

Wrapping the Naive Bayes Classifier to Relax the Effect of Dependences

Jose Carlos Cortizo^{1,2}, Ignacio Giraldez², and Mari Cruz Gaya²

¹ Artificial Intelligence & Network Solutions S.L.

jccp@ainetsolutions.com; <http://www.ainetsolutions.com/jccp>

² Universidad Europea de Madrid

Villaviciosa de Odon, 28670, Madrid, Spain

{josecarlos.cortizo, ignacio.giraldez, mcruz}@uem.es

Abstract. The Naive Bayes Classifier is based on the (unrealistic) assumption of independence among the values of the attributes given the class value. Consequently, its effectiveness may decrease in the presence of interdependent attributes. In spite of this, in recent years, Naive Bayes classifier is worked for a privilege position due to several reasons [1]. We present DGW (Dependency Guided Wrapper), a wrapper that uses information about dependences to transform the data representation to improve the Naive Bayes classification. This paper presents experiments comparing the performance and execution time of 12 DGW variations against 12 previous approaches, as constructive induction of cartesian product attributes, and wrappers that perform a search for optimal subsets of attributes.

Experimental results show that DGW generates a new data representation that allows the Naive Bayes to obtain better accuracy more times than any other wrapper tested. DGW variations also obtain the best possible accuracy more often than the state of the art wrappers while often spending less time in the attribute subset search process.

Keywords: DGW, Naive Bayes, (In)Dependence Assumption, Wrapper, Feature Evaluation and Selection

1 Introduction and Motivation

There exist many approaches to the classification problem, from induction of decision trees, nearest neighbours approaches, etc. but the statistical approach seems to be the most intuitive and simple, in fact, there existed statistical approaches to classification previous than the machine learning ones [2]. From statistical classifiers, Naive Bayes, which is based on the Bayes Theorem [3] is worked for a privilege position [1] due to its simplicity, its resilience to noise, its time and space efficiency [4], [5], its understandability [6], its results both in performance and speed in the area of information retrieval and automated text categorization [7], [8] and also in other areas and because it is well known that when independence assumption is held, no other classifier can outperform Naive

Bayes in the sense of misclassification probability [9]. [10] shows the Naive Bayes classifier is competitive with other learning algorithms as decision trees and neural networks and, in certain situations, outperforms them. [11] and [12] present good experimental results for the Naive Bayes when compared versus other more modern machine learning algorithms. [13] shows that the Naive Bayes outperforms the most state-of-the-art decision-tree based algorithms for ranking.

The Naive Bayes classifier is based upon the simplifying assumption of conditionally independent attributes given the class value (see [14] and [15] for more detailed explanations). This assumption is not very realistic as in many real situations the attributes are, in some manner, dependent.

There are many attempts to relax this assumption in the literature, which can be summarized in three main trends:

1. Attempts that try to relax this assumption by modifying the classifier [16], [17].
2. Feature extraction in order to modify the input data to achieve independent (or pseudo-independent) attributes [4], [18], [15].
3. Approaches that underestimate the independence assumption [19], [5], [20], [21].

This article, that presents an approach in the line of 2, is organized as follows. In the next section general issues related to wrapper methods and its applications to the Naive Bayes classifier and the independence assumption are presented. In Section 3 the Dependence Guided Wrapper (DGW), a correlation guided wrapper that performs an attribute subset selection process for the Naive Bayes is presented. Section 4 explains the experiments made for testing the DGW algorithm and the comparisons to other attribute selection methods. Section 5 discusses the results and the last section presents some conclusions and future work.

