Learning Locally Weighted C4.4 for Class Probability Estimation

Liangxiao Jiang[1], Harry Zhang[2], Dianhong Wang[1], and Zhihua Cai[1]

[1] Faculty of Computer Science, China University of Geosciences
Wuhan, Hubei, P.R. China, 430074
ljiang@cug.edu.cn
[2] Faculty of Computer Science, University of New Brunswick
P.O. Box 4400, Fredericton, NB, Canada E3B 5A3
hzhang@unb.ca

Abstract. In many real-world data mining applications, accurate class probability estimations are often required to make optimal decisions. For example, in direct marketing, we often need to deploy different promotion strategies to customers with different likelihood (probability) of buying some products. When our learning task is to build a model with accurate class probability estimations, C4.4 is the most popular one for achieving this task because of its efficiency and effect. In this paper, we present a locally weighted version of C4.4 to scale up its class probability estimation performance by combining locally weighted learning with C4.4. We call our improved algorithm locally weighted C4.4, simply LWC4.4. We experimentally tested LWC4.4 using the whole 36 UCI data sets selected by Weka, and compared it to other related algorithms: C4.4, NB, KNN, NBTree, and LWNB. The experimental results show that LWC4.4 significantly outperforms all the other algorithms in term of conditional log likelihood, simply CLL. Thus, our work provides an effective algorithm to produce accurate class probability estimation.

Keywords: class probability estimation, C4.4, locally weighted C4.4, locally weighted learning, conditional log likelihood.

1 Introduction

Classification has been extensively studied and various learning algorithms have be developed, such as decision tree, Bayesian network, and k-nearest-neighbor. The predictive performance of a classifier is usually measured by its classification accuracy on the testing instances. In fact, most classifiers, including decision tree, Bayesian network, and k-nearest-neighbor, can also produce probability estimations or "confidence" of the class prediction. Unfortunately, however, this information is completely ignored in classification. This is often taken for granted since the true probability is unknown for the test instances anyway.

In many real-world data mining applications, however, classifiers' classification accuracy are not enough, because they can't express the information how "far-off" (be it 0.45 or 0.01?) is the prediction of each instance from its target.

V. Corruble, M. Takeda, and E. Suzuki (Eds.): DS 2007, LNAI 4755, pp. 104–115, 2007.

For example, in direct marketing, we often need to deploy different promotion strategies to customers with different likelihood (probability) of buying some products. To accomplish these tasks, we need more than a mere classification of buyers and non-buyers, namely an accurate class probability estimation of customers in terms of their likelihood of buying. It is obvious that an accurate probability estimation of class membership is much more desirable than just an accurate classification in these cases.

This fact raises the question of wether is there another better criterion than the classification accuracy to evaluate classifiers that also produce class probability estimation, if we are aiming at an accurate class probability estimation from a classifier? Recent research show that the answer is the conditional log likelihood, simply CLL [1,2]. Now, given a built classifier G and a set of test instances $D = \{e_1, e_2, \ldots, e_i, \ldots, e_N\}$, where $e_i =< a_{i1}, a_{i2}, \ldots, a_{in} >$, N is the number of test instances, n is the number of attributes, and c_i is the true (ideal) class label of the test instance e_i. Then, the conditional log likelihood $CLL(G|D)$ of the built classifier G on the set of test instances D is:

$$CLL(G|D) = \sum_{i=1}^{N} logP_G(c_i|a_{i1}, a_{i2} \ldots, a_{in}) \tag{1}$$

In this paper, we firstly conduct an extensive experiment to compare some state-of-the-art algorithms such as C4.4 [3], NB (naive Bayes) and KNN (k-nearest-neighbor) in terms of class probability estimation (measured by CLL). The experimental results show that C4.4 performs significantly better than NB and KNN. This results indicate that C4.4 is an attractive model for class probability estimation. Motivated by the success of locally weighted linear regression and locally weighted naive Bayes [4], we present a locally weighted version of C4.4 to scale up its class probability estimation performance by combining locally weighted learning (LWNB) [5] with C4.4. We call our improved algorithm locally weighted C4.4, simply LWC4.4.

The rest of the paper is organized as follows. In Section 2, we summarize some related algorithms can be used to class probability estimation. In Section 3, we present our improved algorithm called Locally Weighted C4.4. In Section 4, we describe the experimental setup and results in detail. In Section 5, we make a conclusion and outline our main directions for future research.

2 Related Work

Just as discussed in section 1, many classification models such as decision tree, Bayesian network, and k-nearest-neighbor can also be used for class probability estimation. Now, we simply look back them in this section.

2.1 Decision Tree

Decision tree is one of the most widely used classification models. It classifies an instance by sorting it down the tree from the root node to one leaf node,

which provides the classification of this instance. Each node in the tree specifies a test of one attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given instance. This process is then repeated for the subtree rooted at the new node. After the tree is built, a manipulation called tree pruning is performed to scale up the classification accuracy of the learned tree.

