A Novel Prototype Reduction Method for the K-Nearest Neighbor Algorithm with $K \geq 1$

Tao Yang, Longbing Cao, and Chengqi Zhang

Faculty of Engineering and Information Technology
University of Technology, Sydney, Australia

Abstract. In this paper, a novel prototype reduction algorithm is proposed, which aims at reducing the storage requirement and enhancing the online speed while retaining the same level of accuracy for a K-nearest neighbor (KNN) classifier. To achieve this goal, our proposed algorithm learns the weighted similarity function for a KNN classifier by maximizing the leave-one-out cross-validation accuracy. Unlike the classical methods PW, LPD and WDNN which can only work with $K = 1$, our developed algorithm can work with $K \geq 1$. This flexibility allows our learning algorithm to have superior classification accuracy and noise robustness. The proposed approach is assessed through experiments with twenty real world benchmark data sets. In all these experiments, the proposed approach shows it can dramatically reduce the storage requirement and online time for KNN while having equal or better accuracy than KNN, and it also shows comparable results to several prototype reduction methods proposed in literature.

1 Introduction

We consider a general classification problem with $C(\geq 2)$ classes and n training instances. Each training instance consists of measurements $\mathbf{x} = (x_1, \ldots, x_d)^T$ on d features and a known class label $y = \{1, 2, \ldots, C\}$. The training data set can be represented in the form of $\Omega = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$. The goal of a classification task is to correctly predict the class label of a query $\mathbf{q} \in \Re^d$.

The K-nearest neighbor (KNN) method is a simple and appealing approach to this problem [3]. The KNN algorithm is also known as nearest neighbor (NN) algorithm for $K = 1$. It is well known that the asymptotic error rate of the NN rule is never more than twice the Bayes rate [1]. However, the major outstanding drawback of the KNN algorithm is that the whole training data set must be stored in memory to be used in the test phase. To identify the K nearest neighbors in the test phase, the distances between $\mathbf{q}$ and all training instances $\mathbf{x}_i$ $(i = 1, \ldots, n)$ must be computed. This can result in a prohibitively large storage requirement and slow testing speed. (Note that the term 'testing speed' and 'online speed' can be used interchangeably.) Also, the presence of noisy instances (i.e., those with errors in the feature vector or class label, or those not representative of typical cases) can also degrade the generalization performance of KNN.

M.J. Zaki et al. (Eds.): PAKDD 2010, Part II, LNAI 6119, pp. 89–100, 2010.

Prototype reduction techniques are concerned with reducing the number of training vectors (prototypes) to be used, which can reduce the storage requirement and increase the testing speed simultaneously. Some prototype reduction methods identify the optimal subset of the representative instances from the original data, while the other approaches generate an entirely new set of objects as the artificial prototypes. Suppose the size of the stored training set is n, then the testing time for KNN to classify one query is $O(dn^2)$. Hence, the reduction in storage requirement can subsequently enhance the testing speed of a KNN classifier. A comprehensive survey of the prototype reduction methods for KNN can be found in [12].

A recent and very promising approach for prototype reduction, called Weighted Distance Nearest Neighbor (WDNN) [2], is based on retaining the informative instances and learning their weights for classification. The WDNN algorithm assigns a weight $w_i(\geq 0)$ to each training instance $\mathbf{x}_i$ at the training phase. Only the training instances and the corresponding weights with $w_i > 0$ will be retained (as the prototypes) in the test phase. Although only a fraction of the training set is retained, the generalization performance of WDNN can be equal to or even better than NN. To achieve this goal, the weights w_i are determined by maximizing the leave-one-out cross-validation (LOOCV) accuracy. At each iteration of a LOOCV procedure, each training instance $\mathbf{x}_i$ is regarded as the query and its class label is predicted by using all the other training instances. This procedure is repeated for $i = 1, \ldots, n$. The WDNN algorithm is a hill climbing optimization technique where each w_i is updated by assuming all the other weights w_j $(j \neq i)$ are given and fixed, and the optimal w_i is determined by considering the threshold value for which $\mathbf{x}_i$ will be the nearest neighbor of the other training instances. In [2], it has been shown that the WDNN algorithm can reduce, on average, more than 80% of the training set while retaining or improving the generalization accuracy of a NN classifier over several real data sets. In the same paper, the WDNN algorithm has also been shown to outperform several benchmarking prototype methods including A-NN [11], PW [7] and LPD [6].

