Evolutionary Lazy Learning for Naive Bayes Classification

Yu Bai*, Haishuai Wang*, Jia Wu*, Yun Zhang†, Jing Jiang*, Guodong Long*
*Centre for Quantum Computation & Intelligent Systems (QCIS),
Faculty of Engineering and Information Technology, University of Technology Sydney, Australia
†School of Computer Science and Engineering, University of New South Wales, Australia
Email: Yu.Bai@uts.edu.au; Jia.Wu@uts.edu.au; Haishuai.Wang@uts.edu.au;
Zhangyuny@gmail.com; Guodong.Long@uts.edu.au; Jing.Jiang@uts.edu.au

***Abstract*—Most improvements for Naive Bayes (NB) have a common yet important flaw - these algorithms split the modeling of the classifier into two separate stages - the stage of preprocessing (*e.g.,* feature selection and data expansion) and the stage of building the NB classifier. The first stage does not take the NB's objective function into consideration, so the performance of the classification cannot be guaranteed. Motivated by these facts and aiming to improve NB with accurate classification, we present a new learning algorithm called Evolutionary Local Instance Weighted Naive Bayes or ELWNB, to extend NB for classification. ELWNB combines local NB, instance weighted dataset extension and evolutionary algorithms seamlessly. Experiments on 20 UCI benchmark datasets demonstrate that ELWNB significantly outperforms NB and several other improved NB algorithms.**

I. Introduction

Naive Bayes (NB), which is based on probability theory, is one of the most widely used learning algorithms [1], [2]. NB is computationally highly efficient and thus is suitable for many learning scenarios such as text classification [3], web mining [4], sentiment analysis [5] and image classification [6], [7]. Indeed, NB is a special case of Bayesian network [8], which considers the dependence of attributes to obtain the correct classification. The time consumption of the general Bayesian network can be very high, which affects its application in the real world. To address this issue, NB model adopts a simple and straightforward assumption on the general Bayesian network, *i.e.,* the attributes of the training dataset are independent of each other. Under this assumption, the examination of dependence between attributes is no longer needed and building the NB model only needs linear mathematical computation.

In many real world applications, there is a strong dependence between attributes, the performance of the NB classification can drop sharply. Furthermore, NB treats the different instances with the same weight, but training instances that are closer to the test instance may play a more important role than those that are far from the test instance. For this reason, many approaches have attempted to improve the performance of NB [9], including feature based learning [10], [11], [12], [13], [14], structure extension based learning [15], [16], [17], local learning [18] and data expansion based learning [19], [20].

The above methods of improving NB have achieved good results, however, after carefully investigation, we find that all these methods have a common disadvantage (*i.e.,* the NB models are separated into two steps). The first stage is the preprocessing, and the second is the creation of the NB model. These two stages are independent of each other and the first stage is carried out without consideration of the NB objective function. As a result, the performance of the NB cannot be guaranteed.

To solve the common disadvantage of the above methods, we propose a new method in this paper called Evolutionary Local Instance Weighted Naive Bayes, namely ELWNB. ELWNB constructs an extended training dataset based on the original training dataset using instances cloning, which will be used to build a NB classifier for further classification. Indeed, how the extended training dataset is constructed is determined by two important parameters: $threshold$ (determines whether this instance should be cloned) and $weight$ (determines the number of times that this instance should be cloned). Here, the Differential Evolution (DE) algorithm [21] is used to automatically search for the optimal parameters. To obtain the best classification result, the NB's objective function is directly used as the fitness function of the DE to evaluate the performance of different combinations of the above two parameters. Experiments and comparisons on 20 UCI benchmark datasets demonstrate the performance of proposed ELWNB.

II. Evolutionary Lazy Learning for NB

A. EWLNB Algorithm

First, we define the distance of two instances as

$$d(m,n) = \sqrt{\sum_{i=1}^{k} (m_i - n_i)^2} \tag{1}$$

Algorithm 1 EWLNB

Input:

Training dataset D^a, a test instance x, parameter vector w_i^t;

Output:

The classify result $c(x)$;

```
D*^a ← D^a;
for all y_i in D^a do
    Compute d(x, y_i) using the distance in Eq. (1);
    threshold ← w^t_{i,1}, weight ← w^t_{i,2};
    for all the instances y_i with d(x, y_i) < threshold do
        clonenb ← weight^2 / (weight + weight * d(x, y_i));
        Adding clonenb instance y_i to D*^a;
    end for
end for
Build a Naive Bayes classifier on the new dataset D*^a and use this NB classifier to give the result of c(x);
```

where m and n are two vectors in a K-dimension space with the value $\{m_1, \cdots m_k\}$ and $\{n_1, \cdots n_k\}$, respectively.

