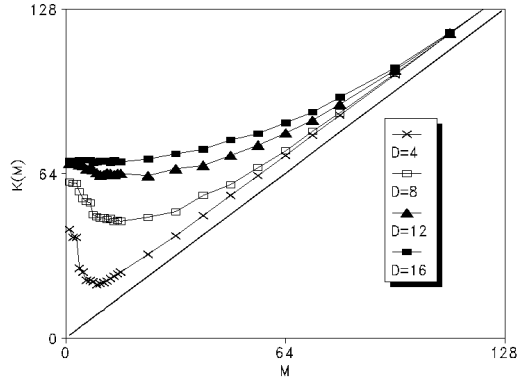
4 Examples of the Computational Complexity Curve

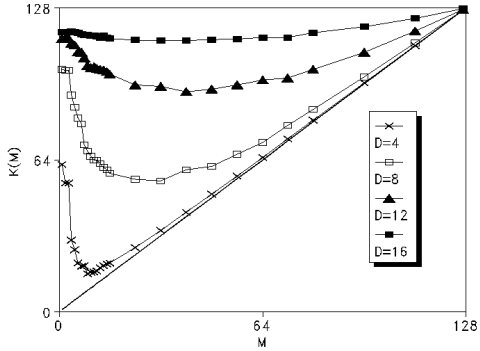
We obtained the computational complexity curves of the proposed algorithm for the problems with real-numbered input spaces with dimension D ∈ {4, 8, 12, 16} and 128 class vectors, which are plotted in Fig. 4. Here, the x-axis represents the number of anchor vectors and the y-axis represents the expected search time K(M). The unit of expected search time is the time required for the distance calculation per class vector. This represents the computational complexity of the RCNN classifier with a real-valued number between zero and N. The 128 class vectors and 1,000 test vectors were selected randomly and uniformly in the R^D space. As D increases, the effectiveness of the fast search algorithm decreases rapidly due to the "curse of dimensionality" as shown in Fig. 4b. Selecting data with uniform distribution results in very noisy data in the classification problem. If we select the test vectors near the class vectors, thus reducing the noise in the data, we obtain a different computational complexity curve, as shown in Fig. 4c, where the expected search time is much smaller than that for the noisy data. This is mainly because the number of distance calculations in phase III is reduced. Therefore, the expected search time, K(M), of the classifier can be decomposed into two terms: a linear term and a nonlinear term:

$$K(M) = M + f_\sigma(M). \quad (5)$$

The linear term corresponds to the number of distance calculations in phase I, which is constant for all input vectors. The nonlinear term corresponds to the number of distance calculations in phases II and III, which depends on the properties of the input







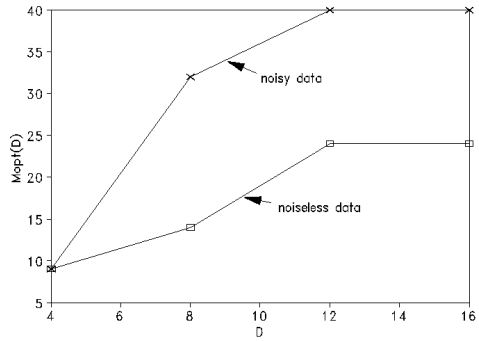


Fig. 4. Computational complexity curves of the RCNN classifiers. (a) Top left: Two components of the searches. (b) Top right: Computational complexity curve for noiseless data. (c) Bottom left: Computational complexity curve for noisy data. (d) Bottom right: M_{min} as a function D and noise.

vector. We do not require any detailed knowledge about the nonlinear component $f_\sigma(M)$. It is conjectured, however, to be nondecreasing function of M. Note that the number of anchor vectors, M_{min}, that minimizes the search time is located near the M value of the intersection point of the two curves. As shown in Fig. 4d, M_{min} decreases if the noise in the data is reduced since the nonlinear term $f_\sigma(M)$ that depends on the noise in input data is reduced.

5 Optimization of the RCNN Classifier

It takes a large amount of computation to obtain the computational complexity of the RCNN classifier with many input data for each M. The time involved in obtaining a computational complexity curve is proportional to the number of complexity computation. Fortunately, we do not have to obtain the locations of all the points in the computational complexity curve if we have a priori knowledge on the general shape of the computational complexity curve in Fig. 4a. By reducing the number of complexity computations, we can reduce the classifier design time substantially, although we cannot reduce the computation time of K(M) much for each M.