2 Wrappers for Attribute Subset Selection

The prediction accuracy of classifiers may degrade in prediction accuracy when faced with many irrelevant and/or redundant attributes. The explanation to this phenomenon may be found in the “Curse of the Dimensionality” [22] which refers to the exponential growth of the number of instances needed to describe the data as a function of dimensionality (number of attributes). $\phi(\mathbf{X}, \mathbf{A})$ is a function that transforms a dataset \mathbf{X} to contain only the attributes included in \mathbf{A} . Given a dataset \mathbf{X} containing a set of features \mathbf{A} and given a certain learning algorithm L , Attribute Subset Selection tries to achieve a subset of the original attributes $A_F \subset \mathbf{A}$ such that when running L using $\phi(\mathbf{X}, A_F)$ as data to learn, L obtains the highest possible accuracy. There are two main trends in Attribute Subset Selection: the **filter model** and the **wrapper model**.

While the **filter model** [23] is based upon the idea of Relevance [24] and selects the attributes independently of what classifier will be used, the **wrapper**

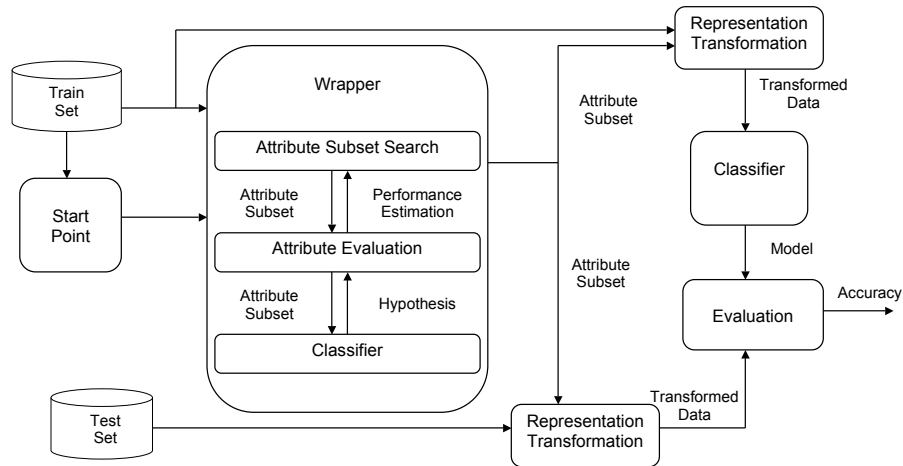


Fig. 1. General structure of a Wrapper.

model [25] conducts a search in the space of possible attributes using the performance of the classifier as the evaluation function (or at least as an important part of the evaluation function). Figure 1 shows the general structure of a wrapper where the initial training set is given to the wrapper with an initial starting point of attributes (all the attributes, none of the attributes or a random set of attributes) and then the wrapper conducts a search (according to the attribute selection search selected) where the induction algorithm is used as a black box to measure the performance of each evaluated subset. Finally, the initial data representation is transformed to comply the final attribute subset achieved.

[26] studied a wrapper performing a forward greedy search using the Naive Bayes as induction algorithm (FSS, Forward Sequential Selection). FSS tries to deal with highly correlated attributes by incorporating only some attributes in the final attributes set. [27] compared FSS and an adaption of Kittler's work [28] called BSE (Backward Sequential Elimination) very similar to FSS but performing a backwards greedy search. [18] proposed joining as an operation that creates a new compound attribute that replaces the original two dependent attributes, then explored two alternative methods related to FSS and BSE: FSSJ (Forward Sequential Selection and Joining) and BSEJ (Backward Sequential Elimination and Joining). Both, FSSJ and BSEJ, maintain a set of attributes to be used by the classifier and operates in a similar way: at each step the algorithm considers a set of possible operations (eliminate/select one attribute or joining two attributes) and studies the effect of each operation in the accuracy obtained by the classifier. The change that makes the most improvement is retained and the process is repeated until no accuracy improvement is achieved.

Experimental results show that the inductive construction of attributes helps the wrapper to achieve greater accuracies. FSS, BSE, FSSJ and BSEJ improve the Naive Bayes performance but due to the structure of a wrapper (uses the accuracy of the classifier as a metric of each possible subset of attributes) other wrappers might also improve the Naive Bayes performance.