Given a training dataset D^a and a test instance x, we first use Eq. (1) to calculate the distance $d(x, y_i)$ between the test instance x and each training instance y_i in D^a. Then, if $d(x, y_i)$ satisfies the requirements of the parameter $threshold$, we get the $clonenb$ of the i th individual through $d(x, y_i)$, and add $clonenb$ clones of y_i into D_*^a to expand the training dataset D^a. We deploy a NB classifier on the expanded D_*^a to calculate the class classification $c(x)$. The detailed EWLNB is depicted in Algorithm 1.

Here D_*^a is the extended training dataset, $w_{i,1}^t$ and $w_{i,2}^t$ are the two dimensions of the parameter vector, in our algorithm, they represent $threshold$ and $weight$ respectively. $clonenb$ is a integer which represents the number of times that should be cloned of the training instance y_i.

It is clear that EWLNB algorithm's main procedure is expanding the training instances D^a for a test instance x. We call our learning algorithm Evolutionary Local Instance Weighted Naive Bayes, or EWLNB, because it spends no effort during training time, delays all computation until classification time and the evolutionary algorithm is used to achieve the optimal parameter vector. Our learning algorithm deals with NB's shortcomings by cloning some of the training instances to produce an expanded training dataset.

B. Evolution of the Parameter $threshold$ and $weight$

To obtain the optimal parameters to get the extended dataset D_*^a, we use DE to learn two important parameters for EWLNB classification. In our solution, the different combinations of the two parameters act as candidates, presented by parameter vector W. The main stages of the evolution are initialization, cloning, mutation, and selection. The evolving optimization will assist the discovery of the optimal W vector with the best classification accuracy. Before introducing details of the algorithm, we define a number of notations.

- . W represents the set of candidates, which we call the population. $W = \{w_1, \cdots, w_L\}$ where L represents the size of the population, $w_i = \{w_{i,1}, \cdots, w_{i,n}\}$ represents a single candidate, which we call an individual, and where n is the size of the parameter vector. In our algorithm, the value of n is two.
- . $w_{i,j}$ denotes the j th value of the i th individual.
- . w_c represents the individual which has the best fitness (*i.e.,*best classification accuracy) on the test instances.

The detailed DE process is descripted as following steps:

1) Initialization of the Parameter Vectors: For individuals in $W = \{w_1, \cdots, w_L\}$ with a population size L, we should ensure that every individual $w_i = \{w_{i,1}, \cdots, w_{i,n}\}$ in the population is generated through certain random mechanisms. We therefore set w_i^j value of w_i for each individual as a uniformly distributed random number within the range (0, 1]. In the experiment, 90% of the original training instances D are used as the training dataset D^a to learn w_c and the remaining instances are used as the test dataset D^b, with the population size L set to 50.

2) Evaluation of EWLNB: The process of Evaluation of EWLNB can be divided into following steps:

Calculation of Fitness Function: the fitness of the i th individual of the t th generation is the classification accuracy that is obtained by EWLNB using the w_i^t to carry out the probability estimation. The calculation of fitness function can be described as:

$$f[w_i^t] = \frac{1}{N^b} \sum_{1}^{N^b} \delta[c(x_i^b), y_i^b] \tag{2}$$

where $c(x_i^b)$ is the classification result of the i th instance in test dataset D^b with N^b instances, using Algorithm 1 based on individual w_i^t. y_i^b is the actual class value of the i th instance. $\delta[c(x_i^b), y_i^b]$ is one if $c(x_i^b) = y_i^b$ and zero otherwise.

Individual Selection: we sort the individuals in every population according to the fitness of each individual, and choose the individual w_c^t with the best fitness performance in the t th generation to use as the base of the DE mutation.

Individual Mutation: we use the mutate operation to get the mutation individuals in the t th generation through small changes over the best individual of the current generation. For any individual w_i^t from the t th generation, the new variation individual v_i^t can be generated as followings.

$$v_i^t = w_c^t + F * (w_i^t - w_j^t) \tag{3}$$

where w_i^t and w_j^t are two different individuals of generation t, F is a normally distributed random variable within the range [0,1].

