Instance Cloning Local Naive Bayes

Liangxiao Jiang[1] *, Harry Zhang[2], and Jiang Su[2]

[1] Faculty of Computer Science, China University of Geosciences
Wuhan, China 430074
[2] Faculty of Computer Science, University of New Brunswick
P.O. Box 4400, Fredericton, NB, Canada E3B 5A3

Abstract. The instance-based k-nearest neighbor algorithm (KNN)[1] is an effective classification model. Its classification is simply based on a vote within the neighborhood, consisting of k nearest neighbors of the test instance. Recently, researchers have been interested in deploying a more sophisticated local model, such as naive Bayes, within the neighborhood. It is expected that there are no strong dependences within the neighborhood of the test instance, thus alleviating the conditional independence assumption of naive Bayes. Generally, the smaller size of the neighborhood (the value of k), the less chance of encountering strong dependences. When k is small, however, the training data for the local naive Bayes is small and its classification would be inaccurate. In the currently existing models, such as LWNB [3], a relatively large k is chosen. The consequence is that strong dependences seem unavoidable.
In our opinion, a small k should be preferred in order to avoid strong dependences. We propose to deal with the problem of lack of local training data using sampling (cloning). Given a test instance, clones of each instance in the neighborhood is generated in terms of its similarity to the test instance and added to the local training data. Then, the local naive Bayes is trained from the expanded training data. Since a relatively small k is chosen, the chance of encountering strong dependences within the neighborhood is small. Thus the classification of the resulting local naive Bayes would be more accurate. We experimentally compare our new algorithm with KNN and its improved variants in terms of classification accuracy, using the 36 UCI datasets recommended by Weka [8], and the experimental results show that our algorithm outperforms all those algorithms significantly and consistently at various k values.

1 Introduction

Classification is a fundamental issue in machine learning. A typical problem setting for classification is that, given a set of training instances with class labels, a classifier is trained and used to predict the class of an unseen instance. An instance is represented by a vector of attributes. In this paper, an instance x

* This work was done when the author was a visiting scholar at University of New Brunswick.

is described by the attribute vector $< a_1(x), a_2(x), \ldots, a_n(x) >$, where $a_i(x)$ denotes the value of the ith attribute A_i of x.

Instance-based learning methods [1] are often used for classification. The most basic instance-based method is the instance-based k-nearest neighbor algorithm (KNN). In KNN, an instance of n attributes corresponds to a point in the n-dimensional Euclidean space $\Re^n$. The standard Euclidean distance is often used as the distance between two instances x and y, defined as follows.

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

When all attributes are nominal, this function can be simplified as:

$$d(x, y) = \sum_{i=1}^{n} (1 - \delta(a_i(x), a_i(y))), \quad (2)$$

where δ is a function that $\delta(u, v) = 1$ if $u = v$.

KNN classifies an instance x by finding its k nearest neighbors y_1, $\cdots$, y_k, and then assigning the most common class of the k nearest neighbors to x, as shown below:

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, c(y_i)), \quad (3)$$

where $c(y_i)$ is the class of y_i. Essentially, the classification of KNN is based on a vote within the neighborhood that consists of the k nearest neighbors of the test instance.

An obvious approach to improving KNN is to weight the contribution of each of k neighbors according to their distance to the test instance x, by giving greater weight to closer neighbors. The resulting classifier is called instance-based k-nearest neighbor with distance weighted (KNNDW), defined as follows.

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \frac{\delta(c, c(y_i))}{d(y_i, x)^2}. \quad (4)$$

KNN has been widely used for decades due to its simplicity, effectiveness and robustness. However, the classification of KNN based on voting is quite simple. It is believed that a more sophisticated local model within the neighborhood, instead of voting, would improve classification performance.

Naive Bayes is a simple, but effective classification model [5], in which all the attributes are assumed independent given the class (the conditional independence assumption). It has been observed that naive Bayes performs well when the training data is small [4]. Thus it is suitable to be a local model within another model, such as a decision tree.

