Adaptive Bayes

João Gama and Gladys Castillo

[1] LIACC, FEP – University of Porto
Rua Campo Alegre, 823
4150 Porto, Portugal
jgama@liacc.up.pt
[2] Department of Mathematics
University of Aveiro
Aveiro, Portugal
gladys@mat.ua.pt

Abstract. Several researchers have studied the application of Machine Learning techniques to the task of user modeling. As most of them pointed out, this task requires learning algorithms that should work on-line, incorporate new information incrementality, and should exhibit the capacity to deal with concept-drift. In this paper we present Adaptive Bayes, an extension to the well-known naive-Bayes, one of the most common used learning algorithms for the task of user modeling. Adaptive Bayes is an incremental learning algorithm that could work on-line. We have evaluated Adaptive Bayes on both frameworks. Using a set of benchmark problems from the UCI repository [2], and using several evaluation statistics, all the adaptive systems show significant advantages in comparison against their non-adaptive versions.

Keywords: User Modeling, Machine Learning, Adaptive Bayes, Incremental Systems

1 Introduction

The task of user modeling is a challenging one for machine learning. Observing the user behavior is a source of information that machine learning systems can use to build a predictive model of user future actions. This is a challenging task because it requires incremental techniques that should work on-line. Moreover, as pointed out in [14]:

> User modeling is known to be a very dynamic modeling task - attributes that characterize a user are likely to change over time. Therefore, it is important that learning algorithms be capable of adjusting to these changes quickly.

Nowadays, with the explosion of the World Wide Web, has increased the need of tools for automatic acquisition of user profiles, retrieval of relevant information, personalized recommendation, etc. All these tasks could use learning techniques. One of the machine learning algorithms most used in these tasks is naive Bayes

F.J. Garijo, J.C. Riquelme, and M. Toro (Eds.): IBERAMIA 2002, LNAI 2527, pp. 765–774, 2002.

[13,11,10]. Naive Bayes has been studied both on pattern recognition literature [5] and in machine learning [9]. Suppose that $P(Cl_i|\boldsymbol{x})$ denotes the probability that example $\boldsymbol{x}$ belongs to class i. The zero-one loss is minimized if, and only if, x is assigned to the class Cl_k for which $P(Cl_k|\boldsymbol{x})$ is maximum [5]. Formally, the class attached to example x is given by the expression:

$$argmax_i P(Cl_i|\boldsymbol{x}) \tag{1}$$

Any function that computes the conditional probabilities $P(Cli|x)$ is referred to as discriminant function. Given an example $\boldsymbol{x}$, the Bayes theorem provides a method to compute $P(Cl_i|\boldsymbol{x})$: $P(Cl_i|\boldsymbol{x}) = P(Cl_i)P(\boldsymbol{x}|Cl_i)/P(\boldsymbol{x})$

$P(\boldsymbol{x})$ can be ignored, since it is the same for all the classes, and does not affect the relative values of their probabilities. Although this rule is optimal, its applicability is reduced due to the large number of examples required to compute $P(\boldsymbol{x}|Cli)$. To overcome this problem several assumptions are usually made. Depending on the assumptions made we get different discriminant functions leading to different classifiers. In this work we study one type of discriminant function that leads to the naive Bayes classifier.

1.1 The Naive Bayes Classifier

Assuming that the attributes are independent given the class, $P(\boldsymbol{x}|Cl_i)$ can be decomposed into the product $P(x_1|Cl_i) * ... * P(x_a|Cl_i)$. Then, the probability that an example belongs to class i is given by:

$$P(C_i|\boldsymbol{x}) \propto P(Cl_i) \prod_j P(x_j|Cl_i) \tag{2}$$

which can be written as:

$$P(Cl_i|\boldsymbol{x}) \propto log(P(Cl_i)) + \sum_j log(P(x_j|Cl_i)) \tag{3}$$

The classifier obtained by using the discriminant function 2 and the decision rule 1 is known as the naive Bayes Classifier. The term naive comes from the assumption that the attributes are independent given the class.

1.2 Implementation Details

All the required probabilities are computed from the training data. To compute the prior probability of observing class i, $P(Cl_i)$, a counter, for each class is required. To compute the conditional probability of observing a particular attribute-value given that the example belongs to class i, $P(x_j|Cl_i)$, we need to distinguish between nominal attributes, and continuous ones. In the case of nominal attributes, the set of possible values is a enumerable set. To compute the conditional probability we only need to maintain a counter for each attribute-value and for each class. In the case of continuous attributes, the number of possible values is infinite. There are two possibilities. We can assume a particular distribution for the values of the attribute and usually the normal distribution is assumed. As alternative we can discretize the attribute in a pre-processing phase. The former has been proved to yield worse results than the latter [3]. Several methods for discretization appear in the literature. A good discussion about discretization is presented in [4]. In [3] the number of intervals is fixed to $k = min(10; nr.\ of\ different\ values)$ equal width intervals. Once the attribute has been discretized, a counter for each class and for each interval is used to compute the conditional probability.

