K Nearest Neighbor Classification on Feature Projections*

Aynur Akkuş
Dept. of Computer Engr. and Info. Sci.
Bilkent University
06533 Ankara, Turkey
akkus@cs.bilkent.edu.tr

H. Altay Güvenir
Dept. of Computer Engr. and Info. Sci.
Bilkent University
06533 Ankara, Turkey
guvenir@cs.bilkent.edu.tr

Abstract

This paper proposes a new approach to classification based on a majority voting on individual classifications made by the projections of the training set on each feature. We have applied the k-nearest neighbor algorithm to determine the classifications made on individual feature projections. We called the resulting algorithm k-NNFP, for *k-Nearest Neighbor on Feature Projections*. The most important characteristic of this technique is that the training instances are stored as their projections on each feature dimension. This allows the classification of a new instance to be made much faster than k-NN algorithm. The voting mechanism reduces the negative effect of possible irrelevant features in classification. Also the classification accuracy of k-NNFP increases when the value of k is increased, which indicates that the process of classification can incorporate the learned classification knowledge better when k increases. The k-NNFP algorithm is compared with the k-NN algorithm, in terms of classification accuracy and running time on some real-world and artificial datasets.

1 INTRODUCTION

Learning to classify objects is one of the fundamental problems in machine learning. This paper presents a new approach to classification based on feature projections. In this approach, the classification knowledge is represented in the form of sets of projections of the training data on each feature dimension. The classification of an instance is based on a voting taken on the classifications made on the basis of individual feature projections.

One of the most common classification techniques is the nearest neighbor (NN) algorithm. In the literature, nearest neighbor algorithms for learning from examples have been studied extensively (Duda & Hart, 1973; Dasarathy, 1990). Aha et al. (1991) has demonstrated that instance-based learning and nearest neighbor methods often work as well as other sophisticated machine learning techniques. A recent work by Salzberg et al. (1995) has given the best case results for the nearest neighbor learning. An experimental comparison of the NN and NGE (*Nested Generalized Exemplars*, a Nearest-Hyperrectangle algorithm) is presented by Wettschereck and Dietterich (1995). An average-case analysis of k-NN classifiers for Boolean threshold functions on domains with noise-free Boolean features and a uniform instance distance distribution is given by Okamoto and Satoh (1995). They observed that the performance of the k-NN classifier improves as k increases, then reaches a maximum before starting to deteriorate, and the optimum value of k increases gradually as the number of training instances increases.

In this paper, we present a particular implementation of the classification on feature projections approach. In this implementation, the classification on a feature is done according to the k nearest neighbors of the test instance on that feature. The resulting algorithm is called k-NNFP for *k Nearest Neighbor on Feature Projections*. The final classification of the test instance is determined by a majority voting among the individual classifications of each feature.

The basic motivation for this study comes from the encouraging results of Classification by Feature Partitioning (Guvenir & Sirin, 1993, 1996) and Classification with Overlapping Feature Intervals (Unsal 1995). CFP and COFI algorithms are algorithms based on feature partitioning and overlapping feature intervals, respectively. The most important property of these

*This project is supported by TUBITAK (Scientific and Technical Research Council of Turkey) under Grant EEEAG-153.

studies is that both consider each feature separately. The reported results show that both techniques have become successful by processing each feature separately.

In the next section, a brief introduction to k-NN algorithm and its several extensions are given. In the third section, the k-NNFP algorithm is described. Section 4 presents the complexity analysis and empirical evaluation of the k-NNFP and k-NN algorithms. The final section concludes with some remarks on the k-NNFP algorithm and its applicability.