In KNN, it is a natural thought to train a local naive Bayes for a test instance using only the k nearest neighbors. Recently, researchers have paid considerable

attention to investigate the approach to combining KNN with naive Bayes [12, 11, 3]. Although the conditional independence assumption of naive Bayes is always violated on the training data as a whole, it is expected that the dependences within the neighborhood of the test instance is weaker than that on the whole data and thus naive Bayes classifies better. Generally, keeping the size of the neighborhood (the values of k) small will reduce the chance of encountering strong dependences. When k is small, however, the probability estimation in naive Bayes that is based on frequency in the training data, is not reliable. Thus, its classification would be inaccurate.

In this paper, we propose an method based on sampling to deal with this issue. For each neighbor, a number of clones are generated and added into the local training data in terms of its similarity to the test instance. Then, a naive Bayes is trained from the expanded training data. By that means, a small k value could be chosen and strong attribute dependences could be avoided. Thus a better classifier is expected. Our experimental results show that our new model outperforms KNN and its variants significantly.

The rest of the paper is organized as follows. In Section 2, we introduce the related work on combining KNN with naive Bayes. In Section 3, we present our model and the experimental results in detail. In Section 4, we make a conclusion and outline our main directions for future research.

2 Related Work

Naive Bayes is a simple but effective classifier. Although its conditional independence assumption is often violated, it performs surprisingly well in classification [2]. In addition, it performs well when the size of training data is small [4]. This feature makes it especially fit to be a local model embedded into another model, such as a decision tree, a KNN. Kohavi [4] proposes a model, NBTree, in which a local naive Bayes is deployed on each leaf of a traditional decision tree. An NBTree classifies an instance using the local naive Bayes on the leaf into which it falls.

KNN has attracted much attention from researchers for decades, due in part to its age and simplicity [1]. In recently years, researchers have done considerable work on combining KNN with naive Bayes [12, 11, 3]. The idea for combining KNN with naive Bayes is quite straightforward. Like all lazy learning methods, the training data is simply stored, and learning is deferred until classification time. Whenever a new (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. The classification of the local naive Bayes is based on the following equation.

$$c(x) = \arg\max_{c \in C} p(c) \prod_{i=1}^{n} p(a_i(x)|c), \tag{5}$$

where x is the test instance of n attributes. The parameters of the local naive Bayes are the probabilities $p(c)$ and $p(a_i(x)|c)$ in Equation 5 that are estimated from the local training data (the k nearest neighbors of x) based on frequency.

Frank et al. [3] present an model to combine KNN with naive Bayes, called locally weighted Naive Bayes(LWNB). In LWNB, each of nearest neighbors is weighted in terms of its distance to the test instance. Then a local naive Bayes is built from the weighted training instances. Their experiments show that LWNB outperforms naive Bayes significantly. Our work is inspired by this work.

Zheng and Webb [12] propose an approach, lazy Bayesian rule (LBR). LBR does not directly use the k nearest neighbors of the test instance as the training data for the local naive Bayes. Instead, before classifying a test instance, LBR generates a rule most appropriate to the test instance. The training instances that satisfy the antecedent of the rule are chosen as the training data for the local naive Bayes, and this local naive Bayes only uses those attributes that do not appear in the antecedent of the rule. In this paper, however, we are interested in learning the local model directly based on the k nearest neighbors of the test instance.

Xie et al. [11] propose a model, selective neighborhood naive Bayes (SNNB), in which multiple naive Bayes are learned by using different k values and a local naive Bayes is trained for each k value. The most accurate one is used to classify the test instance. In SNNB, a set of naive Bayes are trained. In this paper, we focus on training a single local model.

In this paper, we use KNNNB to denote the algorithm that directly uses the k nearest neighbors of the test instance to train a local naive Bayes.

3 Learning Local Naive Bayes Using Instance Sampling

3.1 Size of Neighborhood and Attribute Dependences

Most of the existing research works on combining KNN with naive Bayes are motivated by improving naive Bayes through relaxing the conditional independence assumption using lazy learning. It is expected that there are no strong dependences within the k nearest neighbors of the test instance, although the attribute dependences might be strong in the whole data. Essentially, they are looking for a sub-space of the instance space in which the conditional independence assumption is true or almost true. The size of the neighborhood, or the value of k, is critical. In general, small neighborhood helps to reduce the chance of encountering strong dependences [3]. Thus, a small k is preferred.