Unfortunately, traditional decision tree algorithms, such as C4.5 [6], have been observed to produce poor class probability estimation [7]. Aiming at this fact, Provost and Domingos [3] presented an improved algorithm simply called C4.4 to improve C4.5's performance on class probability estimation. In C4.4, two techniques are used to improve C4.5's class probability estimation:

1. Smooth class probability estimation by Laplace estimation: Assume that there are p instances of the class at a leaf, N total instances, and C total classes. The frequency-based estimation calculates the estimated probability of class membership as $\frac{p}{N}$. The Laplace estimation calculates the estimated probability of class membership as $\frac{p+1}{N+C}$.
2. Turn off pruning: Provost and Domingos show that pruning a large tree damages the probability estimation. Thus, a simple strategy to improve the probability estimation is to build a large tree without tree pruning.

2.2 Bayesian Network

A Bayesian network consists of a structural model and a set of conditional probabilities. The structural model is a directed acyclic graph in which nodes represent attributes and arcs represent attribute dependencies. Attribute dependencies are quantified by conditional probabilities for each node given its parents. Bayesian networks are often used for classification problems, in which a learner attempts to construct a classifier from a given set of training instances with class labels. Assume that A_1, A_2,$\cdots$, A_n are n attributes (corresponding to attribute nodes in a Bayesian network). A test instance e is represented by a vector $(a_1, a_2, , \cdots, a_n)$, where a_i is the value of A_i. Let C represent the class variable (corresponding to the class node in a Bayesian network). We use c to represent the value that C takes and $c(e)$ to denote the class of e. The Bayesian network classifier represented by a Bayesian network is defined in Equation 2.

$$c(e) = \arg\max_{c \in C} P(c)P(a_1, a_2, \cdots, a_n|c). \tag{2}$$

Assume all attributes are independent given the class. Then, the resulting classifier is called naive Bayes, simply NB:

$$c(e) = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(a_i|c). \tag{3}$$

In NB, each attribute node has the class node as its parent, but does not have any parent from attribute nodes. Although naive Bayes is easy to construct, the attribute conditional independence assumption made by the naive approach harms the classification accuracy of naive Bayes when it is violated. In order to relax this assumption effectively, an appropriate language and efficient machinery to represent and manipulate independence assertions are needed [8]. Both are provided by Bayesian networks [9]. Unfortunately, however, it has been proved that learning an optimal Bayesian network is NP-hard [10]. In order to avoid the intractable complexity for learning Bayesian networks, learning improved naive Bayes has attracted much attention from researchers. For example, Kohavi [11] presented an algorithm called naive Bayes tree, simply NBTree. It uses decision trees to scale up the classification accuracy of naive Bayes. Learning an NBTree is similar to C4.5 [6] except for its score function of evaluating split attributes. After a tree is grown, a naive Bayes is constructed for each leaf using the data associated with that leaf. NBTree classifies a test instance by sorting it to a leaf and applying the naive Bayes in that leaf to assign a class label to it.

2.3 K-Nearest-Neighbor

KNN (k-nearest-neighbor) has been widely used in classification problems. KNN is based on a distance function that measures the difference or similarity between two instances. The standard Euclidean distance $d(x, y)$ between two instance x and y is often used as the distance function, defined as follows.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (a_i(x) - a_i(y))^2} \tag{4}$$

Given a test instance x, KNN assigns the most common class of x's k nearest neighbors to x, as shown in Equation 5. KNN is a typical example of lazy learning, which just stores training data at training time and delays its learning until classification time.

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, c(y_i)) \tag{5}$$

where y_1, y_2, $\cdots$, y_k are the k nearest neighbors of x, k is the number of the neighbors, and $\delta(c, c(y_i)) = 1$ if $c = c(y_i)$ and $\delta(c, c(y_i)) = 0$ otherwise.

KNN uses a simple voting to produce the class probability estimation. That is say that the class labels of instances in the neighborhood are treated equally. So, an obvious improved method is to weight the vote of k nearest neighbors differently according to their distance to the test instance x. The resulting classifier is called k-nearest-neighbor with distance weighted defined as follows.

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \frac{\delta(c, c(y_i))}{d(y_i, x)^2} \tag{6}$$

Another most efficient approach is deploying a local probability-based classification model within the neighborhood of the test instance consisting of the k nearest neighbors. Talking of the local probability-based classification models, naive Bayes is absolutely necessary. The idea of combining KNN with naive Bayes is quite straightforward. Whenever a test instance is classified, a local naive Bayes is trained using the k nearest neighbors of the test instance, with which the test instance is classified. Locally weighted naive Bayes [4], simply LWNB, is a state-of-the-art example, which implements the locally weighed manipulation using locally weighted learning [5]. In LWNB, k nearest neighbors of a test instance are firstly found and each of them is weighted in terms of its distance to the test instance. Then a local naive Bayes is built from the locally weighted training instances.

