ISSN (Online) : 0974-5645
ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(1), 70–78, January 2015*
DOI : 10.17485/ijst/2015/v8i1/56659

A Robust Instance Weighting Technique for Nearest Neighbor Classification in Noisy Environments

E. Parvinnia[1]*, M. R. Moosavi[2], M. Zolghadri Jahromi[2] and M. H. Sadreddini[2]

[1]Department of Computer Engineering, Shiraz Branch, Islamic Azad University, Shiraz, Iran; eparvinnia@yahoo.com
[2]Department of Computer Science and Engineering, Shiraz University, Iran

Abstract

The performance of Nearest Neighbor (NN) classifier is highly dependent on the distance (or similarity) function used to find the NN of an input test pattern. Many of the proposed algorithms try to optimize the accuracy of the NN rule using a weighted distance function. In this scheme, a weight parameter is learned for each of the training instances. The weights of training instances are used in the generalization phase to find the NN of an input test pattern. The Weighted Distance Nearest Neighbor (WDNN) algorithm attempts to maximize the leave-one-out classification rate of the training set by adjusting the weight parameters. The procedure simply leads to weights that overfit the train data, which degrades the performance of the method especially in noisy environments.

In this paper, we propose an enhanced version of WDNN, called Overfit Avoidance for WDNN (OAWDNN), that significantly outperforms the algorithm in generalization phase. The proposed method uses an early stopping approach to decrease instance weights specified by WDNN, which implicitly makes the class boundary smooth and consequently more generalized.

In order to evaluate robustness of the algorithm, class label noise is added to a variety of UCI datasets. The experimental results show the supremacy of the proposed method in generalization accuracy.

Keywords: Classification Nearest Neighbor (NN), Instance Weighting, Avoidance Overfit, Robustness, Environment Noisy

1. Introduction

The Nearest Neighbor rule is one of the simplest and most attractive pattern classification methods. The basic rationale for the NN rule is both simple and intuitive: patterns close in feature space are likely to belong to the same class. Therefore, its performance relies on the locally constant class conditional probability[1]. Nearest Neighbor classification was developed to perform discriminant analysis when reliable parametric estimates of probability densities are unknown or difficult to determine[2]. The NN classifier has many advantages over some well known methods such as decision trees and neural nets[3]. It can learn from a small set of examples and it can incrementally add new examples as they become available.

Nevertheless, nearest neighbor rule suffers from two major problems. First, the meaning of closeness various in different feature spaces. Therefore, customized distance functions should be used for each problem. Second, each query instance should be compared with all of the instances to find the nearest one. These cause scalability problems for large datasets.

The first problem could be solved by means of a weight parameter for each instance. For example in[4], a locally adaptive distance measure is used in which a heuristic method specifies the parameters of the distance measure. Many researches have been done to incorporate different kinds of weights. In these methods, the weights may be specified for each class, feature, or individual instances[5–8].

*Author for correspondence

To tackle the second problem mentioned above (i.e., low execution speed and high storage requirement), instance reduction algorithms was proposed[9, 10]. Instance reduction techniques have different objectives: removing noisy instances from the training set, removing instances far from class boundary (i.e., instances that does not affect decision boundary) or removing instances located too near to class boundaries (to increase generalization)[9, 11].

Dealing with both of the mentioned problems, has been the subject of some recent researches. In[12], a prototype reduction algorithm was proposed that simultaneously searches for a reduced subset of prototypes and a suitable local metric for these prototypes.

The Weighted Distance Nearest Neighbor (WDNN) is a novel method, proposed to tackle both mentioned problems, simultaneously[13]. In this method a weight parameter is assigned to each training instance, which is used in the weighted distance function. The learning algorithm tries to maximize the leave one out (LOO) classification accuracy of the NN rule by tuning the weights of the training instances. At the same time, this algorithm tends to decrease the weight parameters which results in zero-weight instances, and consequently can be viewed as a powerful instance reduction technique.

However, in this method the parameters are tuned such that in most cases the distance measure overfits training data. The decision boundary becomes very complex because all of the training instances are considered in adaptation of each parameter. Thus, the main limitation of WDNN is the lack of a mechanism to avoid overfitting.

One way to avoid overfitting is by controlling model complexity[14, 15]. Intuitively, a simpler model is preferred to explain the data (Occam's razor)[16, 17]. This is not applicable in WDNN model since initially model complexity (i.e., the number of parameters) depend on the amount of training data.

