Combining Instance Weighting and Fine Tuning for Training Naïve Bayesian Classifiers with Scant Training Data

Khalil El Hindi
Department of Computer Science, King Saud University, Saudi Arabia

Abstract: *This work addresses the problem of having to train a Naïve Bayesian classifier using limited data. It first presents an improved instance-weighting algorithm that is accurate and robust to noise and then it shows how to combine it with a fine tuning algorithm to achieve even better classification accuracy. Our empirical work using 49 benchmark data sets shows that the improved instance-weighting method outperforms the original algorithm on both noisy and noise-free data sets. Another set of empirical results indicates that combining the instance-weighting algorithm with the fine tuning algorithm gives better classification accuracy than using either one of them alone.*

Keywords: *Naïve bayesian algorithm, classification, machine learning, noisy data sets, instance weighting.*

Received April 4, 2016; accepted June 7, 2016

1. Introduction

Naive Bayesian (NB) learning [16] is a simple and yet effective algorithm for many application domains. It is considered one of the top 10 data mining algorithms [22]. It uses Bayes' conditional probability rule for classifying instances. To classify an instance, NB returns the class that has the maximum probability given the instance's attribute values. It uses the training data to estimate the values of the probability terms used by the Bayes' rule.

However to make things computationally attractable it, naïvely assumes that the attribute values are conditionally independent given the class value. Despite its impressive classification accuracy in many domains, its performance degrades in domains where the independence assumption is violated [8]. Its performance also degrades if the training set is too small to provide accurate estimations of the probability terms [7].

Recently two effective methods were proposed to deal the problem of the lack of training data: Discriminatively Weighting Naïve Bayesian (DWNB) [9] and Fine-Tuning Naïve Bayesian (FTNB) [5, 6]. The DWNB method assigns weights to some instances to increase their influence while the FTNB method augments the NB algorithm with a second phase to reuse the training data to find better estimation for the conditional probability terms used by NB. In this work, we propose a more effective instance weighting algorithms that is also more robust to noise and empirically show that combining the instance weighting method with FTNB gives better results than either of them alone.

In section 2, we review the NB learning algorithm, the instance weighting method [9], and the FTNB method [5, 6]. Section 3 discusses some drawbacks of the instance weighting method, and proposes an improved instance-weighting algorithm. Section 4 presents and analyses the results of the improved instance-weighting algorithm. Section 5 discusses the results of combining the instance weighting method with the FTNB method. Section 6 is the conclusion section.

2. Related Work

To classify an instance of the form $<a_1,a_2,...,a_m>$, the NB algorithm uses Bayes' rule, described in Equation (1), to find the class that has the maximum probability given the instance's attribute values.

$$class = \arg\max_{c \in C} \frac{p(a_1,a_2,\cdots,a_m|c).p(c)}{p(a_1,a_2,\cdots,a_m)} \quad (1)$$

Where:

- C is a vector of all class values.
- $P(c)$is the probability of class c.
- $P(a_1,a_2,...,a_m|c)$is the probability that attributes 1, 2, …, m will take the values $a_1,a_2,...,a_m$, given that the instance is of class c.

The algorithm makes the naïve assumption that all attribute values are conditionally independent given the class values; therefore

$$p(a_1, a_2, \cdots, a_m \mid c) = \prod_j p(a_j \mid c) \quad (2)$$

Also, since given a certain instance, the denominator $P(a_1,a_2,...,a_m)$ is the same for all classes, Equation (1) can be simplified as

$$class = \arg\max_{c \in C} p(c).\prod_j p(a_j \mid c) \quad (3)$$

The training data is usedto estimate the probabilityterms $P(c)$ and $P(a_j|c)$ using formulas Equations (4) and (5), respectively.

$$p(c) = \frac{\sum_{i=1}^{n} \delta(c_i, c) + 1}{\sum_{i=1}^{n} n + n_c} \quad (4)$$

$$p(a_j c) = \frac{\sum_{i=1}^{n} \delta(a_i, a_j)\delta(c_i, c) + 1}{\sum_{i=1}^{n} \delta(c_i, c) + n_j} \quad (5)$$

Where $\delta(x, y)$ is a function that returns 1 if its arguments x and y are the same and 0, otherwise.

Clearly the classification accuracy of the algorithm depends on how accurately we can estimate the values of $P(c)$ and $P(a_j|c)$, which could be a challenge if the training data is too small.

Despite the naïve conditional independence assumption, the classification accuracy of the Naïve Bayesian was shown to be surpassingly good in many domains such as text classifications [4, 9, 17, 19]. Moreover, in a comparative study that compared the robustness to noise of many widely used machine learning algorithms, the Naïve Bayesian algorithm was found to be the most robust algorithm [18].

Most work on improving the NB algorithm focuses on relaxing the independence assumption [3, 11, 20, 23]. The NB algorithm can be seen as a special type of Bayesian Netwroks (BN) were attributes can be conditionally dependent given the class. However, learning a BN is an NP-complete problem [3]. Some approaches attemptto make the problem more tractable by considering only networks of special restricted structures [8, 10, 20, 21, 23]. Other approaches ease the independence assumption by selecting a subset of the features that are more likely to be conditionally independent given the class value [11]. Other approaches use a subset of instances to build the NB classifier. A subset of the training set is more likely to meet the independence assumption than the entire training set. The subset can be chosen using a decision tree [15] or the kNN algorithm [7].