3 Locally Weighted C4.4

Thinking of C4.5's bad performance of class probability estimation, Provost and Domingos [3] presented an improved algorithm simply called C4.4 to improve its class probability estimation performance. In C4.4, two techniques called Laplace correction and turning of tree pruning are used.

Locally weighted learning [5] is meta method, which has been successfully used to improve some efficient and effective algorithms. For example, locally weighted linear regression, which is a locally weighted version of linear regression. It uses a local linear regression to fit to a subset of the training instances that is in the neighborhood of the test instance. The training instances in this neighborhood are weighted according to its distance from the test instance, with less weight being assigned to instances that are further from the test instance. A regression prediction is then obtained from linear regression taking the attribute values of the test instance as input. Similar to locally weighted linear regression, Li [12] uses locally weighted learning to improve SMOreg (a support vector machine algorithm for Regression) for Regression.

For another example, Frank et al. [4] presented a hybrid algorithm called locally weighted naive Bayes, simply LWNB, by combining locally weighted learning with naive Bayes. When call upon to classify a test instance, LWNB firstly finds the k nearest neighbors of this test instance. Then, LWNB assigns different weights to different instances in the neighborhood according to its distance from the test instance. At last, a local naive Bayes is built on these locally weighted training instances, with which this test instance is classified.

For solving the regression problems, the linear regression algorithm is the most popular one. Its locally weighted version demonstrates great improvement. In the same way, naive Bayes performs well in classification [8], its improved algorithm called locally weighted naive Bayes significantly outperforms it in terms of classification accuracy. Thus, we can draw a conclusion: a remarkable character in applying locally weighted learning is that local models all need to be efficient and effective. Fortunately, C4.4 exactly meets this character. These facts raise the question of whether such a locally weighted learning can be used to improve the class probability estimation performance of C4.4.

Responding to this question, we present a locally weighted version of C4.4 by combining locally weighted learning with C4.4. We call our improved algorithm locally weighted C4.4 , simply LWC4.4. LWC4.4 use C4.4 in exactly the same way as naive Bayes is used in locally weighted naive Bayes: a local C4.4 is built on the subset of the training instances that is in the neighborhood of the test instance whose probability of class membership is to be estimated. The training instances in this neighborhood are weighted according to the inverse of its distance from the test instance, with less weight being assigned to instances that are further from the test instance. A class probability estimation is then obtained from C4.4 taking the attribute values of the test instance as input.

The subset of the training instances used to training each locally weighted C4.4 are determined by a k-nearest-neighbor algorithm. A user-specified parameter k controls how many instances are used. So, like locally weighted linear regression and locally weighted naive Bayes, our locally weighted C4.4 also is a k-related algorithm. Fortunately, we get almost same experimental results with LWNB: LWC4.4 is not particularly sensitive to the choice of value of k as long as it is not too small. This makes it a very attractive alternative to the k-related algorithms, which requires fine-tuning of k to achieve good results.

Although our experimental results show that LWC4.4 significantly outperforms the original C4.4 measured by CLL. Our improvements turn an eager learning algorithm into a lazy learning algorithm. Like all the other lazy learning algorithms, LWC4.4 simply stores training instances and defers the effort involved in learning until prediction time. When called upon to predict a test instance, LWC4.4 constructs an C4.4 using a weighted set of training instances in the neighborhood of the test instance. In a word, an obvious disadvantage with LWC4.4 is that it has relatively higher time complexity. So, enhancing LWC4.4's efficiency is one main direction for our future research.

4 Experimental Methodology and Results

We ran our experiments on 36 UCI data sets [13] selected by Weka [14], which represent a wide range of domains and data characteristics. In our experiments, we adopted the following four preprocessing steps.

1. Replacing missing attribute values: We don't handle missing attribute values. Thus, we used the unsupervised filter named *ReplaceMissingValues* in Weka to replace all missing attribute values in each data set.
2. Discretizing numeric attribute values: We don't handle numeric attribute values. Thus, we used the unsupervised filter named *Discretize* in Weka to discretize all numeric attribute values in each data set.
3. Removing useless attributes: Apparently, if the number of values of an attribute is almost equal to the number of instances in a data set, it is a useless attribute. Thus, we used the unsupervised filter named *Remove* in Weka to remove this type of attributes. In these 36 data sets, there are only three such attributes: the attribute "Hospital Number" in the data set "colic.ORIG", the attribute "instance name" in the data set "splice" and the attribute "animal" in the data set "zoo".

4. Sampling large data sets: For saving the time of running experiments (three lazy algorithms are used in our experiments), we used the unsupervised filter named *Resample* with the size of 20% in Weka to randomly sample each large data set having more than 5000 instances. In these 36 data sets, there are only three such data sets: "letter", "mushroom", and "waveform-5000".

