

# DISCOVERING DATA DEPENDENCIES IN WEB CONTENT MINING

José C. Cortizo

*Artificial Intelligence & Network Solutions S.L.  
C/Bélgica 7 2B, Fuenlabrada (28943) Madrid (Spain)*

Ignacio Giráldez

*Departamento de Inteligencia Artificial  
Universidad Europea de Madrid  
C/Tajo s/n, Villaviciosa de Odón (28670) Madrid (Spain)*

## ABSTRACT

Web content mining opens up the possibility to use data presented in web pages for the discovery of interesting and useful patterns. Our web mining tool, FBL (Filtered Bayesian Learning), performs a two stage process: first it analyzes data present in a web page, and then, using information about the data dependencies encountered, it performs the mining phase based on bayesian learning. The Naïve Bayes classifier is based on the assumption that the attribute values are conditionally independent for a given the class. This makes it perform very well in some data domains, but performs poorly when attributes are dependent. In this paper, we try to identify those dependencies using linear regression on the attribute values, and then eliminate the attributes which are a linear combination of one or two others. We have tested this system on six web domains (extracting the data by parsing the html), where we have added a synthetic attribute which is a linear combination of two of the original ones. The system detects perfectly those synthetic attributes and also some “natural” dependent attributes, obtaining a more accurate classifier.

## KEYWORDS

Web Content Mining; Linear Regression; Bayesian Classifier; Attribute Dependencies.

## 1. INTRODUCTION

The Bayesian classifier (Duda and Hart, 1973) studies the most probable target value of a new instance according to its attribute values ( $a_1, a_2, \dots, a_n$ ). The most probable target value will be the one which maximizes  $P(v_j | a_1, a_2, \dots, a_n)$ . For this process we need to calculate  $P(a_1, a_2, \dots, a_n | v_j)$  which is not easy, but if we assume the attribute values are independent, given the target value  $v_j$ , then the probability  $P(a_1, a_2, \dots, a_n | v_j)$  can be factored as the product  $P(a_1 | v_j) \dots P(a_n | v_j)$ , resulting:

$$v_{NaïveBayes} = \underset{v_j \in V}{argmax} P(v_j) \prod_i P(a_i | v_j) \quad (\text{Eq. 2})$$

To study how (Eq. 2) works, we use the accuracy on unseen examples as the metric to study the output of different classifiers. The Naïve Bayes Classifier achieves its best accuracy when the attribute values are independent given the class value, and it can be optimal in other particular domains which present dependencies as can be seen in (Domingos and Pazzani, 1996). It has suffered multiple variations due to the singular problems in some domains and its own characteristics (Webb and Pazzani, 1998). In this paper we study the problem of domains which present dependent attribute values, without considering those which can be optimal even containing dependent attributes, as has been studied in (Langley, 1993), (Langley and Sage, 1994), (Pazzani, 1997). In those approximations, the way to improve the accuracy was to transform the attributes using a greedy algorithm that, in the easiest way, starts with an empty set of attributes  $F$ . In each

iteration, it adds an attribute  $A_i \in S$  when by means of adding  $A_i$  the classifier performs better than by adding any  $A_j \neq A_i / A_i, A_j \in S$ . (Pazzani, 1997) considers also the possibility to join two attributes into one for avoiding dependencies. (Montes, 1994) also detects data dependencies, but then works only with the dependencies and not with the original attributes. In this paper, we present a non greedy algorithm based on recognizing attribute dependencies. We consider that searching for dependencies by means of testing each attribute combination is a rudimentary method, because it doesn't allow to treat each kind of dependency in a different way, and also doesn't extend the benefits to other machine learning algorithms. If we can detect the dependencies and classify them according to some properties, it could be possible to improve the Bayesian Classifier accuracy by transforming the attribute set according to the class of the dependency. For making it possible, we use linear regression, a classical statistical solution to the problem of determining the relationship between two random variables X and Y (or even more variables).

In this paper, we explore a method of deleting dependent attributes by first calculating all the possible dependencies (linear dependencies only), and ordering them by their strength ( $R^2$ ). With that information there is no need of testing all the possible combinations of attributes, you only need to delete attributes according to that list, and stop deleting them when no accuracy performance is reached.