2 THE K-NN ALGORITHM

The NN classification algorithm is based on the assumption that examples which are closer in the instance space are of the same class. Namely, unclassified ones should belong to the same class as their nearest neighbor in the training dataset. After all the training set is stored in memory, a new example is classified with the class of the nearest neighbor among all stored training instances. Although several distance metrics have been proposed for NN algorithms (Salzberg 1991), the most common metric is the Euclidean distance metric. The Euclidean distance between two instances $x =< x_1, x_2, ..., x_n >$ and $y =< y_1, y_2, ...y_n >$ on an n dimensional space is computed as:

$$dist(x, y) = \sqrt{\sum_{f=1}^{n} diff(f, x, y)^2} \quad (1)$$

$$diff(f, x, y) = \begin{cases} |x_f - y_f| & \text{if } f \text{ is linear} \\ 0 & \text{if } f \text{ is nominal and } x_f = y_f \\ 1 & \text{if } f \text{ is nominal and } x_f \neq y_f \end{cases} \quad (2)$$

Here $diff(f, x, y)$ denotes the difference between the values of instances x, and y on feature f. Note that this metric requires the normalization of all feature values into a same range.

Although several techniques have been developed for handling unknown (missing) feature values (Quinlan, 1989, 1993), the most common approach is to set them to the mean value of the values on corresponding feature.

Stanfill and Waltz (1986) introduced the Value Difference Metric (VDM) to define similarity when using symbolic-valued features and empirically demonstrated its benefits.

A generalization of the nearest neighbor algorithm, k-NN, classifies a new instance by a majority voting among its k ($k \geq 1$) nearest neighbors using some distance metrics. This algorithm can be quite effective when the attributes of the data are equally important. However, it can be less effective when many of the attributes are misleading or irrelevant to classification. Kelly and Davis (1991) introduced WKNN, the *weighted* k-NN algorithm, and GA-WKNN, a *genetic algorithm* that learns feature weights for WKNN algorithm. Assigning variable weights to the attributes of the instances before applying the k-NN algorithm distorts the feature space, modifying the importance of each feature to reflect its relevance for classification. In this way, similarity with respect to important attributes becomes more critical than similarity with respect to irrelevant attributes.

3 THE K-NNFP ALGORITHM

In this section we introduce the k-NNFP algorithm for classification based on feature projections using k nearest neighbor algorithm. First, the description of the algorithm is given. Then the algorithm is explained on an example dataset. Later, the behavior of the algorithm on datasets with irrelevant features will be given.

3.1 DESCRIPTION OF THE ALGORITHM

The implementation of the algorithm given here is non-incremental, namely, all training instances are taken and processed at once. An important characteristic of this algorithm is that instances are stored as their projections on each feature dimension. In the training phase, each training instance is stored simply as its projections on each feature dimension. If the value of a training instance is missing for a feature, that instance is not stored on that feature.

In order to classify an instance, a preclassification separately on each feature is performed. In this preclassification, we use the k-NN algorithm for a single dimension. That is, for a given test instance t and feature f, the preclassification for $k = 1$ will be the class of the training instance whose value on feature f is the closest to that of the t. For a larger value of k, the preclassification is a bag (multiset) of classes of the nearest k training instances. In other words, each feature has exactly k votes, and gives these votes for the classes of the nearest training instances. In some cases, especially for nominal features, there may be ties to determine the first k nearest neighbors. In such cases ties are broken randomly. For the final classification of the test instance t, the preclassification bags of each feature are collected using bag union. Finally, the class that occurs most frequently in the collection bag is predicted to be the class of the test instances. In other words, each feature has exactly k votes, and gives these votes for the classes of the nearest training instances. Also note that, since each feature is pro-

```
classify(t, k):
/* t: test instance, k: number of neighbors */
begin
  for each class c
     vote[c] = 0

  for each feature f
     /* put k nearest neighbors of test instance t
        on feature f into Bag */
     Bag = kBag(f, t, k)
     for each class c
        vote[c] = vote[c] + count(c, Bag);

  prediction = UNDETERMINED /* class 0 */
  for each class c
     if vote[c] > vote[prediction] then
        prediction = c

  return (prediction)
end.
```

Figure 1: Classification in the k-NNFP Algorithm.