Some work, on the other hand, addresses the problem of the lack of training data. The proposed methods for dealing with this problem include: cloning instances to increase the size of training data [13, 14], fine tuning NB [5, 6], and instance weighting [9]. The cloning approach expands a training set by cloning instances. It clones instances based on their dissimilarity to the instance that needs to be classified, then an NB classifier is built based on the expanded data set. This process takes place during classification, and thus it increases the classification time [12]. The Fine Tuning approach for NB (FTNB) augments the NB algorithm with a fine tuning stage that aims at finding better estimations for the probability terms [5, 6]. In [1] the fine tuning method was generalized for different types of Bayesian networks. In [9] a different method that assigns weights to instances is proposed. The weights are greater than one for some instances, which increases their influence in estimating the probability terms used by NB.

This work improves the instance weighting method [9] to make it more accurate and more robust to noise. It also shows that combining the improved instance weighting method with the FTNB method gives better classification accuracy than using either one of them alone.

In following subsections, we review two effective methods for dealing with the problem of the lack of data: instance weighting [9] and fine-tuning method [5, 6].

2.1. Discriminatively Weighted Naive Bayes (DWNB)

The DWNB method [9] addresses the problem of the lack of data by assigning a weight to each training instance that is greater or equal to one. It then uses formulas Equations (6) and (7) to estimate the value of $p(c)$ and $p(a_j \mid c)$ instead of formulas Equations (5) and (6).

$$p(c) = \frac{\sum_{i=1}^{n} w_i \delta(c_i, c) + 1}{\sum_{i=1}^{n} w_i + n_c} \quad (6)$$

$$p(a_j c) = \frac{\sum_{i=1}^{n} w_i \delta(a_i, a_j)\delta(c_i, c) + 1}{\sum_{i=1}^{n} w_i \delta(c_i, c) + n_j} \quad (7)$$

Where w_i is the weight of the i training instance. Thus, an instance with a large weight has more influence on the estimation of $P(c)$ and $P(a_j|c)$ than an instance with a small weight. The weight is computed so that difficult instances (i.e., incorrectly classified) are assigned greater weights than easy instances. The aim is to help the NB algorithm classify these instances correctly. The initial weight of an instance, x of class c, is set to one. The weight of an instance is then iteratively increased by an amount that is inversely proportional to the probability of the instance's class given the other attribute values. In other words, it is inversely proportional to $p(c|x)$, where x is a vector of the form$<a_1,a_2,...,a_m>$.Namely, it isincreased by$(1-p(c|x))$.

Since a difficult instance would get a smaller value for $p(c|x)$ than an easy instance, this implies that a difficult instance would get its weight increased by a larger margin than an easy instance. The weighting process is repeated T times, where T is a constant, and according to [9] setting T to 15 gives the

Algorithm 1: DWNB (D, T)

Input: a set of training instances D and the number of iterations T

Output:Discriminatively Weighted Naive Bayes

Step 1. Initialize the weights of all training instances in D to 1
Step 2. Train a naïve Bayes using D
Step 3. Fort=1 to T do
Step 3.1 For each training instance x of class c
Use the trained naïve Bayes to estimate p(c|x)
Weight $(x)_t$ = *Weight* $(x)_{t-1}$ +(1-*p*(*c*|*x*))
Step 3.2 Train an NB again using D *with the current weights*
Endfor t
Step 4 Return the final NB classifier

Best results. In each iteration, an NB classifier is built using the estimated probabilities and is used to compute $p(c|x)$ for each instance x of class c. Algorithm 1 shows the main steps of DWNB [9].

2.2. Fine Tuning the Naïve Bayesian Learning Algorithm (FTNB)

The FTNB algorithm addresses the problem of the lack of data in a different way. It augments the NB algorithm with a fine-tuning stage that aims at finding better estimations for the probability terms used by NB. The fine-tuning process exploits the misclassified training

Instances to decrease some probability values and increase some others.

If an instance x of class c_{actual} is misclassified that means there is another class, $c_{predicted}$, that got a higher probability than c_{actual} (the actual class) given $x's$ other attribute values. The FTNB algorithm then modifies the values of the probability terms responsible for this classification error. Namely, the algorithm increases $p(a_i|c_{actual})$ and decreases $p(a_i|c_{predicted})$ for each attribute valuea_i. This process is gradually performed using the Equations (8) and (9):

$$p_{t+1}(a_i \mid c_{actual}) = p_t(a_i \mid c_{actual}) + \delta_{t+1}(a_i, c_{actual}) \quad (8)$$

$$p_{t+1}(a_i \mid c_{predicted}) = p_t(a_i \mid c_{predicted}) - \delta_{t+1}(a_i, c_{predicted}) \quad (9)$$

Where t is the cycle number, and δ_{t+1} is the size of the update step.

