Learning Semi Naïve Bayes Structures by Estimation of Distribution Algorithms

V. Robles[1], P. Larrañaga[2], J.M. Peña[1], M.S. Pérez[1], E. Menasalvas[1], and V. Herves[1]

[1] Department of Computer Architecture and Technology, Technical University of Madrid, Madrid, Spain, {vrobles,jmpena,mperez,emenasalvas}@fi.upm.es, vherves@datsi.fi.upm.es

[2] Department of Computer Science and Artificial Intelligence, University of the Basque Country, San Sebastián, Spain, ccplamup@si.ehu.es

Abstract. Recent work in supervised learning has shown that a surprisingly simple Bayesian classifier called naïve Bayes is competitive with state of the art classifiers. This simple approach stands from assumptions of conditional independence among features given the class. Improvements in accuracy of naïve Bayes has been demonstrated by a number of approaches, collectively named semi naïve Bayes classifiers. Semi naïve Bayes classifiers are usually based on the search of specific values or structures. The learning process of these classifiers is usually based on greedy search algorithms. In this paper we propose to learn these semi naïve Bayes structures through estimation of distribution algorithms, which are non-deterministic, stochastic heuristic search strategies. Experimental tests have been done with 21 data sets from the UCI repository.

Keywords. Naïve Bayes, semi Naïve Bayes, heuristic search, estimation of distributions algorithms.

1 Introduction

The naïve Bayes classifier [5,12] is a probabilistic method for classification. It can be used to determine the probability that an example belongs to a class given the values of the predictor variables. The naïve Bayes classifier guarantees optimal induction given a set of explicit assumptions [3]. However, it is known that some of these assumptions are not compliant in many induction scenarios, for instance, the condition of variable independence respecting to the class variable. Improvements of accuracy has been demonstrated by a number of approaches, collectively named semi naïve Bayes classifiers, which try to adjust the naïve Bayes to deal with a-priori unattended assumptions.

Previous semi naïve Bayes classifiers can be divided into three groups, depending on different pre/post-processing issues: (i) to manipulate the variables to be employed prior to application of naïve Bayes induction [16,18,23], (ii) to select subsets of the training examples prior to the application of naïve Bayes classification [14,17] and (iii) to correct the probabilities produced by the standard naïve Bayes [8,10,26].

The learning process of the semi naïve Bayes classifiers is usually based on greedy search algorithms. In this paper we propose to learn these semi naïve Bayes structures

Fernando Moura Pires, Salvador Abreu (Eds.): EPIA 2003, LNAI 2902, pp. 244–258, 2003.

through estimation of distribution algorithms, which are non-deterministic, stochastic heuristic search strategies.

All the algorithms evaluated in this paper are shown on table 1. In the left column we have the greedy based algorithms, which are: *Iterative Bayes* [10], Pazzani [23] and *Adjusted Probability naïve Bayes* (APNBC) [26]. In the right column we have their corresponding heuristic based algorithms, which are: *Interval Estimation Naïve Bayes* IENB [24], Pazzani-EDA and APNBC-EDA. These last two algorithms have been developed as part of the contribution of this paper.

Table 1. Evaluated algorithms

Greedy algorithm	**Corresponding heuristic algorithm**
Iterative Bayes [10]	Interval Estimation Naïve Bayes [24]
Pazzani [23]	Pazzani-EDA
APNBC [26]	APNBC-EDA

The outline of this paper is as follows: Section 2 presents the naïve Bayes classifier. Section 3 is a brief introduction to estimation of distribution algorithms. Section 4 presents the algorithms evaluated in this paper. Section 5 illustrates the experimental results obtained with the UCI datasets. To conclude, section 6 gives the conclusions and discusses further future work.

2 Naïve Bayes

The naïve Bayes classifier [5,12] is a probabilistic method for classification. It performs an approximate calculation of the probability that an example belongs to a class given the values of predictor variables. The simple naïve Bayes classifier is one of the most successful algorithms on many classification domains. In spite of its simplicity, it is shown to be competitive with other more complex approaches in several specific domains.

This classifier learns from training data the conditional probability of each variable X_k given the class label c. Classification is then done by applying Bayes rule to compute the probability of C given the particular instance of $X_1, \ldots, X_n$,

$$P(C = c|X_1 = x_1, \ldots, X_n = x_n)$$

Naïve Bayes is based on the assumption that variables are conditionally independent given the class. Therefore the posterior probability of the class variable is formulated as follows,

$$P(C = c|X_1 = x_1, \ldots, X_n = x_n) \propto P(C = c) \prod_{k=1}^{n} P(X_k = x_k|C = c) \quad (1)$$

This equation is highly appropriate for learning from data, since the probabilities $p_i = P(C = c_i)$ and $p^i_{k,r} = P(X_k = x^r_k|C = c_i)$ may be estimated from training data. The result of the classification is the class with highest posterior probability.