Algorithm 2 Evolution Process of EWLNB

Input:

Maximum Evolution Generation $MaxGen$;
Population W;
Training dataset D^a;
Test dataset D^b;

Output:

The best parameter vector w_c;

$W \leftarrow$ the initial $w_{i,j}$ value of w_i for each individual is set to a random number distributed between (0, 1];
while $t < MaxGen$ **do**
  $f[w_i^t] \leftarrow$ using D^a as the training dataset, D^b as the test dataset, w_i^t as the parameter vector, calculating the fitness of each w_i^t for the the whole W^t;
  $f[w_c^t] \leftarrow$ choose the best w_i;
  for all each y_i in D^a **do**
    $v_i^t \leftarrow$ apply w_c^t and a normally distributed random variable $N(0,1)$ to w_i^t and obtain the mutation individual v_i^t ;
    $u_i^t \leftarrow$ crossover w_i^t and v_i^t to obtain the crossover individual u_i^t
    $w_i^{t+1} \leftarrow$ get the new generation through the following Equation.

$$w_i^{t+1} = \begin{cases} w_i^t & f(u_i^t) \leq f(w_i^t) \\ u_i^t & f(u_i^t) > f(w_i^t) \end{cases} \quad (5)$$

  end for
end while
$w_c \leftarrow w_c^{MaxGen}$;
return w_c;

Individual Crossover: we use the crossover operation to exchange the dimensions of vectors in the t th generation of W^t and V^t. For each w_i^t in W^t, there is a corresponding v_i^t in V^t. The crossover vectors are created through W^t and V^t to involve more varieties using the following equation.

$$u_{j,i}^t = \begin{cases} v_{j,i}^t & if(rand_{j,i} \leq CR) or (j = j_{rand}) \\ w_{j,i}^t & otherwise \end{cases} \quad (4)$$

where $w_{j,i}^t$ is the j th dimension of the i th individual in the t th generation. $u_{j,i}^t$ is the j th dimension of the i th mutation vector in the t th generation, $v_{j,i}^t$ is the formed j th dimension of the i th crossover vector in the t th generation. CR is a fixed parameter for the whole algorithm within the range [0,1], and in our algorithm, the value of CR is 0.5. $rand_{j,i}$ is a normally distributed random variable within the range [0,1], which is generated for every dimension of the vector, and j_{rand} is a normally distributed random integer within the range $[0, n]$, where n is the dimension of the individual. In our algorithm, the value of n is two, j_{rand} is generated once for every individual.

3) Update of EWLNB: To determine whether the crossover individual u_i^t can replace the target individual vector w_i^t to be the new individual w_i^{t+1} in the $t+1$ th generation, the EWLNB algorithm uses Eq. (5) to adopt a greedy search strategy. It is chosen as the offspring only if the fitness of u_i^t is better than that of the target individual w_i^t, otherwise, the individual w_i^t is maintained in the tth generation. Following this process, the system again chooses the individual w_c^{t+1} with the best fitness performance in the $t+1$ th generation as the new local optimal individual.

A complete evolutionary process (as shown in Algorithm 2) of the population includes Evaluation and Update, which continuously repeats until the algorithm surpasses the pre-set maximum number $MaxGen$, or the same result is obtained for a number of consecutive iterations. After obtaining the best individual w_c, corresponding to the obtained parameter values of $threshold$ and $weight$, we use the values to construct the extended Training dataset D^* and build a NB classifier over D^* to classify the test instances.

III. EXPERIMENT

A. Benchmark data and parameters

We implement the proposed method using the WEKA [22], [23] data mining tool and validate its performance on 20 benchmark datasets from the UCI data repository [24]. The detailed information of the 20 datasets is shown through table I. Because NB is designed for categorical attributes, we first replace all missing attribute values in our experiment using the unsupervised attribute filter ReplaceMissingValues in WEKA. Then, we apply unsupervised filter Discretize in WEKA to discretize numeric attributes into nominal attributes. The three parameters L, $MaxGen$ and CR in our algorithm are set to 50, 50, and 0.5, respectively. All results are obtained via 10 runs of 10-fold cross validation.

B. Baseline Methods

For comparison purposes, we compare ELWNB with the following baseline methods.

. NB: a standard Naive Bayes classifier with conditional attribute independence assumption.
. LNB: a lazy Naive Bayes classifier which calculates the distances of instances through attribute similarity.

C. Evaluation criterion

In our experiments, the selected algorithms are evaluated on the criterion of classification accuracy (measured by ACC), which is calculated by the percentage of successful predictions on domain specific problems [25], [26], [27]. The detailed experiment results are shown in Table II and table III. We compare our algorithm with each other algorithm via a two-tailed t-test with significantly different probability of 0.95, because we speak of two results for a dataset as being "significantly different" only if the difference is statistically

Table I
20 UCI DATASETS.