First, we note that the time in obtaining the computational complexity curve for the noiseless data is much shorter than that for the noisy data because the input data is limited only to class vectors in the noiseless case. Therefore, we use the optimal number of anchor vectors for noiseless data as the initial guess M_{init}, which is smaller than the optimal number of anchor vectors for noisy data. We denote an ith estimate of M_{min} as M(i) while $M(0) = M_{init}$. We exploit the fact that the optimal point is near the intersection point of the linear and nonlinear terms. If Δ, which is $f_\sigma(M) - M$, is positive, the optimal M is larger than the current estimate M(i). Therefore, we update the current estimate with (6).

$$M(i+1) = M(i) + \alpha \cdot (K(i) - 2 \cdot M(i)). \tag{6}$$

If Δ is negative, the next estimate will be decreased. This procedure continues until the difference Δ becomes too small to move by the step size of 1.0, as shown in Fig. 5. The number of the updating steps should be kept as small as possible because it requires much time to find the expected search time K(M(i)) for M(i) in each updating step. We selected the update parameter α of 0.5, which results in a simple reestimation formula (7) to force the ratio of M(i) / K(i) to 0.5.

$$M(i+1) = 0.5 \cdot K(i). \tag{7}$$

Although we assumed that the optimal M lies near the intersection point of two curves of M and f_σ(M), the optimal point is not located exactly at the intersection point. Consequently, a correction to the update rule (7) is required. From the simulation results in Section 4, we know that the nonlinear component is a monotonically nonincreasing function and that it becomes flat as the noise in the data, σ, or the input space dimension, D, increases. Therefore, we assume that $f_\sigma(M) = N \cdot \exp(-a \cdot M)$, where a should be selected according to the flatness of the nonlinear component $f_\sigma(M)$.

With the simulations, we confirmed that M_{min} lies to the left of the M_{cross} in most nontrivial cases, which corresponds to the condition of $0 < a \cdot M_{cross} < 1$. Therefore, the update rule needs to be modified to (8) because the linear term M is $(50 - \delta)$ percent of the total search time when it is minimum. We used $\delta = 10$ percent in the experiment described in the next section. Note that δ should be selected carefully, based on the noise in the data.

$$\begin{aligned} M(i+1) &= \tfrac{50-\delta}{100} \cdot K(i) \\ &= 0.4 \cdot K(i) \quad \text{for } \delta = 10 \text{ percent} \end{aligned} \tag{8}$$

6 Experiments

We applied the optimization procedure in Section 5 to design an RCNN classifier for the handwritten numerals in the NIST-database HWDB-1 [9]. We segmented the numerals in **box3** through **box5** with connected component analysis [11]. All the numerals were normalized to 24×24 as shown in Fig. 6. The classifier was trained until 100.0 percent classification rate for 1,000 training vectors was achieved. The classifier has eight class vectors for each numerals, summing to 80 class vectors for 10 numerals. This conventional NN classifier achieved 95.2 percent classification rate for 1,000 test vectors.

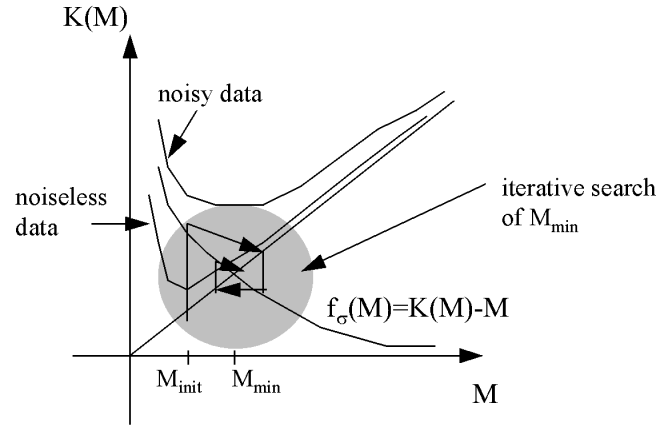


Fig. 5. Sequence of the optimal M's estimate.

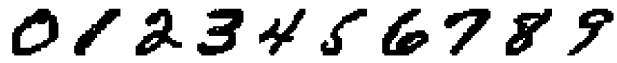
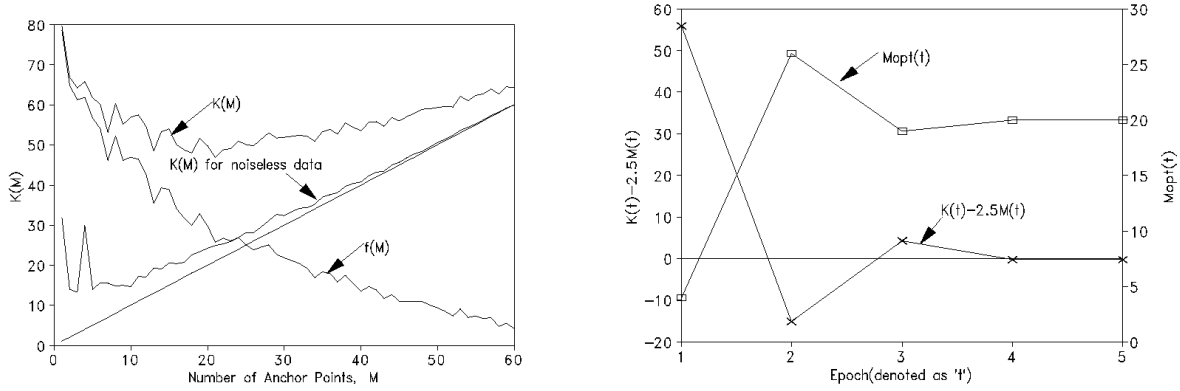


Fig. 6. Numerals extracted from the box4 of hsf_0/f0024_4 in the NIST database.