Although the WDNN algorithm is well formulated and shows encouraging performance in practice, it can only work with $K = 1$. (Similarly, PW [7] and LPD [6] also work with $K = 1$ only). However, it has been shown that the best value of K is neither too large nor too small, and setting $K > 1$ can reduce the sensitivity to noise and smooth the decision boundaries (see [1] and [3]). In the case of $K = 1$, being a nearest neighbor of an instance and classifying this instance to its class are the same thing. For $K > 1$, these two things are different because the decision on the classification of a query is determined by K nearest neighbors. This fact along with the possible tied votes make the weight learning for $K > 1$ complicated and difficult.

In this paper, we extend the WDNN algorithm to a general weight learning algorithm for KNN with $K \geq 1$. Naturally, the developed algorithm is dubbed the Weighted Distance K Nearest Neighbor (WDKNN) algorithm. In fact, WDNN is a special case of WDKNN as NN is a special case of KNN. As with WDNN, the WDKNN algorithm iteratively updates the instance weights by maximizing

the LOOCV accuracy. However, the crucial difference between WDKNN and WDNN is that the weights returned by WDKNN are derived from an explicit objective function and model of the decision function. In addition, it has been shown that the optimal weights can be determined by using a subset of the training set only and the difficulties caused by $K > 1$ have been successfully resolved by considering two threshold values in WDKNN.

The rest of this paper is organized as follows. In Section 2, the KNN classification rule combined with instance weights is introduced. Section 3 presents our proposed WDKNN algorithm. The experiment results are given in Section 4. Some concluding remarks are given in Section 5.

2 KNN Classification with Weighted Instances

In this section, we introduce the KNN classification rule with the specified instance weights w_i $(i = 1, \ldots, n)$. Here, we assume that these weights have already been learned by the WDKNN algorithm and we will present how they are learned in the following section.

The K nearest neighbors of $\mathbf{q}$ are found by using the weighted similarity between $\mathbf{q}$ and $\mathbf{x}_i$:

$$\mu_w(\mathbf{q}, \mathbf{x}_i) = w_i \cdot \mu(\mathbf{q}, \mathbf{x}_i), \quad i = 1, \ldots, n, \tag{1}$$

where

$$\mu(\mathbf{q}, \mathbf{x}_i) = 1 - D(\mathbf{q}, \mathbf{x}_i)/D_{\max}, \tag{2}$$

$$D(\mathbf{q}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{d} (q_j - x_{ij})^2}. \tag{3}$$

Here, $D_{\max}$ is the maximum possible distance between two training instances in the feature space, which is used to allow $\mu(\mathbf{q}, \mathbf{x}_i)$ to fall into the interval of $[0, 1]$. That is, $D_{\max} = \sqrt{\sum_{j=1}^{d} \Delta_j^2}$ and Δ_j represents the difference between maximum and minimum values of feature j. If we denote the neighborhood around $\mathbf{q}$ by $N(\mathbf{q})$ and the Kth largest value of $\mu_w(\mathbf{q}, \mathbf{x}_i)$ $(i = 1, \ldots, n)$ by $\psi(\mathbf{q})$, then $N(\mathbf{q})$ can be represented as $N(\mathbf{q}) = \{\mathbf{x}_l \mid \mu_w(\mathbf{q}, \mathbf{x}_l) \geq \psi(\mathbf{q})\}$.

In the test phase, instead of using the traditional majority voting scheme on the K neighbors, we use the following decision function to classify $\mathbf{q}$:

$$F(\mathbf{q}) = \arg\max_c V_c^K(\mathbf{q}), \tag{4}$$

where $V_c^K(\mathbf{q})$ represents the vote of class c obtained by using K nearest neighbors and it can be determined by

$$V_c^K(\mathbf{q}) = \sum_{\mathbf{x}_l \in N(\mathbf{q})} I(y_l = c)\ \mu_w(\mathbf{q}, \mathbf{x}_l), \tag{5}$$

where $I(.)$ denotes an indicator function, whose value is 1 if its argument is true and 0 otherwise.