The convergence of a large number of parameters (i.e., instance weights) either needs an overall parameter determination method or an iterative parameter tuning. WDNN uses an iterative approach. In each iteration, the weight of one instance is determined, assuming that other parameters are fixed. To find the best weight, the classification improvement of at least one training instance is considered. Therefore, the final model normally overfit the training data and consequently for the most of applications it is not able to generalize to test data.

In the WDNN implementation, the search for the weight parameters is repeated for a fixed number of passes over the entire training set. Increasing the number of passes has a tradeoff between increasing the accuracy and cause of overfit. Randomly reordering the patterns at the end of each training epoch reduces the effect of overfit in subsequent passes, but the experiments show that in WDNN, overfit is expected even in early epochs.

Another approach to prevent train data overfit is early stopping[18, 19]. In model training (usually Neural Networks) a validation set (which is independent of the training set) is used to evaluate the termination criterion of training phase. For example the training procedure is stopped when the result is not expected to have further significant improvement on validation set. In fact, early stopping limits the used weights in the model and thus imposes regularization, effectively lowering the VC dimension[14, 20, 21].

On the other hand, regularization techniques or weight decay ensures that the estimated border is no more curved than necessary[22]. One possible form of weight learning comes from our experimental observations that an overfitted boundary mapped with regions of large curvature requires large weights.

The NN classifier is not robust in noisy environments. Noisy instances (i.e., mislabeled training data) cause misclassification of neighboring instances. Therefore, the class label noise significantly degrades the NN performance[23, 24]. Although WDNN reduces the influence of noisy training instances, by assigning small weights to them, but these instances impact the weight of many other training instances.

In this paper we propose an enhanced version of WDNN (called OAWDNN) to increase its robustness and generalization power. The proposed method has all of the advantages of WDNN over traditional NN. Moreover, its performance (i.e., classification accuracy and reduction rate) is significantly better in noisy environments.

The proposed method uses an early stopping approach to decrease instance weights specified by WDNN, which implicitly causes a smoother and consequently more generalized class boundary.

In the other words, the weight parameter in WDNN may become a large value on benefit of correct classification of few instances, while a much less value suffices for classification of majority of instances. In fact, the basic idea in OAWDNN is classification of few instances does not worth to increase the weight value, since this causes a complex decision boundary.

The rest of this paper is organized as follows. Section 2 describes the weighted similarity metric used in NN classification. In Section 3, the learning method of the weight parameters is presented. Section 4 elaborates the proposed method for overfit prevention. In Section 5, experimental results are presented. Section 6 concludes the paper.

2. The Weighted Similarity Metric

In the NN classification to identify the most similar pattern with a query instance, a similarity function is defined. The similarity measure is defined as follows:

$$\mu(X_i, X_j) = \frac{1}{d(X_i, X_j)} \quad (1)$$

Where d is a distance function, and X_i, X_j belong to the set of training instances $\{(X_i, c_i), i = 1, ..., n\}$. The feature space has d features, m classes and n training instances.

Different types of distance functions have been introduced in the literature[11]. Most popular distance function is based on the Euclidean distance that defines the distance between two instances X_i and X_j as follows:

$$d(X_i, X_j) = \sqrt{\sum_{k=1}^{d} (X_{ik} - X_{jk})^2} \quad (2)$$

The most similar pattern to a query instance Q (denoted as X_p) can be find by:

$$p = \underset{1 \le j \le n}{\operatorname{argmax}} \{\mu(Q, X_j)\} \quad (3)$$

This simple classification technique assumes that all training instances are reliable and has equal importance. The WDNN algorithm is based on the idea that some of the instances should have more influence in the classifiers decision. It accomplishes this by assigning a weight w_k to each training instance X_k. Therefore, the equation (3) should be extended to following form:

$$r = \underset{1 \le j \le n}{\operatorname{argmax}} \{w_j . \mu(Q, X_j)\} \quad (4)$$

3. Learning Weights of Training Instances

The WDNN method attempts to minimize LOO classification error rate on the given training set by specifying the weights of training instances.

