



Fig. 3. $P_{N,sym}^*$ versus N .

observations). Direct calculation of $P_N^*(\infty)$ was possible for small values of N . For larger values of N it became necessary to upper-bound $P_N^*(\infty)$. A Chernoff bound on $P_N^*(\infty)$ is

$$P_N^*(\infty) \leq (2\sqrt{pq})^N/2.$$

Fig. 3 shows $P_{N,sym}^*(m)$ for various numbers of observations N and various sizes of memory m . The Chernoff bound on $P_N^*(\infty)$ was used for $N \geq 32$.

Quite naturally one does better with more memory. The $P_{N,sym}^*(m)$ curve for any given value of m follows the $P_N^*(\infty)$ line for low values of N , diverges from it for larger values of N , and approaches a nonzero limit $P_\infty^*(m)$ as $N \rightarrow \infty$. This behavior is easily explained. Any given machine can "remember" all of the observations for low values of N . Here infinite memory offers no advantages. For larger values of N , a finite-state machine necessarily loses some information and thus does not do so well as one with infinite memory. As $N \rightarrow \infty$, $P_{N,sym}^*(m)$ approaches $P_\infty^*(m)$, the infinite-time lower bound on the probability of error, since from [1] we know that for $N = \infty$ the optimal machine is symmetric.

CONCLUSIONS

Experimentally it has been found to be true that the optimal machine has the conjectured form except for the symmetry requirement. Transitions are only between adjacent states. Randomization is found only in transitions to states with a higher probability of error. As $N \rightarrow \infty$, the probability of a transition out of an end state tends monotonically to 0. It was found that P_N^* and $p_{12}(H)$ approach their respective limits (P_∞^* and 0) as $(\ln N)/N$. A more fundamental argument was given for this behavior.

It was felt that further experiments would only tend to confirm the findings presented. It would be more interesting at this point to attempt mathematical proofs of behavior, specifically of the conjectures. It is hoped that these experimental results will aid in this endeavor. Investigation of deterministic machines for continuous-probability distributions is also of interest and is being pursued [3].

REFERENCES

- [1] M. Hellman and T. M. Cover, "Learning with finite memory," *Ann. Math. Statist.*, vol. 41, pp. 765-782, 1970.
- [2] M. E. Hellman, "Learning with finite memory," Ph.D. dissertation, Stanford Univ., Stanford, Calif., Mar. 1969.
- [3] M. A. Freedman, "A finite memory, finite time, Gaussian hypothesis testing problem," M.S. thesis, Dep. Elec. Eng., Massachusetts Inst. Technol., Cambridge, Sept. 1971.

The Reduced Nearest Neighbor Rule

GEOFFREY W. GATES

Abstract—A further modification to Cover and Hart's nearest neighbor decision rule, the reduced nearest neighbor rule, is introduced. Experimental results demonstrate its accuracy and efficiency.

The nearest neighbor rule was originally proposed by Cover and Hart [1], [2] and is currently being used by several workers. One reason for the use of this rule is its conceptual simplicity, which leads to straightforward, if not necessarily the most efficient, programming. In a subsequent paper, Hart [5] suggested a means of decreasing memory and computation requirements. This paper introduces a technique, the reduced nearest neighbor rule, that can lead to even further savings. The results of this new rule are demonstrated by applying it to the "Iris" data [6].

The nearest neighbor rule (NN) is described in several places in the literature [1], [2]. For background a simple statement will be included here. First, some notation must be defined. Assume there are M pattern classes, numbered $1, 2, \dots, M$. Let each pattern be defined in an N -dimensional feature space and let there be K training patterns. Each training pattern is a pair (x^i, θ_i) , $1 \leq i \leq K$, where $\theta_i \in \{1, 2, \dots, M\}$ denotes the correct pattern class and

$$x^i = (x_1^i, x_2^i, \dots, x_N^i)$$

is the set of feature values for the pattern. Let $T_{NN} = \{(x^1, \theta_1), (x^2, \theta_2), \dots, (x^K, \theta_K)\}$ be the nearest neighbor training set. Given an unknown pattern x , the decision rule is to decide x is in class θ_j if

$$d(x, x^i) \leq d(x, x^j), \quad 1 \leq i \leq K,$$

where $d(\cdot, \cdot)$ is some N -dimensional distance metric.

