Genetic Algorithms to Learn Feature Weights for the Nearest Neighbor Algorithm

Gülşen Demiröz and **H. Altay Güvenir**
Dept. of Computer Engr. and Info. Sci.
Bilkent University, 06533 Ankara, Turkey
{demiroz, guvenir}@cs.bilkent.edu.tr

BU-CEIS-9612

Abstract

In this paper we use genetic algorithms to learn feature weights for the *Nearest Neighbor* classification algorithm. We represent feature weights as real values in [0..1] and their sum is 1. A new crossover operation, called *continuous uniform crossover*, is introduced where the legality of chromosomes is preserved after the crossover operation. This new crossover operation is compared with three other crossover operations—*one-point crossover*, *two-point crossover*, and *uniform crossover*—all of which require normalization after the crossover operation. Four genetic algorithms using each of these four crossover operations are implemented. Each genetic algorithm tries to learn feature weights to improve the classification accuracy of the Nearest Neighbor algorithm. Then the learned weights are assigned to features to be used in distance calculations of the *Weighted Nearest Neighbor* classification algorithm and the resulting classification accuracies in our datasets are compared.

1 Introduction

In recent years, instance-based learning algorithms (Dasarathy 1990, Aha et al. 1991, Wettschereck 1994) have been shown to work as well as other sophisticated machine learning methods despite their simplicity. One of the most common instance-based scheme is the *Nearest Neighbor* (NN) algorithm which classifies new instances depending on some distance measure between instances. The classical *Nearest Neighbor* classification algorithm treats all features as equivalent, thus performs poorly in the presence of irrelevant, weakly relevant, and/or noisy features.

To reduce the impact of irrelevant and weakly relevant features and to increase the impact of the strongly relevant features on the distance measure, several feature weighting methods have been proposed. In Wettschereck and Aha (1995), a five-dimensional framework that categorizes automated weight-learning methods is introduced. The first dimension, called *feedback* dimension, concerns whether the feature weighting method receives feedback from the induction algorithm trying to be improved. The methods that receive feedback are called *feedback* methods and the others that do not receive any feedback are

called *ignorant* methods. In Kohavi et al. (1995), feedback methods are called as *wrapper* methods. One group of feedback methods, called *incremental hill-climbers* (Wettschereck and Aha, 1995), modify feature weights incrementally to increase similarity between a test instance and nearby training instances in the same class, and to decrease its similarity with nearby training instances in other classes. IB4 (Aha, 1992), EACH's weighting method (Salzberg, 1991), and CFP's weighting method (Güvenir and Şirin, 1996) are examples of incremental hill-climbers. The other group of feedback methods, called *continuous optimizers* (Wettschereck and Aha, 1995), iteratively update feature weights using only training instances. GA-WKNN (Kelly and Davis, 1991), GA-CFP (Güvenir and Şirin, 1993), and $k - NN_{VSM}$ (Wettschereck, 1994) are examples of continuous optimizers.

In this paper, we present a new weight learning algorithm applied to the nearest neighbor classification algorithm which can be categorized into continuous optimizers under feedback weighting methods defined in Wettschereck and Aha (1995)'s framework. In this approach, we use a genetic algorithm to search in the space of feature weights.

Genetic algorithms are search and optimization algorithms based on the mechanics of natural selection and natural genetics. They have been developed by John Holland (Holland, 1975) and have drawn increasing interest. In our work, a genetic algorithm is designed to optimize feature weights used in the distance computation of NN. A new crossover operator (called *continuous uniform crossover*) is proposed, such that it produces valid chromosomes given that the parent chromosomes are valid. Unlike GA-WKNN (Kelly and Davis, 1991), our genetic algorithm does not use many genetic operators, instead only crossover operation is used. The aim for doing so is to evaluate the continuous uniform crossover alone and for this purpose we compared it by three common crossover operators: one-point crossover, two-point crossover, and uniform crossover.

The experimental results show that assigning weights to features have improved the accuracy of the Nearest Neighbor algorithm in real-world datasets. The amount of improvement in accuracy seems to be worth for the effort of learning the weights. We observe that the genetic algorithm using our new crossover operator, continuous uniform crossover, learns weights which improve the NN algorithm more than the genetic algorithms using other crossover operators in most of the domains.