The size of the update step is proportional to the error, which is computed as:

$$error = \left| p(c_{actual}) - p(c_{predicted}) \right| \quad (10)$$

Where

$$P(c \mid x) = \frac{p(c \mid x)}{\sum_{k=1}^{nc} p(c_k \mid x)} \quad (11)$$

Algorithm 2: FTNB(Training_instances)

phase 1
Use Training_instances to build a Naïve Bayesian classifier
phase 2
let t=0
while training classification accuracy improves do
for each training instance, inst, do
let c_{actual} *be the actual class of inst*
let $c_{predicted=classify\ (inst)}$
if $c_{predicted} \neq c_{actual}$
compute classification error for inst
for each attribute value,a_i *, of instdo*
compute $\delta_{t+1}(a_i, c_{actual})$
let $P_{t+1}(a_i|c_{actual})$=
$P_t(a_i|c_{actual})$+ $\delta_{t+1}(a_i, c_{actual})$
Compute $\delta_{t+1}(a_i, c_{predicted})$
Let $P_{t+1}(a_i| c_{predicted})$=
$P_t(a_i| c_{predicted})$- $\delta_{t+1}(a_i, c_{predicted})$
Let $P_{t+1}(a_i| c_{predicted})$=
$P_t(a_i| c_{predicted})$- $\delta_{t+1}(a_i, c_{predicted})$
endfor
endif
endfor
let t=t+1
end while

Where *nc* is the number of class values. Equation (11) is used to normalize the probabilities and increase their values.

Equations (12) and (13) show how to compute the update step for $\delta_{t+1}(a_i,c_{actual})$ and $\delta_{t+1}(a_i, c_{predicted})$, respectively.

$$\delta_{t+1}(a_i, c_{actual}) = \eta \cdot (\alpha \cdot p(\max_i \mid c_{actual}) - p(a_i \mid c_{actual})) \cdot error \quad (12)$$

$$\delta_{t+1}(a_i, c_{predicted}) = \eta \cdot (\alpha \cdot p(a_i \mid c_{predicted}) - p(\min_i \mid c_{predicted})) \cdot error \quad (13)$$

The Equations use a learning rate, η, that is a value between zero and one, to decrease the update step. Equation (12), which computes the increment for $p(a_i|c_{actual})$ is designed so that the update step (the increment) is large for small terms and small for larger terms. This is why the update step is proportional to $a \cdot p(max_i|c_{actual})$-$p(a_i|c_{actual})$, where α is a constant. While equation 13, which computes the decrement for $p(a_i|c_{predicted})$, ensures that large probability terms are decreased by a large value while small terms are decreased by smaller steps. This is why we used $p(a_i|c_{predicted})$-$p(mini_i|c_{predicted})$ in Equation (13). Following [6], we set a to 2 in all experiments performed in this work. Algorithm 2 shows the main steps of FTNB.

3. An Improved DWNB (IDWNB)

Although the DWNB method substantially improves the classification accuracy of the NB learning algorithm for many applications [9], we believe that the method suffers from two main drawbacks, discussed in the next subsections.

3.1. Robustness to Noise

Since the DWNB method assigns larger weights for difficult instances, we believe that it is sensitive to noisy instances (outliers). Noisy instances are instances with the wrong class values, which makes them hard to classify. We believe that the DWNB algorithm would assign them large weights and thus increase their

adverse effect on the final NB classifier. In other words, we believe that the DWNB algorithm makes the NB classifier less robust to noise. This sacrifices one of the main advantages of the NB learning algorithm, namely its robustness to noise [18].

To test this hypothesis, we performed several empirical experiments using 49 benchmark data sets obtained from the UCI ML repository [2] with and without inserting artificial noise. We inserted random noise in the class attribute of some randomly selected training instances. We used 0%, 5%, 10%, 15% and 20% noise. The noise was inserted in the training data only; the test sets were left untouched. We used 10-fold cross validation in each experiment. We also usedpaired t-test with 95% confidence level to test if the differences are statistically significant. Table 1 shows the average classification accuracy of the ten folds of the NB learning algorithm and the DWNB method for the different noise ratios. The last two rows in the table present the number of data sets for which each method achieved better accuracy and significantly better accuracy. The better results are highlighted in bold, and the significantly better results are highlighted in bold and underlined.

It is evident from the table that while the performance of DWNB is much better than the performance of NB at 0% noise, its performance degrades as the noise ratio increases. At 0% noise ratio, the DWNB method achieved significantly better results for 12 data sets, while the NB algorithm achieved significantly better results for only 3 data sets. At 20% noise, DWNB achieved significantly better results for 11 data sets while NB achieved significantly better results for 12 data sets. Also, the performance of DWNB degraded much more quickly than the performance on NB in terms of the average classification accuracy. The average classification accuracy of DWNB dropped from 81.49% at 0% noise down to 77.18% at 20% noise, a drop of 4.31%. However, the average classification accuracy of NB dropped at a much slower rate from 80.40% at 0% noise to 78.45% at 20% noise, a drop ofonly 1.95%. This clearly indicates that the DWN method is less robust to noise than NB. This is evident also by comparing the number of data sets at which DWNB achieves better (not just significantly better results) than NB. This figure dropped from 35 data sets at 0% noise down to 20 data sets at 20% noise. At the same time, the number of data sets for which NB achieved better results than DWNB rose from seven data sets at 0% noise to 21 data sets at 20% noise. All of these findings support our claim that the DWNB

Method for instance weighting is sensitive to noise and scarifies one of the main advantages of the NB algorithm, namely its robustness to noise [18].

We believe that noisy instances have this dramatic effect on DWNB, because they are more likely to be misclassified (i.e., to have a small value for $p(c|x)$) and, therefore, have larger weights and more influence on the subsequent classifiers than genuine instances (not noisy).

