Nearest Neighbors in Random Subspaces

Tin Kam Ho

Bell Laboratories, Lucent Technologies
700 Mountain Avenue, 2C-425, Murray Hill, NJ 07974, USA

Abstract. Recent studies have shown that the random subspace method can be used to create multiple independent tree-classifiers that can be combined to improve accuracy. We apply the procedure to k-nearest-neighbor classifiers and show that it can achieve similar results. We examine the effects of several parameters of the method by experiments using data from a digit recognition problem. We show that the combined accuracies follow a trend of increase with increasing number of component classifiers, and that with an appropriate subspace dimensionality, the method can be superior to simple k-nearest-neighbor classification. The method's superiority is maintained when smaller number of training prototypes are available, i.e., when conventional knn classifiers suffer most heavily from the curse of dimensionality.

1 Introduction

Nearest-neighbor classifiers have been widely used in pattern recognition, mainly because of its conceptual simplicity and the assertion that its error is bounded by twice the Bayes error when the training set size approaches infinity [2]. Moreover, it is often observed that its accuracy matches or surpasses those of more sophisticated classifiers [9]. The classifier's popularity has motivated many prior works on optimizing its speed by exploiting geometrical constraints. Those works do not aim at accuracy improvement because it is assumed that using all the training data as prototypes will yield the best accuracy under the same metric.

Some other researchers have noted that for many practical problems, even when all training samples are used as prototypes, the accuracy of nearest-neighbor classifiers can be far from optimal because of the lack of sufficient prototypes in a very high dimensional feature space [3]. To overcome the difficulty with the sparsity of data in such spaces, researchers have investigated using a decision threshold [1][4] and enhancing the training set by bootstrapping [5].

A method that can exploit the advantages of nearest-neighbor classifiers without suffering from the sparsity of high-dimensional data would be valuable. Recently, it was shown [6] [7] that multiple tree-classifiers constructed in randomly selected subspaces can be combined to improve accuracy nearly monotonically as the number of trees increase. Contrary to the common difficulty encountered by other methods due to *the curse of dimensionality*, the random subspace method effectively takes advantages of high dimensionality. By systematically constructing and combining a set of classifiers that are mutually independent to a certain extent, the method can achieve a very high accuracy without the need for an

"intelligent" similarity metric. In this paper we discuss an application of the method to k-nearest-neighbor classifiers with k $\geq$ 1.

2 The Random Subspace Method for kNN Classification

The random subspace method relies on a stochastic process that randomly selects a number of components of the given feature vector in constructing each classifier. In the case of k-nearest-neighbor classifiers (knn), that means when a test sample is compared to a prototype, only the selected features have nonzero contributions to the distance. Geometrically this is equivalent to projecting all the points to the selected subspace, and the k nearest neighbors are found using the projected distances. Each time a random subspace is selected, a new set of k nearest neighbors are computed. The k nearest neighbors in each selected subspace are then assembled for a majority vote on the class membership of the test sample. The same training sample may appear more than once in this ensemble if it happens to be among the k nearest neighbors in more than one selected subspace.

Formally, given a set of N points in an n-dimensional feature space

$$\{(x_1, x_2, ..., x_n) | x_i \text{ is real for all } 1 \leq i \leq n\},$$

we consider the m-dimensional subspaces

$$\{(x_1, x_2, .., , x_n) | x_i = 1 \text{ for } i \in I, x_i = 0 \text{ for } i \notin I\}$$

where I is an m-element subset of $\{1, 2, ..., n\}$, and $m < n$. In each pass, a subspace is chosen by randomly selecting an I from $C(n, m)$-many choices. All points are projected onto the chosen subspace. For each testing point, k nearest neighbors ($1 \leq k \leq N$) among the projected training points are found using Euclidean distance, and the class labels of those k neighbors $\{c_1, c_2, ..., c_k\}$ are appended to a list C. After p ($p \geq 1$) passes, the test point is assigned to the class that has the most frequent occurrences in the list C.