Actually, the preceding rule is more properly called the 1-NN rule, since it uses only one nearest neighbor. An obvious generalization of this is the k -NN rule, which takes the k nearest patterns i_1, i_2, \dots, i_k and decides upon the pattern class that appears most frequently in the set $\theta_{i_1}, \theta_{i_2}, \dots, \theta_{i_k}$.

Hart [5] describes a revised rule called the condensed nearest neighbor rule (CNN). Actually, this is not a new decision rule since it still chooses the class of the nearest neighbor. Rather, the word condensed refers to a procedure for choosing a subset of T_{NN} , which we call T_{CNN} , that should perform almost as well as T_{NN} in classifying unknown patterns. In this case, a possible drop in performance is being traded for greater efficiency, both in the amount of memory required to store the training set and in the computation time required to reach a decision. Simulation can be used to decide whether the increased efficiency is worth the degradation in performance.

In the formation of T_{CNN} , the notion of a consistent subset of T_{NN} is important. It is simply a subset of T_{NN} that will classify all the patterns in T_{NN} correctly. Then the minimal consistent subset is the smallest and therefore the most efficient subset of T_{NN} that will properly classify every pattern in T_{NN} . The minimal consistent subset of T_{NN} is important since, in some sense, it has generalized all the important information out of T_{NN} . It will turn out that T_{CNN} must be consistent, but cannot be guaranteed to be minimal.

The algorithm for constructing T_{CNN} proceeds as follows.

- 1) The first sample pattern is copied from T_{NN} to T_{CNN} .
- 2) T_{CNN} is used as the training set to classify each pattern of T_{NN} , starting with the first. This is done until one of the following two cases arises:

Manuscript received June 18, 1971; revised September 28, 1971.
The author is with the Department of Computer Science, Michigan State University, East Lansing, Mich. 48823.

TABLE I
RESULTS OF IRIS DATA SIMULATIONS RUN ON THE CDC3600 AT
MICHIGAN STATE UNIVERSITY

Experi- ment	Data Set	Eigen- vector Trans- forma- tion	Com- pressed	<i>k</i>	NN Error Prob- ability	CNN Error Prob- ability	RNN Error Prob- ability
1	1			1	120 0.0	20 0.0	15 0.0
2	1	X		1	120 0.067	31 0.067	28 0.067
3	1	X	X	1	120 0.067	22 0.033	15 0.033
4	1			3	120 0.0	40* 0.0*	30* 0.0*
5	1	X		3	120 0.100	62* 0.067*	56* 0.067*
6	1	X	X	3	120 0.033	44* 0.033*	30* 0.033*
7	2			1	132 0.667	7 0.611	4 0.611
8	2	X		1	132 0.500	24 0.556	20 0.389
9	2	X	X	1	132 0.722	11 0.889	8 0.889
10	2			3	132 0.778	13* 0.444*	8* 0.444*
11	2	X		3	132 0.333	48* 0.556*	40* 0.389*
12	2	X	X	3	132 0.833	22* 0.889*	16* 0.889*
13	3			1	18 0.833	5 0.811	3 0.833
14	3	X		1	18 0.667	6 0.636	4 0.598

* Marks the *k*-CNN and *k*-RNN rules with *k* ≠ 1.

a) every pattern in T_{NN} is correctly classified, in which case, the process terminates;

b) one of the patterns in T_{NN} is classified incorrectly, in which case, go to 3).

3) Add the pattern from T_{NN} that was incorrectly classified to T_{CNN} . Go to 2).

