Edited Naive Bayes

J. M. Martínez-Otzeta, B. Sierra, E. Lazkano,

M. Ardaiz, E. Jauregi

Department of Computer Science and Artificial Intelligence
University of the Basque Country
P. Manuel Lardizabal 1, 20018 Donostia-San Sebastián
Basque Country, Spain
http://www.sc.ehu.es/ccwrobot
ccbmaotj@si.ehu.es

Abstract

Naive Bayes is a well-known and studied algorithm both in statistics and machine learning. Bayesian learning algorithms represent each concept with a single probabilistic summary. This paper presents a variant of the Naive Bayes method, in which the original training set is augmented in the following fashion: Leave-One-Out procedure is applied over the training set, and incorrectly classified instances according to Naive Bayes model are duplicated. The augmented dataset is used to induce the model. The motivation behind this idea is that giving more importance to *hard* instances (in this case, duplicating them) might contribute to make the model more accurate over that subset of the instance space. We have tested this algorithm over 41 UCI datasets. The results suggest that the chance of obtaining a significant better performance than with the original Naive Bayes approach are much greater than the opposite.

Keywords: Naive Bayes, Data Mining, Machine Learning, Supervised Classification.

1 Introduction

In the supervised learning task [17], the main goal is to induce a classification model from a given training database for which we know the labeling of every sample. Existing paradigms come from the world of the Artificial Intelligence and are grouped in the family of *Machine Learning* (ML).

The goal of supervised learning is to predict the class labels of examples that have not been seen; for this purpose a subset of cases are left out of the model construction and constitute the so called test database. The training-test division of the whole database is performed more than once for a number of random partitions in order to show the soundness and generalization capabilities of the induced classifiers.

The Naive Bayes algorithm is one of the oldest and most well-known machine learning approaches [16]. To make the probability based classification computation feasible, Naive Bayes assume independence among the variables, in a *naive* fashion. Such assumption in real problems is certainly a simplistic one. But, despite this objection, Naive Bayes works rather well when tested on actual databases, particularly when combined with some attribute selection procedure.

On the other hand, many lines of research have

dealt with the task of editing the training set in order to achieve better results. We could include in that family, for example:

- Prototype selection. Techniques have been developed to select a subset of the whole set of instances while maintaining accuracy. Widely used as a computational load improvement in Nearest Neighbors methods [3].
- Attribute selection [9, 7, 8, 19]. Detection of irrelevant or redundant attributes can lead to better results in accuracy as well as in computational load.
- Bagging [2]. Building models over different subsamples of a unique database and then combining the different outcomes, typically by voting, has shown to be a useful way of improving accuracy with respect to the base model.
- Boosting [4]. Instead of making subsampling as in bagging, the whole dataset is used to induce several classifiers in an iterative manner. The difference among each execution is that misclassified instances in Nth step, will have a greater weight in $(N+1)$th step, in order to make the algorithm focus on *hard* instances. The final outcome of the algorithm is the result of a voting among all the previously built models.

In this paper a new variant of the Naive Bayes algorithm is presented and tested. An edition of the training set is made, in the following fashion: first, Leave-One-Out procedure is applied over the training set, and incorrectly classified instances according to Naive Bayes model are duplicated. The set built adding the duplicated instances to the original training set is used to induce the model. This procedure can be also viewed as a way of *auto-boosting* where *hard* instances are given more importance (duplicating them in the training set) in order to make the algorithm focus on the instance subset that has turned to be more difficult to correctly classify. Experimental results have been obtained using all the UCI datasets of medium size (from 100 to 1000 cases), and show that the presented paradigm significantly outperforms standard Naive-Bayes.

The rest of the paper is organized as follows. Section 2 describes the Naive Bayes classifier In section 3 related work about Naive Bayes modifications and improvements is presented. The new proposed approach is presented in section 4 and results obtained are shown in section 5. Final section is dedicated to conclusions and points out the future work.