All the probabilities required by equation 3 can be computed from the training set in one step. The process of building the probabilistic description of the dataset is very fast. Another interesting aspect of the algorithm is that it is easy to implement in an incremental fashion because only counters are used.

Domingos and Pazzani [3] show that this procedure has a surprisingly good performance in a wide variety of domains, including many where there are clear dependencies between attributes. They argue that the naive Bayes classifier can approximate optimality when the independence assumption is violated as long as the ranks of the conditional probabilities of classes given an example are correct. Some authors[7,8] suggest that this classifier is robust to noise and irrelevant attributes. They also note that the learned theories are easy to understand by domain experts, most due to the fact that the naive Bayes summarizes the variability of the dataset in a single probabilistic description, and assumes that these are sufficient to distinguish between classes.

2 Iterative Bayes

In a previous article [6] we have presented an extension to naïve Bayes. The main idea behind Iterative Bayes is to improve the probability associated with predictions. The naive Bayes classifier builds for each attribute a two-contingency table that reflects the distribution on the training set of the attribute-values over the classes. Iterative Bayes iterates over the training set trying to improve the probability associated with predictions on the training examples.

2.1 An Illustrative Example

Consider the Balance-scale dataset. This is an artificial problem available at the UCI repository [2]. This data set was generated to model psychological experimental results. This is a three-class problem, with four continuous attributes. The attributes are the left weight, the left distance, the right weight, and the right distance. Each example is classified as having the balance scale tip to the right, tip to the left, or to be balanced. The correct way to find the class is the greater of $left\_distance \times left\_weight$ and $right\_distance \times right\_weight$. If they are equal, it is balanced. There is no noise in the dataset.

Because the attributes are continuous the discretization procedure of naive Bayes applies. In this case each attribute is mapped to 5 intervals. In an experiment using 565 examples in the training set, we obtain the contingency table for the attribute *left_W* that is shown in Table 1.

Table 1. A naive Bayes contingency table

Attribute: left_W (Discretized)					
Class	I1	I2	I3	I4	I5
Left	14.0	42.0	61.0	71.0	72.0
Balanced	10.0	8.0	8.0	10.0	9.0
Right	86.0	66.0	49.0	34.0	25.0

After building the contingency tables from the training examples, suppose that we want to classify the following example:

```
left_W:1, left_D: 5, right_W: 4, right_D: 2, Class: Right
```

The output of the *naive Bayes* classifier will be something like:

```
Observed Right Classified Right  [ 0.277796 0.135227 0.586978 ]
```

It says that a test example that it is observed to belong to class *Right* is classified correctly. The following numbers are the probabilities that the example belongs to each one of the classes. Because the probability $p(Right|\boldsymbol{x})$ is greater, the example is classified as class *Right*. Although the classification is correct, the *confidence* on this prediction is low (59%). Moreover, taking into account that the example belongs to the training set, the answer, although correct, does not seems to fully exploit the information in the training set.

The method that we propose begins with the contingency tables built by the standard naive Bayes scheme. This is followed by an iterative procedure that updates the contingency tables. The algorithm iteratively cycle through all the training examples. For each example, the corresponding entries in the contingency tables are updated in order to increase the confidence on the correct class. Consider again the previous training example. The value of the attribute `left_W` is 1. This means that the values in column $I1$ in table 1 are used to compute the probabilities of equation 2. The desirable update will increase the probability $P(Right|\boldsymbol{x})$ and consequently decreasing both $P(Left|\boldsymbol{x})$ and $P(Balanced|\boldsymbol{x})$. This could be done by increasing the contents of the cell (I1;Right) and decreasing the other entries in the column I1. The same occurs for all the attribute-values of an example. This is the intuition behind the update schema that we follow. Also the amount of correction should be proportional to the difference $1 - P(C_{predict}|\boldsymbol{x})$. The contingency table for the attribute *left_W* after the iterative procedure is given in figure 2[1]. Now, the same previous example, classified using the contingency tables after the iteration procedure gives:

```
Observed Right Classified Right  [ 0.210816 0.000175 0.789009 ]
```

[1] The update rules tries to maintain constant the number of examples, that is the total sum of entries in each contingency table. Nevertheless we avoid zero or negative entries. In this case the total sum of entries could exceed the number of examples.