Frank et al. [3] presents a problem that predicts whether an instance belongs to a black or white square on a checker board given its x and y coordinates. In that problem, strong dependences between the two attributes exist. They show that KNN, KNNDW and LWNB performs well at $k \leq 5$, and degrade as k increases. That example shows that small k value is desirable when attribute dependences are strong. In addition, small neighborhood conforms closer to the data. In fact, KNN and KNNDW generally degrade in performance as k increases.

When k is small, such as 5 or 10, however, the training data is small. The parameters of naive Bayes cannot be accurately estimated from the training data. Thus, the classification of a local naive Bayes would be inaccurate. In NBTree[4], a threshold on the size of the training data on a decision node is set to avoid this problem. That is, there should be at least 30 training instances at a decision node. In LWNB, Laplace correction has been used to smooth probability estimation, and a relatively large k, such as $k = 30, 40, 50$, is chosen.

We believe that keeping the size of the neighborhood small would help reducing the chance of having strong dependences and thus improving classification accuracy. We propose to a novel approach to handling the issue of lack of training data by expanding the neighborhood. We"clone" each neighbor of the testing instance and add the clones to the training data. Thus, the parameters in naive Bayes can be estimated more accurately and reliably, and the classification of the local naive Bayes is more accurate.

There has been similar work on expanding or reducing training data by sampling to deal with the issue of class unbalance in machine learning [9]. When the class distribution of training data is highly unbalanced, instances in the majority class are eliminated (under-sampling), or instances are replicated in the minority class (over-sampling). Either way alters the class distribution of the training data. Our sampling is similar in the sense that copies of instances are added into the training data. The difference is that the sampling in dealing with class unbalance is typically based on a probability distribution, and our sampling is based on an explicit distance function and does not alter the class distribution.

3.2 A Learning Algorithm Based on Instance Cloning

At first, let us define a function to measure the similarity between two instances with nominal attributes. Let x and y are two instances, the similarity, denoted by $s(x, y)$, between them is defined as:

$$s(x, y) = \sum_{i=1}^{n} \delta(a_i(x), a_i(y)). \tag{6}$$

$s(x, y)$ is a function that simply counts the number of identical attributes of x and y.

Given a test instance x, we find its k nearest neighbors and put them into the local training data. For each neighbor y, we use Equation 6 to compute the similarity $s(x, y)$. Then, $s(x, y)$ clones of y are added to the local training data. A local naive Bayes is learned from the expanded training data with which x is classified. We call our method *instance cloning local naive Bayes*, or simply ICLNB. The ICLNB algorithm is depicted below.

Algorithm ICLNB(**T**, k, x)
Input : a set **T** of training instances, integer k, and a test instance x.
Output : the class of x

1. Use the distance function in Equation 2 to find x's k nearest neighbors $y_1, \cdots, y_k$, from **T**.

2. Local training set $\mathbf{L} = \{y_1, \cdots, y_k\}$
3. For each neighbor y_i of x
   - Compute $s(x, y_i)$ using the similarity function in Equation 6.
   - Add $s(x, y_i)$ clones of y_i to $\mathbf{L}$.
4. Create a local naive Bayes **NB** using $\mathbf{L}$ as the training data.
5. Classify x using **NB** and return the class label of x.

It is interesting to notice the similarity and difference between instance sampling (or cloning) and instance weighting [3]. In instance weighting, each training instance is assigned a different weight and thus plays a different role in classification. In instance sampling, each training instance is "cloned" different times. More clones generated, more important for that instance in classification. Thus, both change the importance of an instance. However, instance weighting generally does not aim at helping to improve probability estimation, while instance sampling does. "Cloning" expands the training data, and thus leads to more reliable probability estimation. Specifically, our instance sampling is simpler than the instance weighting in [3] in that our similarity function is very simple and there is no need to normalize the instance weights.

3.3 Experimental Methodology and Results

We ran our experiments on the 36 UCI data sets recommended by Weka [10], which are listed in Table 1. All these data sets come from the UCI repository [6]. We downloaded these data sets in format of *arff* from main web of Weka [8].