We conducted extensive experiments to compare LWC4.4 on CLL with other related algorithms: C4.4, NB, KNN, NBTree, and LWNB. In our experiments, the CLL score of each classifier is computed using Equation 1. We use the implementation of C4.4 (J48 with Laplace smoothing but without tree pruning), NB (NaiveBayes), KNN (IBk without distance weighting), NBTree, and LWNB (LWL with NaiveBayes as the basic classifier) in Weka system. We use the LWL with C4.4 as the basic classifier for the implement of LWC4.4. Besides, we set

Table 1. The detailed experimental results on CLL and standard deviation. C4.4: C4.5 with laplace correction and without tree pruning; NB: naive Bayes; KNN: K-Nearest-Neighbor; NBTree: naive Bayes tree; LWNB: locally weighted naive Bayes; LWC4.4: locally weighted C4.4.

Dataset	C4.4	NB	KNN	NBTree	LWNB	LWC4.4
anneal	-7.72±2.17	-14.08±8.72	-8.25±3.33	-16.61±18.73	-11.95±8.13	-6.5±1.63
anneal.ORIG	-22.25±4.16	-23.85±7.26	-28.34±4.13	-29.84±12.46	-22.18±6.96	-20.25±4.3
audiology	-14.62±3.68	-65.71±21.12	-31.82±7.66	-87.29±33.27	-59.9±21.59	-14.47±3.47
autos	-12.83±2.52	-44.52±20.45	-18.89±6.38	-31.58±12.78	-39.65±19.65	-11.5±2.82
balance-scale	-52.4±3.99	-31.77±1.44	-66.96±2.28	-31.77±1.44	-31.63±1.4	-49.79±5.54
breast-cancer	-18.47±3.22	-18.12±6.01	-18.05±4.99	-18.89±4.99	-18.01±6.03	-19.49±3.63
breast-w	-11.23±4.59	-18.2±16.18	-9.18±5.19	-15.41±12.81	-18.21±16.23	-10.89±4.67
colic	-16.84±4.29	-30.29±9.6	-18.49±4.71	-35.19±15.06	-29.61±9.35	-17.65±5.2
colic.ORIG	-18.07±3.48	-20.41±5.55	-25.63±5.35	-33.22±11.57	-19.96±5.35	-18.73±3.11
credit-a	-27.72±3.16	-28.52±7.77	-30.82±7.76	-32.74±12.46	-28.44±7.83	-29.26±3.67
credit-g	-61.8±6.98	-52.42±7.29	-61.16±8.28	-58.95±17.18	-52.26±7.35	-63.4±6.9
diabetes	-43.91±4.94	-40.86±8.11	-44.54±6.41	-40.86±8.11	-40.78±8.08	-44.29±4.98
glass	-20.31±2.12	-24.16±4.21	-23.46±4.83	-33.31±10.14	-23.65±4.17	-20.21±2.03
heart-c	-15.7±4.69	-13.66±5.08	-14.52±5.9	-14.73±4.39	-13.64±5.14	-16.38±4.95
heart-h	-14.75±4.4	-13.69±5.2	-14.1±5.56	-14.87±5.61	-13.69±5.24	-15.14±5.2
heart-statlog	-13.95±3.45	-12.17±4.52	-12.04±4.46	-15.6±5.99	-12.21±4.53	-13.87±3.87
hepatitis	-5.7±2.13	-8.57±4.11	-7.43±4	-7.38±4.33	-8.55±4.08	-5.94±2.28
hypothyroid	-90.86±7.72	-97.44±19.4	-133.81±29.3	-97.81±19.8	-96.61±20.2	-91.6±9.1
ionosphere	-11.04±2.34	-35.01±13.73	-13.53±6.16	-24.27±12.88	-34.92±13.58	-10.65±2.51
iris	-3.67±1.33	-2.56±2.77	-3.1±2.63	-2.76±2.97	-2.53±2.87	-3.53±1.41
kr-vs-kp	-8.61±3.69	-93.21±8.36	-58.71±7.16	-34.67±19.94	-85.73±7.67	-7.45±4.11
labor	-2.47±1.45	-0.96±1.11	-1.67±0.99	-1.63±2.95	-1.01±1.2	-2.47±1.45
letter	-320.96±8.1	-564.72±52.8	-429.4±42.2	-618.49±64.8	-505.48±52.3	-294.91±6.7
lymph	-7.57±3.03	-6.43±3.16	-7.05±3.21	-9.67±7.7	-6.3±3.21	-7.12±2.67
mushroom	-2.53±0.87	-34.7±16.35	-0.55±0.78	-2.61±5.85	-20.64±11.49	-2.13±0.9
primary-tumor	-51.58±2.82	-65.27±10.04	-94.05±11.89	-73.04±15.37	-65.98±10.48	-50.58±2.97
segment	-49.66±6.05	-124.26±38.23	-56.84±7.02	-115.81±62.16	-109.49±35.34	-41.97±6.01
sick	-20.57±3.31	-45.74±11.62	-26.2±3.84	-41.58±13.58	-42.21±11.12	-20.49±4.61
sonar	-11.98±2.28	-20.87±12.2	-9±1.94	-34.68±23.7	-20.5±11.91	-12.21±2.1
soybean	-17.84±2.47	-26.41±9.7	-15.51±4.38	-30.65±15.5	-23.74±8.92	-16.61±2.45
splice	-66.6±8.6	-46.67±8.63	-178.56±18.2	-46.67±8.63	-45.69±8.82	-66.09±7.68
vehicle	-53.61±2.85	-169.76±27.29	-60.78±9.34	-131.69±26.73	-160.81±24.86	-52.37±3.81
vote	-7.31±4.78	-27.08±12.99	-10.03±4.48	-5.43±5.23	-24.33±11.9	-7.22±5.14
vowel	-70.91±4.74	-87.41±8.91	-61.38±5.2	-42.52±11.4	-66.63±6.95	-64.9±4.94
waveform-5000	-67.31±6.21	-74.37±17.55	-69.42±8.55	-104.36±47.13	-69.95±16.39	-66.48±6.32
zoo	-2.96±1.56	-1.21±1.12	-1.61±1.06	-0.96±0.89	-0.99±1.04	-2.67±1.56
Mean	**-34.62±3.84**	-55.14±11.62	-46.25±7.21	-53.82±15.52	-50.77±11.15	**-33.31±4.02**