Although the graph in Fig. 7a shows local minima unlike those in Fig. 4, we obtained a solution of good quality with the proposed design algorithm. This algorithm was finished in five steps, with the sequence M(i), as shown in Fig. 7b, finding an RCNN classifier with $M_{min} = 20$ and $K_{min} = 48.7$ while $M_{opt} = 21$, $K_{opt} = 46.8$ for the optimal classifiers obtained, using exhaustive search for all possible values of M. With the proposed fast design of the classifier, the design time is decreased by 93 percent while the search time of the classifier is increased by about 2 percent compared to the optimal classifier. The search time and design time of the RCNN classifier compared to the conventional NN classifier are listed in Table 1. The RCNN classifier obtained with the proposed fast design has 38 percent reduction in the search time compared with the full search NN classifier. In contrast, although the classifier obtained with Li's algorithm of three anchor vectors shows a very good result in image VQ coding applications [5], it has only 20 percent reduction in the search time for this classification problem, which corresponds to the RCNN classifier with three anchor vectors. This result supports the necessity of wide-range search for anchor vectors, especially in the classification problem. Note that the number of anchor vectors required in the classification problem, especially with high

TABLE 1
Search Time and Design Time in the RCNN Classifier

	Exhaustive design	Proposed design
No. of distance calculation (relative to full search %)	46.8 (60%)	48.7 (62%)
No. of M values checked in classifier design	80	5
The number of anchor vectors (M)	21	20



(a) (b)

Fig. 7. Computational complexity curves for the NIST database and design trace. (a) Computational complexity curve for N = 80. (b) Design trace of the classifier.

input dimension and properly edited class vectors, is much larger than that in the image VQ coding problem in order to obtain substantial reduction in classification time.

7 Conclusion

For a given NN classifier with high-input space dimension and a large number of class vectors, we addressed the problem of the fast design of an RCNN classifier of near-minimal computational complexity with minimal efforts. We explained an algorithm of the RCNN classifier that exploits the triangular inequality and the concept of anchor vectors that are limited to the class vectors. We employed a new discriminative measure and a greedy algorithm in determining the sequence of anchor vectors which can be used to select an anchor vector set for a given number of anchor vectors. Based on the shape of the computational complexity curve obtained by experiment, we introduced an algorithm for determining a near-minimal RCNN classifier efficiently by recursively locating the intersection points of the linear and nonlinear terms instead of directly finding the minimum value. We applied the proposed method to the classification problem of handwritten numerals in the NIST database and obtained an RCNN classifier with the search time of 62 percent compared with that of the given full-search NN classifier. This is quite close to the 60 percent search time of the optimal RCNN classifier obtained by exhaustive search. The results showed the effectiveness of the proposed method and its potential for the fast design of the RCNN classifiers. Although we circumvented the problem of computing the computational complexity for all M with simplification of K(M), further studies on its analytic form are required to tackle a more challenging NN classifier design problem.

Acknowledgments

The authors are grateful to the reviewers for their helpful comments about this paper. This research was supported in part by the Institute of Information Technology Assessment (IITA), Korea, under research grant 96060-IT2-I2.

References

[1] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley, 1973.
[2] B.V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* Los Alamitos, Calif.: IEEE CS Press, 1991.
[3] F.P. Fisher and E.A. Patrick, "A Preprocessing Algorithm for Nearest Neighbor Decision Rules," *Proc. Nat'l Electronic Conf.*, vol. 26, pp. 481-485, Dec. 1970.
[4] J.H. Friedman, F. Baskett, and L.J. Shustek, "An Algorithm for Finding Nearest Neighbors," *IEEE Trans. Computers*, vol. 24, pp. 1,000-1,006, Oct. 1975.
[5] A. Gersho and R.M. Gray, *Vector Quantizer and Signal Compression*. Boston: Kluwer Academic Publisher, 1992.
[6] W. Li and E. Salari, "A Fast Vector Quantization Encoding Method for Image Compression," *IEEE Trans. CAS for Video Tech.*, vol. 5, no. 2, pp. 119-123, Apr. 1995.
[7] G. Poggi, "Fast Algorithm for Full-Search VQ Encoding," *Electron. Lett.*, vol. 29, pp. 1,141-1,142, June 1993.
[8] C.-M. Huang, Q. Bi, G.S. Stiles, and R.W. Harris, "Fast Full Search Equivalent Encoding Algorithm for Image Compression Using Vector Quantization," *IEEE Trans. Image Processing*, vol. 1, no. 3, pp. 413-416, July 1992.
[9] "NIST Special Database 1(HWDB)," Standard Reference Data, National Institute of Science and Technology, 221/A323, Gaithersburg, Md.
[10] S.-W. Ra and J.-K. Kim, "A Fast Mean-Distance-Oriented Partial Codebook Search Algorithm for Image Vector Quantization," *IEEE Trans. CAS*, vol. 40, no. 9, pp. 576-579, Sept. 1993.
[11] W.K. Pratt, *Digital Image Processing*, 2nd ed. New York: John Wiley, 1991.