There are two things that must be noted about this algorithm. First, the number of nearest neighbors to be considered *k* is not mentioned. Hart [5] mentions *k* ≠ 1 as being a possibility for future research. However, this suggestion contains a fundamental flaw. Take *k* to be $2n + 1$. To guarantee a unique solution, at least $n + 1$ of the nearest neighbors must be from the correct pattern class in order for the new pattern to be correctly classified. In forming T_{CNN} , an incorrectly classified pattern from T_{NN} is added to T_{CNN} . In particular, for the worst case, it would have to be added to T_{CNN} $n + 1$ times. While this does not contradict any assumptions that have been made, it is undesirable in the sense that the *k* ≠ 1 condensed nearest neighbor rule will always make the same decision as the 1-CNN rule. A question that this discussion prompts is, "How many times does this worst case occur?" Experimental evidence will be presented that shows that in 11 out of 12 cases tried, the worst case applied.

An alternative would be to use *k* = 1 to construct T_{CNN} and then use $k = 2n + 1$ for actual classification of unknown patterns. This method would avoid the problem of ending up with $n + 1$ copies of each training pattern. The results of this procedure are not easy to visualize and would bear further study.

The second item of importance is that the algorithm will terminate for *k* = 1 in all but one case. If T_{NN} is already a minimal consistent set, then T_{CNN} will equal T_{NN} . If this happens, the algorithm will not terminate only if there are two or more patterns (x^i, θ_i) and (x^j, θ_j) for which $i \neq j$, $x^i = x^j$, but $\theta_i \neq \theta_j$. Although this case is not specifically disallowed, it violates the consistency of T_{CNN} . That is, T_{CNN} must classify all its member patterns correctly.

One can ask whether this procedure is effective or whether it really reduces the size of the training set or whether T_{CNN} contains a minimal consistent subset. The answers to these questions come from simulations. In the simulation performed using well-separated data, savings in storage of as great as 90 percent were realized. While this is encouraging, it should be noted that the next rule to be presented, the RNN rule, improved upon CNN in every case tried. This means that T_{CNN} is not minimal, though how far from minimal it actually is would be difficult to determine.

The last rule to be considered is one that has not appeared in the literature, the reduced nearest neighbor rule RNN. It is an extension of the CNN rule and, like the CNN rule, reduces T_{NN} . Since it is based on the CNN rule, *k* ≠ 1 will not be considered in this paper. The

algorithm to produce T_{RNN} , the reduced nearest neighbor training set, is 1) copy T_{CNN} into T_{RNN} ; 2) remove the first pattern from T_{RNN} ; 3) use T_{RNN} to classify all the patterns in T_{NN} :

- a) if all patterns are classified correctly, go to 4),
- b) if a pattern is classified incorrectly, return the pattern that was removed and go to 4);

4) if every pattern in T_{RNN} has been removed once (and possibly replaced) then halt. Otherwise, remove the next pattern and go to 3).

One question of some interest is whether or not T_{RNN} is a minimal consistent subset of T_{NN} . Obviously, since T_{RNN} is a subset of T_{CNN} , then if T_{CNN} does not contain the minimal consistent subset of T_{NN} , neither does T_{RNN} . An example of a case where T_{CNN} does not contain a minimal subset for $M = 2$ pattern classes, $N = 1$ features, and $K = 13$ training samples is

$$T_{NN} = \{(-13,1), (-16,1), (-19,1), (-6,2), (-4,2), (-2,2), (0,2), (2,2), (4,2), (6,2), (13,1), (16,1), (19,1)\}$$

$$T_{CNN} = \{(-13,1), (-6,2), (13,1), (4,2)\}$$

$$T_{RNN} = T_{CNN}$$

$$T_{min} = \text{minimal consistent subset} \\ = \{(-13,1), (0,2), (13,1)\}.$$

But what happens in the case where T_{CNN} does contain a minimal consistent subset? The following example demonstrates just such a case:

$$T_{NN} = \{(0,2), (-3,1), (-2,2), (3,1), (2,2)\}$$

$$T_{CNN} = T_{NN}$$

$$T_{RNN} = \{(-3,1), (-2,2), (3,1), (2,2)\}$$

$$T_{min} = T_{RNN}.$$

If this were always true when $T_{min} \subseteq T_{CNN}$, the RNN rule would be a very useful tool. It can be shown that this is indeed the case, as follows. Let (x^i, θ_i) be any pattern not in T_{min} , and let (x^m, θ_m) be that pattern in T_{min} that causes (x^i, θ_i) to be correctly classified. The case that must be considered is when both (x^i, θ_i) and (x^m, θ_m) are in T_{CNN} . By the construction of T_{CNN} , (x^i, θ_i) must appear before (x^m, θ_m) , since if (x^m, θ_m) appears first, (x^i, θ_i) would not have been misclassified and therefore would not have been added to T_{CNN} . Notice that if there are several elements of T_{min} that could properly classify (x^i, θ_i) , they must all follow (x^i, θ_i) in order for this pattern to be included in T_{CNN} .