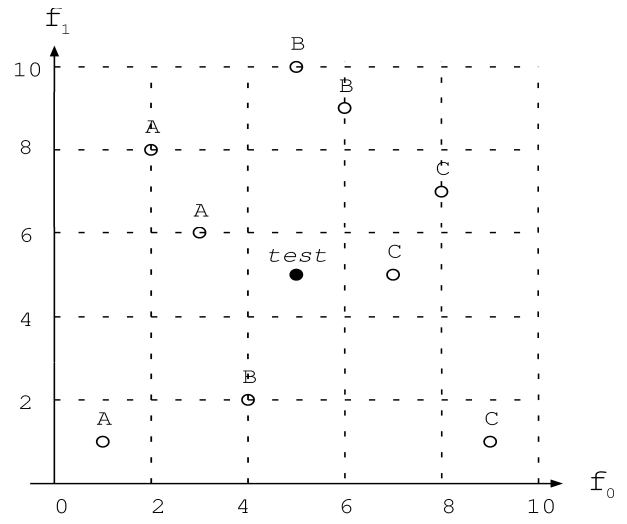

Figure 2: A sample training dataset and a test instance.

cessed separately, no normalization of feature values is needed. The k-NNFP algorithm is outlined in Figure 1.

All the projections of training instances on linear features are stored in memory as sorted values. In Figure 1, the votes of a feature is computed by the function $kBag(f, t, k)$, which returns a bag of size k containing the classes of the k nearest training instances to the instance t on feature f. Distance between the values on a feature dimension is computed using $diff(f, x, y)$ metric given in (2). Note that the bag returned by $kBag(f, t, k)$ does not contain any *UNDETERMINED* class as long as there are at least k training instances whose f values are known. Then, the number of votes for each class is incremented by the number of votes that a feature gives to that class, which is determined by the *count* function. The value of $count(c,\ Bag)$ is the number of occurrences of class c in bag Bag.

The k-NNFP algorithm handles unknown feature values in a straight forward manner. If the value of a test instance for a feature f is missing, then feature f does not participate in the voting for that instance. The final voting is done between the features for which the test instance has a known value. That is, unknown feature values are simply ignored.

3.2 AN EXAMPLE

In order to describe the classification in the k-NNFP algorithm, consider the sample training dataset in Figure 2. In this dataset, the feature f_0 is the only relevant feature, and f_1 is an irrelevant feature. There are three instances of each class A, B, and C in the training set. The test instance $(< 5, 5 >)$ is of class B.

For the test instance in Figure 2, the k-NN classification, $kBag$ values and final prediction for the k-NNFP algorithm are given in Table 1.

As seen in Table 1, the k-NN algorithm will classify the test instance as C if $k = 1$, as C or A if $k = 2$, as C, or A or B if $k = 3$, and as C if $k = 4$. On the other hand, the k-NNFP algorithm will classify the test instance correctly if $k > 1$. This example shows that the k-NNFP algorithm will be unaffected in the presence of irrelevant features.

3.3 HANDLING IRRELEVANT FEATURES

The conclusion about the irrelevant features from the previous example can be generalized. For an irrelevant feature f, the number of occurrences of a class c in a bag returned by $kBag(f, t, k)$ is proportional to the number of instances of class c in the training set. If there are equal number of instances of each class in the training set, than the votes of an irrelevant feature will be equal for each class, and the final prediction will be determined by the votes of the relevant features. If the training instances are not equally distributed among the classes, then the votes of an irrelevant feature will be for the most frequently occuring class.

Table 1: For the test instance ($< 5, 5 >$) in Figure 2 the k-NN classification, kBag values and final prediction of the k-NNFP algorithm.

		k-NNFP			
k	k-NN	f_0	f_1	Sum of Votes	Prediction
1	[C]	[B]	[C]	[B,C]	B or C
2	[C,A]	[B,B]	[C,A]	[A,B,B,C]	B
3	[C,A,B]	[B,B,B]	[C,A,C]	[A,B,B,B,C,C]	B
4	[C,A,B,C]	[B,B,B,A]	[C,A,C,B]	[A,A,B,B,B,B,C,C]	B

3.4 HANDLING MISSING FEATURE VALUES

The k-NNFP algorithm handles the unknown (missing) feature values by not taking them into account. The features containing missing values are simply ignored. This is a natural approach because in real life if nothing is known about a feature, it is usually ignored. If all class dimensions give no prediction, then no prediction is made and the resulting prediction for the class is UNDETERMINED. This is an unexpected case since at least one feature value should be known.