3 Estimation of Distributions Algorithms

3.1 Introduction

EDAs [22,20] are non-deterministic, stochastic heuristic search strategies that form part of the evolutionary computation approaches, where number of solutions or individuals are created every generation, evolving once and again until a satisfactory solution is achieved. In brief, the characteristic that differentiates most EDAs from other evolutionary search strategies such as GAs is that the evolution from a generation to the next one is done by estimating the probability distribution of the fittest individuals, and afterwards by sampling the induced model. This avoids the use of crossing or mutation operators, and the number of parameters that EDAs require is considerably reduced.

EDA

$D_0 \leftarrow$ Generate M individuals (the initial population) randomly

Repeat for $l = 1, 2, \ldots$ until a stopping criterion is met

$D_{l-1}^N \leftarrow$ Select $N \leq M$ individuals from D_{l-1} according to a selection method

$\rho_l(\boldsymbol{x}) = \rho(\boldsymbol{x}|D_{l-1}^N) \leftarrow$ Estimate the probability distribution of an individual being among the selected individuals

$D_l \leftarrow$ Sample M individuals (the new population) from $\rho_l(\boldsymbol{x})$

Fig. 1. Pseudocode for the EDA approach

In EDAs, individuals are not said to contain genes, but variables which dependencies have to be analyzed. Also, while in other heuristics from evolutionary computation the interrelations between the different variables representing the individuals are kept in mind implicitly (e.g. building block hypothesis), in EDAs the interrelations are explicitly expressed through the joint probability distribution associated with the individuals selected at each iteration. The task of estimating the joint probability distribution associated with the database of the selected individuals from the previous generation constitutes the hardest work to perform, as it requires the adaptation of methods to learn models from data developed in the domain of probabilistic graphical models.

Figure 1 shows the pseudocode of a generic EDA algorithm, in which we distinguish four main steps in this approach:

1. At the beginning, the first population D_0 of M individuals is generated, usually by assuming an uniform distribution (either discrete or continuous) on each variable, and evaluating each of the individuals.
2. Secondly, a number N ($N \leq M$) of individuals are selected, usually the fittest.
3. Thirdly, the n–dimensional probabilistic model that better expresses the interdependencies between the n variables is induced.

4. Next, the new population of M new individuals is obtained by simulating the probability distribution learnt in the previous step.

Steps 2, 3 and 4 are repeated until a stopping condition is verified. The most important step of this new paradigm is to find the interdependencies between the variables (step 3). This task will be performed using techniques from the field of probabilistic graphical models.

3.2 EDAs in Discrete Domains

In the particular case where every variable is discrete, the probabilistic graphical model is called *Bayesian network*.

All the EDAs are classified depending on the maximum number of dependencies between variables that they accept (maximum number of parents that any variable can have in the probabilistic graphical model):

- Without interdependencies
  The Univariate Marginal Distribution Algorithm (UMDA) [21] is the most representative example of this category.
- Pairwise dependencies
  An example of this category is the greedy algorithm called MIMIC (Mutual Information Maximization for Input Clustering) [2]. The main idea in MIMIC is to describe the true mass joint probability as closely as possible by using only one univariate marginal probability and $n-1$ pairwise conditional probability functions.
- Multiple interdependencies
  EBNA (Estimation of Bayesian Network Algorithm) will be used as an example of this category. The EBNA approach firstly introduced in [6], where the authors use the Bayesian Information Criterion (BIC) as the score to evaluate the goodness of each structure found during the search.

3.3 EDAs in Continuous Domains

In the particular case where every variable in the individuals are continuous and follows a gaussian distribution, the probabilistic graphical model is called *Gaussian network*, and the EDA algorithms are named CEDAs *Continuous EDAs*.

Next, an analogous classification of continuous EDAs as for the discrete domain is done, in which these continuous EDAs are also classified depending on the number of dependencies they take into account:

- Without dependencies
  In this case, the joint density function is assumed to follow a n–dimensional normal distribution, and thus it is factorized as a product of n unidimensional and independent normal densities. UMDA$_c$ [19] is an example of continuous EDAs.
- Bivariate dependencies
  MIMIC$_c^G$ [19] is a representative example of this type of algorithms, which is basically an adaptation of the MIMIC algorithm [2] to the continuous domain.

– Multiple dependencies
  Algorithms in this section are approaches of EDAs for continuous domains in which there is no constraint in the learning of the density function every generation. EGNA$_{BGe}$ (Estimation of Gaussian Network Algorithm) [19] is a clear example of this category. The method used to find the Gaussian network structure is a Bayesian score+search. In EGNA$_{BGe}$ a local search is used to search for good structures.