## 2. FINDING AND DELETING DEPENDENT ATTRIBUTES

Let  $S = \{A_1, A_2 \dots A_n\}$  be the original set of attributes and  $C = \{y_1, y_2 \dots y_m\}$  the possible classes.  $E_j = \{\vec{x}, y\}$  is a training example where  $\vec{x} = \{a_1, a_2 \dots a_n\}$  is a point of the input space and  $y \in \{y_1, y_2 \dots y_m\}$  is a point in the output space; then the Naïve Bayes classifier receives as input a set of training examples  $T = \{E_1, E_2 \dots E_k\}$  and produces, as output, a function h (hypothesis) mapping the input space onto the output space,  $h: X \rightarrow C$  according to the probabilistic model defined before which can be used to predict the class of previously unseen entries ( $E_j$ ).

Our method modifies S deleting all the dependencies  $A_i = a + bA_v + e$  and  $A_i = a + bA_v + cA_w + e$  where  $A_i \neq A_v \neq A_w \wedge A_i, A_v, A_w \in S$  and  $a, b, c, e \in \mathfrak{N}$ . For this purpose, we create a list of all possible dependencies  $L = \left\{ [D_q, R_q] / D_q = ([A_i, A_v, a, b, e] \vee [A_i, A_v, A_w, a, b, c, e]) \wedge q < t \text{ if } R_q > R_t \right\}$  where  $R_q$  is the square of the correlation and measures how well the linear regression explains the relation. We calculate all the possible dependencies between pairs or trios of attributes for each class value, that's why, finally, we have  $m \times ((n \times (n - 1)) \times (1 + (n - 2)))$  possible dependencies.

L is a list of attributes dependencies ordered by its strength. After the computation of L, we obtain  $F = \{A_i, A_j \dots A_z \mid A_k \in S\}$  which is a subset of S where each  $A_i$  is linearly independent given the class. Finally we modify each element in T deleting each attribute value that belongs to an attribute deleted in this process. The complete process can be summarized as follows.

- (0)  $F = S, L = \{ \}$ 
  - (1) For each  $y_i \in C$ 
    - (2) For each  $A_i \in S$ 
      - (3) For each  $A_j \in S - \{A_i\}$ 
        - (4)  $L = L \cup g(A_i, A_j)$
        - (5) For each  $A_k \in S - \{A_i, A_j\} : L = L \cup g(A_i, A_j, A_k)$

- (7) For each  $L_s \in L$
- (8) If  $accuracy(F' = F - \{A_i \in L_s\}) > accuracy(F)$  then  $F = F'$
- (9) Else break
- (10)  $t(T)/t : S \rightarrow L$
- (11)  $Näive(t(T))$

Where  $g$  is the linear regression method (we have used  $g$  as a general function for a more general algorithm because it could be possible that  $g$  were a non linear regression method) applied to attribute groups that returns the  $R^2$  factor for the best dependency between those variables, and  $t$ , a function that transforms each training example deleting all the attribute values that belongs those attributes that are in  $S$  but not in  $F$ .

## 2.1 Experimental Results

We ran some experiments comparing our system to the Näive Bayes algorithm. We selected five domains from (Blake and Merz, 1998) and added one synthetic attribute to each domain (so that each domain had, at least, one attribute linear relationship). We created one html file for each domain for the system to consult the data on-line. The system accessed the file and parsed the html tags, extracting the data and created an internal representation of that data for applying the entire process. On each problem we ran the experiments using tenfold cross-validation to obtain the more objective results.

Then, for each domain we ran the Näive Bayes algorithm with all the attributes including the synthetic ones (with synthetic deps.), with only the original ones and using FBL. The results are shown in Table 1.

Table 1. Accuracy comparison between Näive Bayes classifier and FBL including number of deleted attributes

Domain	N.B. (with synth. deps.)	N.B. (original)	FBL	Deleted Attrs.
Balance-scale	0.8336	0.9136	0.9136	1
Contraceptive	0.4840	0.5017	0.5078	3
Glass Identification	0.8178	0.8411	0.8925	3
Wine Recognition	0.8983	0.9719	0.9719	1
TAE	0.4569	0.4967	0.5099	2
Breast Cancer	0.9728	0.9742	0.9742	1

As can be seen in Table 1, our system is capable of detecting all the synthetic dependencies avoiding the low accuracy of the original. But in three of the six domains, the system performs better than the Näive Bayes classifier working on the original data, this improvement is caused by some natural attribute relationships that also filtrated on the complete process. This better performance also indicates the system works with less attributes than the Näive classifier over the original data (see Table 1) and that's a good signal because it is obtaining better results with simpler classifiers.

But dependencies are not only a Bayesian matter. As can be seen in Table 2 (the bolded values are the best values for that domain), some other Machine Learning algorithms are affected in a similar way by this dependencies. In this table we present the difference between the accuracy in systems using the original attribute set and the accuracy adding the synthetic attributes.