3 The Dependency Guided Wrapper

This paper presents a wrapper attribute selection method that iteratively discards the top-N linearly correlated features (with N equal to 1, 2 and 3 in experiments) until the accuracy of the Naive Bayes classifier does not improve. Then it transforms the original data set so it complies with the new data representation. This method is called Dependency Guided Wrapper (DGW) as it tries to find a representation free of dependent attributes to satisfy the independence assumption. Figure 1 shows the general structure of a Wrapper. The DGW Wrapper is presented by describing each of the components presented in the general structure.

Let $\mathbf{A} = \{A_1, A_2, \dots, A_n\}$ be the original set of attributes and $\mathbf{Y} = \{y_1, y_2, \dots, y_m\}$ the possible class values, $\mathbf{x}_k = \{\mathbf{x}, y\}$ is a training example where $\mathbf{x} = (a_1, a_2, \dots, a_n)$ is a point that belongs to the input space \mathbf{X} and $y \in \mathbf{Y}$ is a point belonging to the output space \mathbf{Y} .

- ▷ **Starting Point:** The DGW Wrapper starting point contains all the original attributes.
- ▷ **Classifier:** The classifier used to estimate the performance of each attribute subset is the Naive Bayes Classifier
- ▷ **Attribute Selection Search:** We perform a two stage search.
 - The first stage is the **Dependency Based Wrapping**. The DGW tries to avoid all the dependences present on the original attributes. For that, DGW measures the Coefficient of Determination (R^2) of the linear dependence between each pair of attributes as a way to estimate how dependent they are. Each dependence, d_i is defined as the two related attributes and the R^2 values for those attributes, $d_i = (A_i, A_j, R^2(A_i, A_j))$. Then, it constructs a list, L_{d_i} , containing each possible dependence d_i , in a strength-of-dependency decremental order. Then, L_{d_i} is used to obtain the final attributes by deleting the most dependent attributes until no accuracy improvement is achieved.

At each step, the algorithm considers a given number of dependences (N), the first N in L_{d_i} . As each dependence is represented by two attributes A_i and A_j , at each iteration, DGW deals, at most, with $2 * N$ attributes. At each iteration, for each of the $2 * N$ attributes, DGW calculates the impact in the classifier's accuracy of removing the attribute from the attributes set. At the end of each iteration, DGW either deletes the attribute which its impact when removing produces a higher increase on the final accuracy, or stop the Attribute Selection Search when all the final accuracies imply a fall in the accuracy of the classifier.

- The second stage is the **Attribute Evaluator Ranker**. For each attribute not deleted in the previous stage, DGW calculates its value according to a certain Attribute Evaluator and then generate a list L_{AE} containing all these attributes ordered in this metric decremental order. For each attribute on the list, the impact on the final accuracy when deleting that attribute is studied. If DGW obtains a higher or the same accuracy when deleting it, DGW then removes the attribute from the list.

Once finished the whole process, DGW obtains $A_F = \{A_i, A_j, \dots A_z | A_k \in A\}$ that is a subset of the original attributes that contains the best ones.

4 Experiments

In this section some experiments to evaluate the proposed wrapper are presented. These experiments compare DGW to other methods and wrappers that perform a search among the possible subsets of attributes to improve the performance of the Naive Bayes Classifier. For this purpose, 23 datasets have been selected from the University of California at Irvine Machine Learning Repository [29] (see Table 1) trying to show a certain variety in the number of attributes (from 4 to 58) and instances (from a few more than 100 to almost 50.000).

We have selected the wrappers compared in Pazzani’s work [27] [18]: Forward Sequential Selection (FSS), Backward Sequential Elimination (BSE), Forward Sequential Selection and Joining (FSSJ), Backward Sequential Elimination and Joining (BSEJ) but also some other wrappers from the state of the art of wrappers included in Weka [30] (using their default parameters): Best First Forward (BFF), Best First Backward (BFB), Best First Bidirectional (BFBi), Genetic Search (GS), Ranker using Information Gain (IG), Ranking with Gain Ratio (GR), Ranking with Relief (RE), Ranking with Squared-Chi (C2).