2 Naive Bayes

Given a database of cases consisting of a set of predictor variables $X_1, X_2, \cdots, X_n$ and a special variable Y which value must be predicted, being each of the N cases in the form

$$\text{Case}_i : X_1 = x_{1i}, X_2 = x_{2i}, \cdots, X_n = x_{ni}, Y = y_l$$

and being, without lost of generality, $Y = y_1, Y = y_2, \cdots, Y = y_m$ the m possible values the class variable can take (the m categories considered in the classification problem), theoretically, Bayes' rule minimizes error by selecting the class y_j with the largest posterior probability for a given example $\mathbf{X}$ of the form $X =< X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n >$, as indicated below:

$$P(Y = y_j|X) = \frac{P(Y = y_j)P(\mathbf{X}|Y = y_j)}{P(\mathbf{X})}.$$

Since $\mathbf{X}$ is a composition of n discrete values, one can expand this expression to:

$$P(Y = y_j|X_1 = x_1, ..., X_n = x_n) =$$

$$\frac{P(Y = y_j)P(X_1 = x_1, ..., X_n = x_n|Y = y_j)}{P(X_1 = x_1, ..., X_n = x_n)}$$

where $P(X_1 = x_1, ..., X_n = x_n|Y = y_j)$ is the conditional probability of the instance $\mathbf{X}$ given the class y_j; $P(Y = y_j)$ is the a priori probability that one will observe class y_j; $P(\mathbf{X})$ is the prior probability of observing the instance $\mathbf{X}$. All these parameters are estimated from the training set. However, a direct application of these rules is difficult due to the lack of sufficient data in the training set to reliably obtain all the conditional probabilities needed by the model. One simple form of the previous model has been studied [16] that assumes independence of the observations of feature variables $X_1, X_2, ..., X_n$ given the class variable Y, which allows us to use the next equality

$$P(X_1 = x_1, ..., X_n = x_n|Y = y_j) \propto$$

$$\prod_{i=1}^{n} P(X_i = x_i|Y = y_j)$$

where $P(X_i = x_i|Y = y_j)$ is the probability of an instance of class y_j having the observed attribute value x_i. In the core of this paradigm there is an assumption of independence between the occurrence of features values, that is not true in many tasks; however, it is empirically demonstrated that this paradigm gives good results in many real domains, typically in medical tasks. A review of the health of Naive Bayes method at its fortieth birthday can be found in [15].

3 Related work

Some techniques have been developed to improve the performance of the Naive Bayes classifier. Some of them apply different Naive Bayes classifiers to different regions of the input space. Among them:

- Langley developed the recursive Naive Bayes [13], an algorithm that recursively constructs a hierarchy of probabilistic concept descriptions. According to the author the results are mixed and inconclusive, but interesting enough to recommend closer examination.
- Kohavi presented the Naive Bayes tree [10], a hybrid algorithm. It generates a regular univariate decision tree, but the leaves contain a Naive Bayes classifier built from the examples that fall at this node. The approach retains the interpretability of Naive Bayes and decision trees, while resulting in classifiers that frequently outperform both constituents, especially in large datasets.

Other approaches have focused on building new attributes that reflect interdependencies between original attributes. For example:

- Kononenko described the semi Naive Bayes classifier [12]. He attempted to join pairs of attributes, making a cross-product attribute, based on statistical tests for independence. The experimental evaluation was inconclusive.
- Pazzani devised the constructive Bayesian classifier [18]. It employs a wrapper model to find the best Cartesian product attributes from existing nominal attributes, considering deletion of existing attributes. It has been shown to improve the Naive Bayes classifier.

Other works focus in minimizing a quadratic loss function instead of a 0-1 loss function as usual. For example:

- Gama presented an algorithm that iteratively update the contingency tables in order to improve the probability class distribution associated with each training example [6]. Experimental evaluation shows minor but consistent gains in accuracy.

4 Proposed approach

In boosting techniques, a distribution or set of weights over the training set is maintained. On each execution, the weights of incorrectly classified examples are increased so that the base learner is forced to focus on the hard examples in the training set. A good description of boosting can be found in [5].

Following the idea of focusing in the *hard* examples, we have investigated if it is possible to auto-boost a simple and well-known algorithm, as Naive Bayes. Our approach, Edited Naive Bayes, works as follows:

- Leave-One-Out cross-validation is performed over the training set using the Naive Bayes algorithm
- Incorrectly classified instances are duplicated
- Naive Bayes algorithm is executed, for the test cases, estimating the probabilities over the augmented database

A graphical representation of the process can be viewed in Figure 1.

The pseudocode of this approach is shown in Figure 2

Let us note that this approach is equivalent to duplicate the weight of incorrectly classified instances, according to LOO cross-validation with Naive Bayes.

