WEIGHTED K NEAREST NEIGHBOR CLASSIFICATION ON FEATURE PROJECTIONS[1]

H. Altay Güvenir and Aynur Akkuş
Department of Computer Engineering and Information Science
Bilkent University, 06533, Ankara, Turkey
{guvenir, akkus}@cs.bilkent.edu.tr

Abstract. This paper proposes an extension to the k Nearest Neighbor algorithm on Feature Projections, called kNNFP. The kNNFP algorithm has been shown to achieve comparable accuracy with the well-known kNN algorithm. However, kNNFP algorithm has a very low time complexity compared to kNN. The extension to kNNFP introduced here assigns weights to features, therefore it is called WkNNFP, for *Weighted k Nearest Neighbor on Feature Projections*. The paper also introduces a weight learning algorithm, called SFA, for *Single Feature Accuracy*. It is based on the assumption that the weight of a feature is proportional with the accuracy that will be obtained by considering only that feature. The SFA algorithm is not specific to WkNNFP, so it can be used with many other classification algorithms. An empirical evaluation of the SFA algorithm on real-world datasets shows that it achieves an important improvement in the classification accuracy of the WkNNFP algorithm.

1. Introduction

Learning to classify objects, from a given set of classified examples, is one of the fundamental problems in machine learning. Researchers in the field have been working on designing classification algorithms that are accurate and also fast in training and classification.

One of the simplest, yet very accurate, classification methods in the literature is the well-known kNN, for *k nearest neighbor*, algorithm (Duda & Hart, 1973; Dasarathy, 1990). It is based on the assumption that examples that are close in the instance space belong to the same class. Therefore, an unseen instance should be classified as the the majority class of its k $(1 \leq k)$ nearest neighbors in the training dataset. Although the kNN algorithm is quite accurate, the time required to classify an instance is high, since the distance (or similarity) of that instance to all the instances in the training set have to be computed. Therefore, the classification time in kNN algorithm is proportional to the number of features and the number of training instances.

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

On the other hand, the kNNFP (*k Nearest Neighbors on Feature Projections*) algorithm has been proposed to achieve fast classification by storing training instances as their projections on each feature dimension separately (Akkuş & Güvenir, 1996). This allows the classification of a new instance to be made much faster than kNN algorithm, since all projected values can be stored in a data structure that allows fast search. The kNNFP algorithm first makes a set of predictions, one for each feature, using kNN algorithm on the projections on a feature. Then, the final classification of an instance is determined through a majority voting on individual classifications made by each feature.

The kNNFP algorithm assumes that all the features are equally relevant, that is, each feature has the same power in the voting. To a certain extent, the voting approach reduces the hindrance of irrelevant features. However, if there are many irrelevant features, or the relevancies of the features vary a great deal, then voting alone is not sufficient.

This paper presents an extension to the kNNFP algorithm that incorporates weights for features, and a heuristic approach to learn these weights. The resulting algorithm is called WkNNFP. The weight learning approach introduced here is called SFA for *Single Feature Accuracy*, since it determines the weight of a feature as the accuracy obtained by using that feature alone. This approach is not specific to WkNNFP; it can be used with other classification algorithms, as well.

The next section gives an overview of the previous work on learning feature weights. The third section introduces the WkNNFP Algorithm. Then, the SFA feature weight learning algorithm is described. The fifth Section gives the results of our experiments to compare kNNFP and WkNNFP algorithms on some real-world datasets. The last section concludes with some remarks on both WkNNFP and SFA algorithms.

2. Previous Work on Learning Feature Weights

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 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) or feature selection (Langley & Sage, 1994; Skalak, 1994).

The kNN is the basis for many classification algorithms, where all training instances are stored in the memory as points. An instance is classified as the majority class of its k nearest neighbors in the training set. Success of the kNN classifier depends on which instances are deemed least distant, which is determined by its distance function. However, such a distance function allows redundant, irrelevant, interacting, or noisy features to have as much effect on distance computations as other features. kNN can perform poorly when such features exists in the dataset. In order to eliminate the negative effect of such irrelevant features, feature subset selection approaches are investigated, where the space of subsets of feature sets are considered to determine the relevant and irrelevant features (John, Kohavi, Pfleger, 1994). Briefly, the induction algorithm is run on the training data with different subsets of features, using usually 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 kNN. On the other hand, the disadvantage of using a feature selection method is that it treats features as either completely relevant or irrelevant. However, in many real-world datasets the degree of relevance may be in between these two extremes.