Datasets	Instances	Attributes	Classes	Missing	Numeric
anneal	898	39	6	Y	Y
lymph	148	19	4	N	Y
artificial	10218	8	10	N	N
monks	556	7	2	N	N
audiology	226	70	24	Y	N
newthyroid	215	6	3	N	N
energy	768	9	37	N	N
primary-tumor	339	18	21	Y	N
glass	214	10	7	N	Y
qar	1055	42	2	N	N
hypothyroid	3772	30	4	Y	Y
robot	5456	25	4	N	N
ionosphere	351	35	2	N	Y
sick	3772	30	2	Y	Y
iris	150	5	3	N	Y
vehicle	846	19	4	N	Y
labor	57	17	2	Y	Y
vowel	990	14	11	N	Y
letter	20000	17	26	N	Y
zoo	101	18	7	N	Y

significant at the 0.05 level according to the corrected two-tailed t-test [28], as shown in Table II. Table III shows the results of the two-tailed t-test with a confidence level of 95% between each pair of algorithms in terms of accuracy. Each entry w/t/l in the tables means that the algorithm in the corresponding row wins in w datasets, ties in t datasets, and loses in l datasets, compared to the algorithm in the corresponding column. The results displayed in Tables II and III show that ELWNB significantly outperforms LNB and NB measured by ACC. We summarize the highlights as

1. It is clear that ELWNB significantly outperforms NB measured by ACC: the w/t/l value on ACC between ELWNB and NB is 8/12/0. The average ACC for ELWNB is 83.11%, which is higher than the average ACC 76.23% of NB.
2. It can also be seen that ELWNB significantly outperforms LNB measured by ACC: the w/t/l value on ACC between ELWNB and LNB is 8/12/0. The average ACC for ELWNB is 83.11%, which is higher than the average ACC 80.25% of LNB.

IV. CONCLUSIONS

In this paper, we first investigated the classification performance of improved NB methods and found that they all have a common disadvantage, that is, the building of the NB classifier is carried out without consideration of the NB's objective function. Motivated by this observation, we presented a new evolutionary lazy learning algorithm called ELWNB. We experimentally tested the proposed algorithm ELWNB, using 20 UCI datasets recommended by WEKA, and compared our algorithm ELWNB with NB and LNB. The experimental results show that our algorithm significantly outperforms NB and LNB in yielding accurate classification. We believe that our work provides an effective

Table II
EXPERIMENTAL RESULTS: ACC AND STANDARD DEVIATION.

Datasets	ELWNB	LNB		NB	
anneal	97.77±1.82	97.44±1.58	○	94.32±2.38	○
lymph	84.33±8.16	86.33±8.80		85.67±9.55	
artificial	68.58±1.84	47.72±1.19	○	36.40±1.00	○
monks	97.29±2.75	75.36±2.53	○	74.64±2.15	○
audiology	77.39±5.55	78.32±7.12		71.23±7.03	
newthyroid	95.30±4.98	92.99±5.08		92.08±4.46	
energy	65.76±4.86	58.85±4.30	○	45.05±3.92	○
primary-tumor	47.49±5.20	47.50±4.90		46.89±4.32	
glass	58.40±7.05	59.85±7.39		60.32±9.69	
qar	80.29±4.91	81.05±3.37		79.81±4.43	
hypothyroid	92.82±1.37	92.84±0.88		92.79±1.02	
robot	89.08±1.49	83.84±1.79	○	80.57±2.02	○
ionosphere	91.17±3.42	91.44±3.82		90.89±3.49	
sick	97.64±0.63	97.08±0.54	○	96.74±0.53	○
iris	95.33±8.34	96.67±4.71		94.67±8.20	
vehicle	64.18±4.54	67.86±4.73		61.82±3.54	
labor	91.67±14.16	90.00±14.05		93.33±11.65	
vowel	90.61±2.70	87.68±2.22	○	67.07±4.21	○
letter	80.95±2.30	76.00±2.23	○	66.15±2.15	○
zoo	96.18±6.54	96.18±6.54		94.18±6.60	
AVERAGE	**83.11±4.60**	**80.25±3.89**		**76.23±4.62**	

○: Statistically significant degradation.

Table III
SUMMARY OF TWO-TAILED T-TEST RESULTS

	LNB	**NB**
NB	**0/13/7**	
ELWNB	**8/12/0**	**8/12/0**

data mining classification algorithm. An aspect of ELWNB that could clearly be investigated further is the method of calculating the number of times that a specific instance should be copied. Extending ELWNB to calculate the times that a specific instance should be copied in a more efficient way is our main direction for future research.

ACKNOWLEDGMENTS

This work was supported in part by the China Scholarship Council Foundation (No. 201206410056), and the Australian Research Council (ARC) Discovery Projects under Grant No. DP140100545.

REFERENCES

[1] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2, pp. 131–163, 1997.