Increasing the weight of an instance, assuming that the weights of all other instances are fixed, will increase its decision area and influence. In the WDNN algorithm, initially the weights of all instances are set to 1.0. The algorithm specifies the best weight w_k (where w_k is a number in the interval $[0, \infty]$) of a training instance X_k, assuming that the weights of all other instances are given and fixed[13]. Suppose that c_k = *class T* and the classification is a two class problem in which all instances with label not equal to *class T*, belongs to. *class* $\overline{T}$

The learning algorithm requires identifying instances that their correct/incorrect classifications depend on the value of w_k. In the first step, the weight of X_k is set to zero. This effectively removes the X_k from feature space and it is not used to classify any training instances. Then, the algorithm removes two groups of instances that their classification do not depend on the value of w_k:

- Correctly classified instances of *ClassT*
- Misclassified instances of *class* $\overline{T}$

The classification of remained instances depends on the value of w_k. The WDNN algorithm defines a score for each of these instances, X_t:

$$S(X_t) = \frac{\max_{1 \le j \le n} \{w_j . \mu(X_t, X_j), j \ne t\}}{\mu(X_t, X_k)} \quad (5)$$

Using any value $w_k > S(X_t)$ as the weight of X_k, makes X_k to be the nearest neighbour of X_t (i.e., X_k is the classifier of X_t). The following relation can be easily derived from (5).

$$w_k . \mu(X_t, X_k) > \max_{1 \le j \le n} \{w_j . \mu(X_t, X_j), j \ne t\} \quad (6)$$

This in turn means that using any value $w_k > S(X_t)$, pattern X_t is classified as *ClassT*. To find the optimal value of w_k, remained instances are ranked in ascending order of their scores in a list denoted by S_k.

By choosing any value of w_k between two consecutive scores (i.e., $S(X_p) < w_k < S(X_{p+1})$), all instances that their scores are smaller than w_k will be classified as *ClassT*[13].

The best value of w_k is the one minimizing LOO classification error rate. This algorithm is summarized in Figure 1.

The overall classification rate of the algorithm could be improved using several passes of weight tuning over the training data. In fact, there is a trade-off between improvement of the classification rate on training data (by applying more iterations of the algorithm) and the overfit of learned weights. This kind of overfit could be avoided by limiting the number of passes (in practice 2 to 5 iterations). Nevertheless, the experiments show that

```
1. for a number of iterations.
2. for $k = 1$ to $no.\ of\ train\ instances$ { assuming $X_k \in ClassT$ }.
3. Assign zero to weight of $X_k$ .
4. Mark examples that have $ClassT$ and classified correctly.
5. Mark examples that have $class\overline{T}$ and are misclassified.
6. Rank the score of unmarked training examples (i.e. $X_t \in X_{U,k}$ ) in ascending order using (5).
7. Choose the best value for weight of $X_k$ by using the best-weight algorithm (see Figure 2).
```

Figure 1. Algorithm for finding the weight of training instances.

the WDNN overfit is expected in early epochs. On the other hand, the overfit is caused by large values assigned to instance weights. To avoid this, a variation of WDNN algorithm is presented in the next section.

4. Overfit Avoidance

In the basic WDNN algorithm, each different threshold (i.e., between any two consecutive scores in the sorted list S_k) is considered as a candidate for w_k and consequently the accuracy is calculated. The threshold that results in the highest train accuracy is selected as the weight of instance X_k (i.e., w_k). In the proposed technique, any candidate weight less than WDNN threshold is investigated by considering its impact on the expected overfit. Parameter w_k may become a large value on benefit of correct classification of few instances (located near decision boundary of X_k), while a small value of w_k suffices for classification of majority of instances. Therefore, the generalization of classifier is improved. Weight decreasing will increase the chance of having large margin classifier which in turn results in a simplified decision boundary.

On the other hand, in the original version of WDNN, a non-zero weight is assigned to the instance if it can improve the LOO classification of at least one training instance. This increases the complexity of the boundary. Here again, the weight decreasing technique is useful: An instance should be removed if its weight improves the classification of few instances.

In the proposed algorithm shown in Figure 2, we consider a minimum value for the weight of X_k. This weight should improve the overall classification of at least G training instances. This means that the search for best threshold is limited, both from the beginning and the end of the list $X_{U,k,}$ using the G parameter and the original threshold of WDNN, respectively.