All the preprocessing stages of data sets were carried out by the Weka system. They mainly include the following three processes:

1. We used the filter of ReplaceMissingValues in Weka to replace the missing values of attributes.
2. We used the filter of Discretize in Weka to discretize numeric attributes. Thus, all the attributes are treated as nominal.
3. It is well-known that, if the number of values of an attribute is almost equal to the number of instances in a data set, this attribute does not contribute any information to classification. So we use the filter of Remove in Weka to delete these attributes. In the 36 data sets, there only exists three this type of attributes, namely Hospital Number in data set horse-colic.ORIG, Instance Name in data set Splice and Animal in data set zoo.

In our experiments, we use the Laplace estimation to avoid the zero-frequency problem. Assume that there are p instances of class c, N total instances, and C total classes in the training data. The frequency-based estimation calculates the estimated probability $p(c) = \frac{p}{N}$. The Laplace estimation calculates the estimated probability $p(c) = \frac{p+1}{N+C}$. In the Laplace estimation, $p(a_i(x)|c) = \frac{1+N_{ic}}{N_i+N_c}$, where N_{ic} is the number of instances in class c and with $A_i = a_i(x)$, N_c is the number of instances in class c, and N_i is the number of values for attribute A_i.

Table 1. Description of the data sets used in the experiments.

dataset	Size	Number of Attribute	classes	missing value	Numeric
anneal	898	39	6	Y	Y
anneal.ORIG	898	39	6	Y	Y
audiology	226	70	24	Y	N
autos	205	26	7	Y	Y
balance	625	5	3	N	Y
breast	286	10	2	Y	N
breast-w	699	10	2	Y	N
colic	368	23	2	Y	Y
colic.ORIG	368	28	2	Y	Y
credit-a	690	16	2	Y	Y
credit-g	1000	21	2	N	Y
diabetes	768	9	2	N	Y
Glass	214	10	7	N	Y
heart-c	303	14	5	Y	Y
heart-h	294	14	5	Y	Y
heart-s	270	14	2	N	Y
hepatitis	155	20	2	Y	Y
hypoth.	3772	30	4	Y	Y
ionosphere	351	35	2	N	Y
iris	150	5	3	N	Y
kr-vs-kp	3196	37	2	N	N
labor	57	17	2	Y	Y
letter	20000	17	26	N	Y
lymph.	148	19	4	N	Y
mushroom	8124	23	2	Y	N
p.-tumor	339	18	21	Y	N
segment	2310	20	7	N	Y
sick	3772	30	2	Y	Y
sonar	208	61	2	N	Y
soybean	683	36	19	Y	N
splice	3190	62	3	N	N
vehicle	846	19	4	N	Y
vote	435	17	2	Y	N
vowel	990	14	11	N	Y
waveform	5000	41	3	N	Y
zoo	101	18	7	N	Y

We conducted experiments to compare our algorithm ICLNB with KNN, KNNDW, KNNNB and LWNB in terms of classification accuracy. We implemented ICLNB, KNNDW, and KNNNB within the Weka framework [10], and used the implementation of KNN and LWNB (LWL with a base classifier being Naive Bayes) in Weka. In all experiments, the accuracy of each algorithm was based on the percentage of correct classifications on the test sets of each data set. The accuracy of each algorithm was measured via the ten-fold cross validation for all data sets. Runs 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 were the same for all the experiments on each data set. Finally, we conducted two-tailed t-test with significantly different probability of 0.95 to compare our algorithm with other algorithms. That is, we speak of two results for a data set as being "significantly different" only if the difference is statistically significant at the 0.05 level according to the corrected two-tailed t-test [7].