Table 2. Comparison between some machine learning algorithms and FBL when using only original attributes and when using also synthetic ones in 3 domains (Contraceptive Method, Breast Cancer and Balance Scale) and proofing with several synthetic dependencies in each one

Algorithm	CM1	CM2	CM3	BC1	BC2	BC3	BC4	BS1	BS2	BS3
FBL	0.07	0.07	0.07	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>2.30</b>	<b>2.30</b>	<b>2.30</b>
N. Bayes	-1.15	-2.45	-1.02	-0.14	-0.14	-0.14	-0.14	-9.11	-5.23	-0.68
C4.5	<b>-0.61</b>	<b>1.77</b>	<b>0.68</b>	1.72	3.72	2.72	0.14	0.00	0.45	0.68
C4.5 rules	0.34	0.81	1.08	0.72	1.58	0.29	0.86	-2.74	-1.60	-0.91
KNN(K=1)	-1.29	-0.61	-1.29	0.04	0.28	0.28	-0.15	-0.23	0.91	1.14
KNN(K=4)	-0.81	-0.88	-0.88	0.15	0.72	0.72	-0.43	-1.14	-0.91	-2.5
SMO	0.06	0.06	0.00	-0.14	-0.14	-0.14	0.00	0.22	-0.23	-0.46

The results are promising because the system can detect all the synthetic linear dependencies, but also detects five natural linear dependencies on the test domains, which let us think about how good it would be avoiding the linear limitation on the regression method for detecting the relations. It is also interesting to see (Table 2) that in all the algorithms tested it happens that in some domains the accuracy decreases when using dependent attributes.

### 3. LIMITATIONS AND FUTURE WORK

The promising results on the selected domains, as it can be seen in Table 1, are a good starting point because it shows regression is a reliable method for detecting attribute relations. But there is a huge variety of methods, each one related to one kind of function (polynomial, exponential, logistic, etc. and even non regression relations as shown in (Montes, 1994)) and using that methods allows us to detect much more dependencies, because in real problems the relationships are not always linear. In this way, allowing relationships with more than two attributes would also increase the possibilities space.

Detecting attributes dependencies would also help some tree based models, as is explained in (Robnik and Kononenko, 1999) and it could be interesting to extend the benefits of this system to other inductive algorithms (because as shown in Table 2, they present problems with dependencies in some domains).

### 4. CONCLUSION

We have shown that searching for dependencies among attributes when learning Näive Bayesian classifiers results in increases in accuracy. We proposed a non greedy method for detecting and eliminating attributes relationships, and demonstrated it works on synthetic and natural domains. It is also interesting to remark that three out of the five domains presented natural linear dependencies among attributes, which exposes in some common databases or natural problems, the Näive assumption is violated and the Bayesian Classifier needs some help not to decrease its normal accuracy.

Results show the convenience of studying the detection of dependencies using regression methods for avoiding the loss of accuracy caused by hidden relationships among attributes. This process is specially useful in web content mining applications, because data definitions are, either not accesible, or difficult to understand by web content mining tools. As a consequence, for these tools, it is difficult to take advantage of data dependencies, unless they can detect them themselves, as FBL does.

### REFERENCES

- Blake, C.L. and Merz, C.J., 1998. UCI Repository of machine learning databases. Irvine, CA.
- Domingos, P. and Pazzani M., 1996. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *International Conference on Machine Learning*. Bari, Italy, pp. 105-112.
- Duda, R. and Hart, P., 1973. *Pattern classification and scene analysis*. Jonh Wiley and Sons, New York, USA.
- Langley, P. 1993. Induction of recursive Bayesian classifiers. *Proceedings of the Eight European Conference on Machine Learning*. Vienna, Austria, pp. 153-164.
- Langley, P. and Sage, S. 1994. Induction of selective Bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence*.. Seattle, USA, pp. 339-406.
- Montes, C., 1994. *MITO: Método de Inducción Total*. PhD Thesis, Universidad Politécnica de Madrid, Spain.
- Pazzani M., 1997. Searching for dependencies in Bayesian Classifiers. *Artificial Intelligence and Statistics IV*. Springer Verlag, New York, USA.
- Robnik, M. and Kononenko, I., 1999. Attribute dependencies, understability and split selection in tree based models. *International Conference on Machine Learning*. Bled, Slovenia. Morgan Kauffman, pp. 344-353.
- Webb, G.I., & Pazzani, M.J. (1998). Adjusted probability naive Bayesian induction. *Proceedings of the Eleventh Australian Joint Conference on Artificial Intelligence*. Berlin. Springer-Verlag. pp. 285-295.