In fact, a few instances placed in the beginning of the list $X_{U,k}$ should not be considered. This is achieved by lines 11 and 12 of the algorithm. In line 12 the list element, q, is found based on the G parameter. Obviously, the parameter G should be a function of the number of instances in the training set rather than using a fixed value for all datasets. In the experiments, we used G equal to 9 percent of the number of training data. In fact, the WDNN algorithm is a special case where $G = 0$. The parameter G decreases the number of prototypes (i.e., having non-zero weight). When dealing with noisy datasets (i.e., include mislabeled data), it is expected that it achieves better results by assigning zero weight to noisy instances.

The OAWDNN algorithm, in the first step finds the original WDNN threshold (lines 5-10). Suppose that the best threshold to classify training instances is between two consecutive patterns $X_{\text{best-train-index}}$ and $X_{\text{best-train-index+1}}$ in the list $X_{U,k}$.

Afterwards, the OAWDNN algorithm finds the best weight, in the list interval from element q to *best_train_index*. This interval is used in the main loop of the algorithm started in line 15. In this loop, the algorithm revisit the elements of list $X_{U,k}$ to find the first element before $X_{\text{best-train-index}}$ holding the condition $(1 - S(X_p)/best\_train\_index) > a$.

Using this criteria we seek to find a significantly smaller value for w_k that could improve generalization and meanwhile results in correct classification of the majority of instances in the list $X_{U,k}$.

This condition is illustrated by an example. Consider the WDNN algorithm is used to find the best weight of a training instance X_k. To do this, the patterns scores are sorted in Table 1. In this example, X_k is effective in classification of 9 patterns (i.e. $|X_{u,k}| = 9$). Since the marked instances are removed from the list, placing the split point after an instance X_j will cause a change in the classification of X_j. In other words, this change in the threshold, either cause a misclassified instance X_j to be classified correctly (signed "+" in Table 1), or a correctly classified instance X_j to be misclassified (signed "-"

```
**Inputs:** $X_k$, $X_{U,k}$, $S$
$X_k$: the pattern for which the weight parameter should be determined
$X_{U,k}$: unmarked patterns, assume that $X_t$ and $X_{t+1}$ are two successive patterns in the ranked list of $\underline{X_{U,k}}$
$S$: list of scores for unmarked patterns
**Output:** $w_k$
$w_k$: the best weight for $X_k$
1. current = accuracy (leave one out classification rate) corresponding to the $w_k = 0$
2. optimum = current
3. best-train-index = 0
4. best_train_th = 0
     {assume that $S(X_{last})$ is the score of the last pattern in the ranked list and $\tau$ is a very small
      positive number}
5. for each different threshold $th = (S(X_t)+S(X_{t+1}))/2$, and $th = ( S(X_{last}) + \tau )$
6. current = train accuracy assuming $w_k = th$
7. if current > optimum then
8. optimum = current
9. best-train-index = t
10. best_train_th = th
11. $G = 0.09 \times$ (no of training data in ClassT)
     {Determine parameter G according to number of instances in ClassT}
12. Find first instance $X_q \in X_{U,k}$ such that assuming $w_k = S(X_q) + \tau$ improves the classification of at least
G patterns
13. optimum = 0
14. new-weight = 0
15. for each p in [q.. best-train-index] {p is a train index}
16. if $(1 - S(X_p) / best\_train\_index) > \alpha$
17. current = train accuracy assuming $w_k = ( S(X_p)+S(X_{p+1}) ) / 2$
18. if current > optimum then
19. optimum = current
20. new-weight = $(S(X_p)+S(X_{p+1}) ) / 2$
21. return $w_k$ = new-weight
```

Figure 2. AOWDNN algorithm for finding a weight for an instance.

Table 1. An example for explaining AOWDNN algorithm

Example no.	Impact on Classification	Score List	1- (*score*/*best_train_th*)	overfit criterion
1	+	0.20	0.753086	TRUE
2	+	0.20	0.753086	TRUE
3	-	0.25	0.691358	TRUE
4	-	0.30	0.62963	TRUE
5	+	0.35	0.567901	TRUE
6	+	0.76	0.061728	FALSE
7	+	0.80	0.012346	FALSE
8	-	0.82	–0.01235	FALSE
9	+	0.90	–0.11111	FALSE

in Table 1). The best threshold (i.e., the split point with minimum classification error) is between instances 7 and 8, which results in correct classification of 5 instances and misclassification of 2 others.