Table 2. A naive Bayes contingency table after the iteration procedure

Attribute: left_W (Discretized)					
Class	I1	I2	I3	I4	I5
Left	7.06	42.51	75.98	92.26	96.70
Balanced	1.06	1.0	1.0	1.0	1.08
Right	105.64	92.29	62.63	37.01	20.89

The classification is the same, but the confidence level of the predict class increases while the confidence level on the other classes decreases. This is the desirable behavior.

The iterative procedure uses a hill-climbing algorithm. At each iteration, all the examples in the training set are classified using the current contingency tables. The evaluation of the actual set of contingency tables is done using equation 4:

$$\frac{1}{n}\sum_{i=1}^{n}(1.0 - max_j p(C_j|\boldsymbol{x_i})) \tag{4}$$

where n represents the number of examples and j the number of classes. The iterative procedure proceeds while the evaluation function decreases. To escape from local minimum we allow some more iterations till the maximum of a user-defined look-ahead parameter.

The pseudo-code for the adaptation process is shown in Figure 1. To update the contingency tables, we use the following heuristics:

1. If an example is correctly classified then the increment is positive, otherwise it is negative. To compute the value of the increment we use the following heuristic: $(1.0 - p(Predict|\boldsymbol{x}))$ if the example is correctly classified and $-P(Predict|\boldsymbol{x})$ if the example is misclassified.
2. For all attribute-values observed in the given example, the increment is added to all the entries for the predict class and $increment/\#Classes$ is subtracted to the entries of all the other classes. That is, the increment is a function of the confidence on predicting class $Predict$ and of the number of classes.

The contingency tables are updated each time a training example is seen. This implies that the order of the training examples could influence the final results. This update schema guarantees that after one example is seen, the probability of the prediction in the correct class will increase. Nevertheless, there is no guaranty of improvement for a set of examples.

The starting point for Iterative Bayes is the set of contingency tables built by naïve Bayes. In these work we study Adaptive Bayes, an algorithm that use the update schema of Iterative Bayes in an incremental mode.

```
Function AdaptModel (Model, Example, Observed, Predicted)
//Compute the increment
If(Predicted<>Observed) Then  delta = (1-P(Predicted|Example))
Else                          delta = - (P(Predicted|Example))

// the increment is used to update the contingency tables
For each Attribute
  For each Class
    If (Class == Predicted) Then
           Model(Attribute,Class,AttributeValue) += delta
      Else
           Model(Attribute,Class,AttributeValue) -= delta/#Classes
    Endif

  Next Class
Next Attribute
return: Model
End
```

Fig. 1. Pseudo-code for the Adaptation Model function.

3 Adaptive Bayes

Given a decision model and new classified data not used to built the model what should we do? Most of the time a new decision model is built from the scratch. Few systems are able to adapt the decision model given new data. The naïve Bayes classifier is naturally incremental. Nevertheless most interesting problems are not stationary, that is the concepts to learn could change over time. In these scenarios forget old data to incorporate concept drift is a desirable property. An important characteristic of Iterative Bayes is the ability to adapt a given decision model to new data. This property can be explored in the context of concept-drift. The update schema of Iterative Bayes could be used to an incremental adaptive Bayes able to deal with concept drift.

We consider two adaptive versions of naïve Bayes: incremental adaptive Bayes and on-line adaptive Bayes. The former built a model from the training set updating the model (the set of contingency tables) once after seeing each example (there is no iterative cycling over the training set). The latter works on an on-line framework: for each example the actual model makes a prediction. Only after the prediction the true decision (the class of the example) is known.

The base algorithm for the incremental adaptive Bayes is presented in figure 2. The base algorithm for the on-line adaptive Bayes is presented on figure 3[2].

[2] In the initialization step the contingency tables are initialized randomly with the constrain that for all attributes the sum of the entries for each class is constant.

```
Function IncrementalBayes(Training Set, Adaptive)
inputs: The training set, Adaptive Mode

Initialization: Model = initialise all counters to zero
For each Example in the Training Set
    IncrementCounters(Model, Example, Observed)
    If Adaptive=TRUE Then AdaptModel(Model, Example, Observed, Predicted)
   Next Example
Return Model
End
```

Fig. 2. Pseudo-code for the Incremental Adaptive Bayes.

3.1 Discussion

The adaptive step of our algorithm has clear similarities with the gradient descent method used to train a perceptron. For example, both use an additive update schema. The perceptron learning procedure uses the gradient vector of partial derivatives of the parameters. Adaptive Bayes don't use derivatives, but uses a *likelihood* function that increases at each step. Under this perspective, the method could be related with the generalized Expectation-Maximization approach[12] that only requires an improvement of the likelihood function. As pointed out in [1] *"the gradient-based approach is closely related to a GEM algorithm, but this relation remains unclear."*.