[2] D. Berend and A. Kontorovich, "A finite sample analysis of the naive bayes classifier," *Journal of Machine Learning Research*, vol. 16, pp. 1519–1545, 2015.

[3] J. Chen, H. Huang, S. Tian, and Y. Qu, "Feature selection for text classification with naïve bayes," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5432–5435, 2009.

[4] C. Zhang, G.-R. Xue, Y. Yu, and H. Zha, "Web-scale classification with naive bayes," in *Proceedings of the 18th International Conference on World Wide Web (WWW)*. ACM, 2009, pp. 1083–1084.

[5] P. Shanmuganathan and C. Sakthivel, "An efficient naive bayes classification for sentiment analysis on twitter," *Data Mining and Knowledge Engineering*, vol. 7, no. 5, pp. 180–185, 2015.

[6] R. Timofte, T. Tuytelaars, and L. Van Gool, "Naive bayes image classification: beyond nearest neighbors," in *Proceedings of the 11th Asian Conference on Computer Vision (ACCV)*. Springer, 2012, pp. 689–703.

[7] S. McCann and D. G. Lowe, "Local naive bayes nearest neighbor for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 3650–3656.

[8] J. Hernández-González, I. Inza, and J. A. Lozano, "Learning bayesian network classifiers from label proportions," *Pattern Recognition*, vol. 46, no. 12, pp. 3425–3440, 2013.

[9] L. Jiang, D. Wang, Z. Cai, and X. Yan, "Survey of improving naive bayes for classification," in *Advanced Data Mining and Applications*. Springer, 2007, pp. 134–145.

[10] P. Langley and S. Sage, "Induction of selective bayesian classifiers," in *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann Publishers Inc., 1994, pp. 399–406.

[11] J. Wu and Z. Cai, "Attribute weighting via differential evolution algorithm for attribute weighted naive bayes (wnb)," *Journal of Computational Information Systems*, vol. 7, no. 5, pp. 1672–1679, 2011.

[12] N. A. Zaidi, J. Cerquides, M. J. Carman, and G. I. Webb, "Alleviating naive bayes attribute independence assumption by attribute weighting," *Journal of Machine Learning Research*, vol. 14, pp. 1947–1988, 2013.

[13] J. Wu, S. Pan, X. Zhu, Z. Cai, P. Zhang, and C. Zhang, "Self-adaptive attribute weighting for naive bayes classification," *Expert Systems With Applications*, vol. 42, no. 3, pp. 1487–1502, 2015.

[14] J. Wu and Z. Cai, "A naive bayes probability estimation model based on self-adaptive differential evolution," *Journal of Intelligent Information Systems*, vol. 42, no. 3, pp. 671–694, 2014.

[15] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[16] L. Jiang, H. Zhang, and Z. Cai, "A novel bayes model: Hidden naive bayes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 10, pp. 1361–1371, 2009.

[17] J. Wu, S. Pan, X. Zhu, P. Zhang, and C. Zhang, "SODE: self-adaptive one-dependence estimators for classification," *Pattern Recognition*, vol. 51, pp. 358–377, 2016.

[18] E. Frank, M. Hall, and B. Pfahringer, "Locally weighted naive bayes," in *Proceedings of the Nineteenth International Conference on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann Publishers Inc., 2002, pp. 249–256.

[19] L. Jiang and Y. Guo, "Learning lazy naive bayesian classifiers for ranking," in *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2005, pp. 5–pp.

[20] J. Wu, S. Pan, Z. Cai, X. Zhu, and C. Zhang, "Dual instance and attribute weighting for naive bayes classification," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 1675–1679.

[21] K. V. Price, "Differential evolution vs. the functions of the 2 nd iceo," in *IEEE International Conference on Evolutionary Computation (CEC)*. IEEE, 1997, pp. 153–157.

[22] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[23] R. R. Bouckaert, E. Frank, M. A. Hall, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "Weka—experiences with a java open-source project," *The Journal of Machine Learning Research*, vol. 11, pp. 2533–2541, 2010.

[24] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[25] J. Wu, X. Zhu, C. Zhang, and P. S. Yu, "Bag constrained structure pattern mining for multi-graph classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2382–2396, 2014.

[26] J. Wu, S. Pan, X. Zhu, and Z. Cai, "Boosting for multi-graph classification," *IEEE Transactions on Cybernetics*, vol. 45, no. 3, pp. 416–429, 2015.

[27] J. Wu, S. Pan, X. Zhu, C. Zhang, and X. Wu, "Positive and unlabeled multi-graph learning," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–12, 2016.

[28] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, no. 3, pp. 239–281, 2003.