In the next section, a brief introduction to the NN algorithm and its weighted version is given. In the third section, the use of genetic algorithms in weight learning and continuous uniform crossover is described. Section 4, presents the experimental results comparing the genetic algorithms using four different crossover operators. Finally, the last section will summarize the conclusions.

2 The Nearest Neighbor Algorithm

The Nearest Neighbor classification algorithm is based on the assumption that the nearest neighbor of an unclassified instance in the training dataset should belong to the same class as that instance. To measure the distance between two instances, several distance metrics have been proposed by Salzberg (1991), of which the Euclidean distance metric is the most common. Given two instances $x =< x_1, x_2, \ldots, x_n >$ and $y =< y_1, y_2, \ldots, y_n >$ on an n dimensional space, the Euclidean distance between x and y is computed as:

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

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

Here w_f denotes the weight for feature f and for all features $w_f = 1$ in NN and $diff(f,x,y)$ denotes the difference between the values of instances x, and y on feature f. Feature values are normalized to ensure that they have the same range.

The NN algorithm is a specialization of the k-NN classification algorithm where the classification of a new instance is done by a majority voting among its k ($k \geq 1$) nearest neighbors.

Nearest Neighbor algorithms are reported to be effective on many real-world domains, although they are simple and require no training time. Unfortunately, they are less effective when the dataset contains irrelevant, weakly relevant, and/or noisy features. The reason is that all the features have equal impact on the distance computations (1). To improve NN, weights can be assigned to features, w_f in (1), and there are several variants of Weighted Nearest Neighbor (WNN) which try to assign higher weights to more relevant features and lower weights to less relevant or irrelevant features such as GA-WKNN (Kelly and Davis, 1991) and $k - NN_{VSM}$ (Wettschereck, 1994).

3 Weight Learning Genetic Algorithms for NN

The genetic algorithm is used to assign real-valued weights to features in order to reduce the impact of irrelevant and weakly relevant features and increase the impact of strongly relevant features on the classification of the new instance in NN. Among the weighting methods, which receive feedback from the induction algorithm trying to be improved, continuous optimizers are more robust than incremental hill-climbers (Wettschereck and Aha, 1995) which are sensitive to the presentation order of training instances. Weight learning genetic algorithms are continuous optimizer feedback weight learning algorithms (Wettschereck and Aha, 1995) which search for the optimum feature weights that maximizes the classification accuracy of WNN. Four weight learning genetic algorithms using different crossover operators are designed and compared in this work.

In Section 3.1, the representation of the points in our search space and the fitness measure of chromosomes will be described. In Section 3.2, the crossover operators, mainly the continuous uniform crossover is explained.

3.1 Representation and Fitness Measure

Chromosomes are vectors of positive real-valued feature weights between 0 and 1 inclusive such that they sum up to 1. The length of a chromosome is equal to the number of features. The initial population is randomly generated taking care of alleles being between 0 and 1 and the sum of alleles being equal to 1.

The fitness function used to evaluate the chromosomes is the cube of the classification accuracy of the *Weighted Nearest Neighbor* algorithm using the feature weights encoded in the chromosome in (1). In order to get the classification accuracy, five-way cross-validation is used. That is, the whole dataset is partitioned into 5 subsets. One of the subsets is used as the training set, and the other subset is used as the test set, and this process is repeated 5 times once for each subset being the test set. Classification accuracy is the average of these 5 runs.

3.2 Genetic Operators

The main operators of a genetic algorithm are reproduction, crossover, and mutation. We use only reproduction and crossover operators and not the mutation operator in order to see the ability of several crossover operators alone.

Although the initial population is setup in a way that all chromosomes represent legal codings (each allele is between 0 and 1 inclusive and the sum of the alleles is 1), standard crossover operations can result in illegal codings. Here a new form of uniform crossover, called continuous uniform crossover (CUCO) is proposed, that guarantees the legality of the offsprings.