DGW has been tried using 1, 2 or 3 dependences at each step and using four different attribute evaluators (Information Gain, Relief, OneR and Symmetrical Uncertainty), which means 12 possible combinations: Information Gain and 1 (1IG), 2 (2IG) or 3 (3IG) dependences studied at each step, Relief combined with 1 (1RE), 2 (2RE) or 3 (3RE) dependences at each step, OneR with 1 (1OR), 2 (2OR) or 3 (3OR) dependences and Symmetrical Uncertainty combined with 1 (1SU), 2 (2SU) or 3 (3SU) dependences at each step.

In order to evaluate all the studied methods, we have performed a 10 times 10-fold cross validation obtaining the average accuracy and the standard deviation for each wrapper and each dataset. We have also performed a two-tailed t-test at the .05 level to determine whether each wrapper has a significant effect on the accuracy of the Naive Bayes classifier. Table 2 resumes the results, showing for each algorithm and for each dataset whether the preprocessing algorithm obtains a significantly better data representation (marked as +), worse (-) or if the preprocessing algorithm does not achieve a significantly different data representation (blank). N means the algorithm is too slow and there is no results

Table 1. Descriptions of the datasets used. #att means the number of attributes of the dataset, #inst the number of instances, #nom the number of nominal attributes, #num the number of numeric attributes, NB Acc. the average accuracy of the Naive Bayes classifier and NB Discrete the average accuracy of the Naive Bayes classifier when discretizing all the attributes. The numbers after the \pm indicate the standard deviation of the reported accuracy. All the results are obtained by 10 times 10-fold cross validation.

| Dataset | #atts. | #inst. | #nom. | #num. | NB Acc | NB Discrete |
|-----------------|--------|--------|-------|-------|-----------------|-----------------|
| Haberman | 4 | 306 | 1 | 3 | 74.80 \pm 0.2 | 74.38 \pm 0.5 |
| Hayes-Roth | 5 | 132 | 1 | 4 | 72.18 \pm 4.0 | 81.13 \pm 1.3 |
| Iris | 5 | 150 | 1 | 4 | 95.46 \pm 0.6 | 93.06 \pm 0.7 |
| Tae | 6 | 151 | 5 | 1 | 51.56 \pm 1.2 | 52.25 \pm 1.3 |
| Bupa | 7 | 345 | 1 | 6 | 55.14 \pm 0.8 | 57.39 \pm 1.7 |
| Yeast | 7 | 1.479 | 1 | 6 | 57.32 \pm 0.4 | 50.92 \pm 0.5 |
| Machine | 8 | 209 | 1 | 7 | 66.80 \pm 2.4 | 44.11 \pm 0.7 |
| Ecoli | 8 | 336 | 1 | 7 | 85.61 \pm 0.7 | 84.09 \pm 0.5 |
| Pima Indians D. | 9 | 768 | 1 | 8 | 75.75 \pm 0.5 | 75.01 \pm 0.2 |
| Abalone | 9 | 4.177 | 2 | 7 | 23.99 \pm 0.1 | 24.27 \pm 0.1 |
| Nursery | 9 | 12.960 | 9 | 0 | 90.28 \pm 0.0 | 90.28 \pm 0.0 |
| Glass | 10 | 214 | 1 | 9 | 48.14 \pm 1.3 | 50.88 \pm 1.7 |
| TicTacToe | 10 | 958 | 10 | 0 | 69.81 \pm 0.3 | 69.81 \pm 0.3 |
| Cmc | 10 | 1.473 | 8 | 2 | 50.63 \pm 0.3 | 48.53 \pm 0.4 |
| Wine | 14 | 178 | 1 | 13 | 97.46 \pm 0.4 | 97.12 \pm 0.5 |
| Adult | 15 | 48.842 | 9 | 6 | 82.69 \pm 0.0 | 81.54 \pm 0.0 |
| Crx | 16 | 653 | 10 | 6 | 77.75 \pm 0.3 | 86.02 \pm 0.3 |
| PenDigits | 17 | 10.992 | 1 | 16 | 85.77 \pm 0.1 | 85.59 \pm 0.1 |
| Letter Recog. | 17 | 20.000 | 1 | 16 | 64.07 \pm 0.1 | 59.98 \pm 0.1 |
| Segmentation | 20 | 2.310 | 1 | 19 | 80.10 \pm 0.1 | 89.58 \pm 0.1 |
| WdbcCancer | 31 | 569 | 1 | 30 | 93.32 \pm 0.2 | 94.06 \pm 0.2 |
| Ionosphere | 34 | 351 | 1 | 33 | 82.48 \pm 0.6 | 88.92 \pm 0.3 |
| Spam | 58 | 4.610 | 1 | 57 | 79.51 \pm 0.1 | 75.41 \pm 0.1 |