4 Experimental Evaluation

We have evaluated all variants of Adaptive Bayes in a set of benchmark problems from the UCI repository [2]. The design of experiments for the incremental

```
Initialisation: Model = Randomly  initialise all counters

Function OnLineBayes(Example, Model, Adaptive)
  Predicted <- PredictClass(Model, Example)
  Observed <- Class of Example
  IncrementCounters(Model, Example, Observed)
  If Adaptive = TRUE Then AdaptModel(Model, Example, Observed, Predicted)
  Return Model
End
```

Fig. 3. Pseudo-code for the On-Line Adaptive Bayes.

versions of the algorithms is the standard 10-fold cross validation. All the algorithms incrementally built a model from the training set and the model is used to classify the test set. The evaluation statistics is the average of the 10 error rates. While the incremental naive Bayes should generate exactly the same model as its batch version, the adaptive Bayes could generate a different model. For the on-line versions, the experimental set-up was designed as follows. All the available data is presented to the algorithm in sequence. Each example is classified with the actual model. After the prediction the algorithm modifies its decision model. The evaluation statistic is the percentage of misclassified examples. This process is repeated ten times using different permutations of the dataset.

For each dataset the algorithm has access to some information about the problem domain. This information is similar to the information existing in the file *.names* used in C4.5. For each attribute it is known the name, the type, and the set of possible values for each nominal variable. Moreover, for each continuous variable the algorithm also knows the *range* of possible values.

The results are presented on table 3. The best accuracy achieved on each dataset is shown in bold. A summary of evaluation statistics is presented on table 4. The first and second lines present the average and geometric mean of the error across all datasets. The third line shows the average rank of incremental and on-line models, computed for each dataset by assigning rank 1 to the best algorithm and 2 to the second best. The fourth line shows the average ratio of the error rate. This is computed for each dataset as the ratio between the error of the adaptive model and the non-adaptive model. The sixth line shows the number of significant differences using the *Wilcoxon Matched-Pairs signed-rank test* with p-value less than 0.99. The Wilcoxon Test is also used to compare the error rate of pairs of algorithms across datasets[3]. The last line shows the *p values* associated with this test for the results on all datasets.

All evaluation statistics shows the advantage of using the proposed adaptation process. The adaptive process seems to be more advantageous in the on-line framework. The reader should take into account that we cannot compare the results between the incremental and on-line versions: the performance statistics are very different.

We should note that the computational complexity of all the algorithms is the same: $O(n)$ where n represents the number of examples[4].

5 Conclusions and Future Work

Several researchers have studied the application of Machine Learning techniques to the task of user modeling. Learning algorithms for user modeling should work on-line, incorporate new information in an incremental way, and with the capacity to deal with concept-drift.

[3] Each pair of data points consists of the estimate error on one dataset and for the two learning algorithms being compared.

[4] Assuming that the number of examples is much greater than the number of attributes.

Table 3. Comparison between adaptive versus non-adaptive naive Bayes on incremental and on-line frameworks.

