Weighting Features in k Nearest Neighbor Classification on Feature Projections[1]

Aynur Akkuş and H. Altay Güvenir

Department of Computer Engineering and Information Science
Bilkent University
Email : {akkus, guvenir}@cs.bilkent.edu.tr

BU-CEIS-9616

Abstract

This paper proposes two methods for learning feature weights to improve the classification accuracy of the k-NNFP algorithm. In the k-NNFP algorithm, instances are stored as their projections on each feature dimension. The classification of unseen examples are made on the basis of feature projections by a majority voting among the k ($k \geq 1$) predictions of each feature separately. We have treated all features as equivalent in this algorithm. However, all features may not have equal relevancy, even some features may be completely irrelevant. In order to determine features' relevances, the best method is to assign them weights. The first method is based on the assumption of homogeneous feature projections for which the number of consequent values of feature projections of a same class supports an evidence for increasing the probability of correct classification in the k-NNFP algorithm. The second method is based on the individual accuracies of features. In this approach, the k-NNFP algorithm is run on the basis of a single feature, once for each feature. The resulting accuracy is taken as the weight of that feature since it is a measure of contribution to classification for that feature. Empirical evaluation of these feature weighting methods in the k-NNFP algorithm on real world datasets is given.

1 Introduction

One of the central problems when classifying objects is discriminating between features that are relevant to the target concept and that are irrelevant. Many researchers have addressed the

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

issue of features' relevancies in order to reduce the impact of irrelevant features and to increase the impact of more relevant features in classification task by investigating feature weighting (Aha, 1990), feature selection (Langley & Sage, 1994; Skalak, 1994). One of the most common machine learning algorithms for which these methods are employed is the nearest neighbor (NN) algorithm (Dasarathy, 1990). In k nearest neighbor (k-NN) algorithm, all training instances are stored in the memory as points. It tests a new example finding the nearest examples and makes prediction using k of them based on some similarity or distance measure. Simple k-NN algorithm gives equal weight to all features. However, in real life, the relevance of features may not be the same, some of them may be more important than the others. The algorithms which assign equal weights to all features are more sensitive to the presence of irrelevant features. In order to remove the negative effect of irrelevant features, feature subset selection approaches are investigated in which the space of subsets of feature sets are considered to determine the relevant and irrelevant features (John, Kohavi, Pfleger, 1994). Briefly, the algorithm is run on the training data with different subsets of features, using cross-validation to estimate its accuracy with each subset. These estimates are used as an evaluation metric for directing search through the space of feature sets. Aha and Bankert (1994), Langley and Sage (1994), Skalak (1993) have reported improved results in accuracy over simple NN. On the other hand, the disadvantage of using feature selection method is that it treats features as completely relevant or irrelevant. The degree of relevance may not be 0 or 1, it may be a value between them.

In previous distance-based classification methods, a numerical value is assigned to features to modify the distance measure for each feature relevancy. Wettshchereck and Aha (1995) presents a work on feature weighting methods introducing a five-dimensional framework. The *feedback dimension* is the first dimension which concerns whether the feature weighting method receives feedback from the induction algorithm trying to be improved. The weighting methods that receive feedback are called *feedback* methods, and the ones which doesn't receive any feedback from are called *ignorant* methods. Incremental hill-climbers (Wettschereck & Aha, 1995), IB4 (Aha, 1992), and EACH's weighting method (Salzberg, 1991) are categorized as feedback method according to this framework. *Continues optimizers* under *feedback* method consist of GA-WKNN (Kelly & Davis, 1991) and k-NN$_{VSM}$ (Wettschereck, 1994). The work presented by Dietterich and Wettschereck (1995) consists of feature weighting by mutual information categorized as *ignorant* methods.

In this paper, we propose two methods to investigate the effect of weight assigning to features in k-NNFP algorithm. In these methods, domain-specific knowledge is not used. These methods can be categorized according to this framework's first dimension as ignorant and feedback since homogeneity of feature projections weight setting does not use any feedback from the k-NNFP algorithm whereas the second one uses feedback from k-NNFP algorithm. These methods modify the voting mechanism of k-NNFP algorithm by incrementing the vote of the predicted class by using the feature weight. We can easily incorporate the second method into our own approach using cross-validation on a basis of single feature.