for the dataset (some algorithms are hundreds of times slower than the Naive Bayes).

At the end of Table 2 there is a summary showing the number of times each algorithm achieves a significantly better data representation (Total-23). As for some algorithms there are only results for 15 datasets, there is also a summary (Total-15) counting the number of times each algorithm achieves a better data representation only taking care about the 15 datasets for which all algorithms have been computed.

For allow comparisons, Figure 2 and Figure 3 show the number of times each wrapper achieves the best accuracy. Figure 2 groups the state of the art wrappers and Figure 3 shows the results for the DGW variations. Dark bars represents the number of times each wrapper achieves the best accuracy when only the 15 datasets for which all wrappers have results, and light bars show the results when taking into account all the 23 datasets.

Table 2. Summary table showing when a wrapper achieves a significantly better data representation (+) or worse (−) than the original one. *N* means there are no results available for that wrapper and dataset. A blank space means the data representation achieved by the wrapper is not significantly distinct than the original one. All the variations of the DGW algorithm are condensed in DGW(All) as all of them achieve significantly better data representations in the same datasets.

| Data Set | FSS | BSE | FSSJ | BSEJ | BFF | BFB | BFBi | GS | GR | IG | RE | C2 | DGW(All) |
|-----------|----------|----------|----------|----------|-----|-----|------|----|----|----|----|----|----------|
| Tae | | | + | + | | | | | | | | | |
| Hayes-R. | | | + | + | | | | | | | | | + |
| Haber. | | | | | | | | | | | | | |
| Iris | | | | | | | | | | | | | |
| Bupa | + | + | + | + | + | + | + | + | + | + | + | + | + |
| Wine | | + | | + | + | + | + | + | | | | | + |
| Machine | + | + | − | − | + | + | + | + | + | + | + | + | + |
| Glass | + | + | + | + | + | + | + | + | + | + | | | + |
| Ecoli | | | − | | | | | | | | | | |
| Pima Ind. | + | + | + | + | + | + | + | + | | | | | + |
| TicTacT. | | + | + | + | + | + | + | + | + | + | + | + | + |
| Crx | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | + | + | + | + | + | + | + | + | + |
| Cmc | + | + | | | + | + | + | + | + | + | | + | + |
| Yeast | | | − | − | | | | | | | | | |
| Ionosp. | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | + | + | + | + | + | + | + | + | + |
| Wdbc. | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | + | + | + | + | + | + | + | + | + |
| Abalone | + | + | + | + | + | + | + | + | + | + | + | + | + |
| Segmen. | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | + | + | + | + | + | + | + | + | + |
| Spam | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | + | + | + | + | + | + | + | + | + |
| Letter R. | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | + | + | + | + | + | + | + | + | + |
| Pendigits | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | + | + | + | + | | | | | + |
| Nursery | − | | + | + | | | | | | | | | |
| Adult | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | − | + | − | + | | + | + | + | + |
| Total-15 | 5 | 8 | 5 | 7 | 8 | 8 | 8 | 8 | 6 | 6 | 4 | 5 | 9 |
| Total-23 | <i>N</i> | <i>N</i> | <i>N</i> | <i>N</i> | 14 | 16 | 14 | 16 | 12 | 13 | 11 | 11 | 17 |