In this manner, the core of this new approach consists of inflating the training database adding the

instances misclassified by the Naive Bayes algorithm, and then execute again Naive Bayes over the new database obtained. It has to be said that this approach increases the computational cost only in the model induction phase, while the classification costs are similar as in the original Naive Bayes paradigm.

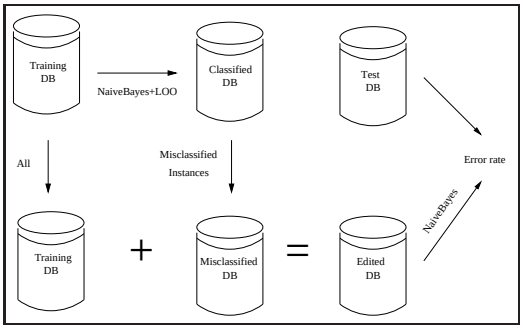

Figure 1: Edition process over the original training set

begin EDITED NAIVE BAYES
a) Let T be the training set
b) Perform Leave-One-Out cross-validation over T using *Naive Bayes*
c) Let $Mis(T)$ be the set of misclassified instances in the previous step
d) $Augmented(T) = T \cup Mis(T)$ (duplication of the misclassified instances)
e) Create a model M using *Naive Bayes* over $Augmented(T)$
f) Classify the test cases using M
end EDITED NAIVE BAYES

Figure 2: The pseudocode of Edited Naive Bayes

5 Experimental Results

Our aim has been to test this algorithm over a high number of the well-known UCI repository databases [1]. To do so, we have selected all the databases of medium size (between 100 and 1000 instances) among those converted to the $\mathcal{MLC}$++ [11] format, and located in this public repository: *[http://www.sgi.com/tech/mlc/db/]* This amounts to 59 databases, from which we have selected one of each family of problems. For example, we have chosen *monk1* and not *monk1-cross*, *monk1-full* or *monk1-org*. After this final selection, we were left with the 41 databases shown in Table 1.

In order to give a real perspective of applied methods, we have performed five times a 10-Fold Cross-validation [20] in all experiments (50 different data-test evaluation for each dataset). In each of the five times, all databases have been randomly partitioned into ten sets. One of these sets has been hold out as a test set and the other nine have been joined together to build the training set over which the experiment is performed. This procedure has been repeated ten times, one for each subset. Obviously all the validation files used have been always the same for the two algorithms: Naive Bayes and our approach, Edited Naive Bayes. We have made use of the $\mathcal{MLC}$++ libraries to perform Leave-One-Out cross-validation, 10-fold cross-validation as well as the Naive Bayes classification algorithm. The $\mathcal{MLC}$++ outcome has been the basis to edit the original training set and this edited training set has been presented again to $\mathcal{MLC}$++ .

In Table 1 the characteristics of the 41 databases along with the results of the experiments are shown. The error rates under the label Naive Bayes are the mean and the standard deviation of the fifty executions (five times 10-fold cross-validation) of the experiment using Naive Bayes. Likewise, the error rates under the label Edited Naive Bayes are those corresponding to Edited Naive Bayes over the same data. In boldface is shown the best result for each database. An arrow pointing upwards (↑) means the method labeling that column outperforms the other significantly under a confidence level of 95% according to Wilcoxon signed rank test [21] . For example, Naive Bayes outperforms Edited Naive Bayes on *Anneal* database while Edited Naive Bayes outperforms Naive Bayes on *Australian* database.

From Table 1, we can extract the following conclusions:

- In 22 out of 41 databases, the average error rate of Edited Naive Bayes is lower than with Naive Bayes, while in 14 databases Naive Bayes outperforms Edited Naive Bayes.
- In 14 out of 41 databases that difference is significative in favor of Edited Naive Bayes and in 8 cases is significative against our algorithm.

Table 1: The characteristics of the 41 databases used in this experiment, along with the error rate of Naive Bayes (NB) and Edited Naive Bayes (ENB). The best result is written in boldface. An ↑ means that the method labeling that column outperforms the other according to Wilcoxon signed rank test with a confidence level of 95%. For the sake of completeness, a row with boosting NB (as implemented in $\mathcal{MLC}$++) is included, performing roughly similar to ENB (ENB vs. BoostingNB: +17 -16 =7).