The method is a derivative of *stochastic discrimination* where many stochastically created weak classifiers are combined for nearly monotonic increase in accuracy [11] [12]. The individual classifiers do not have full discriminative power but they generalize very well to unseen data for the same problem. A stochastic procedure is used to introduce independence among the classifiers. It is in combining their decisions that the discriminative power is developed. The random subspace method follows the same approach. By ignoring some dimensions of the feature space, invariance of classification is maintained for samples that differ only in the ignored dimensions. By randomly selecting the combination of dimensions to be ignored, certain independence is introduced among the component classifiers. By combining the individual decisions, discriminative power is improved, just like in other methods for classifier combination [8]. By using knn classifiers as individual components, we avoid algorithmic difficulties in achieving a uniform cover of the feature space that is needed in a former SD algorithm using simple weak models [10].

The method gives a kernel estimate of the posterior probability $P(c|x)$ for a vector x belonging to class c. The kernels are defined by the decision regions of the component classifiers so that their sizes and shapes are irregular. For knn classification, the kernels are defined by the locations of the k nearest neighbors of a test sample, thus they may be different for each test sample. In this aspect the method is very similar to simple k-nearest-neighbor classification. But with the combination of many component classifiers, there is another dimension for averaging, which makes it interesting to compare the two methods. In the rest of this paper, we will compare the empirical results of the proposed method and those of simple knn classification using data from a digit recognition problem.

Since the method uses random subsets of components of a feature vector, it will be good only for problems with a relatively large number of features. Such problems typically arise in signal processing tasks such as image and speech recognition, or large-scale data mining applications. For problems with smaller number of features, it will be necessary to augment the feature vector with certain simple functions (e.g. pairwise sums, differences, or products) of the raw features.

3 Comparison with k-Nearest-Neighbors in Full Feature Space

We tested the performance of the method using data from a handwritten digit recognition problem. Our choice of this problem was motivated by the suitability of the method to the domain, the accessibility of previous results on the problem, and the publicity of the dataset. The dataset contains 7291 training vectors and 2007 testing vectors. Each vector has 256 dimensions and it stores the gray-levels (originally in [0, 255], normalized to be in [-1,1]) of a 16×16 image of a handwritten digit. The images were scanned from mail pieces processed by the U.S. Postal Services. Recognition accuracies of several methods on this datasets have been previously reported in [13] (Table 1).

Table 1. Reported accuracies of other methods on USPS data.

Source: V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

classifier	error rate (%)
human performance	2.5
C4.5 decision tree	16.2
two-layer neural network	5.9
five-layer neural network	5.1
support vector machines (degree 3 polynomials)	4.0
support vector machines (radial basis functions)	4.1
support vector machines (neural network)	4.2

We tested the effects of several parameters of the subspace method: (1) the number of subspace dimensions (referred to as `f`); (2) the value `k` in k-nearest-

neighbor voting; and (3) the number of prototypes (`n`) used in distance comparison. For each combination of these three parameters (`f,k,n`), we built 100 knn classifiers each differing only by the random choices of feature components (Figure 1). In each component classifier, k nearest neighbors of a test sample are found among the prototypes using Euclidean distance, but voting for class membership is delayed until decision combination. We measured the accuracy of the combined decisions on the test samples as each classifier was added. Ambiguous decisions were resolved randomly. These accuracies, as well as those of simple knn classification using each (`k,n`) combination (`f=256`, the entire feature vector is used), are plotted in Figure 2.

From Figures 2 and 3, it can be observed that, regardless of the parameter values, the trend of accuracies of the subspace method is like all other variants of stochastic discrimination, i.e., as the number of component classifiers increases, test set accuracy increases, and the rate of increase is largest at the beginning.