We have also computed the execution times of the wrappers in order to allow a comparison among them (see Figures 4 and 5). For each dataset, we have computed the number of times each wrapper is slower than the Naive Bayes (as a way to avoid absolute times and normalize the data). Then, for each wrapper, we have computed the average of these values, resulting in a metric that shows the number of times each wrapper is slower than the Naive Bayes. Figure 4 groups the results for the DGW variations and Figure 5 shows the results for the state of the art wrappers tested.

5 Discussion of Results

Comparing results presented in Table 2, it can be concluded that DGWrapper achieves significantly better data representations for the Naive Bayes more often

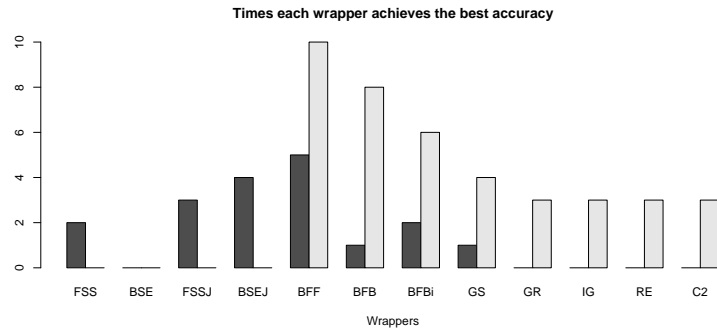


Fig. 2. Number of times each wrapper from the state of the art achieves the best accuracy. Dark bars represents the results for 15 datasets and light bars show the results for all the 23 datasets.

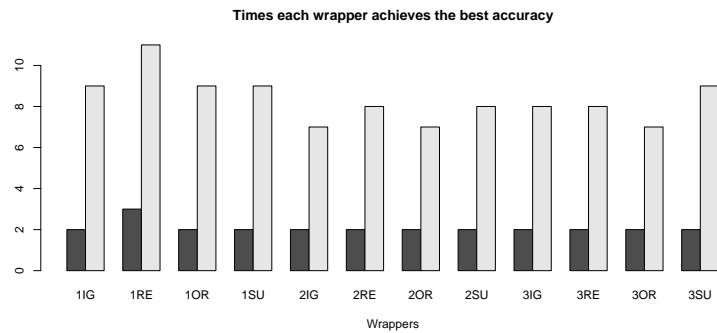


Fig. 3. Number of times each DGW variation achieves the best accuracy. Dark bars represents the results for 15 datasets and light bars show the results for all the 23 datasets.

than any other wrapper tested in the experiments: 17 times when considering all the datasets and 9 times when considering only 15. DGW (1R variation) obtains the best accuracy in 11 of the 23 datasets proven, more often than the rest of the wrappers. BFF obtains the best accuracy 10 times, not very far from DGW results, but DGW is 40% faster than BFF. Using these experimental results we can conclude that DGW is the best wrapper when using the Naive Bayes Classifier as learner, in terms of final predictive accuracy. Moreover, in the execution time graphics, we can notice that the DGW is among the fastest wrappers, more than 100 times faster than the FSSJ and BSEJ proposed by Pazzani. The DGW is equiparable, in terms of running time, to the most simple and faster wrappers: the rankers.