In the example of Table 1, the best weight for w_k is 0.81 (i.e., average value of $S(X_7)$ and $S(X_8)$). However, improving the classification of few instances (i.e., instances 6 and 7) does not worth considering. Since increasing the weight of X_k may cause a complex decision boundary. It is apparent from the sorted list that the greatest difference between scores of two consecutive examples is between instances 5 and 6.

Overfit criterion (used in line 16 of overfit avoidance algorithm presented in Figure 2) is considered to find which instances increased the threshold more than expected. In this example, supposing the overfit parameter α is 0.1, the X_k weight is significantly decreased to 0.56. In Table 1, the horizontal dashed lines are used to depict the threshold displacement caused by overfit avoidance algorithm.

Normally, the parameter α is a very small positive value and is proportional to the amount of noise. An easy way to find out a proper value for α is a cross-validation technique. In this technique, the dataset is partitioned into three parts, training set, validation set and test set. The validation set is used to get an unbiased estimate of the generalization (i.e., to estimate the value of α). To determine the value of a parameter, different values for the parameter is used in the learning phase and then the best one (i.e., provides the best result on the validation data) is chosen.

5. Experimental Results

The effectiveness of the AOWDNN algorithm has been evaluated through two different types of experiments. In the first one, a 2-dimensional dataset was generated to illustrate the decision boundaries created by the WDNN and AOWDNN algorithms. In the second, a number of UCI datasets is used to evaluate the classification accuracy of the algorithm. The aim of these experiments is to show the robustness and generalization power of AOWDNN in noisy environments.

4.1 Artificial Data

An artificial 2-class dataset was generated in a 2-dimensional feature space bounded to the region of ^{0,1} in each dimension. The instances are generated from a Gaussian distribution for each of the classes.

Figure 3(a), (b) show the training data points and the selected prototypes by WDNN and AOWDNN algorithms, respectively. In this figure the straight line shows the optimal class boundary and the selected prototypes are denoted with star. Using these weighted prototypes, the decision boundary between two classes is also shown in the figures. As seen in (a) the WDNN decision boundary is very complex and curved on behalf of few instances.

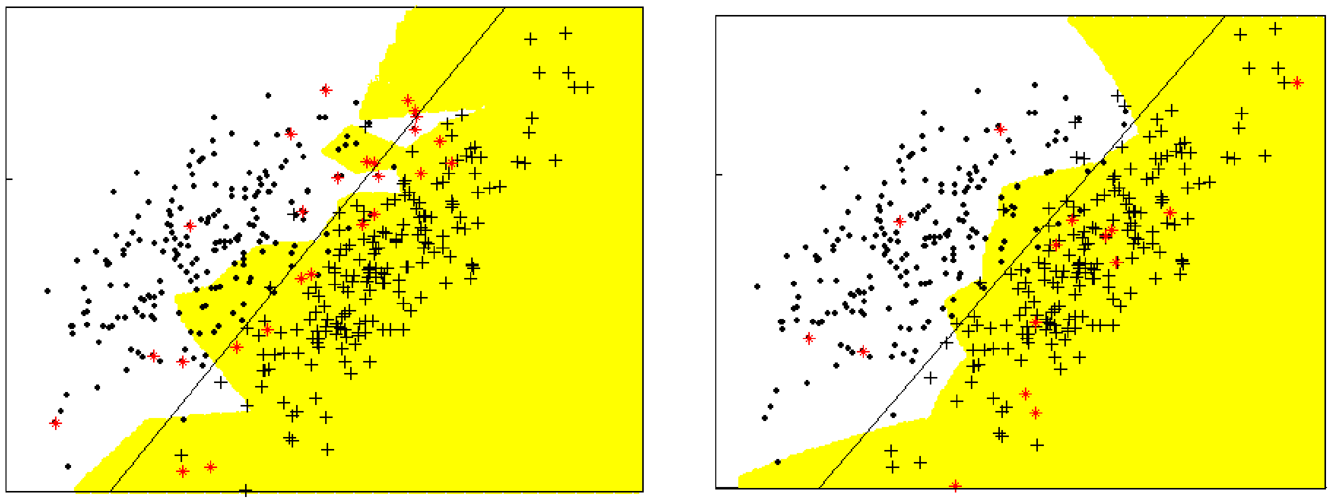

Figure 3. The results of applying the (a) WDNN and (b) AOWDNN algorithms on an artificial dataset. The decision boundaries between the classes are shown with two algorithms.