4 Algorithms

In this section all the algorithms evaluated in the next section are described.

4.1 Iterative Bayes and Interval Estimation Naïve Bayes

The *Iterative Bayes* [10] algorithm begins with the a priori conditional probabilities obtained by naïve Bayes. Those probabilities are iteratively updated in order to improve the probability class distribution associated with each training example.

The iterative procedure uses a hill-climbing algorithm. At each iteration, all the examples in the training set are classified using the current conditional probabilities. The evaluation of the actual set of conditional probabilities is done by the next expression:

$$\frac{1}{n}\sum_{i=1}^{n}(1.0 - argmax_j\ p(C = c_j|X_1 = x_1, \ldots, X_n = x_n)) \tag{2}$$

where n represents the number of instances and j the number of classes. The iterative procedure proceeds while the evaluation function decreases till the maximum of 10 iterations.

To update the conditional probabilities, it is used the following heuristic:

1. The value of each conditional probability never goes below 0.01.
2. If an example is correctly classified then the increment is positive, otherwise it is negative. The value of the increment is $1.0 - p(Predict|X_1 = x_1, \ldots, X_n = x_n)/num.classes$. That is, the increment is a function of the confidence on predicting class $Predict$ and the number of classes.
3. For all attribute-values observed in the given example, the increment is added to all the entries for the predicted class and half of the increment is subtracted to the entries of all the other classes.

The conditional probabilities are incrementally updated each time a training example is presented. This implies that the order of the training examples could influence the final results.

Iterative Bayes tries to improve the conditional probabilities of naïve Bayes in an iterative way with a hill-climbing strategy. On the other hand, we have the algorithm *Interval Estimation naïve Bayes* (IENB) [24], that belongs to the approaches that correct the probabilities produced by the standard naïve Bayes.

In this approach, instead of calculating the point estimation of the conditional probabilities from data, as simple naïve Bayes makes, confidence intervals are calculated. After that, by searching for the best combination of values into these intervals, it is aimed to relieve the assumption of independence among variables the simple naïve Bayes makes. This search is carried out by EDAs and is guided by the accuracy of the classifiers.

There are three main important aspects in IENB algorithm:

1. **Calculation of Confidence Intervals**
   Given the dataset, the first step is to calculate the confidence intervals for each conditional probability and for each class probability. For the calculation of the intervals first the point estimations of these parameters must be computed.
   This way, each conditional probability $p^i_{k,r} = P(X_k = x^r_k | C = c_i)$, that has to be estimated from the dataset must be computed with the next confidence interval.

   For $k = 1, \ldots, n; i = 1, \ldots, r_0; r = 1, \ldots, r_k$ the next formula:

$$\left(\hat{p}^i_{k,r} - z_\alpha \sqrt{\frac{\hat{p}^i_{k,r}(1-\hat{p}^i_{k,r})}{N_i}}; \hat{p}^i_{k,r} + z_\alpha \sqrt{\frac{\hat{p}^i_{k,r}(1-\hat{p}^i_{k,r})}{N_i}} \right) \tag{3}$$

   denotes the interval estimation for the conditional probabilities $p^i_{k,r}$, where,
   r_k denotes the possible values of variable X_k
   r_0 represents the possible values of the class
   $\hat{p}^i_{k,r}$ denotes the point estimation of the conditional probability $P(X_k = x^r_k | C = c_i)$
   z_α denotes the $(1 - \frac{\alpha}{2})$ percentil in the $\mathcal{N}(0,1)$ distribution
   N_i is the number of cases in dataset where $C = c_i$

   Also, in a similar way, the probabilities for the class values $p_i = P(C = c_i)$, are estimated with the next confidence interval,

$$\left(\hat{p}_i - z_\alpha \sqrt{\frac{\hat{p}_i(1-\hat{p}_i)}{N}}; \hat{p}_i + z_\alpha \sqrt{\frac{\hat{p}_i(1-\hat{p}_i)}{N}} \right) \tag{4}$$

   where, $\hat{p}^i_i$ is the point estimation of the probability $P(C = c_i)$
   z_α is the $(1 - \frac{\alpha}{2})$ percentil in the $\mathcal{N}(0,1)$ distribution
   N is the number of cases in dataset

2. **Search Space Definition**
   Once the confidence intervals are estimated from the dataset, it is possible to generate as many naïve Bayes classifiers as needed. The parameters of these naïve Bayes classifiers must only be taken inside theirs corresponding confidence intervals.
   In this way, each naïve Bayes classifier is going to be represented with the next tupla of dimension $r_0(1 + \sum_{i=1}^{n} r_i)$

$$(p^*_1, \ldots, p^*_{r_0}, p^{*1}_{1,1}, \ldots, p^{*r_0}_{1,1}, \ldots, p^{*r_0}_{1,r_1}, \ldots, p^{*r_0}_{n,r_n}) \tag{5}$$

   where each component in the tupla p^* represents the selected value inside its corresponding confidence interval.