3 Learning the Instance Weights by WDKNN Algorithm

The WDKNN algorithm presented in this section is an extension of the WDNN algorithm for $K \geq 1$. The WDKNN algorithm assigns a weight $w_i (\geq 0)$ to each training instance $\mathbf{x}_i$ $(i = 1, \ldots, n)$ by maximizing the LOOCV accuracy. This procedure is done in the training phase. Only the training instances and the associated weights with $w_i > 0$ are retained in the test phase.

The WDKNN algorithm is a hill climbing optimization technique for solving w_i, where the optimal weight $w_i^\star$ is determined by assuming all the other weights w_j $(j \neq i)$ are given and fixed. We set $w_i = 1\ \forall i$ as the initial values. At iteration i $(i = 1, \ldots, n)$, the optimal weight $w_i^\star$ can be found by maximizing the objective function related to the LOOCV accuracy:

$$J(w_i) = \sum_{\{\mathbf{x}_m \in \Omega, \mathbf{x}_m \neq \mathbf{x}_i\}} I(F(\mathbf{x}_m|w_i) = y_m), \tag{6}$$

where $F(\mathbf{x}_m|w_i)$ is the decision function of instance $\mathbf{x}_m$ given that the weight for $\mathbf{x}_i$ is w_i. Here, $\mathbf{x}_m$ is treated as a query in the LOOCV procedure. Obviously, the obtained $w_i^\star$ is only suboptimal as the other weights change during the optimization process. Thus, the algorithm will be restarted after a fixed number of runs over the training data set. We follow [2] to restart the optimization process after three runs over the entire data in all experiments conducted in Section 4.

To optimize the weight w_i for the training instance $\mathbf{x}_i$, we assume that $\mathbf{x}_i$ is *consistent* on all other training instances $\mathbf{x}_m (m \neq i)$ in the LOOCV test. Below is the precise definition.

Definition: A training instance is *consistent* on a query if it can either make this query be classified into its class or it is irrelevant of its prediction.

In our proposal, the decision function $F(\mathbf{x}_m|w_i)$ of instance $\mathbf{x}_m$ is modeled as follows:

$$F'(\mathbf{x}_m|w_i) = I(w_i \leq \theta_m)\ F(\mathbf{x}_m|w_i = 0) + I(w_i > \theta_m)\ y_i, \tag{7}$$

where $F(\mathbf{x}_m|w_i = 0)$ is the decision function with $w_i = 0$, θ_m is a threshold for w_i. This model essentially means that $\mathbf{x}_m$ will be classified without using $(\mathbf{x}_i, w_i)$ unless they can make $\mathbf{x}_m$ be classified into its class y_i. It is easy to see that $F'(\mathbf{x}_m|w_i) = F(\mathbf{x}_m|w_i)$ for $K = 1$. This model is illustrated in Figure 1, where a three-class classification problem in two-dimensional space has been plotted. The class memberships of the training instances are distinguished by the various shapes of the data points. The query $\mathbf{q}$ is denoted by a black square and the instance $\mathbf{x}_i$ is the data point with a green '+' label. If we set $w_i = 0,\ w_j = 1, \forall j \neq i$, then the $K(= 7)$ nearest neighbors of $\mathbf{q}$ are drawn with connection lines and $\mathbf{q}$ will be classified into the 'triangle' class. On the other hand, if $w_i > 1.18,\ w_j = 1, \forall j \neq i$, the nearest neighbor with the dashed connection line will be replaced by $\mathbf{x}_i$ and thus WDKNN will assign $\mathbf{q}$ into the 'diamond' class. Hence, we have $\theta_m = 1.18$ in this example.

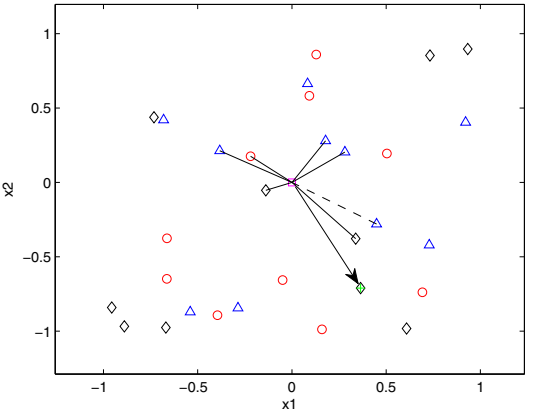

Fig. 1. An illustration of the model in equation (7)

The value of threshold θ_m can be specified for each $\mathbf{x}_m$ by considering three cases regarding the relationship between $\mathbf{x}_m$ and $(\mathbf{x}_i, w_i)$.