Now applying RNN to T_{CNN} , observe what happens to (x^i, θ_i) when it is removed. Since $T_{min} \subseteq T_{CNN} - \{(x^i, \theta_i)\}$ all patterns will be correctly classified and the extraneous pattern will be removed. But what happens when (x^m, θ_m) is removed? All patterns that depend on (x^m, θ_m) to be correctly classified have already been removed. If they can all be

correctly classified at this point, then (x^m, θ_m) is not required in T_{\min} which contradicts our initial assumption. Therefore, (x^m, θ_m) must be kept and so only those patterns that are not in T_{\min} are kept.

There were several alternatives possible in the simulations used to study the k -NN, CNN, and RNN rules. Different combinations of these alternatives lead to 14 different experiments. The data used were the so-called "Iris" data [6], which consist of four measurements on each of 150 flowers. There were three pattern classes, Virginica, Setosa, and Versicolor, corresponding to three different types of iris. The 150 samples were divided to form a training set and a test set. This was done in three different ways to give the following three different data sets.

1) *Random*: 120 training patterns were chosen, 40 from each pattern class. The remaining 30 patterns were used to test the training algorithms.

2) *Pathological*: There were 18 of the patterns that were misclassified by the ISODATA clustering algorithm [7] using a Euclidean distance metric. These were used for test patterns. The remaining 132 patterns were used as a training set.

3) *Inverted Pathological*: In an attempt to find out whether all 18 of the patterns mentioned in 2) were misfits, these 18 were used as a training sample to classify the remaining 132 patterns.

A second option that was available was to transform the raw data so that the axes of the data were aligned with the eigenvectors of the data covariance matrix. One can reduce the number of features by selecting only the eigenvectors corresponding to the principal or largest eigenvalues. Finally, the value of k , the number of nearest neighbors, was varied, adding another alternative. The results from these experiments are tabulated in Table I.

Experiment 1 gives some idea of how the 1-NN, CNN, and RNN rules might work in the typical case of well-separated data. Since improvement is impossible, the error probabilities stay the same when going to the three nearest neighbors in experiment 4. In this case the CNN rule offered an 83-percent improvement in memory and time efficiency, and RNN offered an additional 4 percent with no degradation of performance. The average improvement over all the experiments was 84 percent for the CNN rule and an additional 4 percent for the RNN rule. This improvement cost an average increase in the probability of error of 0.0225 for CNN and an average decrease of 0.00037 for the RNN rule.

In only two out of the eight applicable cases did the CNN rule do worse than the NN rule, and in four of the remaining six cases it did better. Similarly, the RNN rule was at least as good as the CNN and NN rules in a majority of the experiments.

As was mentioned above, the k -CNN and k -RNN rules do not appear to be directly useful for $k \neq 1$. It was also stated that if $k = 2n + 1$, then the k -CNN and k -RNN training sets could each contain $n + 1$ copies of each training pattern in the corresponding $k = 1$ training set. This was considered to be the worst case, and the question was posed of how many times the worst case would come up. Looking at the entries of Table I marked with an asterisk, signifying the k -CNN and k -RNN rules, $k \neq 1$, the worst case can be seen to appear in 11 out of 12 possible cases.

To conclude, CNN and RNN are two revised versions of an intuitively appealing decision rule. The question of whether or not they offer any general advantage over a k -NN rule must be answered with more detailed simulations. With the well-separated "Iris" data, the CNN rule appears to be very beneficial, confirming Hart's contention. The RNN rule gives an added advantage, but it is doubtful that the additional time to compute T_{RNN} required, usually more than twice as much, is worth the extra 3-4-percent gain in efficiency. However, for extremely large training sets, this 3-4 percent might be crucial.