Table 2. The compared results of two-tailed t-test on CLL with a 95% confidence level. An entry w/t/l in the table means that the algorithm at the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the algorithm at the corresponding column.

	C4.4	NB	KNN	NBTree	LWNB
NB	4/19/13				
KNN	2/23/11	13/17/6			
NBTree	4/21/11	4/29/3	6/21/9		
LWNB	5/19/12	19/17/0	5/19/12	5/27/4	
LWC4.4	**9/26/1**	13/18/5	10/25/1	11/21/4	12/19/5

the weighting kernel function to the inverse of their distance in LWNB[1] and LWC4.4, the number of neighbors to 5 in KNN, 50 in LWNB and LWC4.4. The CLL of each classifier on each data set was obtained via 10-fold cross validation. Run with the various algorithms were carried out on the same training sets and evaluated on the same test sets. In particular, the cross-validation folds are the same for all the experiments on each data set. Finally, we compare each pair of algorithms via two-tailed t-test with a 95% confidence level. According to the statistical theory, we speak of two results for a data set as being "significantly different" only if the probability of significant difference is at least 95%.

Table 1 shows the CLL and standard deviation of each algorithm on the test sets of each data set, the average CLL and standard deviation are summarized at the bottom of the table. Table 2 shows the results of two-tailed t-test with a 95% confidence level between each pair of algorithms in terms of CLL. each entry $w/t/l$ in Table 2 means that the algorithm at the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the algorithm at the corresponding column.

The detailed results displayed in Table 1 and Table 2 show that our improved algorithm LWC4.4 significantly outperforms all the other algorithms used to compare measured by CLL. Now, let's summarize the highlights as follows:

1. C4.4 significantly outperforms NB. In the 36 data sets we test, C4.4 wins in 13 data sets, only loses in 4 data sets. C4.4's average CLL is -34.62, much higher than that of NB (-55.14). This fact proves that C4.4 is an attractive alternative for class probability estimation.
2. C4.4 significantly outperforms KNN. In the 36 data sets we test, C4.4 wins in 11 data sets, only loses in 2 data sets. C4.4's average CLL is -34.62, much higher than that of KNN (-46.25). This fact also proves that C4.4 is an attractive alternative for class probability estimation.
3. LWC4.4 significantly outperforms C4.4. In the 36 data sets we test, LWC4.4 wins in 9 data sets, surprisingly loses in 1 data sets. LWNB's average CLL is -33.31, much higher than that of C4.4 (-34.62). This fact proves that locally weighted learning is an effective method for scaling up the class probability estimation performance of C4.4.
4. LWC4.4 significantly outperforms other two algorithms: NBTree (11 wins and 4 losses) and LWNB (12 wins and 5 losses). This fact is another

[1] It is a little different from LWNB published in UAI 2003.

Table 3. The detailed experimental results on AUC and standard deviation. C4.4: C4.5 with laplace correction and without tree pruning; NB: naive Bayes; KNN: K-Nearest-Neighbor; NBTree: naive Bayes tree; LWNB: locally weighted naive Bayes; LWC4.4: locally weighted C4.4.