Case 1: If we have $F(\mathbf{x}_m|w_i = 0) = y_i$, then $w_i^\star$ will not depend on $(\mathbf{x}_m, y_m)$.

Proof. If $F(\mathbf{x}_m|w_i = 0) = y_i$, then according to the definition given in equation (4), we have[1]

$$V_{y_i}(\mathbf{x}_m|w_i = 0) > V_{c\neq y_i}(\mathbf{x}_m|w_i = 0). \tag{8}$$

But, according to equation (5), we also have

$$V_{y_i}(\mathbf{x}_m|w_i > 0) \geq V_{y_i}(\mathbf{x}_m|w_i = 0), \tag{9}$$

$$V_{c\neq y_i}(\mathbf{x}_m|w_i > 0) \leq V_{c\neq y_i}(\mathbf{x}_m|w_i = 0), \tag{10}$$

which leads to

$$V_{y_i}(\mathbf{x}_m|w_i > 0) > V_{c\neq y_i}(\mathbf{x}_m|w_i > 0), \tag{11}$$

and thus

$$F(\mathbf{x}_m|w_i > 0) = y_i. \tag{12}$$

Now,

$$F(\mathbf{x}_m|w_i > 0) = F(\mathbf{x}_m|w_i = 0), \tag{13}$$

which implies that $F(\mathbf{x}_m|w_i)$ and hence $I(F(\mathbf{x}_m|w_i) = y_m)$ will be a constant for all values of $w_i \geq 0$. Therefore, the optimal value of w_i is irrelevant of $(\mathbf{x}_m, y_m)$. (Note that this proof is valid regardless the appropriateness of the model given in equation (7).)

Case 2: If we have $F(\mathbf{x}_m|w_i = 0) \neq y_i$, $F(\mathbf{x}_m|w_i = 0) \neq y_m$ and $y_i \neq y_m$, then $w_i^\star$ will not depend on $(\mathbf{x}_m, y_m)$.

[1] For the sake of clarity, the notation $V_c^K(\mathbf{q})$ is replaced by $V_c(\mathbf{q})$ for the proof given here.

Proof. From the above given conditions, it is easy to arrive at

$$F'(\mathbf{x}_m|w_i) \neq y_m, \quad \forall w_i \geq 0, \tag{14}$$

leading to

$$I(F'(\mathbf{x}_m|w_i) = y_m) = 0, \quad \forall w_i \geq 0. \tag{15}$$

Hence, it can be seen that $\mathbf{x}_m$ will be misclassified for all values of w_i in this case.

Case 3: If we let M_i denote the collection of all training instances $\{(\mathbf{x}_m, y_m) \in \Omega \mid m \neq i\}$ that do not belong to Case 1 and 2. Then, the threshold value θ_m can be determined for all $(\mathbf{x}_m, y_m) \in M_i$ as follows:

$$\theta_m = \max\{\ \alpha_m,\ \beta_m\}, \tag{16}$$

$$\alpha_m = \psi(\mathbf{x}_m)/\mu(\mathbf{x}_m, \mathbf{x}_i), \tag{17}$$

$$\beta_m = (\max_c V_c^{K-1}(\mathbf{x}_m) - V_{y_i}^{K-1}(\mathbf{x}_m))/\mu(\mathbf{x}_m, \mathbf{x}_i). \tag{18}$$

where $\psi(\mathbf{x}_m)$ is the Kth largest value of $\mu_w(\mathbf{x}_m, \mathbf{x}_j)$ for $j = 1, \ldots, n$ and $j \neq i$.

Proof. Equation (7) tells us that $F'(\mathbf{x}_m|w_i) = y_i$ if and only if $w_i > \theta_m$. Also, according to the classification rule introduced in Section 2, $F(\mathbf{x}_m|w_i) = y_i$ if and only if two two conditions are satisfied: (a) $(\mathbf{x}_i, y_i)$ is selected as one of the K nearest neighbors of $\mathbf{x}_m$; (b) the class vote for class y_i is the largest among all classes.