Database	*#Cases*	*#Attrib*	*#Classes*	*Naive Bayes*	*Edited NB*	*Boosting NB*
Anneal	798	38	6	↑ **8.46±0.60**	8.80±0.61	8.80±0.56
Audiology	226	69	24	**38.89±3.13**	38.91±3.54	37.57±3.72
Australian	690	14	2	22.90±1.36	↑**20.72±1.33**	22.75±1.38
Auto	205	26	7	41.02±2.72	**40.05±3.37**	40.95±2.14
Balance-Scale	625	4	3	**9.14±1.52**	9.30±1.67	9.30±1.52
Banding	238	29	2	↑**21.43±3.26**	24.82±3.41	23.95±2.64
Breast	699	10	2	**3.86±0.77**	4.01±0.79	3.86±0.77
Breast-cancer	286	9	2	↑**26.93±2.45**	30.43±1.90	28.65±2.22
Cars	392	8	3	0.51±0.34	0.51±0.34	0.51±0.34
Cars1	392	7	3	↑**31.16±3.46**	34.45±3.29	32.19±3.52
Cleve	303	14	2	16.49±1.38	↑**15.52±1.79**	16.17±1.51
Corral	129	6	2	15.58±4.12	↑**9.29±2.76**	13.97±3.57
Crx	690	15	2	22.32±1.56	↑**21.45±1.41**	21.74±1.43
Diabetes	768	8	2	24.34±1.40	↑**23.43±1.18**	24.34±1.40
Echocardiogram	132	7	2	28.96±2.91	**26.65±3.40**	28.96±2.91
German	1000	20	2	24.60±1.70	**24.30±1.13**	25.50±1.60
Glass	214	9	7	**52.79±2.36**	55.56±3.50	53.25±2.74
Glass2	163	9	2	39.38±4.42	↑**35.66±3.68**	38.12±4.84
Hayes-Roth	162	4	3	31.25±3.49	↑**21.88±3.39**	32.50±2.76
Heart	270	13	2	↑**16.30±1.85**	17.04±1.93	15.93±1.92
Hepatitis	155	19	2	16.12±2.92	**15.46±2.73**	16.08±2.56
Horse-colic	368	28	2	21.51±2.65	↑**18.50±2.09**	18.24±2.48
Hungarian	294	13	2	15.59±2.70	↑**15.24±2.34**	15.57±2.78
Ionosphere	351	34	2	15.10±1.76	↑**7.40±0.86**	14.82±1.85
Iris	150	4	3	4.67±1.42	4.67±1.42	4.67±1.42
Liver-disorder	345	6	2	44.32±3.34	**41.44±2.28**	42.33±2.99
Lymphography	148	19	4	19.57±2.88	↑**18.81±3.08**	20.14±3.25
Monk1	432	6	2	↑**25.03±1.95**	51.14±2.17	30.10±2.02
Monk2	432	6	2	↑**33.78±2.80**	49.50±2.50	36.56±2.82
Monk3	432	6	2	2.78±0.47	2.78±0.47	0.00±0.00
Pima	768	8	2	24.10±1.88	↑**22.79±1.92**	24.36±1.75
Primary-org	339	17	22	↑**51.35±2.53**	54.60±3.07	52.53±3.17
Solar	323	11	6	**32.81±2.46**	33.13±2.36	32.17±2.37
Sonar	208	59	2	31.21±3.60	↑**24.02±2.83**	33.12±3.31
ThreeOf9	512	9	2	19.74±1.92	**19.55±2.02**	18.76±1.74
Tic-tac-toe	958	9	2	30.28±1.75	↑**27.77±1.47**	19.42±1.95
Tokyo1	963	44	2	11.57±1.76	**11.47±1.96**	11.47±1.78
Vehicle	846	18	4	**54.59±1.41**	55.09±1.56	Not applicable[1]
Vote	435	16	2	9.66±0.68	**9.43±0.65**	4.36±0.86
Wine	178	13	3	2.81±1.25	2.81±1.25	2.81±1.25
Zoo	101	16	7	12.91±4.96	12.91±4.96	12.91±4.96

In Table 2 and 3 are summarized these data.

When restricted to problems with just two classes, the comparison between the two algorithms yields better figures for Edited Naive Bayes. In this case, the rate of wins-losses of

[1] $\mathcal{MLC}$++ fails to build a categorizer with error below 50%

Edited Naive Bayes against Naive Bayes is 19-6 from a total of 26 databases, while the number of these wins which are significative is 12-5.