Thus, the search space for the heuristic optimization algorithm is composed of all the valid tuplas. A tupla is valid when it represents a valid naïve Bayes classifier. Formally,

$$\sum_{i=1}^{r_0} p_i^* = 1; \forall k \forall i \sum_{r=1}^{r_k} p_{k,r}^{*i} = 1 \tag{6}$$

Finally, each generated individual must be evaluated with a fitness function. This fitness function is based on the percentage of successful predictions on each dataset, which means that we are carrying out one wrapper approach.

3. **Heuristic Search for the Best Individual**
   Once the individuals and the search space are defined, one heuristic optimization algorithm is ran in order to find the best individual.

4.2 Pazzani and Pazzani-EDA

Pazzani [23] tries to improve the naïve Bayes classifier by searching for dependencies among attributes. He proposes two algorithms for detecting dependencies among attributes: *Forward Sequential Selection and Joining* (FSSJ) and *Backward Sequential Elimination and Joining*.

The FSSJ algorithm initializes the set of attributes to be used by the naïve Bayes classifier to the empty set. A naïve Bayes with no attributes simply classifies all examples to the most frequent class that occurs in the training data. Next, two operators are used to generate new classifiers:

1. To add a given attribute (not used by the current classifier) as a new attribute conditionally independent of all the others attributes (used by the classifier).
2. To join a given attribute (not used by the current classifier) with another possible attribute (currently used by the classifier).

At each step in the classifier, every addition and every joining of an unused attribute with a used attribute is considered and evaluated using leave-one-out on the training data. If no change makes an improvement, the current classifier is returned. Otherwise, the change that makes the most improvement is retained and the process of modifying the classifier is repeated.

The BSEJ algorithm initially creates a naïve Bayes classifier treating all attributes as conditionally independent. It uses two operators for considering new hypotheses:

1. To replace a given pair of attributes (used by the classifier) with a new attribute that joins the pair of attributes.
2. To delete one attribute (used by the classifier).

Like the FSSJ algorithm, the BSEJ algorithm considers all possible single-step operators, evaluates these using leave-one-out cross validation in the training data and permanently makes the change with the greatest improvement. If no changes results in an improvement, the current classifier is returned.

In this paper, we propose to make a heuristic search of the Pazzani structure with the target of maximize the percentage of successful predictions. We will do this heuristic search with EDAs.

The figure 2 contains two Pazzani structures and their corresponding individuals.

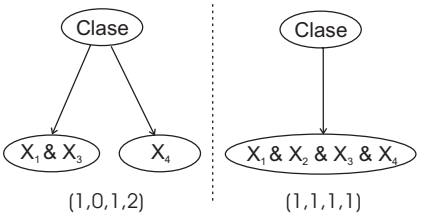

Fig. 2. Two Pazzani structures and their corresponding individuals

Thus, for a dataset with n attributes, individuals will have n genes, each one with a integer value between 0 an n. The value 0 represents that the corresponding attribute is not part of the Pazzani structure. A value between 1 an n means that the corresponding attribute belongs to that group in the Pazzani structure.

4.3 APNBC and APNBC-EDA

Adjusted probability naïve Bayes induction (APNBC) [26] is just a simple extension to the naïve Bayes classifier. A numeric weight is inferred for each class. During discriminative classification, the naïve Bayes probability of a class is multiplied by its weight to obtain an adjusted value. The lineal adjust proposed by the authors can be expressed as:

$$P(C = c_i|X_1 = x_1, \ldots, X_n = x_n) \propto w_i P(C = c_i) \prod_{k=1}^{n} P(X_k = x_k|C = c_i) \quad (7)$$

In a two class case, it is only necessary to find an adjustment value for one of the classes. This is because for any combination of adjustments a_1 and a_2 for the classes c_1 and c_2, the same effect will be obtained by setting the adjustment for c_1 to a_1/a_2 and the adjustment for c_2 to 1. In the multiple class case, the search for suitable adjustments is similar, setting one value to 1 and making a search for the rest of the other values. In this context, a simple hill-climbing search is employed. All adjustment values are initialized to 1, and a single adjustment that maximizes resubstitution accuracy is found. It is important to emphasize that the APNBC approach can achieve worsen results than naïve Bayes classifiers.