In order to satisfy condition (a), we must have

$$w_i \cdot \mu(\mathbf{x}_m, \mathbf{x}_i) > \psi(\mathbf{x}_m), \tag{19}$$

leading to

$$w_i > \psi(\mathbf{x}_m)/\mu(\mathbf{x}_m, \mathbf{x}_i). \tag{20}$$

(b) After condition (a) is satisfied, the previous Kth neighbor of $\mathbf{x}_m$ will be replaced by $(\mathbf{x}_i, y_i)$. By equation (4) - (5),

$$w_i \cdot \mu(\mathbf{x}_m, \mathbf{x}_i) + V_{y_i}^{K-1}(\mathbf{x}_m) > V_c^{K-1}(\mathbf{x}_m) \quad \forall\ c = 1, \ldots, C, \tag{21}$$

which leads to

$$w_i > (\max_c V_c^{K-1}(\mathbf{x}_m) - V_{y_i}^{K-1}(\mathbf{x}_m))/\mu(\mathbf{x}_m, \mathbf{x}_i). \tag{22}$$

According to the above three cases, the optimal $w_i^\star$ can be found based on all $\mathbf{x}_m \in M_i$ by maximizing the following criterion:

$$J'(w_i) = \sum_{\{\mathbf{x}_m \in M_i\}} I(F'(\mathbf{x}_m|w_i) = y_m). \tag{23}$$

Assume there are L threshold values θ_m ranked in ascending order $\theta_1 < \theta_2 < \ldots < \theta_L$. Then, $L + 1$ values of w_i are examined and the best one can be found

by using equation (23). The first and last values examined are 0 and $\theta_L + \pi$, respectively (π is a very small positive number). The rest values are chosen in the middle of two successive θ_m.

The compression rate (CR) is used to compute the rate at which the prototypes are reduced for a prototype reduction method:

$$CR = 1 - r/n, \tag{24}$$

where r and n represent the reduced and original number of instances, respectively. The KNN (and hence NN) classifier retains all training instances and thus it has $CR = 0$.

4 Experiments

To validate the proposed WDKNN algorithm, we compared it with the traditional KNN (including NN) algorithm and several other state-of-the-art prototype reduction methods in literature: learning vector quantization [4], learning prototypes and distances (LPD) [6], adaptive nearest neighbor (A-NN) [11], prototype-dependent weighting (PW) [7], WDNN [2] and MWDNN [2]. Twenty real world benchmark data sets taken from the UCI Machine Learning Repository[2] are used throughout the experiments (see Table 1). We performed the implementation using MATLAB R2007a on Windows XP with 2Duo CPU running on 3.16 GHz PC with 3.25 GB RAM.

4.1 Experiments on UCI Data Sets

We follow [7] [6] and [2] by using five-fold cross-validation (5-CV) to test the performance of various methods. The average accuracy and compression rate for various methods are compared. Both the KNN and WDKNN algorithms have a tuning parameter: K (neighborhood size). They are determined by selecting the integer ranging from 1 to 41 that corresponds to the maximal 5-CV accuracy on each data set. The selected values of K for both algorithms on each data set are reported in Table 1.

Figure 2 shows the LOOCV accuracies of WDNN and WDKNN during the weight learning progress at the first iteration. This figure was plotted based on one fold of a 5-CV test on the ionosphere data set and $K = 4$ was used for WDKNN. The LOOCV accuracy was computed after each one of the weights had been updated. It can be seen that the LOOCV accuracy for WDKNN never increases during the learning process. Also, WDKNN has shown superior performance to WDNN during the whole optimization process.

The mean classification accuracies and compression rates of the NN, KNN, WDNN and WDKNN methods are summarized in Table 1. We noted that the accuracy of WDKNN is higher than that of WDNN over all data sets. However, this higher accuracy is achieved at the expense of lower compression rates. This

[2] `http://archive.ics.uci.edu/ml/index.html`

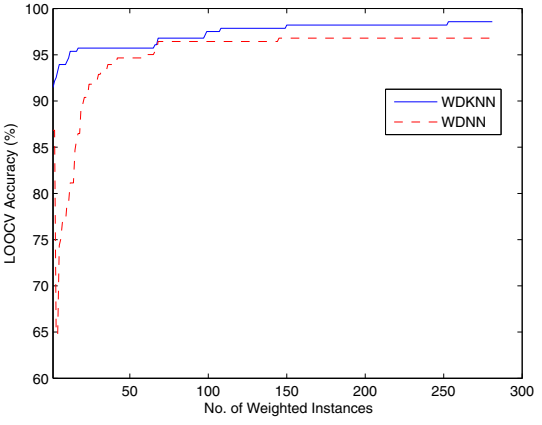

Fig. 2. Leave-one-out cross-validation accuracy during the first iteration of the WDNN and WDKNN ($K = 4$) algorithms on ionosphere data