This improvement when restricted to two-classes databases, can be noticed more graphically when comparing the entries for *Glass* and *Glass2* in Table 1. *Glass2* is a subset of *Glass* where the seven original classes have been put together in just two different classes. The results for *Glass* are better (but not significantly) for the original Naive Bayes, while in the case of *Glass2* Edited Naive Bayes outperforms Naive Bayes significantly.

We have performed another statistical test, now regarding the number of times one algorithm defeats the other when run over the same database. Thus, we can see each of the executions of the five times 10-fold cross-validation as a *match* and record the number of matches an algorithm defeats the other. If this is done for every database, it is possible to check again these results according to Wilcoxon test, and obtain the following results: over the 41 databases, the difference is significative for a confidence level of 68%, while when applied to the 26 two-classes databases, that confidence level raises to 90%.

6 Conclusions and Further Work

In this paper a modification of Naive Bayes is presented. The main idea is to augment the training test duplicating the incorrectly classified cases according to Naive Bayes when performing Leave-One-Out cross-validation.

The experimental results support the idea that such edition of the training set can lead to good results, frequently outperforming Naive Bayes. In a number of cases these results are significant according to Wilcoxon signed rank test. Such results seems to be improved when applied to two-classes problems.

Further work involves a characterization of the databases for which this procedure performs better, as well as other more elaborated ways of editing the training set. For example, not just duplicating the misclassified instances, but repeating them other number of times. More complex joint probability distributions in form of Bayesian networks structures are also to be tested with this database edition new idea, as well as new combinations of the proposed model with some standard paradigms [14].

7 Acknowledgments

This work has been supported by the University of the Basque Country under grant 1/UPV00140.226-E-15412/2003 and by the Gipuzkoako Foru Aldundia OF-838/2004.

Table 2: Summary of the results of Table 1

	Naive Bayes	*Edited Naive Bayes*	*#Databases*
Better in average	14	22	41
Significantly better (95%)	8	13	41

Table 3: Summary of the results of Table 1 when restricted to two-classes databases

	Naive Bayes	*Edited Naive Bayes*	*#Databases*
Better in average	6	19	26
Significantly better (95%)	5	12	26

References

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Recognition Classification Techniques.* IEEE Computer Society Press, 1991.

[4] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[5] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[6] J. Gama. Iterative bayes. *Theoretical Computer Science*, 292(2):417–430, 2002.

[7] I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra. Feature subset selection by bayesian networks based optimization. *Artificial Intelligence*, 123(1-2):157–184, 2000.

[8] I. Inza, P. Larrañaga, and B. Sierra. Feature subset selection by bayesian networks: a comparison with genetic and sequential algorithms. *International Journal of Approximate Reasoning*, 27(2):143–164, 2001.

[9] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.

[10] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.

[11] R. Kohavi, D. Sommerfield, and J. Dougherty. Data mining using $\mathcal{MLC}$++, a machine learning library in $\mathcal{C}$++. *International Journal of Artificial Intelligence Tools*, 6(4):537–566, 1997.

[12] I. Kononenko. Semi-naive bayesian classifier. *European Working Session on Learning-EWSL91. Lecture Notes in Computer Science*, 482:206–219, 1991.

[13] P. Langley. Induction of recursive bayesian classifiers. *Proceedings of the European Conference on Machine Learning. Lecture Notes in Computer Science*, 667:153–164, 1993.

[14] E. Lazkano and B. Sierra. Bayes-nearest:a new hybrid classifier combining bayesian network and distance based algorithms. *11th Protuguese Conference on Artificial Intelligence, EPIA 2003. Lecture Notes in Artificial Intelligence*, 2902:171–183, 2003.

[15] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, Germany, 1998. Springer-Verlag.

[16] M. Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30, 1961.

[17] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[18] M. Pazzani. Constructive induction of cartesian product attributes. In *Proc. Conf. ISIS96: Information, Statistics and Induction in Science*, pages 66–77, 1996.

[19] B. Sierra, E. Lazkano, I. Inza, M. Merino, P. Larrañaga, and J. Quiroga. Prototype selection and feature subset selection by estimation of distribution algorithms. a case study in the survival of cirrhotic patients treated with tips. *Artificial Intelligence in Medicine. 8th Conference on Artificial Intelligence in Medicine in Europe, AIME 2001. Lecture Notes in Artificial Intelligence*, 2101:20–29, 2001.

[20] M. Stone. Cross-validation choice and assessment of statistical procedures. *Journal Royal of Statistical Society*, 36:111–147, 1974.

[21] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.