Given two chromosomes $x =< x_1, x_2, \ldots, x_n >$ and $y =< y_1, y_2, \ldots, y_n >$ such that n is the number of genes in a chromosome, the offsprings are defined as $x' =< x'_1, x'_2, \ldots, x'_n >$ and $y' =< y'_1, y'_2, \ldots, y'_n >$, where

$$x'_i = s \times x_i + (1 - s) \times y_i \tag{3}$$
$$y'_i = s \times y_i + (1 - s) \times x_i \tag{4}$$

Here s, called *stride*, is constant through a single crossover operation. This crossover operation guarantees that the sum of the alleles of an offspring is still 1 given that the sum of the parents' alleles is 1. Given that $\sum_{i=1}^{n} x_i = 1$ and $\sum_{i=1}^{n} y_i = 1$,

$$\sum_{i=1}^{n} x'_i = s \times \sum_{i=1}^{n} x_i + (1 - s) \times \sum_{i=1}^{n} y_i = s + (1 - s) = 1$$

and similarly for the sum of y_i.

In order to preserve further legality of the offsprings such as the alleles being between 0 and 1, the choice of the *stride* s should be restricted. Since, for any value of s, the sum of the alleles is 1, it will be guaranteed that each allele would be less than 1 as long as each allele is ensured to be greater than 0. In order to have $x'_i \geq 0$ and $y'_i \geq 0$ for all i $(1 \leq i \leq n)$,

$$\frac{-x_i}{y_i - x_i} \leq s \leq \frac{y_i}{y_i - x_i}$$

where $y_i \geq x_i$. Each allele pair (x_i, y_i) brings an upper bound, call it $upper_i$ and a lower bound, call it $lower_i$, on s where

$$upper_i = \frac{y_i}{y_i - x_i} \geq 1,$$

$$lower_i = \frac{-x_i}{y_i - x_i} \leq 0$$

Among n upper and lower bounds we choose a global upper bound, call it *upper* and a global lower bound, call it *lower*, for the stride such that

$$upper = min_i(\frac{y_i}{y_i - x_i})$$

$$lower = max_i(\frac{-x_i}{y_i - x_i})$$

The stride s should be in the range [*lower*..*upper*] to preserve the legality of the offsprings.

There is a symmetry between two strides, s_1 and s_2, when $s_1 + s_2 = 1$. Here symmetry means that continuous uniform crossover on two given parents produces same offsprings in both use of the strides s_1 and s_2. For example, the strides $s_1 = 0.5$ and $s_2 = 0.5$, $s_1 = -2$ and $s_2 = 3$ are symmetric. There is always a symmetric stride value for each value of strides. Further, the symmetric stride of a given stride greater (less) than 0.5 is less (greater) than 0.5. So we can discard the stride values less than 0.5 in the global range for s, since there is a stride in the range [0.5..*upper*] symmetric to the stride in the range [*lower*..0.5].

If we think of some special values of s, we see that when $s = 1$ or $s = 0$, the offsprings are same as their parents. When $s = 0.5$, the alleles of the offsprings are the average of the alleles of their parents. When $s > 1$ or $s < 0$, the alleles of the offsprings are greater than the maximum of the corresponding allele pair of their parents and less than the minimum of the corresponding allele pair of their parents. Hence, the stride being greater than 1 or less than 0 enables the crossover operation to try the outer values of the alleles of the parents. Because of the symmetry we have discarded the strides less than 0.5, having a range [0.5..*upper*] for s. When *upper* value is greater than 1, we restrict s to be in the range [1..*upper*] in order to try the outer values of the parent alleles. But when upper bound is equal to 1, s is restricted to be in the range [0.5..1].

Let us see an example CUCO operation. Let the parent chromosomes be $x =< 0.4, 0.6 >$, and $y =< 0.5, 0.5 >$. The first allele pair ($x_1 = 0.4$, $y_1 = 0.5$) requires $-4 \leq s \leq 5$, and the second allele pair ($y_2 = 0.6$, $x_2 = 0.5$) requires $-5 \leq s \leq 6$. From the lower bounds we choose the maximum (i.e. -4) and from the upper bounds we choose the minimum (i.e. 5). We get the requirement for s to be in the range [-4..5]. By using the symmetry property, s is restricted to be in the range [0.5..5]. Since *upper* > 1, s is randomly selected from the range [1..5].