Table 2 and 3 show the accuracy and standard deviations of each algorithm on each data set, and the average accuracy and deviation over all the data sets are summarized at the bottom of the table. Table 4 and 5 shows the results of two-tailed t-test between each pair of algorithms, and each entry $w/t/l$ 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 2 and 3 show that our algorithm outperforms all the other algorithms significantly. Now, we summarize the highlights as follows:

1. ICLNB outperforms the traditional k-nearest algorithms KNN and KNNDW significantly. From our experiments, KNNDW is significantly better than KNN. Compared to KNNDW, ICLNB wins in 6 data sets, ties in 30 data sets and loses in 0 data set, when $k = 5$; and ICLNB wins in 5 data sets, ties in 31 data sets and loses in 0 data set, when $k = 10$.
2. ICLNB outperforms the existing algorithms of combining KNN with naive Bayes: KNNNB and LWNB significantly. Compared to KNNNB, ICLNB wins in 6 data sets, ties in 30 data sets and loses in 0 data set, when $k = 5$; and ICLNB wins in 7 data sets, ties in 28 data sets and loses in 1 data set, when $k = 10$. Compared to LWNB, ICLNB wins in 5 data sets, ties in 31 data sets and loses in 0 data set, when $k = 5$; and ICLNB wins in 6 data sets, ties in 30 data sets and loses in 0 data set, when $k = 10$.
3. In terms of the average classification accuracy, ICLNB is the best among all the algorithms compared. When $k = 5$, ICLNB's average classification accuracy is 84.71%, and the highest average classification accuracy of the other algorithms is 84.09% from KNNNB. When $k = 10$, ICLNB's average classification accuracy is 85.01%, and the highest average classification accuracy of the other algorithms is 84.36% from KNNDW.

Table 2. Experimental results for instance cloning local naive Bayes (ICLNB) versus instance-based k-nearest neighbor (KNN), instance-based k-nearest neighbor with distance weighted (KNNDW), instance-based k-nearest neighbor naive Bayes (KNNNB), locally weighted naive Bayes (LWNB) and naive Bayes (NB): percentage of correct classifications and standard deviation when $k = 5$.

Datasets	ICLNB	KNN	KNNDW	KNNNB	LWNB	NB
anneal	98.66±1.26	96.88±2.15	98.55±1.05	97.33±2.24	98.78±1.11	94.32±2.38
anneal.O	91.2±3.08	87.31±3.35	90.09±2.93	88.09±2.96	91.53±3.04	87.53±4.69
audiology	77.81±9.23	60.57±7.87	75.57±8.92	68.58±6.77	77.37±9.21	71.23±7.03