The Effect of k

We first focus our discussion on the first column of Figure 2. There we show the accuracies of the method with different choices of `k` and `f`, using the entire training set for prototypes in distance computation (`n=7291`). For this dataset

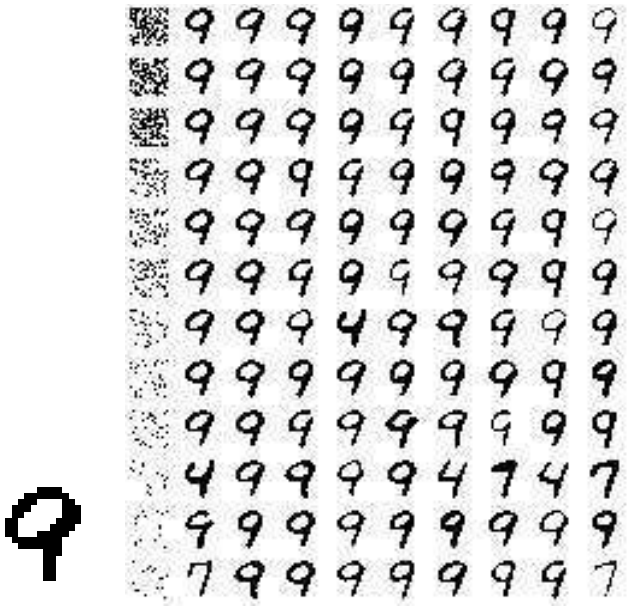


Fig. 1. A test sample and its 9 nearest neighbors under 12 random masks of 128, 64, 32, or 16 features each.