To reduce this influence, we propose reducing the weight update that takes place in late iterations. This is performed using Equation (14).

$$weight(x)_t = weight(x)_{t-1} + \frac{1}{t^2}(1 - p(c \mid x)) \quad (14)$$

Where t is the iteration number. Formula Equation (14) is designed to reduce the effect of noisy instances in later iterations. None noisy instances are likely to get correctly classified in a few number of iterations (i.e. large value for $p(c|x)$) and thus get smaller weight update anyway. However, noisy instances are likely to continue to be misclassified by the subsequent classifiers. In fact, as the subsequent classifiers become more accurate, $p(c|x)$ will get smaller for noisy instances and thus noisy instances will have even greater weight and thus more influence. By multiplying by $\frac{1}{t^2}$ formula Equation (14) ensures that the weight of noisy instances will onlybe slightly increased in later iterations and thus will not continue to have an increasing adverse effect on the classifiers built in late iterations. This should help avoid overfitting the training data.

3.2. Making the Method More Discriminative

We believe that $p(c|x)$ is too small to discriminate between difficult and easy instances in the training data in a sufficient way, which may assign similar weights to the different instances. To address this issue and to make $p(c|x)$ sufficiently large, we normalize $p(c|x)$ using Equation (11) (repeated below as Equation (15) for the reader's convenience). The equation divides $p(c|x)$ by the total probability of all classes given x.

Table 1. The effect of noise on DWNB.

Data Set	0% Noise		5% Noise		10% Noise		15% Noise		20% Noise	
	NB	DWNB	NB	DWNB	NB	DWNB	NB	DWNB	NB	DWNB
anneal	96.99	97.77	97.10	94.54	97.10	93.10	95.88	90.20	96.88	90.20
AnnealORIG	79.06	89.42	85.30	79.62	88.31	62.36	87.97	42.54	87.42	45.32
arrhythmia	54.20	74.56	54.20	75.22	54.20	74.78	54.20	74.56	54.20	74.34
autos	60.00	62.44	60.00	60.49	60.98	60.49	59.51	53.17	57.56	50.73
breast-cancer	73.08	72.38	70.98	70.28	72.38	72.38	70.63	69.93	68.88	69.23
breast-w	97.28	97.42	97.57	97.28	97.42	97.42	97.00	97.14	97.42	97.28
bridges_ver1	59.81	60.75	59.81	59.81	65.42	57.94	58.88	56.07	59.81	47.66
bridges_ver2	57.94	56.07	54.21	56.07	52.34	52.34	47.66	43.93	52.34	50.47
car	73.15	73.38	73.38	73.32	71.93	71.88	72.74	72.74	72.28	72.16
colic	72.01	72.01	70.92	71.47	69.57	69.84	69.02	69.02	67.66	67.93
colic.orig	70.65	68.48	71.74	64.67	69.57	67.39	69.02	60.05	62.50	54.89
credit-a	84.06	84.35	84.49	84.35	84.06	83.91	83.91	83.62	84.20	83.77
credit-g	75.50	75.70	74.70	74.80	76.20	76.60	75.00	75.00	71.30	71.20
Cylinder bands	69.44	69.63	71.48	69.63	68.15	66.67	67.96	65.93	68.70	67.96
dermatlogy	97.27	99.18	98.09	98.91	98.36	98.36	96.72	96.45	97.27	97.27
diabetes	77.34	77.34	77.21	77.47	76.69	76.82	76.04	76.04	76.69	76.82
ecoli	74.40	78.87	73.21	79.46	71.13	77.38	74.40	80.95	76.19	80.36
flags	60.31	56.19	59.28	52.58	58.76	48.45	56.70	53.61	60.31	52.06
glass	70.09	74.30	71.03	72.43	70.56	72.43	71.96	72.90	68.22	72.43
haberman	74.18	74.18	73.86	74.18	74.18	73.20	72.88	72.22	74.51	73.86
heart-c	85.15	84.82	83.17	83.17	84.16	83.17	84.49	83.17	85.15	83.50
heart-h	83.33	76.87	83.67	77.21	83.67	69.73	82.99	76.87	84.69	72.45
heart-statlog	83.70	84.07	83.70	83.70	83.33	83.70	84.44	84.44	83.70	83.70
hepatitis	83.23	84.52	83.87	81.29	81.29	80.65	80.00	79.35	80.00	80.00
hypothyroid	92.95	93.35	90.99	91.07	90.22	89.87	88.65	87.96	87.91	87.59
ionosphere	90.88	91.45	90.31	91.17	89.74	90.03	90.60	90.88	91.17	90.88
iris	95.33	96.67	94.67	94.67	96.00	95.33	95.33	95.33	94.67	94.67
letter	74.11	74.86	73.48	74.00	72.85	73.29	72.61	72.95	72.00	72.30
liverDisorder	54.78	54.78	53.04	53.04	58.84	59.71	53.33	52.46	54.78	53.91
lung-cancer	81.25	84.38	78.13	81.25	75.00	75.00	81.25	84.38	65.63	75.00
lymph	83.78	84.46	83.11	81.08	85.14	82.43	82.43	81.08	82.43	79.05
mushroom	94.33	97.03	92.07	92.29	91.65	91.85	91.04	91.22	90.85	90.90
nursery	81.37	81.41	81.41	81.42	81.19	81.18	81.40	81.40	82.03	82.04
optdigits	92.12	92.08	91.60	91.69	91.28	91.39	91.30	91.33	90.84	90.93
pendigits	87.92	88.37	87.21	87.34	86.64	86.77	86.22	86.34	85.53	85.72
segment	91.77	93.25	90.69	91.17	89.57	90.56	89.09	89.91	87.92	88.61
sick	96.85	96.85	93.74	93.66	91.76	91.89	88.12	88.57	85.15	85.95
solar-flare_1	91.64	94.43	92.57	94.74	93.50	95.67	93.19	94.12	91.64	94.12
solar-flare_2	97.00	97.09	96.34	97.65	96.53	98.03	96.90	98.12	97.28	98.22
sonar	82.21	83.17	83.65	84.13	85.58	85.58	86.06	87.50	83.17	83.65
spambase	89.35	89.37	89.22	89.31	89.18	89.26	89.50	89.50	89.18	89.20
splice	94.92	95.24	93.54	93.70	92.35	92.51	91.47	91.54	90.50	90.75
trains	70.00	70.00	70.00	70.00	70.00	70.00	70.00	60.00	60.00	60.00
vehicle	63.36	63.48	62.17	62.17	62.29	62.65	63.95	63.83	62.17	62.17
vote	89.43	89.66	89.43	89.43	90.34	90.34	89.66	89.66	89.43	89.43
vowel	61.31	62.22	59.09	59.29	58.48	59.49	58.28	58.08	57.98	58.69
Waveform-5000	80.64	80.76	80.16	80.18	79.88	79.92	79.56	79.56	78.90	78.90
wine	98.88	98.88	98.31	99.44	98.88	99.44	97.75	97.75	97.75	97.19
zoo	91.09	95.05	90.10	94.06	91.09	94.06	87.13	92.08	87.13	86.14
average	80.40	81.49	79.96	80.20	79.95	79.33	79.28	78.07	78.45	77.18
#better	7	35	13	28	17	24	21	18	21	20
#Sig Better	3	12	8	15	9	14	7	7	12	11