autos	83.83±6.63	66.29±8.28	83.4±7.61	78.55±9.2	81.4±5.23	64.83±11.2
balance	83.84±4.41	83.84±4.71	83.84±4.71	84.16±4.03	83.99±4.22	91.36±1.38
breast	73.85±8.84	73.78±4.38	75.55±7.2	75.55±3.78	72.09±9.52	72.06±7.97
breast-w	96.85±2	94.99±2.81	95.28±2.79	96.57±2.54	95.99±2.52	97.28±1.84
colic	77.72±3.55	80.68±6.65	82.33±5.35	81.51±7.01	78.56±5.38	78.81±5.05
colic.O	76.09±4.91	70.63±5.06	73.35±5.64	73.08±6	75.81±5.5	75.26±5.26
credit-a	82.61±2.73	85.07±3.62	84.93±2.99	85.22±3.19	82.75±4.24	84.78±4.28
credit-g	73.5±3.03	71.5±2.42	72.6±3.41	73.3±3.33	70.9±5.17	76.30±4.76
diabetes	72.14±4.68	69.14±1.84	70.58±3.33	71.23±3.25	69.15±4.66	75.40±5.85
glass	65.41±8.64	58.92±7.8	62.23±6.79	64.52±7.69	61.21±7.77	60.32±9.69
heart-c	79.14±8.59	81.41±12.7	81.42±11.4	82.75±8.94	80.77±9.36	84.14±4.16
heart-h	82.7±6.35	81.36±6.65	81.34±6.27	81.34±6.3	81.31±5.99	84.05±6.69
heart-s	81.11±5.91	80.74±6	81.48±5.79	81.85±5.64	80.37±7.42	83.70±5.00
hepatitis	84.42±7.61	84.46±6.25	82.54±4.41	83.88±6.2	80.63±8.57	83.79±8.79
hypoth.	93.05±0.93	93.03±0.89	93.05±0.84	93.03±0.89	92.29±0.86	92.79±1.02
ionosphere	92.31±3.31	89.44±3.34	90.02±2.8	90.02±3.11	91.73±3.43	90.89±3.49
iris	95.33±5.49	93.33±6.29	93.33±7.03	94.67±5.26	94±5.84	94.67±8.20
kr-vs-kp	97.97±0.71	96.03±1.19	96.97±0.96	96.9±0.85	97.31±0.86	87.89±1.81
labor	86.33±13.3	91.67±11.8	91.67±11.8	91.67±11.8	90±11.65	93.33±11.7
letter	92.16±0.32	88.02±0.63	90.17±0.55	89.87±0.4	90.8±0.4	70.00±0.81
lymph.	81.67±9.18	82.33±9.81	82.29±10.9	83.62±9.52	82.9±11.18	85.67±9.55
mushroom	100±0	100±0	100±0	100±0	100±0	95.57±0.45
p.-tumor	43.65±3.26	41.26±8.05	42.44±4.98	44.22±5.71	40.38±5.71	46.89±4.32
segment	94.85±0.95	90.74±1.61	93.38±1.55	91.6±1.26	94.63±1.49	88.92±1.95
sick	98.2±0.48	97.51±0.59	97.72±0.49	97.72±0.49	98.01±0.53	96.74±0.53
sonar	80.81±7.38	80.79±10.06	80.79±8.72	81.74±6.72	82.24±10.3	77.50±12.0
soybean	93.11±2.42	90.76±3.76	91.79±3.36	91.94±3.22	92.38±3.02	92.08±2.34
splice	86.18±2.02	79.81±2.81	82.23±2.63	85.86±1.47	82.6±2.3	95.36±1.00
vehicle	72.45±3.78	70.57±3.02	71.27±4.46	72.46±3.41	70.57±5.64	61.82±3.54
vote	95.18±2.51	94.03±2.69	93.81±3.06	94.95±2.37	93.58±4.27	90.14±4.17
vowel	94.14±1.7	81.31±1.73	92.42±3.06	90.1±2.17	94.65±2.52	67.07±4.21
waveform	74.22±1.59	73.42±1.55	73.92±1.86	74.42±1.85	69.98±1.94	79.96±1.92
zoo	97.09±4.69	92.09±6.3	96.09±5.05	95.09±5.18	97.09±4.69	94.18±6.60
Mean	84.71±4.32	82.05±4.68	84.12±4.58	84.09±4.30	83.83±4.88	82.41±4.88

