Learning Bayesian network classifiers for credit scoring using Markov Chain Monte Carlo search

B. Baesens[1], M. Egmont-Petersen[2], R. Castelo[2], and J. Vanthienen[1]
[1]K.U.Leuven Dept. of Applied Economic Sciences
Naamsestraat 69, B-3000 Leuven, Belgium
{Bart.Baesens;Jan.Vanthienen}@econ.kuleuven.ac.be
[2]Utrecht University, Institute of Information and Computing Sciences
Padualaan 14, De Uithof, The Netherlands
{Michael;Roberto}@cs.uu.nl

Abstract

In this paper, we will evaluate the power and usefulness of Bayesian network classifiers for credit scoring. Various types of Bayesian network classifiers will be evaluated and contrasted including unrestricted Bayesian network classifiers learnt using Markov Chain Monte Carlo (MCMC) search. The experiments will be carried out on three real life credit scoring data sets. It will be shown that MCMC Bayesian network classifiers have a very good performance and by using the Markov Blanket concept, a natural form of input selection is obtained, which results in parsimonious and powerful models for financial credit scoring.

1. Introduction

The problem of financial credit scoring is a very challenging and important financial analysis problem. Many techniques have already been proposed to tackle this problem, ranging from statistical classifiers to decision trees, nearest-neighbour methods and neural networks. Although the latter are powerful pattern recognition techiques, their use for practical problem solving (and credit scoring) is rather limited due to their intrinsic opaque, black box nature. Recently, Bayesian network classifiers have been introduced in the literature as probabilistic white-box classifiers with the capacity of giving a clear insight into the structural relationships in the domain under investigation. In this paper, we will investigate the performance and complexity of various kinds of Bayesian network classifiers for financial credit scoring. The experiments will be carried out on three real life credit scoring data sets.

This paper is organised as follows. Section 2 discusses several types of Bayesian network classifiers. The experimental setup and the results are presented in section 3. Conclusions are drawn in section 4.

2. Bayesian network classifiers

A Bayesian network (BN) represents a joint probability distribution over a set of discrete attributes. It is to be considered as a probabilistic white-box model consisting of a qualitative part specifying the conditional (in)dependencies between the attributes and a quantitative part specifying the conditional probabilities of the data set attributes [8]. Formally, a Bayesian network consists of two parts $B = \langle G, \Theta \rangle$, the directed acyclic graph G consisting of nodes and arcs and the conditional probability tables $\Theta = (\theta_{x_1}, ..., \theta_{x_n})$. The nodes are the attributes $X_1, ..., X_n$ whereas the arcs indicate direct dependencies. The graph G then encodes the independence relationships of the domain. Each conditional probability table has the form $\theta_{x_i|\Pi_{x_i}} = P_B(x_i|\Pi_{x_i})$ for each possible value x_i of X_i, and Π_{x_i} of Π_{X_i}, where Π_{X_i} denotes the set of direct parents of X_i in G. The network B then represents the following joint probability distribution:

$$P_B(X_1, ..., X_n) = \prod_{i=1}^{n} P_B(X_i|\Pi_{X_i}) = \prod_{i=1}^{n} \theta_{X_i|\Pi_{X_i}}. \quad (1)$$

The first task when learning a Bayesian network is to find the structure G of the network. Once we know the network structure G, the parameters Θ need to be estimated. In this paper, we will use the empirical frequencies from the data D to estimate these parameters[1]:

$$\hat{\theta}_{x_i|\Pi_{x_i}} = \hat{P}_D(x_i|\Pi_{x_i}). \quad (2)$$

[1]Note that we hereby assume that the data set is complete, i.e., no missing values.

It can be shown that these estimates maximise the log likelihood of the network B given the data D [4].

A Bayesian network is essentially a statistical model that makes it feasible to compute the (joint) posterior probability distribution of any subset of unobserved stochastic variables, given that the variables in the complementary subset are observed. This functionality makes it possible to use a Bayesian network as a statistical classifier by applying the winner-takes-all rule to the posterior probability distribution for the (unobserved) class node [3].