In most early distance-based classification methods, a numerical value is assigned to features to modify the distance measure for each feature relevancy. Wettschereck 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. The weighting methods that receive such feedback are called *feedback* methods, and the ones which doesn't receive any feedback 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. *Continuous 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.

3. The Weighted k-NNFP Algorithm

The kNNFP algorithm was introduced for classification based on feature projections using k nearest neighbor algorithm (Akkuş & Güvenir, 1996). Our motivation for kNNFP was the encouraging results of the knowledge representation technique based on feature partitions, introduced by Güvenir and Şirin (1996). The basic assumption of the kNNFP algorithm is that each feature can contribute the classification process and the majority voting provides a correct classification.

The implementation of the kNNFP algorithm 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 kNN 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 processed separately, no normalization of feature values is needed.

The kNNFP 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.

Although the majority voting for final classification can reduce the effect of irrelevant and noisy features to some extent, we wanted to investigate the effects of incorporating feature weights to the voting in this paper. So, if we multiplied the votes of each feature with its weight. Here, we assumed that these feature weights would be provided externally, (Section 4 will describe a simple method for learning these weights). We called the resulting algorithm Weighted kNNFP (WkNNFP, for short). The classification algorithm in WkNNFP 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 $kBag(f, t, k)$, that returns a bag of size k containing the classes of the k nearest training instances to the instance t on feature f. The distance between the values x and y on a feature dimension f is computed using $diff(f, x, y)$ metric as follows:

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

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.

4. Weight Learning based on Single Feature Accuracies

Here we propose a simple method for estimating the weights of features. This method is motivated by the work of Holte (1993) since each feature is processed independently in kNNFP algorithm. Holte 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 motivated us to examine the classification accuracy of the kNNFP algorithm on the basis of a single feature. Therefore, those accuracies can be used directly as weight of the corresponding feature, since those accuracies reflect how much each feature can contribute to the final classification. We call this weight learning algorithm SFA, for *Single Feature Accuracy*.

The SFA algorithm is also efficient interms of its time complexity. It has been shown that the training time complexity of kNNFP algorithm is $O(n \cdot m$ḑ$ot \log m)$, and time complexity for testing an instance is $O(n \cdot \log m)$, where n is the number of features and m is the number of training instances (Akkuş & Güvenir, 1996). Therefore, the execution of one fold in cross-validatation in SFA algorithm has the complexity of $O(m$ḑ$ot \log m)$, since only one feature is used in each fold. The time complexity of learning the weight of one feature is $O(f \cdot m \cdot \log m)$, where f is the number of folds in cross-validation. Finally, the time complexity of SFA is $O(n \cdot f \cdot m \cdot \log m)$.