Let the randomly chosen value of s be 3:

$$x'_1 = 3 \times x_1 + (1 - 3) \times y_1 = 1.2 - 1.0 = 0.2 \quad (5)$$

$$x'_2 = 3 \times x_2 + (1 - 3) \times y_2 = 1.8 - 1.0 = 0.8 \quad (6)$$

Hence, one of the offsprings became $x' =< 0.2, 0.8 >$ where $\sum_{i=1}^{2} x'_i = 1$. The other offspring alleles are as follows:

$$y'_1 = 3 \times y_1 + (1 - 3) \times x_1 = 1.5 - 0.8 = 0.7 \quad (7)$$

$$y'_2 = 3 \times y_2 + (1 - 3) \times x_2 = 1.5 - 1.2 = 0.3 \quad (8)$$

Hence, the other offspring became $y' =< 0.7, 0.3 >$ where $\sum_{i=1}^{2} y'_i = 1$. Note that the symmetric stride $s = -2$ would give the offsprings $x' =< 0.7, 0.3 >$ and $y' =< 0.2, 0.8 >$ which are the same as the offsprings produced by using $s = 3$.

Three other operators that are chosen to compare with CUCO are *one-point crossover* (1PCO), *two-point crossover* (2PCO), and *uniform crossover* (UCO). *One-point crossover* cuts the parent chromosomes at one site randomly and exchanges their cross parts. *Two-point crossover* cuts the parent chromosomes at two random sites and the middle part between the sites are exchanged. *Uniform crossover* swaps two corresponding alleles of parent chromosomes with some probability to get offsprings. All of these three crossover operations require a normalization of the offsprings.

In addition to the crossover operator, the *elitism* operator is used, that is the fittest chromosome in the current generation is always copied to the next generation.

4 Experimental Results

One of the purposes of the experiments was to compare each of the four weighted nearest neighbor algorithms listed as follows:

1. Weighted NN learning feature weights using a genetic algorithm which uses one-point crossover (1PCO-WNN)
2. Weighted NN learning feature weights using a genetic algorithm which uses two-point crossover (2PCO-WNN)
3. Weighted NN learning feature weights using a genetic algorithm which uses uniform crossover (UCO-WNN)
4. Weighted NN learning feature weights using a genetic algorithm which uses continuous uniform crossover (CUCO-WNN)

with the Nearest Neighbor which does not assign weights to features. The other purpose was to compare each of the above four weighted versions of NN (i.e. 1PCO-WNN, 2PCO-WNN, UCO-WNN, CUCO-WNN) with each other.

In the following subsections, we will give the experimental parameters, the datasets used, and the results with some discussion.

4.1 Experiment Parameters and Datasets Used

The representation of chromosomes was defined in Section 3.1. In each run of the genetic algorithms a population of size 100 was used. The runs terminated after 200 generations. The fitness proportionate roulette wheel selection in reproduction was used. In each of the four genetic algorithms using different crossover operators, the crossover probability was given as 0.8. The swap probability for uniform crossover was given as 0.5.

The algorithms NN, 1PCO-WNN, 2PCO-WNN, UCO-WNN, CUCO-WNN have been tested using four of the real-world datasets from the collection of datasets provided by the machine learning group at the University of California at Irvine (Murphy 1995). The properties of the four datasets chosen is given in Table 1. Since Nearest Neighbor algorithms use geometric distance between feature values, we preferred the datasets with only continuous features.

Table 1: Properties of the real-world datasets used in the comparisons.