$$P(c \mid x) = \frac{p(c \mid x)}{\sum_{k=1}^{nc} p(c_k \mid x)} \qquad (15)$$

Where *nc* is the number of class values. In all experiments reported in the following sections, we use (1-$P(c|x)$) instead of (1-$p(c|x)$) to better discriminate between instances.

4. The Empirical Results of IDWNB

In this section, we discuss the results of the Improved IDWN, which incorporates the two modifications proposed in the last section.

Table 2 presents the results we obtained using IDWN compared to the results of NB. The results show that even at noise free data sets (i.e., at 0% artificial noise) the Improved DWNB (IDWNB) outperforms NB and DWNB. IDWNB outperforms NB for 35 data sets 23 of which are significantly better results, while NB outperformed IDWNB for 8 data sets only 2 of them are significantly better results. Recall that DWNB outperformed NB for 35 data sets only 12 of them are

Statistically significant improvements (see Table 1). While NB outperformed DWNB for 7 data sets, 3 of them are statistically significant results. Moreover, IDWNB outperforms DWNB and NB regarding the average classification at 0% noise. The average classification accuracy of IDWNB, DWNB and NB at 0% noise are 81.80%, 81.49% and 80.40%, respectively.

It is also evident from Table 2 that IDWNB is more robust to noise than DWNB. Its performance remained better than NB even at 20% noise, outperforming NB for 26 data sets, 18 of them are significantly better results. The average classification accuracy of IDWNB also declined more slowly than the average classification accuracy of DWNB. It declined from 81.80% at 0% noise to 79.20% at 20% noise, a drop of 2.6%. While the average classification accuracy of DWNB declined from 81.49% at 0% noise down to77.18% at 20% noise, adrop of 4.31% (see Table 1). The average classification accuracy of NB declined from 80.4% at0% noise down to 78.45% at 20% noise ratio, a drop of 1.95%. Moreover, the average classification accuracy of IDWNB remained the best average at all noise ratios.

Table 2. The results of IDWNB compared to NB.