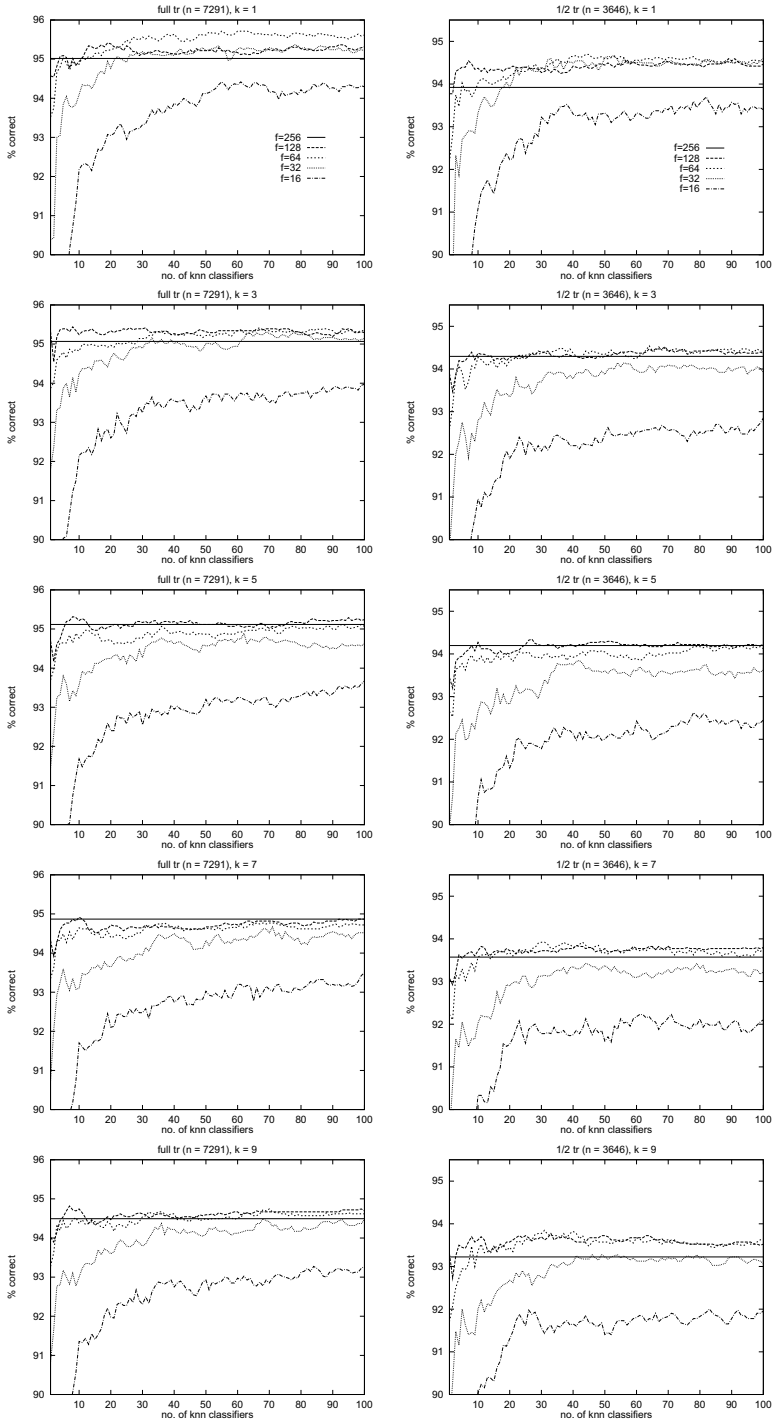


Fig. 2. Test set accuracies of combined knn classifiers in random subspaces, with full and half the training set as prototypes.