In previous work, we have introduced a new classification algorithm, called k-NNFP, based

on majority voting on individual classifications made by the projections of the training set on each feature (Akkuş & Güvenir, 1996). K-nearest neighbor algorithm is used to determine the classifications made on individual feature projections. We have compared k-NNFP algorithm with the k-NN algorithm in terms of classification accuracy and time complexity on both real-world and artificially generated datasets.

In k-NNFP algorithm, training instances are stored as their feature projections on each feature dimension. The final classification is based on a majority voting taken on the classifications made on the basis of individual feature projections by giving them equal weight for decision ($w_f = 1$ for all features). To a certain extent, the voting mechanism reduces the negative effect of irrelevant features. However, after a certain extent, this bias can be a disadvantage allowing redundant, irrelevant and other imperfect features to influence voting mechanism used. In this study, our aim was to investigate the importance of features' contribution to final classification since to assign higher weights to more relevant features increase the reliability of voting. This paper focused on the empirical evaluations of feature weighting methods proposed.

Comparison of similar algorithms highlights dissimilarities that can explain observed performance differences. Our experimental results show that weighting features in the k-NNFP algorithm improves the accuracy effectively in real-world datasets in some domains, especially for smaller k values. to be worth, especially for smaller k values. An explanation of observed performance differences is presented in section 4.

In the next section, the k-NNFP algorithm is given with its weighted version, briefly. In the subsequent section, a detailed descriptions of feature weighting methods studied are given. The forth section presents the empirical comparison of these methods on real-world datasets along with a summary of results obtained from the experiments. The last section concludes with an overview of possible extensions to the work reported here.

2 The Weighted k-NNFP Algorithm

The k-NNFP algorithm is introduced for classification based on feature projections using k nearest neighbor algorithm. Our motivation for this algorithm is that the encouraging results of classification by feature partitioning technique (Güvenir, Şirin, 1996). The basic assumption of the k-NNFP algorithm is that each feature can contribute the classification process and the majority voting provides a correct classification. The implementation of it is non-incremental. Since all feature values are treated independently, there is no need for normalization of feature values. In the learning phase, each training instance is stored 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. The k-NNFP algorithm stores the feature projections of training instances in a sorted order. Therefore, the classification of a new instance requires a simple search of the nearest training instance values on each feature. The classification of an instance is based on a majority voting taken on the classifications made on the basis of individual

feature projections. In general, with the majority voting for final classification, the effect of irrelevant features may be reduced. On the other hand, each feature can contribute to the classification by its relevance. So, if we place weights on features before voting, this can supply more accurate result for final class by reflecting each feature's relevance in the classification. The weighted 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 as follows:

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

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 multiplying the weight of that feature by 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.

3 Feature Weight Learning Methods

Two feature weighting methods are proposed for k-NNFP algorithm to see the effect of irrelevant, and relevant features with varing relevancies. In Section 3.1, the first method, called homogenity, is discussed and the subsequent section presents the second method which is based on the individual accuracies of the features .