Dataset	C4.4	NB	KNN	NBTree	LWNB	LWC4.4
anneal	93.78±2.9	95.9±1.3	93.66±5.92	96.45±0.28	96.1±1.2	96.1±1.31
anneal.ORIG	92.69±3.15	94.49±3.67	93.95±1.44	94.71±3.74	94.63±3.61	94.27±2.1
audiology	70.58±0.63	70.96±0.73	70.59±0.66	71.14±0.71	71.03±0.73	70.88±0.59
autos	90.73±4.52	89.18±4.93	89.29±3.84	93.93±2.68	90.77±5.1	94.1±3.3
balance-scale	63.06±6.18	84.46±4.1	65.84±2.89	84.46±4.1	84.01±4.4	62.24±5.37
breast-cancer	59.3±12.03	69.71±15.21	62.14±13.51	68.95±11.27	69.37±14.79	57.86±12.17
breast-w	97.85±1.86	99.19±0.87	98.71±1.38	99.21±0.73	99.21±0.86	98.29±1.6
colic	85.02±7.03	83.71±5.5	85.3±5.09	85.92±6.3	83.98±5.41	83.21±9.56
colic.ORIG	80.56±8.94	80.67±6.98	71.35±7.56	80.06±8.69	81.45±6.19	79.66±6.75
credit-a	89.42±3.1	92.09±3.43	91±3.14	91.48±3.52	92.22±3.41	88.24±2.89
credit-g	69.62±5	79.27±4.74	74.36±5	77.75±5.97	79.5±4.65	69.06±4.71
diabetes	75.5±5.76	82.31±5.17	77.57±3.98	82.31±5.17	82.44±5.19	75.46±5.87
glass	82.36±4.38	80.5±6.65	83.36±5.86	82.53±8.46	82.23±6.2	85.1±4.48
heart-c	83.1±1.19	84.1±0.54	83.85±0.84	83.96±0.51	84.1±0.56	83±1.24
heart-h	83.04±0.85	83.8±0.7	83.47±0.99	83.78±0.62	83.8±0.71	83.15±0.95
heart-statlog	81.36±9.15	91.3±4.19	89.79±4.36	89.66±3.42	91.06±4.24	82.76±9.13
hepatitis	82.03±14.04	88.99±8.99	83.14±12.51	88.03±8.29	88.99±8.99	81.7±12.83
hypothyroid	81.58±8.8	87.37±8.52	83.12±11.13	87.01±9.1	87.52±8.61	81.85±9.9
ionosphere	93.1±3.76	93.61±3.36	93.85±3.99	96.84±2.16	94.24±3.14	93.06±4.42
iris	97.33±2.63	98.58±2.67	97.75±3.22	98.08±2.67	98.58±2.67	99.25±1.39
kr-vs-kp	99.95±0.06	95.17±1.29	99.33±0.36	99.17±0.68	96.18±1.08	99.96±0.07
labor	74.17±31.04	98.33±5.27	92.5±7.03	100±0	98.33±5.27	88.33±31.48
letter	88.83±1.12	95.51±0.78	96.38±0.58	96.38±0.76	96.35±0.69	90.83±0.89
lymph	87.26±3.75	89.69±1.49	88.41±3.09	89.08±2.08	89.77±1.34	88.63±3.05
mushroom	99.98±0.02	99.59±0.18	99.97±0.02	99.97±0.1	99.86±0.09	100±0
primary-tumor	75.48±2.33	78.85±1.35	77.1±2.08	78.26±1.75	79.08±1.45	76.62±2.3
segment	98.85±0.32	98.51±0.46	99.01±0.16	99.09±0.43	98.73±0.39	99.36±0.2
sick	99.07±0.35	95.91±2.35	98.55±0.54	94.46±2.65	96.46±2.07	99.11±0.5
sonar	77.01±8.59	85.48±10.82	88.32±7.39	79.72±12.51	85.48±10.82	77.64±7.29
soybean	91.43±2.6	99.53±0.6	96.16±1.8	99.33±0.64	99.54±0.61	99.2±0.87
splice	98.14±0.72	99.41±0.22	96.99±0.97	99.41±0.22	99.43±0.22	98.16±0.63
vehicle	86.5±2.28	80.81±3.51	88.48±2.05	85.83±2.9	81.94±3.41	87.24±2.78
vote	96.77±2.96	96.56±2.09	97.39±1.49	98.82±1.61	96.77±1.92	98.21±1.8
vowel	91.28±2.46	95.81±0.84	97.58±0.64	98.66±0.68	97.46±0.55	93.28±2.02
waveform-5000	79.22±3.91	95.26±1.4	85.49±3.18	91.3±4.35	95.83±1.17	80.27±3.06
zoo	88.88±4.5	89.88±4.05	89.7±4.17	89.88±4.05	89.88±4.05	89.88±4.05
Mean	**85.69±4.80**	89.57±3.58	87.87±3.69	89.88±3.44	89.90±3.49	**86.83±4.49**

Table 4. The compared results of two-tailed t-test on AUC with a 95% confidence level. An entry w/t/l in the table means that the algorithm at the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the algorithm at the corresponding column.

	C4.4	NB	KNN	NBTree	LWNB
NB	12/20/4				
KNN	8/25/3	6/24/6			
NBTree	13/21/2	6/30/0	11/24/1		
LWNB	14/18/4	13/23/0	8/25/3	3/30/3	
LWC4.4	**8/28/0**	5/19/12	7/23/6	2/24/10	5/19/12

evidence to prove that LWC4.4 is an effective algorithm for addressing the class probability estimation problems.