4 EVALUATION OF THE K-NNFP ALGORITHM

In this section, the training and classification complexities of the k-NNFP and the k-NN algorithms are given. Next, an empirical evaluation of the algorithm is presented along with its comparison with the k-NN algorithm.

4.1 COMPLEXITY ANALYSIS

Since all the training instances are stored in the memory, the space required for training with m instances on a domain with n features is proportional to $m \cdot n$.

In the training, all instances are stored on each feature dimension as their feature projections. And then they are sorted once at the end. For a dataset containing m instances and n features the training time complexity of the k-NNFP is $O(n \cdot m \cdot \log m)$.

The $kBag(f, t, k)$ function, to determine the votes of a feature, first finds the nearest neighbor of t on f and then next $k - 1$ neighbors around the nearest neighbor. The time complexity of this process is $O(\log m + k)$. Since $m >> k$, the time complexity of $kBag$ is $O(\log m)$. The final classification requires the votes of each of n features. Therefore, the classification time complexity of the k-NNFP algorithm is $O(n \cdot \log m)$.

On the other hand, in the k-NN algorithm, the classification of a test instance requires the computation of its distance to m training instance on n dimensions. Therefore, the classification time complexity of the k-NN algorithm is $O(n \cdot m)$, assuming $m >> k$.

4.2 EMPIRICAL EVALUATION

In this section we present an empirical evaluation of the k-NNFP algorithm on both real-world data sets and artificially generated datasets in order to show the effect of irrelevant features on the classification accuracy. The results will be compared with that of the k-NN algorithm.

4.2.1 Experiments with Real-World Datasets

The k-NNFP and k-NN algorithms are evaluated on some real-world datasets which are widely used in the machine learning field, therefore comparisons will be possible with other similar methods in future. The real-world datasets are selected from the collection of datasets provided by the machine learning group at the University of California at Irvine (Murphy 1995).

Several measures of performance are possible. One performance measure of a classification algorithm can be found in terms of classification accuracy. For supervised concept learning tasks, the most commonly used classification accuracy metric is the percentage of correctly classified instances over all test instances. The other performance measures are time and space complexities. The space required by both of these algorithms is the same. Therefore, in this section we will present the classification accuracy for increasing values of k, and average running time.

Accuracy of an algorithm is a measure of correct classifications on a test set of unseen instances. There are several ways of measuring the accuracy of an algorithm. In this study, we chose the 5-way cross-validation techniques. That is, the whole dataset is partitioned into 5 subsets. The four of the subsets is used as the training set, and the fifth is used as the test set, and this process is repeated 5 times once for each subset being the test set. Therefore, each instance appears once in the test set, and four times in the training set. Classification accuracy is the average of these 5 runs.

An overview of the datasets is shown in Table 2. In this table, name of the real-world datasets are shown with the size of the dataset, number of features, number of classes, and number of missing feature values.

The accuracy of the k-NNFP in Table 3 and k-NN in Table 4 were obtained for the specified datasets for k = 1, 2, ... 10.

These experiments show that the classification accuracy of the k-NNFP algorithm usually increases when the value of k increases. This suggests that the k-NNFP algorithm can exploit the knowledge represented in the form of feature projections for higher values of k. On the other hand, increase in the value of k does not result in a parallel increase in the accuracy of the k-NN algorithm. Langley and Sage's works on NN classifiers suggest that many of the UCI datasets have few irrelevant features, if any. Our experimental results also suggest this claim.

4.2.2 Experiments on Artificial Data

As indicated in Section 3.2, the k-NNFP algorithm is, in general, unaffected from the presence of irrelevant features in the dataset. Experiments with artificial datasets have important roles to play in the study of irrelevant features. Hence, in order to empirically prove this claim, we have generated six datasets with increasing number of irrelevant features from zero to ten. Each of the datasets contain four relevant features, three classes with 100 instances each. A class is represented by a hyperrectangle in four (relevant) dimensional space, the values for irrelevant features are randomly generated. We have conducted 5-way cross-validation experiments on these six datasets, and compared the results of k-NNFP and k-NN algorithms. The accuracy results are plotted in Figure 3. Almuallim and Dietterich (1991) studied learning with many irrelevant features in the space of all binary functions defined over Boolean input features, examining the MIN-FEATURE inductive bias.