Data Set	0% Noise		5% Noise		10% Noise		15% Noise		20% Noise	
	NB	IDWNB	NB	IDWNB	NB	IDWNB	NB	IDWNB	NB	IDWNB
anneal	96.99	98.00	97.10	97.55	96.77	97.10	96.21	96.44	96.66	96.88
AnnealORIG	79.06	82.63	86.30	90.31	86.86	87.19	85.86	85.41	88.42	83.85
arrhythmia	54.20	61.95	54.20	62.39	54.20	62.83	54.20	62.83	54.20	61.50
autos	60.00	60.98	61.46	59.02	60.98	59.02	56.59	54.63	55.12	50.24
breast-cancer	73.08	70.63	72.73	69.58	69.93	65.38	70.63	67.83	67.48	66.08
breast-w	97.28	97.14	97.57	97.42	97.57	97.28	97.14	96.85	97.14	96.57
bridges_ver1	59.81	61.68	60.75	57.01	61.68	58.88	59.81	55.14	56.07	56.07
bridges_ver2	57.94	59.81	51.40	56.07	52.34	55.14	56.07	49.53	56.07	49.53
car	73.15	78.82	72.45	75.69	73.15	76.10	72.74	74.25	74.65	76.16
colic	72.01	75.54	70.38	74.46	69.57	75.00	68.48	77.45	67.66	75.54
Colic-orig	70.65	70.38	71.74	71.20	67.66	66.03	64.95	61.41	67.39	61.96
credit-a	84.06	83.62	84.06	82.90	84.20	83.77	83.04	82.61	84.06	82.32
credit-g	75.50	76.60	76.10	75.40	74.30	75.30	73.80	73.50	74.00	74.30
Cylinder bands	69.44	72.22	69.63	74.44	68.33	73.33	70.93	73.15	69.63	67.41
dermatology	97.27	97.54	98.09	98.36	96.72	95.08	96.99	96.72	96.45	94.81
diabetes	77.34	78.65	77.34	78.78	76.69	77.99	75.39	78.78	76.69	78.78
ecoli	74.40	79.17	75.30	81.25	73.81	80.95	73.21	79.17	72.92	74.40
flags	60.31	57.73	58.76	58.76	58.76	56.70	59.28	57.22	59.28	55.15
glass	70.09	74.30	69.16	72.90	69.16	73.83	67.29	71.96	69.63	70.56
haberman	74.18	74.18	72.88	72.88	72.55	72.88	74.84	73.53	72.55	73.20
heart-c	85.15	84.16	84.82	85.15	84.82	83.83	85.48	85.48	84.82	83.17
heart-h	83.33	82.99	82.65	83.33	83.67	83.33	82.65	80.95	82.65	77.55
heart-statlog	83.70	82.96	82.96	82.96	82.96	83.70	82.59	83.70	81.48	81.11
hepatitis	83.23	85.16	83.23	82.58	80.65	81.94	78.71	76.77	81.94	83.23
hypothyroid	92.95	98.12	90.99	92.02	89.50	92.23	89.18	92.84	87.06	92.10
ionosphere	90.88	91.45	91.45	92.59	90.60	92.02	90.60	92.59	89.46	91.17
iris	95.33	96.00	96.00	97.33	95.33	94.67	96.00	96.00	96.00	95.33
letter	74.11	76.73	73.42	76.05	73.05	75.60	72.34	75.21	72.12	74.69
liver-disorders	54.78	54.78	54.78	54.78	51.88	51.88	52.75	52.75	56.23	56.23
lung-cancer	81.25	81.25	81.25	81.25	78.13	75.00	75.00	71.88	78.13	78.13
lymph	83.78	84.46	85.81	84.46	83.78	83.11	80.41	81.76	82.43	81.08
mushroom	94.33	97.33	92.06	96.52	91.52	97.56	91.41	98.24	91.05	98.60
nursery	81.37	82.65	81.35	82.72	81.71	82.57	80.81	81.82	81.91	82.82
optdigits	92.12	92.97	91.83	92.54	91.51	92.31	91.17	91.57	90.77	90.91
pendigits	87.92	90.72	87.23	89.66	86.70	89.04	86.23	88.70	85.70	88.28
segment	91.77	93.64	90.56	91.69	89.48	91.08	89.18	90.74	88.31	89.57
sick	96.85	97.00	93.48	96.61	90.27	96.58	89.32	96.74	86.90	96.13
solar-flare_1	91.64	93.81	91.95	95.36	91.64	96.59	94.43	96.59	90.71	95.05
solar-flare_2	97.00	97.37	96.53	98.59	96.25	99.25	97.37	99.34	96.06	98.50
sonar	82.21	82.21	83.65	83.65	82.69	82.21	81.73	78.37	83.65	80.77
spambase	89.35	90.46	89.07	90.65	89.39	91.13	89.55	91.91	89.42	91.02
splice	94.92	94.98	93.73	92.70	92.51	91.10	91.38	90.00	90.03	87.96
trains	70.00	70.00	70.00	70.00	70.00	70.00	70.00	70.00	60.00	60.00
vehicle	63.36	65.60	62.41	65.25	62.53	64.54	61.82	65.48	62.77	65.84
vote	89.43	90.34	89.89	90.80	89.89	90.80	90.34	91.26	89.43	93.56
vowel	61.31	63.94	60.20	62.42	60.20	63.33	59.19	59.39	57.78	59.39
Waveform-5000	80.64	81.74	80.12	81.54	79.60	81.52	79.22	81.84	79.00	82.38
wine	98.88	98.88	98.88	98.31	99.44	98.88	98.31	94.94	96.07	94.94
zoo	91.09	95.05	90.10	94.06	89.11	91.09	90.10	91.09	86.14	86.14
Average	80.40	81.80	80.16	81.47	79.47	80.75	78.08	79.93	78.65	79.20
#better	8	35	11	31	16	31	18	27	18	26
#sig better	2	23	2	26	5	19	6	20	8	18

5. Combining IDWNB and FTNB

Since both methods, FTNB and IDWNB, deal with the problem of the lack of training data, in this section, we compare between the two methods to find out which one is more effective in improving the classification accuracy. We also address the issue of whether or not combining both of them would give better results than using either one of them alone. Combining both methods is straightforward; we first use IDWNB tobuild a classifier then we use FTNB to fine-tune this classifier. Table 3 shows the result of comparing both methods and the result of the combined method compared with each one of them alone.