Table 1. Classification accuracy (%) and compression rates (%) of four classifiers on twenty real data sets

Data	NN	KNN	CR	WDNN	CR	WDKNN	CR	K_{knn}	K_{wdknn}
Australian	79.28	86.52	0	82.75	91.88	**86.67**	79.67	33	24
Balance	80.48	90.08	0	86.72	93.32	**91.36**	70.96	20	36
Breast (Original)	95.90	97.07	0	96.63	98.72	**97.22**	96.82	5	7
Breast (Prognostic)	69.21	78.79	0	74.79	93.57	**79.81**	63.26	6	31
Dermatology	94.54	96.18	0	95.91	94.06	**97.00**	83.20	10	21
Diabetes	71.75	**76.43**	0	71.87	89.55	75.65	56.48	18	40
Ecoli	82.73	86.59	0	83.32	93.30	**88.08**	84.52	9	8
Flag	41.30	46.98	0	43.86	79.13	**51.08**	70.88	28	3
German	68.30	73.50	0	69.00	88.95	**74.50**	60.25	22	33
Glass	**73.39**	**73.39**	0	64.41	81.19	73.32	70.68	1	3
Haberman	65.34	74.53	0	70.93	94.28	**76.17**	71.41	11	29
Heart	78.15	84.07	0	82.22	91.02	**84.81**	83.61	29	3
Ionosphere	87.45	87.45	0	90.31	93.59	**93.16**	88.60	1	4
Iris	92.67	96.67	0	94.00	91.00	**97.33**	59.00	13	33
Liver	60.87	69.86	0	66.96	89.28	**70.43**	52.03	7	25
Soybean	95.56	95.56	0	84.67	73.93	**97.78**	55.87	1	7
Vehicle	70.69	**73.88**	0	63.31	84.43	71.52	53.99	5	29
Vote	93.11	93.11	0	90.95	94.72	**95.28**	84.16	1	14
Wine	95.51	97.75	0	95.49	93.96	**98.30**	82.86	33	12
Zoo	94.00	**96.00**	0	95.00	82.43	**96.00**	76.74	6	4
Average	79.51	83.72	0	80.16	89.62	**84.77**	72.25	13	18

The bold numbers represent the highest accuracy values for each data set.

is because setting a larger K usually requires more training instances to be involved in the classification stage. It can be seen that WDKNN obtained the overall highest accuracy and also achieved the best accuracy in 17 out of 20

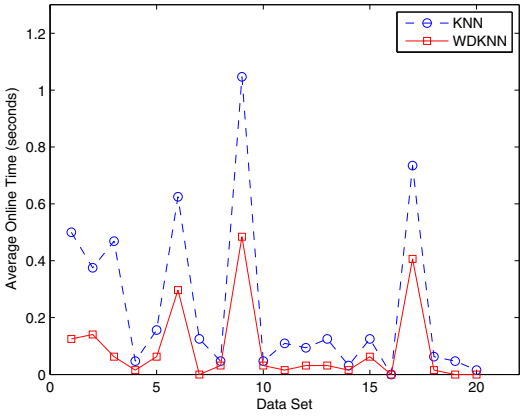

Fig. 3. The comparison of average online time (in seconds) between KNN and WDKNN over the real data sets

cases (with 1 equal best one). Based on the accuracy values obtained by various methods over twenty data sets, we used a one-tailed paired t-test to compare WDKNN's accuracies to those of NN, KNN and WDNN. We have statistical evidence that the average accuracy of WDKNN is significantly higher than the average accuracies of all of NN, KNN and WDNN for each significance level of 0.5%, 1% and 5%. Moreover, WDKNN achieves higher accuracy than KNN with no more than 30% of the training data set on average.

Figure 3 shows the average online time required for KNN and WDKNN over the twenty real data sets. In the figure, each data set was denoted by the order in which they appeared in Table 1. We noted that the online time required for WDKNN is less than that of KNN over all 20 data sets. The online speed of WDKNN is 7.5 times faster than KNN for the breast cancer (original) data set, and the average online speed of WDKNN over the twenty data sets is 2.6 times faster than that of KNN. Although the reduction in online time of KNN is trivial for these small to medium sized data sets, the reduction can be dramatic for large data sets and this property is crucial for a KNN classifier. If the size of either the training set or the test set increases, the advantage of WDKNN over KNN in terms of online execution speed will become more obvious.