In our experiments, we also observe the ranking performance of LWC4.4 in term of AUC (the area under the Receiver Operating Characteristics curve)

Table 5. The detailed experimental results on classification accuracy and standard deviation. C4.4: C4.5 with laplace correction and without tree pruning; NB: naive Bayes; KNN: K-Nearest-Neighbor; NBTree: naive Bayes tree; LWNB: locally weighted naive Bayes; LWC4.4: locally weighted C4.4.

Dataset	C4.4	NB	KNN	NBTree	LWNB	LWC4.4
anneal	99±0.98	94.32±2.38	96.88±2.15	98.33±1.6	95.65±2.13	99.11±0.88
anneal.ORIG	91.76±3.07	87.53±4.69	87.31±3.35	90.98±4.46	87.64±3.57	92.31±3.22
audiology	78.3±8	71.23±7.03	60.57±7.87	79.66±6.6	74.74±6.4	78.28±8.51
autos	81.45±7.48	64.83±11.18	66.29±8.28	78.12±7.02	69.17±8.96	82.9±9.22
balance-scale	69.3±4.25	91.36±1.38	83.84±4.71	91.36±1.38	91.36±1.38	72.02±4.69
breast-cancer	68.57±7.49	72.06±7.97	73.78±4.38	74.53±8.37	71.71±8.35	65.05±7.6
breast-w	92.99±3.66	97.28±1.84	94.99±2.81	96.99±1.85	97.28±1.84	92.99±3.72
colic	80.17±5.95	78.81±5.05	80.68±6.65	83.42±4.49	79.62±4.95	79.08±8.36
colic.ORIG	76.08±8.74	75.26±5.26	70.63±5.06	76.07±5.03	75.53±5.04	75.55±6.1
credit-a	83.19±3.5	84.78±4.28	85.07±3.62	85.07±3.81	85.22±4.36	80.72±4.21
credit-g	68.6±4.3	76.3±4.76	71.5±2.42	75.9±4.48	76.2±4.59	67.2±4.16
diabetes	69.54±5.12	75.4±5.85	69.14±1.84	75.4±5.85	75.4±5.38	69.8±3.85
glass	58.83±7.73	60.32±9.69	58.92±7.8	56.99±10.66	60.35±8.98	56.95±8.11
heart-c	74.26±11.46	84.14±4.16	81.41±12.65	82.16±3.66	84.14±4.16	72.59±11.54
heart-h	72.78±11	84.05±6.69	81.36±6.65	82.36±7.71	84.05±6.07	73.11±9.83
heart-statlog	75.93±8.95	83.7±5	80.74±6	82.59±6.06	83.7±5	74.44±7.29
hepatitis	81.25±11.52	83.79±8.79	84.46±6.25	83.79±9.91	83.13±8.22	79.33±11.15
hypothyroid	92.5±0.58	92.79±1.02	93.03±0.89	93.08±1	92.79±0.99	92.07±0.92
ionosphere	84.63±4.45	90.89±3.49	89.44±3.34	91.45±3.3	90.89±3.49	85.2±5.3
iris	92.67±5.84	94.67±8.2	93.33±6.29	94±7.98	95.33±8.34	92.67±5.84
kr-vs-kp	99.41±0.45	87.89±1.81	96.03±1.19	97.09±2.38	88.86±1.35	99.41±0.43
labor	77.67±15.64	93.33±11.65	91.67±11.79	91.67±11.79	93.33±11.65	79.33±15.22
letter	70.3±1.67	66.15±2.15	73.3±2.24	73.9±1.69	69.7±2.34	72.58±2.28
lymph	74.29±12.56	85.67±9.55	82.33±9.81	83.05±8.01	86.33±8.8	75.67±9.55
mushroom	99.75±0.32	93.84±2.02	99.82±0.3	99.88±0.26	95.57±2.16	99.75±0.32
primary-tumor	38.91±4.97	46.89±4.32	41.26±8.05	46.9±6.22	48.37±4.08	38.03±3.83
segment	92.86±1.39	88.92±1.95	90.74±1.61	92.51±1.77	90±2.14	93.51±1.71
sick	97.83±0.61	96.74±0.53	97.51±0.59	97.96±0.73	96.85±0.48	97.69±0.66
sonar	67.69±10.94	77.5±11.99	80.79±10.06	73.62±13.8	77.98±12.03	66.74±9.25
soybean	92.68±1.56	92.08±2.34	90.76±3.76	92.24±2.08	92.96±2.5	92.67±1.72
splice	91.57±1.37	95.36±1	79.81±2.81	95.36±1	95.42±1	90.78±1.5
vehicle	69.03±2.63	61.82±3.54	70.57±3.02	68.1±5	62.41±4.04	69.5±3.71
vote	94.96±3.83	90.14±4.17	94.03±2.69	95.41±4.03	90.6±3.93	94.95±4.41
vowel	75.66±5.18	67.07±4.21	81.31±1.73	88.59±2.74	75.56±5.08	77.47±4.37
waveform-5000	65.8±3.77	79.7±4	70.4±4.09	79.4±3.31	80.7±3.68	65.8±3.08
zoo	92.18±8.94	94.18±6.6	92.09±6.3	95.09±5.18	96.18±6.54	92.18±8.94
Mean	**80.34±5.55**	82.24±5.02	81.55±4.81	84.53±4.87	83.19±4.83	**80.21±5.43**

Table 6. The compared results of two-tailed t-test on classification accuracy with a 95% confidence level. An entry w/t/l in the table means that the algorithm at the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the algorithm at the corresponding column.