3.1 Weight Learning Based on Homogenity of Projections

The main observation for this method comes from the k-NNFP algorithm itself. The basic assumption of the k-NNFP algorithm is that closer values on a feature dimension are of the same class. Namely, distribution of trainig instances on a feature dimension is homogenious. That is, the projections of all training instances of the same class are grouped together. Therefore, the total number of consequent values of a same class can give a measure for its relevancy for classification prediction. In k-NNFP algorithm, all seen feature values are stored in memory as sorted. We can determine the weight of a feature as follows: Initially, a count is set to 0, then for all sorted feature values, if the consequent feature value's class is same as the previous one, then count is incremented. Therefore, feature weight can be found by

```
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] + weight[f] * 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 weighted k-NNFP Algorithm.

dividing that count by the total number of distinct feature values on that feature. This can be summarized as follows:

$$w_f = \frac{\sum_{v=1}^{V_f} \alpha(f,v)}{V_f} \tag{2}$$

$$\alpha(f,v) = \begin{cases} 1 & \text{if} fC_{v_f} = C_{(v+1)_f} \\ 0 & otherwise \end{cases} \tag{3}$$

All feature weights are computed using this formula. Here C_{v_f} denotes the the class label of v_{th} value on feature dimension f, and V_f denotes the number of distinct values on feature dimension f. This equation always gives a value for a feature between 0 and 1, so it can be the probability of that feature in classification. This is incorporated with feature weights to allow that more important features contribute to classification process effectively.

We have performed some experiments assigning weight to features by this method on real-world taken from the UCI repository (Murphy, 1995). The results of this experiments will be presented in Section 4.

Table 1: Comparison on some real-world datasets.

Data Set:	bcancerw	cleveland	glass	hungarian	ionosphere	iris	liver	wine
No. of Instances	273	303	214	294	351	150	345	178
No. of Features	9	13	9	13	34	4	6	13
No. of Classes	2	2	6	2	2	3	2	3
No. of Missing values	16	6	0	784	0	0	0	0
No. of Linear features	9	5	9	5	34	4	6	13

3.2 Weight Learning Based on Individual Accuracies of Features

The second method is motivated by the work of Holte (1993) since each feature is processed independently in k-NNFP algorithm. Holte (1993) reports the results of experiments measuring the performance of very simple rules on the datasets commonly used in machine learning research. The specific kind of rules studied is called *1-rules*, which classify an object on the basis of a single feature. This study motivates us to examine the classification accuracy of the k-NNFP algorithm on the basis of a single feature. Therefore, those accuracies can be used as weight of that feature since those accuracies reflect how much each feature can contribute to the final classification. However, to avoid random correct classification, we subtract the random accuracy of a feature from the individual accuracies. The random accuracy is taken as $1/No\ of\ Classes$.

In order to investigate this method, we have performed some experiments on real-world. For these experiments, feature weights are learned by running k-NNFP algorithm on the basis of a single feature by 5-way cross-validation for each feature. These estimated accuracies are used for the weights of corresponding features in k-NNFP algorithm. Since it takes feedback from k-NNFP algorithm, it can be categorized as feedback method.

4 Experiments on Real-World Datasets

In this section, we present an empirical evaluation of two weighting methods: homogeneity of feature projections and individual accuracies of features along with its comparison with its unweighted version.

The weighted versions of the k-NNFP algorithm are evaluated on some real-world datasets selected from the collection of datasets provided by the machine learning group at the University of California at Irvine (Murphy 1995). The properties of these datasets can be shown in Table 1. In this table, name of the real-world datasets are shown with the size of the dataset, number of features, number of classes, number of missing feature values, and number of linear features.

Table 2: Accuracies (%) of the k-NNFP (N) and its weighted versions using homogeneity of feature projections (H) and individual accuracies (A).

	Data Set:	bcancerw	cleveland	glass	hungarian	ionosphere	iris	liver	wine
N	k=1	94.0	67.62	57.0	**70.04**	88.04	**90.0**	50.44	79.7
H		**94.28**	67.62	**57.92**	68.7	88.32	89.98	50.42	**87.58**
A		**94.28**	**79.6**	57.0	61.52	**88.6**	89.98	**58.26**	87.0
N	k=3	94.88	72.94	61.18	75.84	**88.02**	91.34	55.68	90.96
H		**95.02**	72.92	62.14	**77.88**	**88.02**	94.02	56.52	94.36
A		**95.02**	**77.24**	**62.58**	77.18	**88.02**	**94.68**	**60.58**	**94.9**
N	k=5	**96.16**	78.88	60.72	76.16	87.46	91.3	58.26	93.24
H		96.02	80.18	59.78	**77.86**	87.18	93.32	57.96	94.38
A		**96.16**	**80.5**	**65.84**	74.78	**87.74**	**94.0**	**63.5**	**94.9**
N	k=7	**96.0**	79.52	62.58	74.8	87.74	92.0	61.46	95.48
H		94.96	79.2	63.06	**76.86**	**87.8**	93.34	61.44	**95.52**