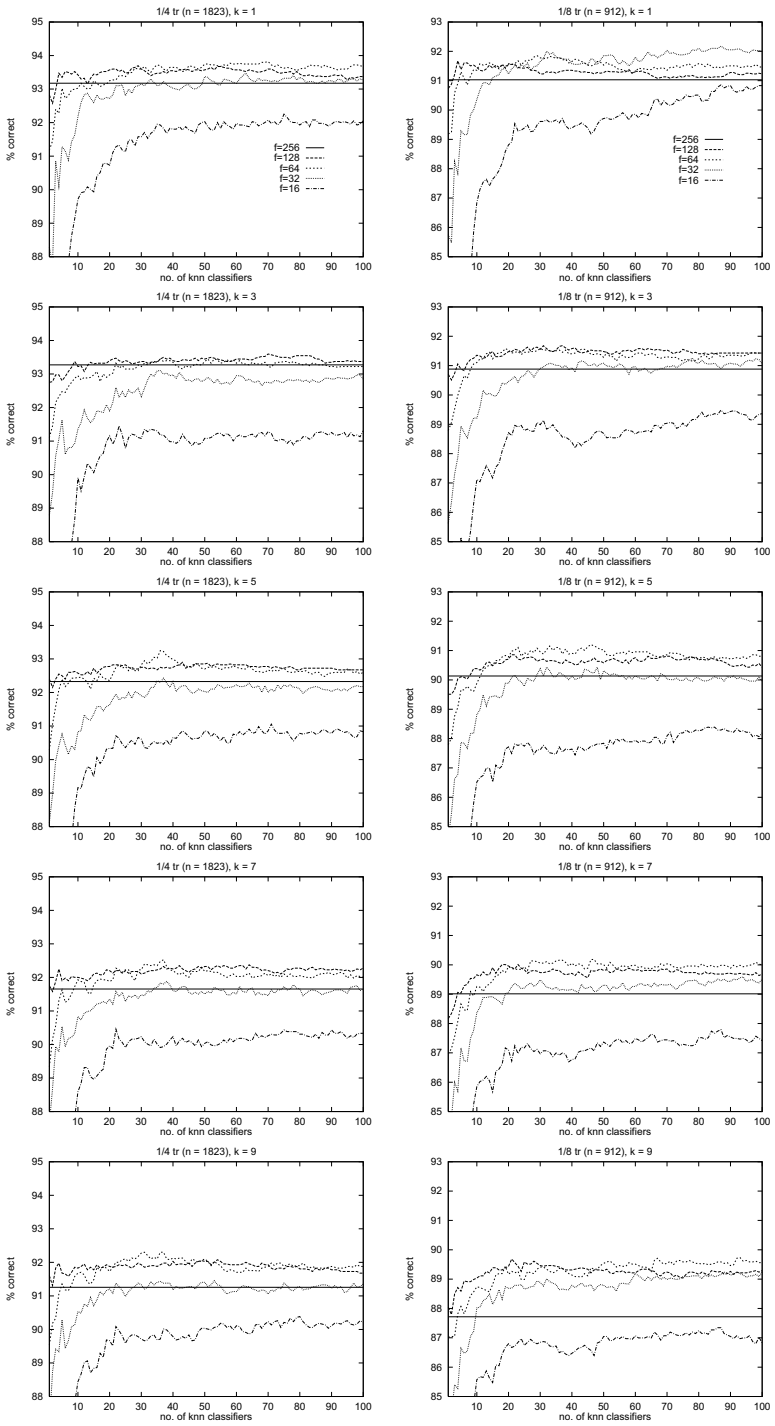


Fig. 3. Test set accuracies of combined knn classifiers in random subspaces, with 1/4 and 1/8 of the training set as prototypes.

it seems that `k=5` would be the best for simple knn classification using the full feature vector (when `f=256`). However, with several settings of `k` and `f` (such as (`k=1, f=64`)), the random subspace method yields a better accuracy. In fact, with all the settings of `k`, there is a choice of `f` so that the accuracy of the subspace method either closely matches or exceeds that of simple knn. With its best setting (`f=64` and `k=1`), the subspace method scores an error rate of 4.285%, matching closely with the best reported accuracy for this dataset. Also, with `k = 1,3,` or `5`, it does not take many (about 10) of the component classifiers for the method to be better than simple knn.

The Effect of Sample Size

We now compare the different columns of plots in both Figures 2 and 3. This comparison is to show the effect of a smaller training set on accuracy and differences between accuracies. Beginning with the entire training set, the set of prototypes to be used is randomly reduced to half in the runs shown in each column to simulate smaller training sets. The prototypes are fixed for all runs plotted in the same column. As expected, the number of prototypes has a very strong effect on accuracy, as the sparsity of data is one of the biggest problem with knn classifiers. Nevertheless, the superiority of the random subspace method persists (with `k=1`) and the margin is maintained between the accuracy of the random subspace method and that of simple knn as the training set decreases in size (Figure 4).

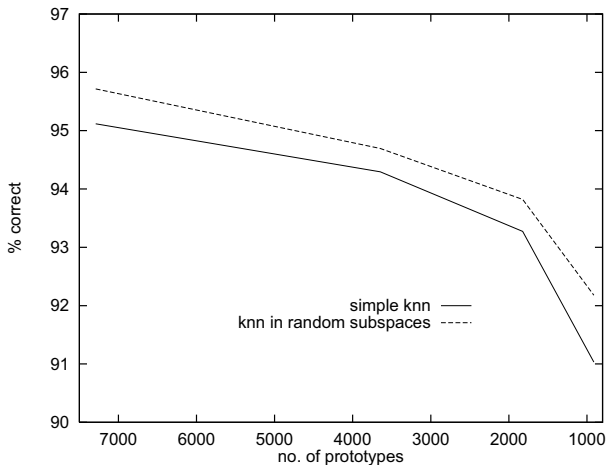


Fig. 4. Comparison of best achieved accuracy of the random subspace method versus that of simple knn as the size of the training set (number of prototypes) changes.

The Effect of Subspace Dimensionality

Some interesting observations can be made about the effect of `f`. When the training set is large (`n=7291`), `f=64` yields the best accuracy. As `n` decreases and with `k=1`, the accuracy with `f=32` gets closer to that with `f=64`, and eventually becomes the best when the training set size reaches one eighth of the original (`n=912`). One may also notice that the rates of increases in accuracy are different for each value of `f` and for all values of `n` and `k`. Classifiers with larger values of `f` tend to produce accuracy curves that rise rapidly when only a few component classifiers are combined, but stay flat afterwards. Those with smaller values of `f`, on the contrary, produce slower increases, but their accuracies get closer to or eventually may exceed those with larger values of `f` (say, at `n=912`).