Backward approaches can not be regarded as superior to forward approaches since although backward approaches (BSE, BSEJ, BFB) obtain significantly bet-

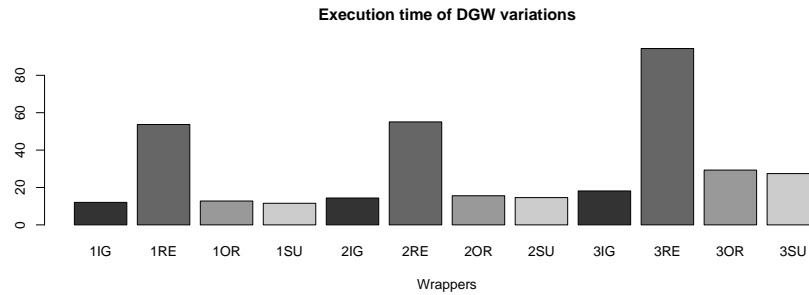


Fig. 4. Execution times (computed as the number of times each algorithm is slower than the Naive Bayes) of the DGW variations.

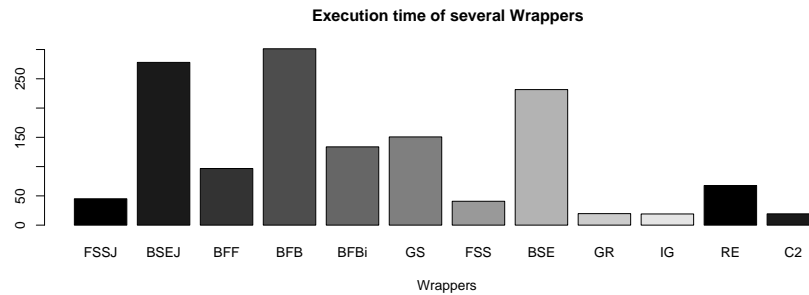


Fig. 5. Execution times (computed as the number of times each algorithm is slower than the Naive Bayes) for the state of the art wrappers compared in this paper.

ter data representations for the Naive Bayes more often than their corresponding forward approaches (FSS, FSSJ, BFF) (see Table 2), results also show that forward approaches obtain the best accuracy more often than the backward ones (see Figure 2).

FSSJ and BSEJ, in average, are as good as FSS and BSE, but in some datasets are quite better or worse than the rest of wrappers. FSSJ and BSEJ need all the attributes to be discrete, and it can be noticed in Table 1 that for those datasets, when discretizing all the attributes, the Naive Bayes Classifier shows a similar behavior. This means that part of the effect of the FSSJ and BSEJ is due to discretization of the attributes. In any case the concept of joining attributes instead of deleting them seems very interesting and would be interesting to integrate in the proposed method allowing the DGWrapper to evaluate at each step whether to delete one of the attributes belonging to the dependence or to join the attributes.

6 Conclusions and Future Work

We have shown that when learning the Naive Bayes Classifier, searching for dependences among attributes results in significant increases in accuracy. We proposed a general algorithm, DGW, that performs a search in the attribute space guided by the information of linear dependences among the values of the attributes. We have tested 12 DGW variations and 12 state of the art wrappers, performing searches in the attribute space in order to avoid the effect of dependences in the Naive Bayes Classifier. Experimental results and evaluation show that DGW provides the most improvement while spending less time than almost any other wrapper tested.

BSEJ and FSSJ rely their performance in the discretization of the attributes, and that makes difficult to compare the results. In future experiments we would test variations of BSEJ and FSSJ that do not need to discretize all the attributes (or that only discretizes the attributes ‘on demand’) in order to allow better comparison. Joining the attributes seems a good way to delete dependences but not loosing some other information and would be interesting to incorporate this concept into the DGW algorithm. Also would be interesting to test other metrics to measure the dependences among attributes like Mutual Information or multivariate measures like CFS [31].