2.1. The Naive Bayes Classifier

A simple Bayesian network classifier, which in practice often performs surprisingly well, is the Naive Bayes classifier [3]. This classifier basically learns the class-conditional probabilities $P(X_i = x_i|C = c_l)$ of each attribute X_i given the class label c_l. A new test case $(X_1 = x_1, ..., X_n = x_n)$ is then classified by using Bayes' rule to compute the posterior probability of each class c_l given the vector of observed attribute values:

$$\begin{aligned} &P(C = c_l|X_1 = x_1, ..., X_n = x_n) = \\ &\tfrac{P(C=c_l)P(X_1=x_1,...,X_n=x_n|C=c_l)}{P(X_1=x_1,...,X_n=x_n)}. \end{aligned} \tag{3}$$

The simplifying assumption behind the Naive Bayes classifier is that the attributes are conditionally independent, given the class label. Hence,

$$\begin{aligned} &P(X_1 = x_1, ..., X_n = x_n|C = c_l) = \\ &\prod_{i=1}^{n} P(X_i = x_i|C = c_l). \end{aligned} \tag{4}$$

This assumption simplifies the estimation of the class-conditional probabilities from the training data. Notice that one does not estimate the denominator in Eq. (3) since it is independent of the class. Instead, one normalises the nominator term to 1 over all classes.

2.2. The Tree Augmented Naive Bayes Classifier

In [4] Tree Augmented Naive Bayes Classifiers (TANs) were presented as an extension to the Naive Bayes Classifier. TANs relax the conditional independence assumption by allowing arcs between the attributes. An arc from attribute X_i to X_j implies that the impact of X_i on the class variable also depends on the value of X_j. In a TAN network, the class variable has no parents and each attribute has as parents the class variable and at most one other attribute. The attributes are thus only allowed to form a tree structure. In [4], a procedure was presented to learn the optional arrows in the structure that forms a TAN network. This procedure is based on an earlier algorithm suggested by Chow and Liu [1].

2.3. Unrestricted Bayesian network classifiers

Several different approaches exist for learning the structure of an unrestricted probabilistic network, for an overview, see, e.g., [6]. In this paper, we use a Bayesian approach to structure learning [5]. The motivation behind this approach is to obtain samples from a (posterior) distribution of probabilistic network structures (the models M) given the (discrete) data D, rather than learning a particular probabilistic network structure that maximizes a certain criterion such as the maximum likelihood. The posterior distribution of the (finite number of) *models* is given by Bayes' formula:

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}, \tag{5}$$

with

$$P(D) = \sum_{M \in \mathcal{M}} P(D|M)P(M). \tag{6}$$

The probability distribution $P(D|M)$ is the likelihood of the data D given the model M, which has a closed form given certain assumptions [6]. The term $P(M)$ is the prior of the network structure (we assume a uniform prior). The posterior probability $P(M|D)$ allows us to account for the uncertainty of every model. Once we account for the uncertainty of the models, it is possible to weigh competing quantities of interest by averaging over all the models in the following way:

$$P(\Delta|D) = \sum_{M \in \mathcal{M}} P(\Delta|M, D)P(M|D), \tag{7}$$

where Δ refers to the quantity of interest and $P(\Delta|D)$ is its posterior density given the data D. For our current purpose, Δ will correspond to a restriction regarding conditional independence. For a given query node C (the *class label* in classification problems), define the two disjoint subsets Z and Y, $X = Z \cup Y \cup \{C\}$, for which it holds that C is independent of Z, when all variables in Y are observed, $C \perp\!\!\!\perp Z|Y$. The set Y is uniquely defined by the graphical structure G, and is called the *Markov Blanket* of node C. If the direct parents of a node C is denoted by Π_C and the direct children by Σ_C, the markov blanket of node C is given by $\Pi_C \cup \Sigma_C \cup \Pi_{\Sigma_C}$.