In the case of the use of CEDAs (*continuous EDAs*) for the search of the adjustment values, it is necessary to define the individuals format. In CEDAs, for each gen, we must define a maximum (real) value and a minimum (real) value. The search will be made between these values. In APNBC, for a dataset of n classes, we will use an individual with $n-1$ genes –the last class will be adjusted to 1– and each individual gen will have a

Table 2. Description of the data sets used in the experiments

	Attributes				Instances	
Name	**Total**	**Continuous**	**Nominal**	**Classes**	**Learning**	**Validation**
breast	10	10	-	2	699	-
chess	36	-	36	2	3196	-
cleve	13	6	7	2	303	-
corral	6	-	6	2	128	-
crx	15	6	9	2	692	-
flare	10	2	8	2	1066	-
german	20	7	13	2	1000	-
glass	9	9	-	7	214	-
glass2	9	9	-	2	163	-
hepatitis	19	6	13	2	155	-
iris	4	4	-	3	150	-
lymphography	18	3	15	4	148	-
m-of-n-3-7-10	10	-	10	2	300	1024
pima	8	8	-	2	768	-
satimage	36	36	-	6	6435	-
segment	19	19	-	7	2310	-
shuttle-small	9	9	-	7	5800	-
soybean-large	35	-	35	19	683	-
vehicle	18	18	-	4	846	-
vote	16	-	16	2	435	-
waveform-21	21	21	-	3	300	4700

value between 0 and 5. Besides, each individual will be validated with the *leave-one-out* method.

5 Experimentation Results

5.1 Datasets

Results are compared for 21 classical datasets –see table 2–, also used by other authors [9]. All the datasets belong to the UCI repository [1], with the exception of *m-of-n-3-7-10* and *corral*. These two artificial datasets, with irrelevant and correlated attributes, were designed to evaluate methods for feature subset selection [13].

5.2 Experimental Methodology

To estimate the prediction accuracy for each classifier our own implementation of a naïve Bayes classifier has been implemented. This implementation uses the Laplace correction for the point estimation of the conditional probabilities [11,13] and deals with missing values as recommended by [3].

However, our new algorithm does not handle continuous attributes. Thus, a discretization step with the method recommended in [4] has been performed using MLC++

Table 3. Experiment results for Iterative Bayes and IENB

Dataset	*NB*	*IENB*	*Improvement*	*Iterative Bayes Improvement*
breast	97.14	97.71 ± 0.00 †	0.57	-0.16
chess	87.92	93.31 ± 0.10 †	5.39	
cleve	83.82	86.29 ± 0.17 †	2.47	0.1
corral	84.37	93.70 ± 0.00 †	9.33	
crx	86.23	89.64 ± 0.10 †	3.41	
flare	80.86	82.69 ± 0.13 †	1.83	
german	75.40	81.57 ± 0.10 †	6.17	-0.05
glass	74.77	83.00 ± 0.20 †	8.23	-1.19
glass2	82.21	88.27 ± 0.00 †	6.06	
hepatitis	85.16	92.60 ± 0.34 †	7.44	1.41
iris	94.67	95.97 ± 0.00 †	1.30	1.4 †
lymphography	85.14	94.56 ± 0.00 †	9.42	
monf-3-7-10	86.33	95.31 ± 0.00 †	8.98	
pima	77.73	79.84 ± 0.09 †	2.11	
satimage	82.46	83.88 ± 0.32 †	1.42	3.59 †
segment	91.95	96.38 ± 0.08 †	4.43	1.5 †
shuttle-small	99.36	99.90 ± 0.00 †	0.54	
soybean-large	92.83	95.67 ± 0.10 †	2.84	
vehicle	61.47	71.16 ± 0.25 †	9.69	4.39 †
vote	90.11	95.07 ± 0.19 †	4.96	1.45 †
waveform-21	78.85	79.81 ± 0.10 †	0.96	

tools [15]. This discretization method is described by Ting in [25] that is a global variant of the method of Fayyad and Irani [7].

Nineteen of the datasets have no division between training and testing sets. On these datasets the results are obtained by a *leave-one-out* method inside of the heuristic optimization loop.

On the other hand, two out of these twenty one datasets include separated training and testing sets. For these cases, the heuristic optimization algorithm uses only the training set to tune the classifier. A *leave-one-out* validation is performed internally inside of the optimization loop, in order to find the best classifier. Once the best candidate is selected, it is validated using the testing set.

All the heuristic experiments were ran in a Athlon 1700+ with 256MB of RAM memory. The parameters used to run EDAs or CEDAs were: population size 500 individuals, selected individuals for learning 500, new individuals on each generation 1000, learning type $UMDA$ (*Univariate Marginal Distribution Algorithm*) or $UMDA_C$ (*UMDA continuous*) [21], depending on the algorithm, and elitism. Each experiment has been ran 10 times.

5.3 Results

This section present the experimental results of the evaluation of the different semi naïve Bayes approaches presented above.