In contrast, as seen in (b) the distance of the prototypes from the optimal boundary is much more than the distance between WDNN prototypes and the boundary. In addition, the number of selected prototypes is reduced in AOWDNN in comparison to WDNN. This experiment shows that a complex decision boundary can be effectively simplified using the modified AOWDNN.

4.2 Real World Problems

We used 10 datasets from UCI repository to evaluate the performance of the proposed scheme. Table 2 gives a short summary of the datasets used in the experiments.

In the preprocessing phase, each categorical attribute was replaced by *P* binary attributes, where *P* is the number of different values that the attribute can assume. Then the feature vectors normalized to the interval[0, 1]. In data sets having missing values (i.e., Heart), missing values were replaced by the average value of that attribute in the dataset.

The impact of the proposed method is more significant in noisy environments, where a number of observations are mislabeled. In this situation, the mislabeled instances located near the decision boundary of a prototype increase the weight of the prototype, significantly. The OAWDNN method decreases the impact of these noisy instances by decreasing the weight of the prototypes. The weight decay simplifies the decision boundary and increases the generalization power. In the UCI repository many of the datasets are created by experts and the training examples could be classified accurately by well-chosen relevant features. Therefore, we uniformly changed the class label of 20% of the training data.

To measure the performance of the OAWDNN algorithm, ten-fold cross validation (10-CV) was used. To provide a better estimate of the accuracy, the 10-CV test was repeated 20 times and the average classification rate on test data is calculated as the performance of the scheme. For tuning the parameter α, the training data is randomly separated into two parts. One third of the instances in the training data is separated as validation data. Different values for the parameter α are used for the prototype weighting.

The validation data is used to determine which of these values performs the best result. Finally, the prototype weights (that are found using the best value of α) are used to evaluate the generalization performance of AOWDNN algorithm using the test data. In the reported experiments, the parameter α is examined in the interval [.001,3] step by .005. The classification rate on validation data is calculated to determine the best value (i.e. the one which maximizes the accuracy on validation data).

Figure 4 shows the average classification rate of all examined data sets. This figure reports the train and test accuracy of both the WDNN and AOWDNN methods during the 5 iterations of learning process. In each iteration of the WDNN method, the classification rate of the training data is slightly increased while the classification of the test data is decreased. In the iterations, the prototype weights are more and more fitted to the training data which causes the test accuracy to be degraded.

On the other hand, in the AOWDNN method, the classification rate of training data is less than the original WDNN. It is expected because the algorithm intentionally selects weight less than the best weight (i.e., in the interval [0, *best_weight*]) instead of the best weight. Meanwhile,

Table 2. Statistics of the data sets used in experiment

Data Set	No. of Instances	No. of Features	No. of Classes	No. of non numeric features
Cancer	685	9	2	0
Pima	768	8	2	0
Heart	294	13	2	7
Ionosphere	351	34	2	0
Iris	150	4	3	0
Bupa	347	6	2	0
Vote	435	16	2	16
Thyroid	215	5	3	0
Yeast	1484	8	10	0
Wine	178	13	3	0

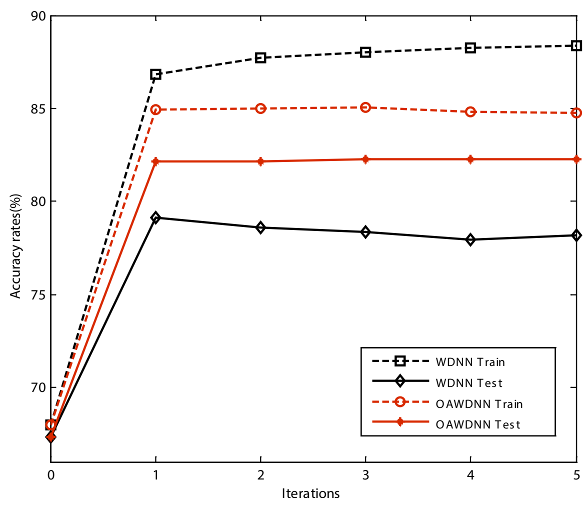

Figure 4. Classification rates of Train and test data in 5 weighting iterations with WDNN and OAWDNN methods.