Finally, answering Hart's question about extending the CNN rule for $k \neq 1$, which applies to the RNN rule as well, if the k nearest neighbors are used to construct T_{CNN} , then T_{CNN} will contain multiple copies of each pattern. On the other hand, if the one nearest neighbor is used to construct T_{CNN} , there is no assurance that there will be enough patterns from a given class to ever guarantee a majority. Therefore, the conjecture is that k -CNN and k -RNN offer no further improvements.

REFERENCES

- [1] P. E. Hart, "An asymptotic analysis of the nearest-neighbor decision rule," Stanford Electron. Lab., Stanford, Calif., Tech. Rep. 1828-2, SFI-66-016, May 1966.
- [2] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 21-27, Jan. 1967.
- [3] T. M. Cover, "Estimation by the nearest neighbor rule," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 50-55, Jan. 1968.
- [4] A. W. Whitney and S. J. Dwyer, III, "Performance and implementation of the k -nearest-neighbor decision rule with incorrectly identified training samples," in *Proc. 4th Allerton Conf. Circuit and System Theory*, 1966.
- [5] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inform. Theory* (Corresp.), vol. IT-14, pp. 515-516, May 1968.
- [6] J. J. Freeman, "Experiments in discrimination and classification," *Pattern Recognition*, vol. 1, no. 3, pp. 207-218.
- [7] G. H. Ball, "Data analysis in the social sciences," in *Proc. 1965 Fall Joint Computer Conf.* Washington, D.C.: Spartan, 1965, pp. 533-560.

On the Minimum Distortion of Block Codes for a Binary Symmetric Source

ANTHONY KERDOCK AND JACK K. WOLF

Abstract—It is shown that the mixture of two codes of block length n can have a distortion strictly less than the distortion of the best possible code of block length n having the same rate. The seeming contradiction is explained by offering two interpretations for the mixture of two codes and then explaining why the bound for the performance of the best code cannot be applied directly to the mixture.

Consider a source that emits a sequence of statistically independent equiprobable binary digits. We are concerned with encoding blocks of n source digits, using as codewords a set of M binary n -tuples. The distortion measure is taken as the normalized Hamming distance between a block of n source digits and its associated codeword.

Let $W = (W_1, W_2, \dots, W_n)$ be a binary n -tuple representing a source output sequence and let $Q(W) = Z = (Z_1, Z_2, \dots, Z_n)$ be the binary n -tuple representing the codeword associated with that sequence. The distortion between W and $Z = Q(W)$ is

$$D(W, Q(W)) = D(W, Z) = \frac{1}{n} \sum_{i=1}^n d_H(W_i, Z_i),$$

where

$$d_H(W_i, Z_i) = \begin{cases} 1, & W_i \neq Z_i \\ 0, & W_i = Z_i. \end{cases}$$

A code is then described by two parameters, its rate R and its average distortion \bar{D} , defined as

$$R = (1/n) \log_2 M$$

$$\bar{D} = E\{D(W, Z)\},$$

where the average is taken over all 2^n possible source sequences. The rate of the code depends only on the number of codewords, while the average distortion depends upon the choice of the codewords and the choice of a deterministic mapping $Q(W)$, which maps the source outputs into codewords.

The following is a short derivation of a well-known lower bound to \bar{D} as a function of R and n . Let α_j , $j = 0, 1, 2, \dots, n$, be the number of source sequences W for which $D(W, Q(W)) = j/n$. Then since all source sequences are equally likely

$$\bar{D} = \frac{1}{n2^n} \sum_{j=0}^n j\alpha_j. \quad (1)$$

For any code

Manuscript received July 29, 1971. This research was supported by the U.S. Air Force Office of Scientific Research under Contract F-44620-71-C-0001.
A. Kerdock is with the Sperry Division, Sperry Rand Corporation, Great Neck, N.Y.
J. K. Wolf is with the Department of Electrical Engineering, Polytechnic Institute of Brooklyn, Brooklyn, N.Y.