A		95.86	**81.5**	**66.76**	74.78	86.9	**94.0**	**64.64**	95.5
N	k=9	96.14	78.52	63.04	75.8	87.44	92.0	62.04	96.62
H		**96.28**	79.18	63.06	**78.2**	**87.74**	94.02	62.32	96.62
A		**96.28**	**81.52**	**66.3**	72.74	**87.74**	**94.68**	**64.94**	**97.2**

In previous work, we have presented the performance of k-NNFP algorithm in terms of classification accuracy, the percentage of correctly classified instances over all test instances. In that study, we have chosen 5-way cross-validation for measuring the accuracy of the k-NNFP algorithm. In this study, we continue to get accuracies of weighted versions by 5-way cross-validation. Our findings emphasize that weighted versions does not improve the k-NNFP algorithm effectively in most of the real-world datasets. Langley & Sage (1994) concluded from their experiments with feature selection that a number of data sets in the UCI repository contain few or no irrelevant features.

The accuracy results of k-NNFP and its two weighted versions were observed and given in Table 2. In this table, the first row of each k value presents the k-NNFP algorithm results, the second row is the results of the homogeneity of feature projections weight learning, and finally the third row presents the results of individual accuracies of features.

These experiments showed that the none of the weight learning algorithms improved the k-NNFP algorithm on the bcancerw and ionosphere datasets significantly. This should be because all the features are equally relevant. On the cleveland, liver, iris and glass (except $k = 1$) datasets, the weights learned by the individual accuracies always performed significantly better than the others. The weight learning method based on the homogeneity performed better than the other on the hungarian dataset, except $k = 1$. There were no significant difference between the two weight learning algorithms on the wine dataset.

5 Conclusions

A version of the famous k-NN algorithm, that stores the classification knowledge as the projections of the training instances on the features, called k-NNFP algorithm, had been shown to be successful. In this paper, we have presented two methods for determining the relative weights of features for use in the k-NNFP algorithm. The first method, called homogeneity, assigns a higher weight to features on which the projections of instances of the same class are located close to each other, resulting in a homogeneous distribution. The second method, on the other hand, assigns a weight as the classification accuracy that would have been obtained if only that feature were used in the classification.

Our experiments revealed that these weighting methods assign low weights to completely irrelevant features, and high weights to relevant ones. Further, among these two weight learning algorithms, the one that is based on the individual accuracies learned weights that helped k-NNFP achieve higher accuracies. The reason for this success is due to the feedback received from the classification algorithm. We conclude that this weight learning method could be successful for other classification algorithms that use feature weights. As a further work we plan to investigate these weight learning methods on artificial datasets.

References

Aha, D. W.(1990) A Study of instance-based algorithms for supervised learning tasks: Mathematical, empirical, and psychological evaluations. Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

Aha, D. W.(1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, **36**(1), 267-287.

Aha, D. W. & Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison. In D. Aha (Ed.) *Case-Based Reasoning: Papers from the 1994 Workshop (TR WS-94-01).* Menlo Park, CA: AAAI Press.

Akkuş, A. & Güvenir, H. A. (1996). k Nearest Neighbor Classification on Feature Projections, Proceedings of the 13^{th} International Conference on Machine Learning. Lorenza Saitta (Ed.), Bari, Italy: Morgan Kaufmann. pp. 12-19.

Dasarathy, B. V., (1990). *Nearest Neighbor (NN) Norms, NN Pattern Classification Techniques.* IEEE Computer Society Press.

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

Holte, C. R. (1993). Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, *Machine Learning*, **11**:63-91.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. Proceedings of the 11^th International Conference on Machine Learning. New Brunswick, NJ: Morgan Kaufmann. pp. 293-301.

Langley, P., & Sage, S. (1994). Oblivious decision trees and abstract cases, in "Working Notes of the AAAI-94 Workshop on Cased-Based Reasoning", AAAI Press, Seattle, pp. 113-117.

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.

Salzberg,S. (1991). A nearest hyperectangle learning method. *Machine Learning, 6*, 251-276.

Skalak, D. B. (1994). Prototype and feature selection by sampling and random mutation hill-climbing algorithms. Proceedings of the 11^th International Conference on Machine Learning. New Brunswick, NJ: Morgan Kaufmann. pp. 293-301.

Wettschereck,D. (1994). A study of Distance-Based Machine Learning Algorithms, PhD Thesis, Oregon State University.

Wettschereck, D., Aha W., D. (1995). Weighting Features, *First International Conference on Case-Based Reasoning*, pp.

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