Table 2 gives the average 5-CV accuracies of WDKNN in comparison with the published results of several state-of-the-art prototype reduction methods. The data sets used by these methods are taken from the Statlog project, so we only display WDKNN's results on these data sets. The experimental procedures for the competing methods are the same as those of WDKNN except that the feature weighting methods have only been used in PW, LPD, MDNN and MWDNN. In this paper, we focus on instance weighting, and thus we only employ the basic Euclidean distance, which implicitly assumes that all features have equal weight. Hence, the accuracies of WDKNN would be consistently higher if the feature weighting scheme had also been used in WDKNN. Despite this unfairness for WDKNN, it still achieves the best average accuracy on all data sets.

Table 2. Classification accuracies (%) of the WDKNN algorithm in comparison with other algorithms in literature

Data	WDKNN	LVQ1	LVQ2	LVQ3	A-NN	PW	LPD	WDNN	MWDNN
Australian	**86.67**	66.7	66.0	68.9	75.91	83.05	86.1	85.48	85.01
Balance	**91.36**	83.7	85.3	83.7	89.88	86.56	83.7	89.14	90.32
Breast (Original)	97.22	95.6	95.5	95.8	97.14	96.68	96.6	97.52	**97.88**
Diabetes	75.65	73.6	74.2	74.0	71.86	72.61	74.0	75.96	**76.31**
German	74.50	69.9	71.5	71.3	61.89	71.68	74.0	**75.84**	73.89
Glass	73.32	60.4	57.6	58.5	71.22	73.72	72.0	71.34	70.81
Heart	84.81	64.0	66.0	66.0	67.45	81.06	81.4	83.91	**84.91**
Liver	**70.43**	67.5	67.3	66.4	65.12	63.78	66.7	68.31	65.39
Vehicle	71.52	61.8	68.0	65.6	66.28	70.69	**72.6**	70.14	69.15
Vote	**95.28**	92.4	93.8	94.3	93.31	94.49	96.3	92.29	91.37
Wine	**98.30**	70.3	74.2	70.8	84.82	98.65	95.0	96.61	96.04
Average	**83.6**	73.3	74.5	74.1	76.8	81.2	81.7	82.4	81.9

The bold numbers represent the highest accuracy values for each data set.

Using the one-tailed paired t-test (with a significance level of 5%), we have statistical evidence that WDKNN outperforms all the competing methods. It must be noted that the A-NN and PW methods do not reduce the training size. In addition, these results obtained by WDKNN are comparable to or better than those obtained by other state-of-the-art methods recently published on the same tasks [9,8,5].

4.2 Effect of Noise

Since WDKNN is designed to be more robust in the presence of noise than WDNN, the experiments conducted in Section 4.1 were repeated with 20% uniform class noise artificially added to each data set. This was done by randomly changing the class label of 20% of the training instances to an incorrect value (with an equal probability for each of the incorrect classes). The class labels of the test instances are not noisy. Note that the experimental settings here are the same as those in [2]. The performance of NN, KNN, WDNN and WDKNN were tested to see their robustness of noise.

Table 3 reports the average accuracies and compression rates of each method over twenty data sets. As can be seen, the accuracy of WDKNN is much better than that of WDNN. This is consistent with our motivation that using $K > 1$ can reduce WDNN's sensitivity of noise. The average values of K for both KNN and WDKNN on the noisy data become larger compared to the real data sets. This also suggests that a larger K is more suitable for the noisy data sets. WDKNN achieves the highest accuracy value averaged over the entire data set. Also, the accuracy given by WDKNN is the best in 10 out of 20 cases. Using the one-tailed paired t-test (for both the 1% and 5% significance level), we have statistical evidence that WDKNN outperforms NN and WDNN in terms of accuracy, and we have no statistical evidence that the average accuracy of WDKN is larger

Table 3. Classification accuracies (%) and compression rates (%) of four classifiers on the noisy data sets