In order to investigate the effects on accuracy, we have performed some experiments on real-world datasets. For these experiments, feature weights are learned by running kNNFP algorithm

```
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 kNNFP Algorithm.

on the basis of a single feature by 10-fold cross-validation for each feature. These estimated accuracies are used as the weights of corresponding features in kNNFP algorithm. Since it takes feedback from kNNFP algorithm, it can be categorized as feedback method.

5. Experiments on Real-World Datasets

In this section, we present an empirical evaluation of the SFA weight learning algorithm on the accuracy of the WkNNFP algorithm. In order to evaluate the effect of SFA, we first ran the WkNNFP on a dataset without any feature weights, that is treating all features equally. Then we used the SFA algorithm to learn feature weights, and run the WkNNFP again with these learned weights. We repeated this for values of k from 1 to 10, in order to see also the effect of k. The SFA algorithm used the same WkNNFP algorithm, that is the same k value is used in weight learning and comparisons.

The datasets used in the experiments were 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 are given 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.

The accuracy results of WkNNFP with and without feature weights are given in Table 2. In this table, the first row of each k value presents the accuracy of the WkNNFP algorithm with equal feature weigths, while the second row shows the accuracy obtained by WkNNFP using

Table 1: Comparison on some real-world datasets.

Data Set:	cleveland	glass	horse	hungarian	iris	liver	sonar	wine
No. of Instances	303	214	368	294	150	345	208	178
No. of Features	13	9	22	13	4	6	60	13
No. of Classes	2	6	2	2	3	2	2	3
No. of Missing values	6	0	1927	784	0	0	0	0
No. of Linear features	5	9	7	5	4	6	60	13

Table 2: Accuracies (%) of the WkNNFP. (U) Unweighted, (W) with weights learned by SFA.

	Data Set:	cleveland	glass	horse	hungarian	iris	liver	sonar	wine
U	k=1	63.5	50.8	69.3	67.2	86.0	51.8	57.7	85.8
W		63.5	52.7	70.9	70.0	89.3	52.7	61.1	88.7
U	k=2	64.6	60.6	70.9	68.7	89.3	51.8	63.9	89.7
W		68.9	63.0	73.1	70.1	90.7	53.0	66.3	92.0
U	k=3	69.9	62.5	69.0	73.1	90.7	54.4	65.5	92.0
W		70.5	63.0	71.2	73.8	94.0	55.9	67.8	94.3
U	k=4	70.2	60.2	67.9	71.1	90.7	52.9	64.1	91.4
W		72.2	61.6	69.0	72.8	94.0	53.8	65.0	94.8
U	k=5	72.5	63.5	70.1	73.1	92.7	55.8	63.1	93.7
W		72.9	63.9	70.9	73.8	93.3	57.3	67.9	94.9
U	k=6	73.8	63.1	70.9	73.1	91.3	59.0	67.9	94.3
W		75.1	63.4	71.2	72.8	92.7	60.5	68.9	96.1
U	k=7	74.5	68.5	70.4	72.7	91.3	61.6	67.9	97.1
W		76.9	68.1	72.8	73.1	93.3	61.3	69.8	96.0
U	k=8	74.5	65.2	69.6	73.7	93.3	58.7	67.4	96.6
W		75.5	66.2	72.3	73.7	94.7	59.6	69.3	97.2
U	k=9	76.2	66.3	70.9	74.1	95.3	59.0	66.4	96.6
W		78.5	67.7	73.1	74.4	96.0	60.1	68.8	97.7
U	k=10	76.2	64.4	70.9	73.1	94.7	60.1	67.8	96.0
W		78.2	67.2	72.3	73.1	94.7	62.7	70.3	97.2

the feature weights learned by the SFA algorithm. The accuracy is measured using the 10-fold cross-validation technique. That is, the whole dataset is partitioned into 10 subsets. The nine of the subsets form the training set, and the last one is used as the test set. Therefore each instance in the dataset is used as training instance for nine times and as test instance for only once.

The experiments on these datasets indicate that the SFA method can increase the accuracy of the WkNNFP algorithm on the average 2 percentage points.

We also observe that the accuracy of both kNNFP and WkNNFP algorithms increase with the increasing value of k. However, no relation between the value of k and accuracy is observed.

6. Conclusions

A version of the well-known kNN algorithm that stores the classification knowledge as the projections of the training instances on the features, called kNNFP algorithm, had been shown to be successful on most real-world datasets in UCI-repository. In this paper, we have presented an extension of kNNFP that incorporated feature weights, called WkNNFP. We also gave a simple heuristic approach for learning relative weights for feature. This weight learning technique, called SFA, assigns a weight to a feature as the classification accuracy that would have been obtained if only that feature were used in the classification.

Our experiments revealed that this weight learning method assigns low weights to completely irrelevant features, and high weights to relevant ones, as expected. Further, WkNNFP can achieve higher accuracies using these weights learned by SFA algorithm in real-world dataset. The reason for this success is due to the feedback received from the classification algorithm. We can 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.

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

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.

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. (1994). "Oblivious decision trees and abstract cases," *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*, Lisbon, Portugal: Springer-Verlag. pp. 347-358

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