	Incremental		**On-Line**	
Dataset	Naive Bayes	Adaptive	Naive Bayes	Adaptive
Adult	17.671 ±0.6 +	**14.748 ±0.5**	17.818 ±0.1 +	**14.870 ±0.1**
Australian	**13.750 ±0.4**	13.839 ±0.5	14.899 ±0.5	**14.841 ±0.6**
Balance	**8.539 ±0.3**	8.747 ±0.4	14.560 ±0.8	**14.160 ±0.7**
Banding	22.822 ±1.1	**21.542 ±1.1**	23.740 ±1.6	**23.529 ±1.1**
Breast	**2.659 ±0.1**	2.774 ±0.1	**3.119 ±0.3**	3.176 ±0.2
Cleveland	**18.035 ±0.6**	18.134 ±1.0	**18.581 ±1.1**	18.977 ±1.1
Credit	14.060 ±0.3	**13.869 ±0.3**	15.652 ±0.8	**15.449 ±0.4**
Diabetes	23.763 ±0.6	**23.203 ±0.7**	25.547 ±0.7	**24.674 ±0.8**
German	**24.400 ±0.5** −	25.510 ±0.6	**26.710 ±0.5**	27.620 ±0.4
Glass	36.900 ±1.5	**35.710 ±1.8**	42.570 ±2.1	**41.916 ±1.4**
Heart	17.407 ±0.7 +	**16.370 ±0.8**	18.556 ±1.7	**18.370 ±2.2**
Hepatitis	19.308 ±1.1 +	**16.348 ±0.9**	21.097 ±1.9 +	**19.226 ±1.7**
Ionosphere	11.158 ±0.6	**10.437 ±0.5**	13.903 ±1.2 +	**12.108 ±0.9**
Iris	**6.133 ±0.5**	6.133 ±0.5	10.133 ±1.6	**9.867 ±1.3**
Letter	29.989 ±1.2 +	**28.262 ±1.3**	33.075 ±0.2 +	**31.907 ±0.3**
Mushroom	4.766 ±0.0 +	**1.307 ±0.0**	5.695 ±0.1 +	**2.008 ±0.1**
Satimage	18.943 ±0.1 +	**15.942 ±0.2**	19.133 ±0.2 +	**16.306 ±0.2**
Segment	10.948 ±0.2 +	**8.965 ±0.3**	13.312 ±0.7 +	**11.416 ±0.5**
Shuttle	3.052 ±0.6	**2.704 ±0.3**	3.173 ±0.2 +	**2.897 ±0.1**
Sonar	24.505 ±1.8 +	**22.759 ±2.3**	27.500 ±2.9	**26.635 ±2.3**
Vehicle	40.191 ±0.8 +	**37.866 ±1.2**	42.411 ±1.1 +	**39.870 ±0.7**
Votes	9.820 ±0.2 +	**8.760 ±0.4**	10.207 ±0.5 +	**9.149 ±0.7**
Waveform	19.076 ±0.2 +	**15.615 ±0.3**	19.961 ±0.6 +	**16.904 ±0.4**
Wine	4.258 ±0.6 +	**3.176 ±1.0**	8.258 ±1.9	**7.303 ±2.0**

Table 4. Summary of Results.

	Incremental		**On-Line**	
Dataset	Naive Bayes	Adaptive	Naive Bayes	Adaptive
Arithmetic Mean	16.76	15.53	18.73	17.63
Geometric Mean	13.40	11.79	15.55	14.03
Average Rank	1.75	1.25	1.88	1.12
Error Ratio	1	0.91	1	0.92
Nr. Wins	6	18	3	21
Nr. Significant Wins	1	12	0	11
Wilcoxon Test	–	0.0002	–	0.0006

In this paper we have studied the behavior of adaptive Bayes, a new incremental algorithm based on naive Bayes that could work on-line. Adaptive Bayes uses the same adaptation strategy used in Iterative Bayes - a batch classifier. The main idea behind Iterative Bayes is to improve the probability associated with predictions. This strategy is used on Adaptive Bayes to guide the adaptation process. In a set of benchmark datasets, the adaptation process shows clear advantages over the non-adaptive naive-Bayes both on incremental and on-line frameworks.

The next step of this work is to incorporate the on-line adaptive Bayes in a WEB based teaching system.

Acknowledgments. Gratitude is expressed to the financial support given by the FEDER, the Plurianual support attributed to LIACC, and ALES project (POSI/39770/SRI/2001).

References

1. John Binder, Daphne Koller, Stuart Russel, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–243, 1997.
2. C. Blake, E. Keogh, and C.J. Merz. UCI repository of Machine Learning databases, 1999.
3. Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–129, 1997.
4. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. Russel, editors, *Machine Learning Proc. of 12th International Conference*. Morgan Kaufmann, 1995.
5. R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. New York, Willey and Sons, 1973.
6. J. Gama. Iterative Bayes. In S. Arikawa and K. Furukawa, editors, *Discovery Science - Second International Conference*. LNAI 1721, Springer Verlag, 1999.
7. I. Kononenko. Semi-naive Bayesian classifier. In Y. Kodratoff, editor, *European Working Session on Learning -EWSL91*. LNAI 482 Springer Verlag, 1991.
8. P. Langley. Induction of recursive Bayesian classifiers. In P.Brazdil, editor, *Proc. of European Conf. on Machine Learning*. LNAI 667, Springer Verlag, 1993.
9. Tom Mitchell. *Machine Learning*. MacGraw-Hill Companies, Inc., 1997.
10. Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39:1–32, 2000.
11. Michael Pazzani and Daniel Billsus. Learning and revising user profiles: the identification of interesting web sites. *Machine Learning*, 27:313, 1997.
12. Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
13. Mia Stern, Joseph Beck, and Beverly Woolf. Naive bayes classifiers for user modelling. In *Proceedings of the User Modelling Conference*. Morgan Kaufmann, 1999.
14. G. Webb, M. Pazzani, and D. Billsus. Machine learning for user modelling. *User Modelling and User-adapted Interaction*, 11:19–29, 2001.