Table 4. Experimental results for APNBC-EDA and APNBC

Dataset	*NB*	*APNBC-EDA*	*Eval.*	*Improvement*	*APNBC Improvement*
breast	97.14	97.57 ± 0.00 †	2600	0.43	-0.20
chess	87.92	88.14 ± 0.00 †	11300	0.22	
cleve	83.82	84.16 ± 0.00 †	7500	0.34	-1.00
corral	84.37	90.63 ± 0.00 †	1500	6.26	
crx	86.23	86.96 ± 0.00 †	10400	0.73	-0.20
flare	80.86	83.77 ± 0.00 †	4000	2.91	
german	75.40	75.90 ± 0.00 †	30600	0.50	
glass	74.77	79.44 ± 0.00 †	7500	4.67	
glass2	82.21	88.34 ± 0.00 †	2200	6.13	
hepatitis	85.16	88.39 ± 0.00 †	3300	3.23	
iris	94.67	96.00 ± 0.00 †	3200	1.33	0.60
lymphography	85.14	86.49 ± 0.00 †	3500	1.35	1.40
monf-3-7-10	86.33	98.24 ± 0.04 †	5500	11.91	
pima	77.73	79.43 ± 0.00 †	5500	1.70	-0.50
satimage	82.46	85.19 ± 0.00 †	25500	2.73	
segment	91.95	92.68 ± 0.00 †	11400	0.73	
shuttle-small	99.36	99.47 ± 0.00 †	6500	0.11	
soybean-large	92.83	93.70 ± 0.00 †	13700	0.87	-1.50
vehicle	61.47	65.25 ± 0.17 †	6600	3.78	
vote	90.11	90.57 ± 0.00 †	5400	0.46	
waveform-21	78.85	80.21 ± 0.05 †	3100	1.36	5.00

Iterative Bayes vs. IENB. Results for IENB and *Iterative Bayes* approaches are shown in table 3. In IENB the results are calculated for the 0.95 percentile ($z_\alpha = 1.96$). Columns in the table (as they appear from left to right) are: first, the value obtained by the simple naïve Bayes algorithm, second, the mean value ± standard deviation from IENB, third, the improvement respect to naïve Bayes (*IENB-Naïve Bayes*) and finally the improvement respect to naïve Bayes obtained by *Iterative Bayes* (*Iterative Bayes-Naïve Bayes*).

Results are really interesting. Respect to naïve Bayes, using IENB, we obtained an average improvement of 4.30%. Besides, although this is not always true, better improvements are obtained in the datasets with less number of cases, as the complexity of the problem is lower. *Iterative Bayes* is not so good, being better than IENB only in one dataset, *satimage*.

The non-parametric tests of Mann-Whitney were used to test the null hypothesis of the same distribution between these approaches and naïve Bayes. This task was done with the statistical package S.P.S.S. release 11.50. The results for the tests applied to all the algorithms are:

- Respect to naïve Bayes algorithm:
  - Naïve Bayes vs. IENB. Fitness value: $p < 0.001$ for all datasets.
  - Naïve Bayes vs. Iterative Bayes. Fitness value: $p < 0.001$ only in 5 of 10 datasets.

Table 5. Experimental results for Pazzani FSSJ and BSEJ algorithms

	FSSJ				*BSEJ*			
Dataset	*FSSJ*	*Eval.*	*(Att,Groups)*	*Impr*	*BSEJ*	*Eval.*	*(Att,Groups)*	*Impr*
breast	97.42	80	(3 , 2)	0.28	97.56	200	(10 , 9)	0.42
chess	94.33	396	(5 , 1)	6.41	97.06	14256	(35 , 26)	9.14
cleve	85.10	169	(4 , 3)	1.28	85.43	676	(12 , 10)	1.61
corral	74.80	18	(1 , 1)	-9.57	93.70	108	(6 , 4)	9.33
crx	86.79	195	(5 , 2)	0.56	88.10	1125	(14 , 11)	1.87
flare	84.04	90	(3 , 2)	3.18	84.04	500	(6 , 6)	3.18
german	77.18	500	(7 , 3)	1.78	77.68	2400	(19 , 15)	2.28
glass	76.53	153	(5 , 3)	1.76	76.53	162	(9 , 8)	1.76
glass2	87.04	99	(4 , 2)	4.83	87.04	162	(9 , 8)	4.83
hepatitis	91.56	456	(7 , 3)	6.40	88.96	1083	(18 , 17)	3.80
iris	95.97	12	(1 , 1)	1.30	95.97	32	(3 , 3)	1.30
lymphography	84.35	252	(4 , 3)	-0.79	90.48	972	(18 , 16)	5.34
monf-3-7-10	77.32	30	(1 , 1)	-9.01	88.27	200	(9 , 9)	1.94
pima	79.79	72	(4 , 1)	2.06	79.79	192	(6 , 6)	2.06
satimage	86.17	828	(7 , 3)	3.71	86.84	12914	(36 , 22)	4.38
segment	96.32	589	(7 , 4)	4.37	95.89	3971	(18 , 10)	3.94
shuttle-small	99.74	72	(3 , 2)	0.38	99.90	324	(9 , 6)	0.54
soybean-large	93.40	3010	(13 , 8)	0.57	94.57	7350	(34 , 30)	1.74
vehicle	70.77	162	(4 , 1)	9.30	72.66	3240	(17 , 9)	11.19
vote	96.77	128	(3 , 2)	6.66	94.47	1792	(15 , 10)	4.36
waveform-21	69.65	189	(4 , 1)	-9.20	79.70	89	(21 , 19)	0.85