Table 3. Experimental results for instance cloning local naive Bayes (ICLNB) versus instance-based k-nearest neighbor (KNN), instance-based k-nearest neighbor with distance weighted (KNNDW), instance-based k-nearest neighbor naive Bayes (KNNNB), locally weighted naive Bayes (LWNB) and naive Bayes (NB): percentage of correct classification and standard deviation when $k = 10$.

Datasets	ICLNB	KNN	KNNDW	KNNNB	LWNB	NB
anneal	99±0.82	95.88±1.97	98.55±1.05	97.1±2.05	98.89±1.05	94.32±2.38
anneal.O	90.98±3.47	84.41±3.3	89.75±3.1	86.64±3.48	91.64±3.04	87.53±4.69
audiology	77.83±7.59	58.79±8.3	74.72±6.74	68.1±6.28	77.79±8.11	71.23±7.03
autos	80.38±6.8	62.52±8.03	82.88±6.63	76.57±8.47	81.88±5.84	64.83±11.2
balance	83.84±4.41	83.84±4.71	83.84±4.71	84.16±4.03	83.99±4.22	91.36±1.38
breast	73.15±8.78	73.09±4.25	75.22±5.58	74.85±4.09	72.8±8.74	72.06±7.97
breast-w	96.99±1.84	93.99±3.42	94.28±3.5	96.57±2.54	96.13±2.45	97.28±1.84
colic	77.98±5.65	83.13±6.29	84.23±5.09	83.13±5.75	80.98±3.82	78.81±5.05
colic.O	76.09±3.57	69.82±3.4	73.09±5.23	73.09±5.54	75.8±4.25	75.26±5.26
credit-a	83.77±2.54	86.09±4.39	85.51±3.74	86.52±4.21	83.33±4.11	84.78±4.28
credit-g	73.5±3.1	71.9±3.28	73.7±3.13	74.4±3.5	72.1±4.18	76.30±4.76
diabetes	72.01±5.47	69.02±2.19	69.27±4.12	72.01±4.5	69.4±4.23	75.40±5.85
glass	67.75±8.15	57.06±8.03	60.82±7.99	64.55±8.76	60.74±5.83	60.32±9.69
heart-c	78.85±7.28	81.09±9.77	81.42±9.55	83.1±7.37	80.11±8.99	84.14±4.16
heart-h	81.33±6.14	82.02±6.06	81.68±5.66	81.67±4.73	82.34±6.06	84.05±6.69
heart-s	80.74±6	82.22±7.37	82.96±6.1	82.59±5.53	80±7.45	83.70±5.00
hepatitis	84.38±7.13	84.5±6.22	83.25±5.35	83.92±6.93	82.54±6.17	83.79±8.79
hypothy.	93.21±0.66	93.08±0.64	93.16±0.58	93.21±0.62	92.37±0.87	92.79±1.02
ionosphere	90.9±4.58	89.74±2.78	89.74±3.37	90.31±3.62	91.44±3.03	90.89±3.49
iris	94.67±6.13	93.33±6.29	93.33±7.03	94±5.84	94±5.84	94.67±8.20
kr-vs-kp	97.59±0.81	95.06±1.34	96.84±0.95	96.46±0.78	97.56±0.76	87.89±1.81
labor	90±14.1	85.67±14.5	89.67±11.9	91.67±11.9	90±11.7	93.33±11.7
letter	92.83±0.33	86.56±0.67	89.68±0.64	89.73±0.55	91.25±0.43	70.00±0.81
lymph.	83±10.8	80.86±12.0	82.86±13.1	85±9.18	82.29±10.92	85.67±9.55
mushroom	100±0	99.91±0.08	100±0	100±0	100±0	95.57±0.45
p.-tumor	43.34±3.7	42.47±5.67	43.03±5.86	46±4.85	40.97±5.6	46.89±4.32
segment	95.45±0.82	89.65±1.84	92.68±1.59	91.6±1.54	94.81±1.43	88.92±1.95
sick	98.12±0.55	97.03±0.73	97.45±0.6	97.56±0.53	98.01±0.53	96.74±0.53
sonar	81.33±9.57	81.33±8.42	80.38±8.44	81.83±9.39	84.14±6.38	77.50±12.0
soybean	93.84±2.39	89.01±2.12	91.94±3.29	91.94±3.06	92.96±2.77	92.08±2.34
splice	91.76±1.45	83.26±2.42	85.2±2.11	90.5±1.25	86.65±1.69	95.36±1.00
vehicle	71.28±4.77	68.68±2.74	70.8±4.48	70.69±3.01	72.46±4.5	61.82±3.54
vote	96.1±2.64	92.9±3.61	93.36±3.28	94.95±2.81	93.57±3.86	90.14±4.17
vowel	93.54±1.73	67.68±4.04	91.62±3.63	89.49±3.02	94.24±2.02	67.07±4.21
waveform	76.76±1.33	76.36±1.15	76.34±1.15	77.2±1.35	72.7±1.86	79.96±1.92
zoo	98.09±4.03	89.18±9.78	96.09±5.05	95.09±5.18	97.09±4.69	94.18±6.60
Mean	85.01±4.42	81.14±4.77	84.15±4.56	84.34±4.35	84.36±4.37	82.41±4.88

4. ICLNB outperforms naive Bayes significantly, just as KNNNB and LWNB do. That verifies the idea that there are weaker attribute dependences within the neighborhood of the test instance.

From our experiments, we have other two interesting observations below, showing that the probability estimation in naive Bayes could not be improved by the instance weighting in KNNDW when k is small.

1. The difference between LWNB and KNNDW in classification accuracy is not significant. When $k = 5$, LWNB wins in 3 data sets and loses in 2 data sets. When $k = 10$, LWNB wins in 2 data sets, and loses in 2 data sets.
2. LWNB does not outperforms KNNNB significantly. When $k = 5$, LWNB wins in 4 data sets and loses in 3 data sets. When $k = 10$, LWNB wins in 6 data sets, and loses in 4 data sets.