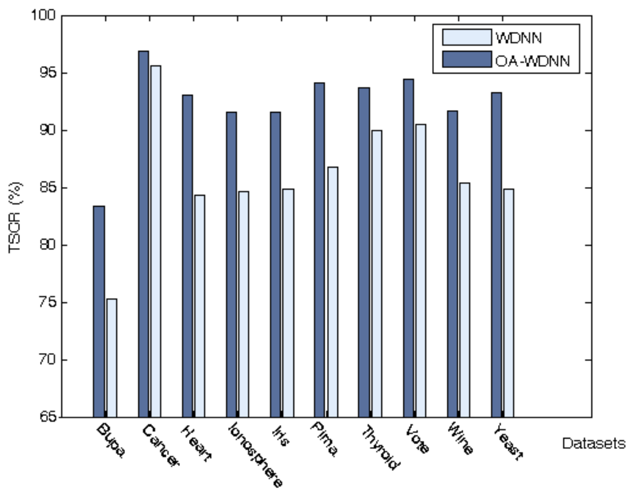

Figure 5. Compression rates of AOWDNN method in comparison with the WDNN method.

Table 3. Classification rate of OA_WDNN method in comparison with 1NN and WDNN methods

Dataset	ALFA	1NN ± STD_DEV	WDNN ± STD_DEV	OAWDNN± STD_DEV
Bupa	0.001	56.60 ± 1.05	59.67 ± 1.38	60.41 ±1.04
Cancer	0.127	77.38 ± 2.09	93.52 ± 1.06	96.71 ±0.27
Heart	0.131	64.99 ± 3.33	73.28 ± 2.90	79.60 ±1.76
Ionosphere	0.058	71.68 ± 2.14	84.44 ± 2.67	87.07 ±1.20
Iris	0.054	75.29 ± 4.29	89.02 ± 2.57	92.80 ±0.88
Pima	0.064	57.77 ± 1.55	69.44 ± 0.98	72.63 ±1.07
Thyroid	0.056	76.13 ± 1.95	92.06 ± 2.14	93.90 ±1.54
Vote	0.104	75.57 ± 2.14	85.97 ± 1.12	88.67 ±1.11
Wine	0.057	76.08 ± 2.41	89.41 ± 1.83	93.61 ±1.57
Yeast	0.030	42.13 ± 1.05	54.77 ± 0.85	56.18 ±0.77
Average	---	67.36 ± 2.20	79.16 ± 1.75	82.16 ±1.12

the test accuracy is increased in comparison to the original method.

Table 3 gives the average of 10-CV test accuracy overall 20 times, for the proposed method, the WDNN algorithm and the basic 1-NN method. Next to each average accuracy value reported in Table 3, the standard deviation (calculated based on 20 runs of 10-CV) is given. The best value for parameter α is also reported in the table. As seen, the OAWDNN improves the generalization accuracy of the basic NN and WDNN method in all data sets. In the last row of this table, average generalization accuracy of each method over all datasets is reported. As seen in the result of Table 3, the OAWDNN method improves the average accuracy of the basic NN and the WDNN 14.8% and 3%, respectively. This verifies the effectiveness of the proposed method compared with the WDNN algorithm.

In Figure 5, the compression rates of the OAWDNN and WDNN methods are shown for various datasets. As seen, The AOWDNN algorithm achieves the best compression rates on all datasets. The reason is that the AOWDNN weight decay causes some prototypes be virtually removed as their weights decrease to zero.

5. Conclusion

The WDNN algorithm is a method to learn nearest neighbor distance measure parameters. In this method a weight parameter is assigned to each training instance, which is used in the weighted distance function. In each iteration, the WDNN learning algorithm maximizes the leave-one-out (LOO) classification rate of the NN rule by determining the best weight for one of the training instances, assuming that the weights of all other instances are given and fixed. This maximization is highly expected to cause weight parameters overfitting to the training data. In this paper we proposed a mechanism to avoid the overfit problem in the WDNN algorithm. This method decreases the instance weights to smooth decision boundary. Experimental results show the effectiveness of the method. In new method, although the error rate is higher on the training data, but its generalization is increased, so test error is significantly lower than original version of WDNN.