These results show that the differences between naïve Bayes and Interval Estimation naïve Bayes are significant in all the datasets, meaning that the behavior of selecting naïve Bayes or IENB is very different. On the other hand, results for Iterative Bayes are statistically significant only in 5 of 10 datasets.

The symbols † in the table show that the result is statistically significant respect to naïve Bayes.

APNBC vs. APNBC-EDA. The experimental results of APNBC –extracted from the original paper [26]– and the results of APNBC-EDA are shown on table 4. The column contents are (from left to right): first, the percentage of correct classification using naïve Bayes; second mean and standard deviation of classification accuracy using APNBC-EDA; third, average number of evaluations to reach the solution; fourth, improvement compared to naïve Bayes classifier; fifth and last columns represent the classifier accuracy and solution improvement using either APNBC-EDA or standard APNBC (the last one has been taken from the original source mentioned above).

There are only eight out of the twenty-one cases with results for APNBC algorithm. In six of these problems APNBC-EDA clearly outperforms APNBC, in one (*lymphography*) results are almost the same. In *waveform-21* APNBC improves 5,00% while APNBC-EDA only reaches 1,36%.

It is remarkable to emphasize that in the eight datasets APNBC-EDA gets a mean improvement of 1.01%, and APNBC only achieves 0.45%. Considering all the twenty-

Table 6. Experimental results for Pazzani-EDA

Dataset	*Pazzani-EDA*	*Eval.*	*(Att,Groups)*	*Impr*
breast	97.82 ± 0.06 †	20650	(10 , 7)	0.68
chess	97.93 ± 0.24 †	44900	(26 , 15)	10.01
cleve	86.98 ± 1.18 †	49600	(9 , 5)	3.16
corral	100.00 ± 0.00 †	7400	(5 , 3)	15.63
crx	89.23 ± 0.07 †	80200	(12 , 7)	3.00
flare	84.04 ± 0.00	9700	(5 , 4)	3.18
german	78.41 ± 0.06 †	90900	(17 , 11)	3.01
glass	77.93 ± 0.00 †	40200	(7 , 5)	3.16
glass2	87.04 ± 0.00	7700	(6 , 5)	4.83
hepatitis	93.64 ± 0.29 †	55800	(11 , 7)	8.48
iris	95.97 ± 0.00	3600	(1 , 1)	1.30
lymphography	93.88 ± 0.00 †	48900	(15 , 7)	8.74
monf-3-7-10	100.00 ± 0.00 †	39600	(8 , 2)	13.67
pima	79.82 ± 0.28	7300	(7 , 4)	2.09
satimage	86.94 ± 0.12	105300	(12 , 7)	4.48
segment	96.15 ± 0.06 †	24300	(14 , 7)	4.20
shuttle-small	99.90 ± 0.00	30200	(7 , 5)	0.54
soybean-large	95.67 ± 0.17 †	32	(20 , 15)	2.84
vehicle	76.50 ± 0.32 †	61300	(12 , 7)	15.03
vote	96.77 ± 0.13 †	30200	(7 , 4)	6.66
waveform-21	79.72 ± 0.07 †	89200	(17 , 9)	0.87

one problems APNBC-EDA performs an average of 2.46% better than naïve Bayes. It is very significative that APNBC only gets better accuracy than naïve Bayes in three out of the eight cases experimented by [26], APNBC-EDA improves in all of the datasets. These results show a clear difference for APNBC-EDA against standard APNBC.

As well as in the previous experimentation a non-parametric Mann-Whitney test has been performed to evaluate the significance of the improvements. The values obtained are:

- Respect to naïve Bayes algorithm:
  - Naïve Bayes vs. APNBC-EDA. Fitness value: $p < 0.001$ for all datasets.

These values show a significative difference between naïve Bayes and APNBC-EDA results for all the datasets.