The Table also shows that combining IDWNB and FTNB produced better results than using either one of them alone. The combined method outperformed IDWNB for 29 data sets, 18 of which are significantly better results. However, IDWNB outperformed the combined method for 12 data sets only 4 of them are statistically better results. Compared with the FTNB method, the combined method produced better results for 28 data sets, 13 of which are significantly better results. FTNB, on the other hand, outperformed the combined method for 12 datasets, only 5 of which are significantly better results. Also the average classification accuracy of the combined method of 82.93% is higher than the average accuracies of IDWNB and FTNB, which are 81.80% and 82.47%, respectively. This shows that the improvement in classification accuracy achieved by the two methods adds up when they are combined. It is worth sayingthat we set the learning rate, η, to 0.01 when we trained FTNB alone, and we had to set it to a much smaller value of 0.005 when we combined it with IDWNB (i.e. when we fine tuned the classifier that we obtained from IDWNB). Having to use a smaller learning rate to fine tune IDWNB, may indicate that the classifier produced by IDWNB is much closer to the optimal estimation than classifier generated by NB.

The experiments reported in Table 3 were performed at 0% noise ratio (i.e., no artificial noise were added). Like DWNB, FTNB is also sensitive to noise [5]. In [5], some methods were proposed to make the FTNB algorithm more robust to noise. It would be interesting to combine these fine-tuning methods with IDWNB and evaluate the performance of the combined method at different noise ratios. This may be an interesting issue for future work.

Table 3. Combining FTNB and IDWNB.

Data Set	IDWNB vs FTNB		IDWNB vs Combined		FTNB vs Combined	
	IDWNB	FTNB	IDWNB	Combined	FTNB	Combined
anneal	98.00	98.22	98.00	98.11	98.22	98.11
anneal.ORIG	82.63	97.66	82.63	97.66	97.66	97.66
arrhythmia	61.95	72.57	61.95	74.56	72.57	74.56
autos	60.98	64.88	60.98	62.93	64.88	62.93
breast-cancer	70.63	67.83	70.63	66.43	67.83	66.43
breast-w	97.14	95.99	97.14	96.28	95.99	96.28
bridges_ver1	61.68	61.68	61.68	63.55	61.68	63.55
bridges_ver2	59.81	60.75	59.81	61.68	60.75	61.68
car	78.82	84.72	78.82	82.23	84.72	82.23
colic	75.54	80.16	75.54	80.16	80.16	80.16
colic.orig	70.38	75.54	70.38	72.55	75.54	72.55
credit-a	83.62	82.17	83.62	82.46	82.17	82.46
credit-g	76.60	72.80	76.60	73.60	72.80	73.60
cylinder-bands	72.22	71.85	72.22	73.70	71.85	73.70
dermatology	97.54	97.54	97.54	97.81	97.54	97.81
diabetes	78.65	77.47	78.65	78.13	77.47	78.13
ecoli	79.17	77.98	79.17	79.76	77.98	79.76
flags	57.73	57.73	57.73	57.73	57.73	57.73
glass	74.30	72.43	74.30	74.30	72.43	74.30
haberman	74.18	70.26	74.18	70.92	70.26	70.92
heart-c	84.16	84.16	84.16	83.83	84.16	83.83
heart-h	82.99	80.27	82.99	80.61	80.27	80.61
heart-statlog	82.96	82.59	82.96	82.96	82.59	82.96
hepatitis	85.16	87.10	85.16	89.03	87.10	89.03
hypothyroid	98.12	99.26	98.12	99.20	99.26	99.20
ionosphere	91.45	92.31	91.45	92.02	92.31	92.02
iris	96.00	96.00	96.00	96.00	96.00	96.00
letter	76.73	77.05	76.73	77.89	77.05	77.89
liver-disorders	54.78	63.19	54.78	63.19	63.19	63.19
lung-cancer	81.25	81.25	81.25	81.25	81.25	81.25
lymph	84.46	85.81	84.46	87.16	85.81	87.16
mushroom	97.33	99.68	97.33	99.64	99.68	99.64
nursery	82.65	84.32	82.65	84.27	84.32	84.27
optdigits	92.97	94.04	92.97	94.06	94.04	94.06
pendigits	90.72	94.71	90.72	94.83	94.71	94.83
segment	93.64	93.90	93.64	93.98	93.90	93.98
sick	97.00	96.98	97.00	97.03	96.98	97.03
solar-flare_1	93.81	95.98	93.81	95.98	95.98	95.98
solar-flare_2	97.37	98.87	97.37	98.59	98.87	98.59
sonar	82.21	78.37	82.21	80.29	78.37	80.29
spambase	90.46	77.57	90.46	81.26	77.57	81.26
splice	94.98	91.63	94.98	93.70	91.63	93.70
trains	70.00	70.00	70.00	70.00	70.00	70.00
vehicle	65.60	68.91	65.60	67.38	68.91	67.38
vote	90.34	93.33	90.34	94.48	93.33	94.48
vowel	63.94	60.61	63.94	62.63	60.61	62.63
waveform-5000	81.74	83.14	81.74	83.68	83.14	83.68
wine	98.88	98.88	98.88	98.88	98.88	98.88
zoo	95.05	91.09	95.05	95.05	91.09	95.05
Average	81.80	82.47	81.80	82.93	82.47	82.93
#better	17	24	12	29	12	28
#sig better	6	18	4	18	5	13

6. Conclusions

This work addresses the problem of training an NB classifier using limited training data. It presents an improved instance-weighting algorithm that is more accurate than the original one [9] and more robust to noise. The algorithm assigns weights to instances that are greater than or equal to one. This magnifies the effect of some training instances especially if the training data is limited. Then we presented a straightforward method for combining this instance weighting method with another method for dealing with the problem of the lack of data, namely, the fine tuning method [5, 6]. Our empirical results using 49 data sets show that the improved instance-weighting algorithm is more accurate and more robust to noise than the original one. Moreover, combining the instance weighting methods with the fine-tuning method give better classification accuracy than using either one of them alone.