7. References

1. Peng J, Heisterkamp DR, Dai HK. Adaptive quasiconformal kernel nearest neighbor classification. IEEE Trans Pattern Anal Mach Intell. 2004; 26:656–61.
2. Hastie T, Tibshirani R. Discriminant adaptive nearest neighbor classification. IEEE Trans Pattern Anal Mach Intell. 2002; 18:607–15.
3. Moosavi MR, Jahromi ZM, Ghodratnama S, Taheri M, Sadreddini MH. A Cost Sensitive Learning Method To Tune The Nearest Neighbour For Intrusion Detection. IJST, Transactions of Electrical Engineering. 2012; 36:109–29.
4. Wang J, Neskovic P, Cooper LN. Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recogn Lett. 2007; 28:207–13.
5. Moosavi MR, Yeganehfard Z, Kazemi A, Sadreddini MH, Jahromi MZ. Distance measure adaptation based on local feature weighting. In Intelligent Systems (IS), 2012 6th IEEE International Conference. 2012 Sept; IEEE. p. 132–7.
6. Paredes R, Vidal E. Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Trans Pattern Anal Mach Intell. 2006; 28:1100–10.
7. Shahparast H, Jahromi MZ, Taheri M, Hamzeloo S. A Novel Weight Adjustment Method for Handling Concept-Drift in Data Stream Classification. Arabian Journal for Science and Engineering. 2014; 39(2):799–807.
8. Shahparast H, Taheri M, Hamzeloo S, Jahromi ZM. An online rule weighting method to classify data streams. 2012 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP); 2012 May; IEEE. p. 407–12.
9. Wilson DR, Martinez TR. An integrated instance-based learning algorithm. Comp Intell. 2000; 16:1–28.
10. Jankowski N. Grochowski M. Comparison of Instances Selection Algorithm I. Algorithms Survey. Rutkowski Let al. (eds), ICAISC, Springer-Verlag LNAI 3070; p. 598–603.
11. Wilson DR, Martinez TR. Reduction techniques for exemplar-based learning algorithms. Mach Learn. 2000; 38:257–86.
12. Paredes R, Vidal E. Learning prototypes and distances: A prototype reduction technique based on nearest neighbor error minimization. Pattern Recogn. 2006; 39:180–8.
13. Jahromi MZ, Parvinnia E, John R. A method of learning weighted similarity function to improve the performance of nearest neighbour. Inform Sci. 2009; 179:2964–73.
14. Cherkassky V, Shao X, Mulier FM, Vapnik VN. Model complexity control for regression using VC generalization bounds. IEEE Trans Neural Network. 1999; 10:1075.
15. Pardo M, Sberveglieri G. Learning from data: A tutorial with emphasis on modern pattern recognition methods. IEEE Sensor J. 2002; 2(3):203–17.
16. Domingos P. The role of Occam's razor in knowledge discovery. Data Min Knowl Discov. Springer. 1999; 3:409–25.
17. Gamberger D, Lavrač N. Conditions for Occam's razor applicability and noise elimination. Springer Berlin Heidelberg, 1997. p. 108–23.
18. Prechelt L. Automatic early stopping using cross validation: quantifying the criteria. Neural Network. 1998; 11:761–7.
19. Wu H, Shapiro JL. Does overfitting affect performance in estimation of distribution algorithms. Proceedings of the 8th annual conference on Genetic and evolutionary computation; 2006; p. 434.
20. Ng AY. Feature selection, L1 vs. L2 regularization and rotational invariance. Proceedings of the twenty-first international conference on Machine learning; 2004; p. 78.
21. Wang C, Venkatesh SS, Judd JS. Optimal Stopping and Effective Machine Complexity in Learning. Advances in NIPS. 1994; 6:303–10.
22. Lauret P, Fock E, Randrianarivony RN, Manicom-Ramsamy JF. Bayesian neural network approach to short time load forecasting. Energ Convers Manag. 2008; 49:1156–66.
23. Moosavi MR, Javan MF, Jahromi MZ, Sadreddini MH. An adaptive nearest neighbor classifier for noisy environments. 18th Iranian Conference on Electrical Engineering (ICEE); 2010; p. 576–80.
24. Parvinnia E, Sabeti M, Jahromi ZM, Boostani R. Classification of EEG Signals using adaptive weighted distance nearest neighbor algorithm. Journal of King Saud University-Computer and Information Sciences. 2013; 26(1):1–6.