As seen from these results, the decrease in the accuracy of the k-NNFP algorithm when the number of irrelevant features increase is much less than that of the k-NN algorithm. Also we observed that the accuracy of the k-NNFP algorithm increases parallel to the increase in the value of k, whereas the accuracy of the k-NN algorithm is not correlated with increase in the value of k.

The time required to train the k-NNFP and the k-NN algorithms with the 80% of the data and test with the remaining 20% for these datasets are given in Table 5. The comparison of the running times in this table agrees with the time complexity analysis of these algorithms given in Section 4.1.

5 CONCLUSIONS

In this paper, a new form of classification method, called k-NNFP, has been presented. This algorithm has been compared with the k-NN algorithm in terms of classification accuracy and time complexity on both real-world and artificially generated datasets.

In the k-NNFP algorithm, the classification knowledge is represented in the form of sets of projections of the training data separately on each feature dimension. The classification of an instance is based on a majority voting taken on the classifications made on the basis of individual feature projections. Since each feature is processed separately, there is no need for normalization of feature values. Also, for the same reason, the algorithm can simply ignore any missing feature values that may appear both in training and test instances. The effect of the missing and noisy feature values on the prediction accuracy of the k-NNFP algorithm will be investigated as a future work. As another direction for future work, we plan to integrate a feature weight learning algorithm to k-NNFP.

The k-NNFP algorithm is based on the assumption that each feature can contribute the classification process and the majority voting provides a correct classification when data contain many irrelevant features. The k-NNFP algorithm can provide better classification accuracy than k-NN algorithm when a dataset contains many irrelevant features with respect to relevant ones. This claim has been justified on artificially generated datasets. On real-world datasets, the k-NNFP algorithm achieves comparable accuracy with the k-NN algorithm. On the other hand, the average running time of the k-NNFP algorithm is much less than that of the k-NN algorithm.

The k-NNFP algorithm treats feature values independently, whereas the k-NN algorithm treats all instances as points in Euclidean n-space. The k-NNFP algorithm stores the feature projection of the training instances in a sorted order. Therefore, the classification of a new instance requires a simple search of the nearest training instance value. On the other hand, in the k-NN algorithm, a new search must be done for each test instance in the whole Euclidean space.

References

Aha, D. W., Kibler, D. & Albert, M. K. (1991). Instance-Based Learning Algorithms, *Machine Learning*, **6**:37-66.

Almuallim, H. & Dietterich, T. G. (1991). Learning with Many Irrelevant Features, Proceedings of the Ninth National Conference on Artificial Intelligence, 547-552.

Dasarathy, B. V., (1990). *Nearest Neighbor (NN)*

Table 2: Comparison on some real-world datasets.

Data Set:	bcancerw	cleveland	glass	hungarian	ionosphere	iris	musk	wine
No. of Instances	273	303	214	294	351	150	476	178
No. of Features	9	13	9	13	34	4	166	13
No. of Classes	2	2	6	2	2	3	2	3
No. of Missing values	16	6	0	784	0	0	0	0

Table 3: Accuracy (%) and average running time (msec) of the k-NNFP algorithm on real-world datasets.

Data Set:	bcancerw	cleveland	glass	hungarian	ionosphere	iris	musk	wine
k=1	94.0	67.62	57.0	70.04	88.04	90.0	69.54	79.7
k=2	94.56	72.28	62.14	70.7	88.02	92.0	71.4	90.4
k=3	94.88	72.94	61.18	75.84	88.02	91.34	70.76	90.96
k=4	95.72	77.56	60.74	73.8	87.46	92.64	71.4	93.24
k=5	96.16	78.88	60.72	76.16	87.46	91.3	71.22	93.24
k=6	96.0	77.86	63.54	72.76	87.78	91.88	69.96	95.48
k=7	96.0	79.52	62.58	74.8	87.74	92.0	70.36	95.48
k=8	96.14	79.18	63.98	73.76	86.9	92.66	69.96	96.04
k=9	96.14	78.52	63.04	75.8	87.44	92.0	69.52	96.62
k=10	96.14	78.86	64.9	72.76	87.46	94.02	69.1	96.62
Avg. Time	340	740	94	266	477	40	2654	282

Table 4: Accuracy (%) and average running time (msec) of the k-NN algorithm on real-world datasets.