References

1. Rish, I.: An empirical study of the naive bayes classifier. In: International Joint Conference on Artificial Intelligence, American Association for Artificial Intelligence (2001) 41–46
2. Fisher, R.: The use of multiple measurements in taxonomic problems. *Annals of Eugenics* **7** (1936) 179–188
3. Bayes, T.: An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions* **53** (1763) 370–418
4. Kononenko, I.: Semi-naive bayesian classifier. In: EWSL-91: Proceedings of the European working session on learning on Machine learning, New York, NY, USA, Springer-Verlag New York, Inc. (1991) 206–219
5. Zhang, H., Ling, C.X., Zhao, Z.: The learnability of naive bayes. *Lecture Notes in Computer Science* **1822** (2000) 432–441
6. Kononenko, I.: Inductive and bayesian learning in medical diagnosis. *Applied Artificial Intelligence* **7**(4) (1993) 317–337
7. Lewis, D.D.: Representation and learning in information retrieval. PhD thesis, Amherst, MA, USA (1992)
8. Lewis, D.D.: Naive (Bayes) at forty: The independence assumption in information retrieval. In Nédellec, C., Rouveirol, C., eds.: Proceedings of ECML-98, 10th European Conference on Machine Learning. Number 1398, Chemnitz, DE, Springer Verlag, Heidelberg, DE (1998) 4–15
9. Mitchell, T.: *Machine Learning*. 1 edn. McGraw Hill (1997)
10. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: *Machine Learning, Neural and Statistical Classification*. Ellis Horwood (1994)
11. Kononenko, I.: Comparison of inductive and naive bayesian learning approaches to automatic knowledge acquisition

12. Langley, P., Iba, W., Thompson, K.: An analysis of bayesian classifiers. In: National Conference on Artificial Intelligence. (1992) 223–228
13. Zhang, H., Su, J.: Naive bayesian classifiers for ranking. In: ECML. (2004) 501–512
14. Cortizo, J.C., Giráldez, J.I.: Discovering data dependencies in web content mining. In Gutierrez, J.M., Martínez, J.J., Isaias, P., eds.: IADIS International Conference WWW/Internet. (2004)
15. Cortizo, J.C., Giráldez, J.I.: Multi criteria wrapper improvements to naive bayes learning. In Corchado, E., Yin, H., Botti, V.J., Fyfe, C., eds.: IDEAL. Volume 4224 of Lecture Notes in Computer Science., Springer (2006) 419–427
16. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. *Machine Learning* **29**(2-3) (1997) 131–163
17. Kohavi, R.: 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. (1996) 202–207
18. Pazzani, M.: Constructive induction of cartesian product attributes. *ISIS: Information, Statistics and Induction in Science* (1996)
19. Domingos, P., Pazzani, M.J.: On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning* **29**(2-3) (1997) 103–130
20. Domingos, P., Pazzani, M.J.: Beyond independence: Conditions for the optimality of the simple bayesian classifier. In: International Conference on Machine Learning. (1996) 105–112
21. Hand, D.J., Yu, K.: Idiot’s bayes - not so stupid after all? *International Statistical Review* **69**(3) (2001) 385–299
22. Bellman, R.: *Adaptive Control Processes: a Guided Tour*. Princeton, University Press (1961)
23. Duch, W.: *Filter Methods*. In: Feature Extraction, Foundations and Applications. Springer Verlag (2004)
24. Langley, P.: Selection of relevant features in machine learning. In: Proceedings of the AAI Fall Symposium on Relevance. (1994)
25. Kohavi, R., John, G.H.: Wrappers for feature subset selection. *Artificial Intelligence* **97**(1-2) (1997) 273–324
26. Langley, P., Sage, S.: Induction of selective bayesian classifiers. (1994) 399–406
27. Pazzani, M.J. In: Searching for Dependencies in Bayesian Classifiers. 5th Workshop on Artificial Intelligence and Statistics (1996)
28. Kittler, J.: *Feature Selection and Extraction*. In: Handbook of Pattern Recognition and Image Processing. Academic Press (1986)
29. Newman, D., Hettich, S., Blake, C., Merz, C.: *UCI repository of machine learning databases* (1998)
30. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd edn. Morgan Kaufmann (2005)
31. Hall, M.A.: *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Department of Computer Science, University of Waikato (1998)