In our experiments, we also tested some relatively large k values, and the experimental results are similar. For example, ICLNB outperforms LWNB in 5 data sets, ties in 30 data sets and loses in 1 data set, when $k = 50$. We have not presented the experimental results in this paper due to the limit of space. In fact, ICLNB consistently outperforms all other algorithms compared in this paper at all the various k values we tested.

Table 4. Summary of experimental results: classification accuracy comparisons when $k = 5$. An entry $w/t/l$ 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.

	NB	KNN	KNNDW	KNNNB	LWNB
KNN	8/23/5				
KNNDW	10/22/4	8/28/0			
KNNNB	10/23/3	9/27/0	3/30/3		
LWNB	10/20/6	10/24/2	3/31/2	4/29/3	
ICLNB	10/23/3	12/24/0	6/30/0	6/30/0	5/31/0

Table 5. Summary of experimental results: classification accuracy comparisons when $k = 10$.

	NB	KNN	KNNDW	KNNNB	LWNB
KNN	5/23/8				
KNNDW	9/22/5	11/25/0			
KNNNB	9/24/3	13/23/0	3/31/2		
LWNB	11/20/5	13/20/3	2/32/2	6/26/4	
ICLNB	10/21/5	15/21/0	5/31/0	7/28/1	6/30/0

4 Conclusions

In this paper, we have proposed a novel model ICLNB for learning local naive Bayes within KNN using instance sampling. Instance sampling leads to relatively large training data for the local naive Bayes, and results in a naive Bayes with more accurate parameters. Thus, the classification of the local naive Bayes is more accurate. Indeed, when the neighborhood size is small, ICLNB deals with the problem of lack of training data effectively. Moreover, ICLNB performs well for various sizes of the neighborhood.

Although considerable work has been done in combining k-nearest neighbor with naive Bayes, some questions still remain unknown. Firstly, the basic assumption in combining k-nearest neighbor with naive Bayes is that, within the small neighborhood, attributes have a less chance to have strong dependences. However, the underlying reason is not clear. Another interesting direction for future research is how to apply KNN and its variants to the problems beyond classification, such as the problems in which accurate probability estimates or an accurate probability-based ranking of instances are required.

References

1. Aha, David W., Dennis Kibler, Marc K. Albert. 1991. Instance-Based Learning Algorithms. Machine Learning, vol. 6, pp. 37-66.
2. Domingos, P., Pazzani M.: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. Machine Learning **29** (1997) 103-130
3. Frank, E., Hall, M., Pfahringer, B.: Locally Weighted Naive Bayes. Proceedings of the Conference on Uncertainty in Artificial Intelligence (2003). Morgan Kaufmann(2003), 249-256.
4. Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press (1996) 202-207
5. Langley, P., Iba, W., Thomas, K.: An Analysis of Bayesian Classifiers. Proceedings of the Tenth National Conference of Artificial Intelligence. AAAI Press (1992) 223-228
6. Merz, C., Murphy, P., Aha, D.: UCI repository of machine learning databases. Dept of ICS, University of California, Irvine (1997). http://www.ics.uci.edu/ mlearn/MLRepository.html
7. Nadeau, C., Bengio, Y.: Inference for the generalization error. Advances in Neural Information Processing Systems 12 (1999) 307-313. MIT Press.
8. http://prdownloads.sourceforge.net/weka/datasets-UCI.jar
9. Weiss, G., M., Provost, F.: Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. Journal of Artificial Intelligence Research, 19 (2003) 315-354.
10. Witten, I. H., Frank, E.: Data Mining –Practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann (2000)
11. Xie, Z., Hsu, W., Liu, Z., Lee, M.: SNNB: A Selective Neighborhood Based Nave Bayes for Lazy Learning. Proceedings of the Sixth Pacific-Asia Conference on KDD. Springer (2002) 104-114
12. Zheng, Z., Webb, G. I.,: Lazy Learning of Bayesian Rules. Machine Learning, **41(1)** (2000) 53-84