	C4.4	NB	KNN	NBTree	LWNB
NB	11/15/10				
KNN	4/26/6	8/23/5			
NBTree	10/26/0	10/26/0	9/27/0		
LWNB	11/17/8	7/29/0	7/23/6	0/28/8	
LWC4.4	**2/33/1**	10/16/10	6/24/6	0/24/12	9/16/11

[15,16,17] shown in Table 3 and Table 4. Fortunately, LWC4.4 also significantly outperforms C4.4. In the 36 data sets we test, LWC4.4 wins in 8 data sets, surprisingly loses in 0 data sets, and ties all the other data sets. Besides, an interested observation is that LWC4.4 almost ties C4.4 in term of classification

accuracy shown in Table 5 and Table 6. So, we can draw a conclusion that locally weighted learning can be used to improve C4.4 for class probability estimation and ranking but not for classification.

5 Conclusions and Future Work

C4.4 is one of the most popular algorithms for addressing the class probability estimation problems. C4.4 is an improved version of C4.5, in which two techniques respectively called Laplace correction and turning of tree pruning are used. Motivated by the success of using locally weighted learning to improve linear regression for regression and using locally weighted learning to improve naive Bayes for classification, we present to apply locally weighted learning to C4.4 to scale up its class probability estimation performance. We call our improved algorithm locally weighted C4.4, simply LWC4.4. Our experimental results show that LWC4.4 is surprisingly effective in class probability estimation and significantly outperforms all the other algorithms used to compare.

Aiming at accurate classification, Friedman et al. [18] presented another lazy decision tree learning algorithm, simply called LazyDT. LazyDT creates a path in a tree for a test instance instead of a neighborhood. According to the experimental results in [18], LazyDT is certainly effective in classification. However, it is not clear whether LazyDT also is effective in class probability estimation. In our future work, we will compare LWC4.4 with LazyDT.

References

1. Grossman, D., Domingos, P.: Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood. In: Proceedings of the Twenty-First International Conference on Machine Learning, Banff, Canada, pp. 361–368. ACM Press, New York (2004)
2. Guo, Y., Greiner, R.: Discriminative Model Selection for Belief Net Structures. In: Proceedings of the Twentieth National Conference on Artificial Intelligence, pp. 770–776. AAAI Press (2005)
3. Provost, F.J., Domingos, P.: Tree Induction for Probability-Based Ranking. Machine Learning 52(3), 199–215 (2003)
4. Frank, E., Hall, M., Pfahringer, B.: Locally Weighted Naive Bayes. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 249–256. Morgan Kaufmann, San Francisco (2003)
5. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally Weighted Learning. Artificial Intelligence Review 11(1-5), 11–73 (1997)
6. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1993)
7. Provost, F., Fawcett, T., Kohavi, R.: The case against accuracy estimation for comparing induction algorithms. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 445–453. Morgan Kaufmann, San Francisco (1998)
8. Friedman, Geiger, Goldszmidt: Bayesian Network Classifiers. Machine Learning 29, 131–163 (1997)

9. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Francisco, CA (1988)
10. Chickering, D.M.: Learning Bayesian networks is NP-Complete. In: Fisher, D., Lenz, H. (eds.) Learning from Data: Artificial Intelligence and Statistics V, pp. 121–130. Springer, Heidelberg (1996)
11. Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pp. 202–207. AAAI Press (1996)
12. Li, C., Jiang, L.: Using Locally Weighted Learning to Improve SMOreg for Regression. In: Yang, Q., Webb, G. (eds.) PRICAI 2006. LNCS (LNAI), vol. 4099, pp. 375–384. Springer, Heidelberg (2006)
13. Merz, C., Murphy, P., Aha, D.: UCI repository of machine learning databases. In Dept of ICS, University of California, Irvine (1997), `http://www.ics.uci.edu/mlearn/MLRepository.html`
14. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005), `http://prdownloads.sourceforge.net/weka/datasets-UCI.jar`
15. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30, 1145–1159 (1997)
16. Hand, D.J., Till, R.J.: A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning 45, 171–186 (2001)
17. Ling, C.X., Huang, J., Zhang, H.: AUC: a statistically consistent and more discriminating measure than accuracy. In: Proceedings of the International Joint Conference on Artificial Intelligence IJCAI03, Morgan Kaufmann, San Francisco (2003)
18. Friedman, J., Kohavi, R., Yun, Y.: Lazy decision trees. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 717–724. The AAAI Press, Menlo Park, CA (1996)