Data Set:	iris	glass	wine	liver
No. of Instances	150	214	178	345
No. of Features	4	9	13	6
No. of Classes	3	6	3	2
No. of Missing values	0	0	0	0

Table 2: Accuracy(%) of NN, 1PCO-WNN, 2PCO-WNN, UCO-WNN, and CUCO-WNN obtained by 5 way cross-validation on four real-world datasets

Data Set:	iris	glass	wine	liver
NN	93.98	68.66	94.4	63.48
1PCO-WNN	95.34	85.96	99.44	71.9
2PCO-WNN	96	84.10	99.44	69.3
UCO-WNN	95.34	85.5	100	68.42
CUCO-WNN	97.34	86.86	98.86	72.2

4.2 Results and Discussion

The experiments have been done on four algorithms listed in the beginning of Section 4. The fitness measure used to evaluate a feature weight set in each of the genetic algorithms was the classification accuracy which is measured by 5-way cross-validation explained in detail in Section 3.1. For this reason we compare the algorithms according to their accuracy measured by 5-way cross-validation in Table 2.

The accuracies in Table 2 show that in all of the four datasets used, weighted versions of nearest neighbor algorithm outperforms unweighted version of the nearest neighbor algorithm. These results indicate that assigning different weights to features in all these domains improves the classification accuracy of the nearest neighbor algorithm. Another important observation from the experiments is that CUCO-WNN generally has higher accuracies than other three weighted nearest neighbor algorithms.

In the iris domain CUCO-WNN has the highest classification accuracy, and UCO-WNN and 1PCO-WNN have the worst accuracy. In the glass domain CUCO-WNN again has the highest classification accuracy, and 2PCO-WNN has the worst accuracy. The accuracy improvement gained by assigning weights to features is very significant in the glass domain where the smallest improvement is 15.44% with the learned weights $< 0.375, 0.083, 0.12, 0.007, 0.018, 0.207, 0.186, 0, 0.004 >$ respectively. In the liver domain, CUCO-WNN again has the highest classification accuracy, and UCO-WNN has the lowest classification accuracy. Only in the wine domain we observed that UCO-WNN has the highest classification accuracy, 1PCO-WNN and 2PCO-WNN follow it, and CUCO-WNN has the lowest classification accuracy; however the differences in the accuracies are insignificant. In Figure 1 the comparison of the four algorithms on four real-world datasets for increasing number of generations is shown.

Experiments have shown that CUCO-WNN generally outperforms other three weighted nearest neighbor algorithms because CUCO-WNN learns the best feature weights by