Data	NN	KNN	CR	WDNN	CR	WDKNN	CR	K_{knn}	K_{wdknn}
Australian	66.67	85.51	0	77.25	88.99	**85.94**	67.07	39	31
Balance	69.60	**88.96**	0	80.96	90.16	88.00	62.36	23	38
Breast (Original)	80.09	**96.48**	0	93.56	89.89	96.05	89.42	28	33
Breast (Prognostic)	64.69	**78.33**	0	69.24	89.77	78.31	66.55	10	14
Dermatology	74.07	**95.36**	0	91.27	91.60	95.08	74.59	11	36
Diabetes	62.11	**73.17**	0	66.67	86.72	72.14	60.41	36	18
Ecoli	65.17	84.51	0	79.74	91.52	**84.81**	69.79	14	39
Flag	36.65	47.49	0	42.31	81.19	**50.05**	64.17	33	11
German	63.60	72.60	0	64.90	88.03	**73.20**	67.13	30	18
Glass	62.20	65.49	0	64.50	82.36	**67.32**	76.52	4	1
Haberman	60.15	**74.53**	0	71.59	92.81	73.55	73.29	23	17
Heart	64.81	**84.44**	0	72.96	86.76	83.70	59.26	26	38
Ionosphere	77.77	85.17	0	85.45	89.74	**88.89**	71.37	10	12
Iris	69.33	95.33	0	88.67	88.67	**96.67**	66.00	13	25
Liver	55.07	**66.09**	0	58.55	85.51	65.80	41.88	26	35
Soybean	80.89	93.33	0	86.89	75.48	**93.56**	49.43	4	11
Vehicle	57.09	**69.98**	0	61.59	84.49	67.49	56.74	15	26
Vote	71.96	93.11	0	85.39	89.22	**93.55**	65.20	16	38
Wine	80.37	**97.17**	0	92.13	91.15	96.63	68.26	37	29
Zoo	71.14	93.05	0	86.14	78.97	**93.10**	73.27	5	6
Average	66.67	82.01	0	75.99	87.15	**82.19**	66.14	20	24

The bold numbers represent the highest accuracy values for each data set.

than that of KNN. On average, WDKNN achieves the same level of accuracy as KNN by using no more than 35% of the training data set. In addition, we also found that the online time required for WDKNN is less than that of KNN over all noisy data sets. Specifically, the average online speed of WDKNN over the twenty noisy data sets is 2.2 times faster than that of KNN.

5 Conclusions

In this paper, a novel prototype reduction method for KNN has been proposed. This method removes the instances that are more of a computational burden but do not contribute to better classification accuracy and it also assigns a weight to each retained training instance. These weights are learned by maximizing the leave-one-out cross-validation classification accuracy, which is the true estimate of the generalization ability of a classifier. Empirical results have shown that WDKNN considerably reduces the size of the training set while retaining or improving the classification accuracy of KNN. In fact, the proposed method may also be useful for other lazy learning methods such as ALH [10] in terms of storage requirement and online speed.

References

1. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
2. Jahromi, M.Z., Parvinnia, E., John, R.: A method of learning weighted similarity function to improve the performance of nearest neighbor. Information Sciences 179(17), 2964–2973 (2009)
3. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer, Heidelberg (2009)
4. Kohonen, T.: Self-Organization and Associative Memory, 3rd edn. Springer, Berlin (1989)
5. Loog, M., Duin, R.P.W.: Linear dimensionality reduction via a heteroscedastic extension of LDA: the chernoff criterion. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(6), 732–739 (2004)
6. Paredes, R., Vidal, E.: Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization. Pattern Recognition 39(2), 180–188 (2006)
7. Paredes, R., Vidal, E.: Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(7), 1100–1110 (2006)
8. Peng, J., Heisterkamp, D.R., Dai, H.: Adaptive quasiconformal kernel nearest neighbor classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5), 656–661 (2004)
9. Ridder, D., Loog, M., Reinders, M.J.T.: Local Fisher embedding. In: Proc. of 17th International Conference on Pattern Recognition, vol. 2, pp. 295–298 (2004)
10. Yang, T., Kecman, V.: Adaptive local hyperplane classification. Neurocomputing 71, 3001–3004 (2008)
11. Wang, J., Neskovic, P., Cooper, L.N.: Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recognition Letters 28(2), 207–213 (2007)
12. Wilson, D.R., Martinez, T.R.: Reduction techniques for exemplar-based learning algorithms. Machine Learning 38(3), 257–286 (2000)