Data Set:	bcancerw	cleveland	glass	hungarian	ionosphere	iris	musk	wine
k=1	95.0	80.52	68.66	75.5	84.62	93.98	73.1	94.4
k=2	93.84	80.2	67.7	79.54	88.06	94.0	77.54	94.42
k=3	96.28	82.5	66.76	81.58	83.78	94.68	70.18	96.6
k=4	95.72	82.84	68.14	80.92	85.2	94.0	74.16	94.38
k=5	96.58	83.8	66.3	82.26	83.2	94.66	67.88	96.04
k=6	96.56	82.82	67.24	83.64	83.76	95.32	69.14	96.08
k=7	96.26	82.5	65.36	83.28	82.34	94.66	65.58	96.04
k=8	95.86	82.16	65.36	83.62	84.06	94.66	67.86	95.48
k=9	95.56	82.82	65.34	82.94	8262	94.66	65.16	96.04
k=10	95.7	81.48	63.96	83.96	84.06	94.66	67.86	96.06
Avg. Time	3216	7786	318	695	2335	105	18520	615

Table 5: The average time (in msec) required to train with 80% and test with the 20% of the artificial datasets for increasing number of features.

	Number of features					
	4	6	8	10	12	14
k-NNFP	85	212	285	367	517	556
k-NN	365	937	1257	1335	1472	1720