The slower increases in accuracy are due to weaker discriminative power of component classifiers using a smaller number of features. But those weaker classifiers tend to generalize better, and when the training set is small, their generalization power becomes more valuable. Unfortunately it is impossible to test the effects of all values of `f` when the number of component classifiers approaches infinity. Therefore, it remains uncertain at which point the effect on accuracy will be dominated by the generalization power rather than the discriminative power of the component classifiers.

4 Implementational Concerns

On a first thought the method seems to be a prohibitively expensive procedure, since the knn calculation is done in multiple times. However, the distance computation can be organized in a way that the per-feature differences and their squares need only be computed once. The random subspace selection can be implemented as masks on the feature vectors. Each component classifier has its own mask, and only the differences of the unmasked features are summed and sorted. Therefore it is only the summation of per-feature differences and the sorting of the sums that are performed multiple times. With `n` training samples, `m`-dimensional feature space, `f`-dimensional subspaces, and `p` component classifiers, the run time for each test sample is $O(\mathtt{mn} + \mathtt{p}(\mathtt{nf} + \mathtt{n}\log\mathtt{n}))$, and that with simple knn is $O(\mathtt{mn} + \mathtt{n}\log\mathtt{n})$. These estimates assume no optimization in finding the nearest neighbors.

When a large number of component classifiers are used, much of the run time is spent in the sorting passes. Therefore an efficient implementation of the method will rely on ways to avoid complete sorting of all the distances by utilizing geometrical constraints such as the triangular inequality. Better speed and accuracy tradeoff may also be obtained by reordering the component classifiers so that those contributing most to accuracy improvement are used first.

5 Conclusions

We described an application of the random subspace method to k-nearest-neighbor classifiers and showed that, on a practical digit recognition problem, the method

achieves a pattern of performance very similar to those of other applications of stochastic discrimination, namely, accuracy improves nearly monotonically with increasing number of component classifiers. We also showed that knn classification, performed in this way, can exceed the apparent upper limit of accuracy given by conventional knn classifiers using the entire training set for prototypes, and such a superiority is well maintained with decreasing number of training samples. For the digit recognition problem studied in our experiments, the accuracy of the method closely matches the best reported accuracy for the same data. It is worth noting that the results are obtained without reliance on a sophisticated similarity metric designed specifically for the problem.

References

1. Chandrasekaran, B., Jain, A.K.: On balancing decision functions. J. of Cybernetics and Information Science, **2**, 3 (1979) 12–15
2. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory, **IT–13**, 1 (1967) 21–27
3. Fukunaga, K., Hummels, D.M.: Bias of nearest neighbor error estimates. IEEE Transactions on Pattern Analysis and Machine Intelligence, **9** (1987) 103–112
4. Fukunaga, K., Hummels, D.M.: Bayes error estimation using Parzen and k-NN procedures. IEEE Transactions on Pattern Analysis and Machine Intelligence, **9** (1987) 634–643
5. Hamamoto, Y., Uchimura, S., Tomita, S.: A bootstrap technique for nearest neighbor classifier design. IEEE Transactions on Pattern Analysis and Machine Intelligence, **19**, 1 (1997) 73–79
6. Ho, T.K.: Random decision forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, Canada, August 14-18 (1995) 278–282
7. Ho, T.K.: C4.5 decision forests. Proceedings of the 14th International Conference on Pattern Recognition, Brisbane, Australia, August 17-20 (1998)
8. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, **16**, 1 (1994) 66–75
9. Ho, T.K., Baird, H.S.: Large-scale simulation studies in image pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, **19**, 10 (1997) 1067–1079
10. Ho, T.K., Kleinberg, E.M.: Building projectable classifiers of arbitrary complexity. Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, August 25-30 (1996) 880–885
11. Kleinberg, E.M.: Stochastic discrimination. Annals of Mathematics and Artificial Intelligence, **1** (1990) 207–239
12. Kleinberg, E.M.: An overtraining-resistant stochastic modeling method for pattern recognition. Annals of Statistics, **4**, 6 (1996) 2319–2349
13. Vapnik, V.: The nature of statistical learning theory. Springer-Verlag (1995)

This article was processed using the LaTeX macro package with LLNCS style