As a future work, it would be interesting to evaluate the combined method under different noise ratios and perhaps combine the IDWNB with the more robust fine-tuning methods proposed in [5].

Acknowledgment

This work was supported by the Research Center of College of Computer and Information Science, King Saud University. The author is grateful for this support.

References

[1] Alhussan A. and El Hindi K., "Selectively Fine-Tuning Bayesian Network Learning Algorithm," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 30, no. 8, 2016.

[2] Blake C. and Merz C., "UCI Repository of Machine Learning Databases," *University of California*, http://archive.ics.uci.edu/ml/, Last Visited, 2016.

[3] Chickering D., "Learning Bayesian Networks Is NP-Complete," *in Proccedings of Learning from Data: Artificial Intelligence and Statistics*, New York, pp. 121-30, 1996

[4] Duwairi R., "Arabic Text Categorization," *The International Arab Journal of Information Technology*, vol. 4, no. 2, pp. 125-31, 2007.

[5] El Hindi K., "A Noise Tolerant Fine Tuning Algorithm for the Naïve Bayesian Learning Algorithm," *Journal of King Saud University-Computer and Information Sciences*, vol. 26, no. 2, pp. 237-246, 2014.

[6] El Hindi K., "Fine Tuning the Naïve Bayesian Learning Algorithm," *AI Communcations*, vol. 27, no. 2, pp. 133-141, 2014.

[7] Frank E., Hall M., and Pfahringer B., "Locally weighted naive Bayes," *in Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, Acapulco, pp. 249-256, 2003.

[8] Friedman N., Geiger D., and Goldszmidt M., "Bayesian Network Classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131-163, 1997.

[9] Jiang L., Wang D., and Cai Z., "Discriminatively Weighted Naive Bayes and Its Application in Text Classification," *International Jorurnal on Artificial Intelligence Tools*, vol. 21, no. 1, 2012.

[10] Jiang L., Cai Z., Wang D., and Zhang H., "Improving Tree Augmented Naive Bayes for Class Probability Estimation," *Knowledge-Based Systems*, vol. 26, pp. 239-245, 2012.

[11] Jiang L., Zhang H., Cai Z., "Evolutional Naive Bayes," *in Proceedings of the 1st International Symposium on Intelligent Computation and Applications*, pp. 344-350, 2005.

[12] Jiang L., Wang D., Cai Z., Yan X., "Survey of Improving Naive Bayes for Classification," *in Advanced Data Mining and Applications*, Harbin, pp. 134-145, 2007.
[13] Jiang L., Wang D., Cai Z., Zhang H., "Using Instance Cloning to Improve Naive Bayesfor Ranking," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 22, no. 6, pp. 1121-1140, 2008.
[14] Jiang L. and Zhang H., "Learning Instance Greedily Cloning Naive Bayes for Ranking," *in Proccedings of 5th IEEE International Conference on Data Mining*, Houston, pp. 202-209, 2005.
[15] Kohavi R.,"Scaling Up the Accuracy of Naive-Bayes Classi Ers : A Decision-Tree Hybrid," *in Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 202-207, 1996.
[16] Langley P. and Sage S., "Induction of Selective Bayesian Classifiers," *in Proceedings of the 10th International Conference on Uncertainty in Artificial Intelligence*, pp. 339-406, 1994.
[17] Mitchell T., *Machine Learning*, McGraw Hill, 1997.
[18] Nettleton D., Fornells A., and Orriols-Puig A., "A Study of the Effect of Different Types of Noise on the Precision of Supervised Learning Techniques," *Artificial Intelligence Review*, vol. 33, no. 4, pp. 275-306, 2010.
[19] Nigam K., McCallum A., Thrun S., and Mitchell T., "Text Classification from Labeled and Unlabeled Documents Using EM," *Machine Learning*, vol. 39, no. 2-3, pp.103-134, 2000.
[20] Palacios-Alonso M., Brizuela A., and Enrique Sucar L., "Evolutionary Learning of Dynamic Naive Bayesian Classifiers," *Journal of Automated Reasoning*, vol. 45, no. 1, pp. 21-37, 2010.
[21] Quinn C., Coleman T., and Kiyavash N., "Approximating Discrete Probability Distributions with Causal Dependence Trees," *in Proccedings of International Symposium on Information Theory and Its Applications*, Taichung, pp. 100-105, 2010.
[22] Wu X., Kumar V., Quinlan J., Ghosh J., Yang Q., Motoda H., McLachlan G., Ng A., Liu B., Yu P., Zhou Z., Steinbach M., Hand D., and Steinberg D., "Top 10 Algorithms in Data Mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1-37, 2008.
[23] Zhang H. and Ling C., "An Improved Learning Algorithm for Augmented Naive Bayes," *Advances in Knowledge Discovery and Data Mining*, Hong Kong, pp. 581-586, 2001.

Khalil El Hindi is a Professor at the department of Computer Science, King Saud University. His research interest includes machine leaning and data mining. He is particularly interested in improving the classification accuracy of Bayesian classifiers and developing new similarity metrics for instance-based learning. He received his BS.C. Degree from Yarmouk University and his MSc and Ph.D. degrees from the University of Exeter, UK.