Pazzani (FSSJ/BSEJ) vs. Pazzani-EDA. The experimental results achieved by the greedy algorithms FSSJ and BSEJ are presented on table 5. The four columns on the left-hand side show the results of FSSJ and the four on the right-hand side show the results of BSEJ. Results of Pazzani-EDA algorithm are shown on table 6. Each of the algorithms have the following values: percentage of instances classified correctly, number of necessary evaluations, number of selected attributes and groups of attributes; and the improvement comparing to naïve Bayes.

Considering all the datasets, FSSJ algorithm shows an average improvement of 1.25% (comparing to naïve Bayes), BSEJ 3.61% and Pazzani-EDA improves 5.51%.

Labelled with the symbol † on table 6 in 15 of the 21 datasets the difference between Pazzani-BSEJ and Pazzani-EDA is statistically significative.

As a summary, another benefit in using heuristic search with EDAs is that Pazzani structures are simpler: less number of attributes (recall, Pazzani performs an attribute selection strategy) and few groups of attributes. BSEJ uses 15 attributes and 12 groups (in average), Pazzani-EDA uses only 12 attributes and 7 groups.

6 Conclusion and Further Work

The experimental results presented on this paper show a clear advantage in the use of heuristical search methods, rather than greedy approaches, as an optimization tool for semi naïve Bayes classifiers.

The better improvements have been achieve using Pazzani-EDA (5.50%) and IENB (4.65%). Both strategies can be combined in the following way: (a) looking for the best Pazzani structure and (b) tuning the probabilities of each group using the IENB approach.

The only drawback of this approach is the increment of the number of tentative naïve Bayes classifier required to find the optimal parameters. For large problems with significative evaluation times each of the individuals could take several seconds to be computed. As a solution for this problem a parallel implementation of these algorithms is an open issue we are working on.

References

1. C. L. Blake and C. J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/, 1998.
2. J. S. De Bonet, C. L. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Vol. 9, 1997.
3. P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning*, pages 105–112, 1996.
4. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202, 1995.
5. R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
6. R. Etxeberria and P. Larrañaga. Global optimization with Bayesian networks. In *II Symposium on Artificial Intelligence. CIMAF99. Special Session on Distributions and Evolutionary Optimization*, pages 332–339, 1999.
7. U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Conference on Artificial Intelligence*, pages 1022–1027, 1993.
8. J.T.A.S. Ferreira, D.G.T. Denison, and D.J. Hand. Weighted naïve Bayes modelling for data mining. Technical report, Deparment of Mathematics, Imperial College, May 2001.
9. N. Friedman, D. Geiger, and D.M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.
10. J. Gama. Iterative Bayes. *Intelligent Data Analysis*, 4:475–488, 2000.
11. I.J. Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. MIT Press, 1965.

12. D.J. Hand and K. Yu. Idiot's Bayes - not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.
13. J. Kohavi, B. Becker, and D. Sommerfield. Improving simple Bayes. Technical report, Data Mining and Visualization Group, Silicon Graphics, 1997.
14. R. Kohavi. Scaling up the accuracy of naïve-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.
15. R. Kohavi, G. John, R. Long, D. Manley, and K.Pfleger. MLC++: A machine learning library in C++. *Tools with Artificial Intelligence. IEEE Computer Society Press*, pages 740–743, 1994.
16. I. Kononenko. Semi-naïve Bayesian classifier. In *Sixth European Working Session on Learning*, pages 206–219, 1991.
17. P. Langley. Induction of recursive Bayesian classifiers. In *European Conference on Machine Learning. Berlin: Springer-Verlag*, pages 153–164, 1993.
18. P. Langley and S. Sage. Induction of selective Bayesian classifiers. In Morgan Kaufmann, editor, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, Seattle, WA, 1994.
19. P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña. Optimization in continuous domains by learning and simulation of gaussian networks. In *Proceedings of the Workshop in Optimization by Building and Using Probabilistic Models. A Workshop within the 2000 Genetic and Evolutionary Computation Conference, GECCO 2000*, pages 201–204, 2000. Las Vegas, Nevada, USA.
20. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publisher, 2001.
21. H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303–346, 1998.
22. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, 1996.
23. M. Pazzani. Searching for dependencies in Bayesian classifiers. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 239–248, 1996.
24. V. Robles, P. Larrañaga, J.M. Peña, E. Menasalvas, and M.S. Pérez. Interval Estimation Naïve Bayes. In *Lecture Notes in Computer Science*, 2003.
25. K.M. Ting. Discretization of continuous-valued attributes and instance-based learning. Technical Report 491, University of Sydney, 1994.
26. G.I. Webb and M.J. Pazzani. Adjusted probability naive Bayesian induction. In *Proceedings of the 11th Australian Joint Conference on Artificial Intelligence*, pages 285–295, 1998.