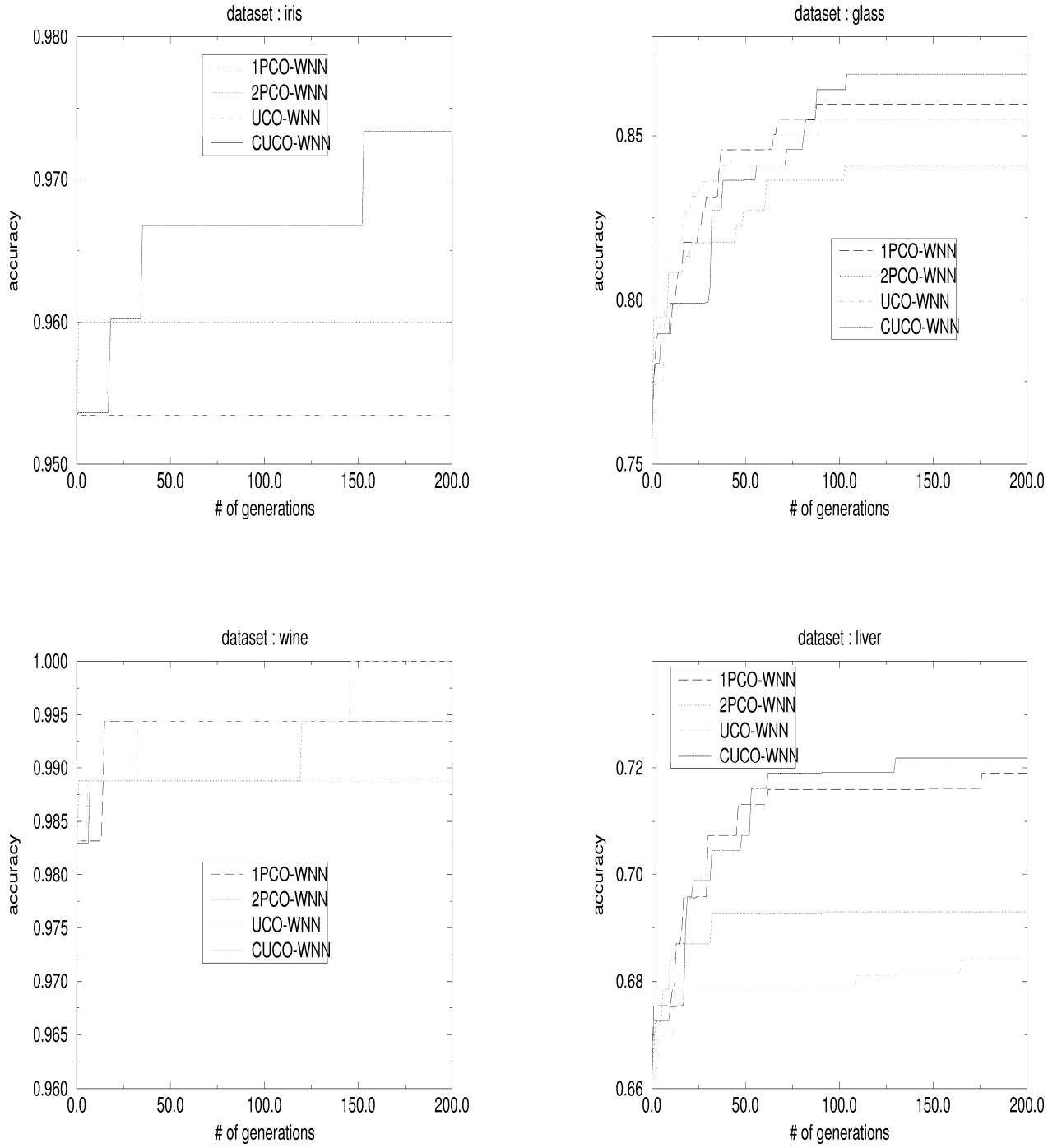

Figure 1: Comparison of 1PCO-WNN, 2PCO-WNN, UCO-WNN, and CUCO-WNN on real-world datasets for increasing number of generations. The accuracy results are obtained by 5 way cross-validation.

which the best classification accuracy is obtained. However, what we also have observed is that the classification accuracies of these four weighted nearest neighbor algorithms are close to each other.

5 Conclusions

In this paper, weight learning genetic algorithms for the Nearest Neighbor classifier are presented. These weight learning approaches receive feedback from the WNN such that the classification accuracy of WNN, given the weights encoded in a chromosome, is used as the fitness measure of that chromosome. The classification accuracies of four different genetic algorithms using either one-point crossover, two-point crossover, uniform crossover, or continuous uniform crossover are compared.

The continuous uniform crossover preserves the legality of the offspring chromosomes given that the parent chromosomes are legal. A legal chromosome has alleles greater than 0 and the sum of the alleles is 1. Other crossover operations destroy these properties during the crossover operation and therefore require normalization after the crossover.

The results have shown that all the weight learning genetic algorithms improved the unweighted version of the nearest neighbor algorithm, thus weighting features is a good idea to further improve the nearest neighbor algorithm. The comparison between the four algorithms shows that CUCO-WNN generally has higher classification accuracies than other three weighted nearest neighbor algorithms.

References

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

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.

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

Güvenir, H. A., & Şirin, İ. (1993). A Genetic Algorithm for Classification by Feature Partitioning. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, 543-548.

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

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

Kelly, J. D., & Davis,L. (1991). A Hybrid Genetic Algorithm for Classification. In *Proceedings IJCAI-91*, 645-650, Sydney, Australia.

Kohavi, R., Langley, P., & Yun, Y. (1995). Heuristic Search for feature weights in instance-based learning. (Unpublished manuscript).

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.

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

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

Wettschereck,D. & Aha, D. W. (1995). Weighting Features. *In Proceedings of the First International Conference on Case-Based Reasoning(ICCBR-95).*