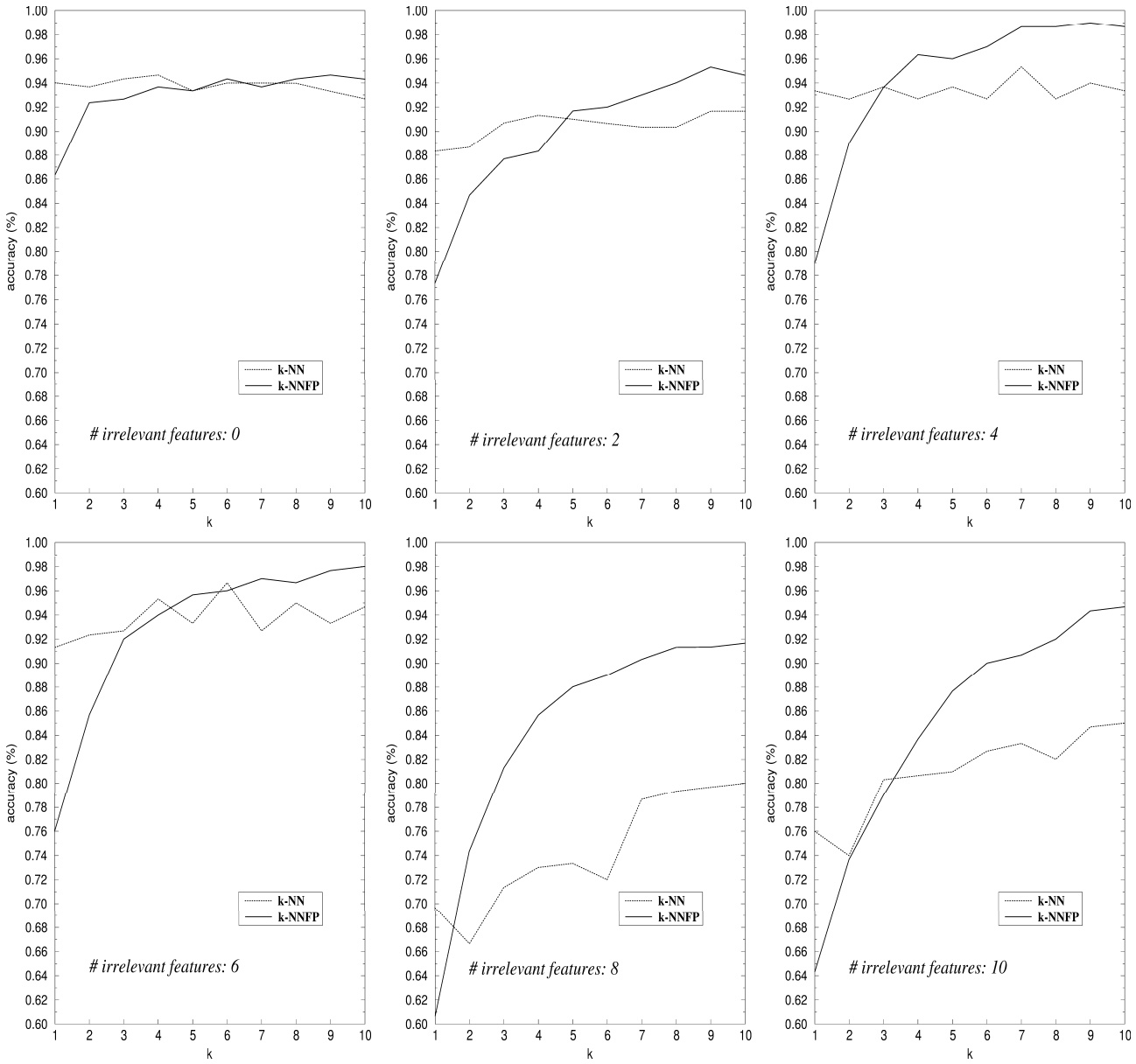

Figure 3: Comparison of k-NN and k-NNFP on artificial datasets for increasing value of k. In all datasets there are 4 relevant features, 3 classes and 100 instances for each class. The accuracy results are obtained by 5 way cross-validation.

Norms, NN Pattern Classification Techniques. IEEE Computer Society Press.

Duda, R.O. & Hart, P.E. (1973). *Pattern Classification and Scene Analysis.* New York: Wiley & Sons.

Güvenir, H. A., & Şirin, İ. (1993). The complexity of the CFP, a method for classification based on feature partitioning. In P. Torasso (Ed.) *Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence, LNAI 728*, 202-207. Berlin: Springer-Verlag

Güvenir, H. A., & Şirin, İ. (1996). Classification by Feature Partitioning, *Machine Learning*, **23**:47-67.

Kelly, J. D., & Davis, L. (1991). *A Hybrid Genetic Algorithm for Classification. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 645-650.

Langley, P., & Sage, S. (In press). Scaling to domains with many irrelevant features. In E. Greiner (ed.), *Computational learning theory and natural learning systems*, (Vol. 4). Cambridge, MA:MIT Press.

Murphy, P. (1995). UCI Repository of machine learning databases - Maintained at the Department of Information and Computer Science, University of California, Irvine, Anonymous FTP from ics.uci.edu in the directory pub/machine-learning-databases.

Okamoto, S. & Satoh, K. (1995). An Average-Case Analysis of k-Nearest Neighbor Classifier. In *Proceedings of the First Internaltional Conference on Case-Based Reasoning*, 243-264.

Quinlan, J. R. (1989). Unknown Attribute Values in Induction, In A. Segre (Ed.), *Proceedings of the 16th International Workshop on Machine Learning* , 164-168, San Mateo, CA:Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* California: Morgan Kaufmann.

Salzberg, S. (1991). Distance Metrics for Instance-Based Learning, ISMIS'91 6th International Symposium, Methodologies for Intelligent Systems, 399-408.

Salzberg, S., Delcher, A., Heath, D., & Kasif, S., (1995) Best-Case Results for Nearest Neighbor Learning, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**: 599–610.

Stanfill, C., Waltz, D., (1986). Toward memory-based reasoning. *Communications of the Association for Computing Machinery*, **29**: 1213–1228.

Ünsal, H. (1995) Classification with Overlapping Feature Intervals, Bilkent University, Dept. of Computer Engineering and Information Science, MSc. Thesis.

Wettschereck, D., Dietterich, T. G. (1995). An Experimental Comparison of the Nearest Neighbor and Nearest-hyperrectangle Algorithms, *Machine Learning*, **9**: 5-28.