The method of Markov chain Monte Carlo (MCMC hereafter) samples directly from the posterior distributions $P(M|D)$ and $P(\Delta|D)$, so the summation (Eq. (7)) is performed implicitly. The Metropolis-Hastings algorithm was first adapted for structural learning of Bayesian and Markov Networks by Madigan and York [7]. Let $\mathcal{M}$ denote the whole class of DAGs given by the complete set of variables X. For each DAG $M \equiv G \in \mathcal{M}$, let $\mathcal{N}(G)$ be the set of *neighbour* models of M. Let q be a transition matrix such that for some other $M' \equiv \mathbf{M}(G') \in \mathcal{M}$, $q(M \rightarrow M') = 0$ if $M' \notin \mathcal{N}(G)$ and $q(M \rightarrow M') > 0$ if $M' \in \mathcal{N}(G)$. Using the transition matrix q we build a Markov chain $M(t, q)$, $t = 1, 2, \ldots, n$, with state space $\mathcal{M}$.

Let the chain $M(t, q)$ be in state M and let's draw a candidate model M' from $q(M \rightarrow M')$. The proposed model M' is accepted with probability

$$\alpha(M', M) = min \left\{ 1, \frac{|\mathcal{N}(G)| P(M'|D)}{|\mathcal{N}(G')| P(M|D)} \right\}, \qquad (8)$$

where $|\mathcal{N}(G)|$ refers to the cardinality of the set of neighbors of the model M. If M' is not accepted, $M(t, q)$ remains in state M. Such a Markov chain $M(t, q)$ has $P(M|D)$ as its equilibrium distribution which means that, after the chain has ran for *enough* time, the draws can be regarded as a sample from the target distribution $P(M|D)$, and one says that the chain has *converged*.

The *convergence* of the Markov chain to the target distribution $P(M|D)$ is guaranteed under the regularity conditions of Irreducibility and Aperiodicity. In our particular case, we should take care on how a new candidate graph is proposed, in order to retain those regularity conditions. For Bayesian Networks [7], it reduces to pick randomly an ordered pair of vertices and propose the addition of an arc if the vertices are non-adjacent and the addition keeps the digraph acyclic. If the vertices are adjacent then the algorithm proposes the removal of the arc, which depends of no further condition as removal always leaves the digraph acyclic.

When the Markov chain $M(t, q)$ converges, the draws from the Markov chain mimic a random sample from $P(M|D)$. Therefore, in order to get an estimate of $P(M|D)$, it suffices to account for the frequency of visits of each model M and divide it by the number iterations n. After all these iterations have been performed, the obtained graphs can be grouped according to their markov blankets in relation to the node of interest, C.

Although the MCMC learning algorithm results in a set of directed acyclic graphs, it is by no means trivial to compare the resulting independence relations that follow from a particular graph. The domain knowledge we have is confined to the correct classification of each case. Assuming a symmetric loss function, this classification should correspond to the most likely outcome of the query node (winner-takes-all rule). Therefore, we decided to use the percentage of correctly classified training cases as criterion for assessing the appropriateness of a particular model. The only drawback of this approach is that inference - computation of the posterior probability distribution of the query node - is required. Inference is known to be computationally complex. Moreover, in the MCMC learning paradigm, each graph in the chain needs to be triangulated before the desired posterior probabilities can be computed.

3. Experiments

All Bayesian network classifiers were applied to three real life credit scoring data sets. The Bene1 and Bene2 data set were obtained from major Benelux financial institutions whereas the German credit data set is publicly available at the UCI repository[2]. After discretisation, Bene1 has 3123 observations with 23 inputs, Bene2 7190 observations with 28 inputs and German credit 1000 observations with 15 inputs. Each data set was randomly split into two-third training set and one-third test set. We compare the performance of the Bayesian network classifiers with the performance of C4.5, which is a well-known decision tree classifier. For the MCMC net classifier, we did 10 runs of each 15000 iterations for the Markov Chain and chose the network with the best training set accuracy.

The performance is quantified using both the percentage of correctly classified (PCC) observations and the area under the receiver operating characteristic curve (AUROC). The receiver operating characteristic curve (ROC) is a 2-dimensional graphical illustration of the sensitivity on the Y-axis versus 1-specificity on the X-axis for various values of the classification threshold [9]. It basically illustrates the behaviour of a classifier without regard to class distribution or misclassification cost. The AUROC then provides a simple figure-of-merit for the performance of the constructed classifier. We will use McNemar's test to compare the PCCs of different classifiers and the nonparametric chi-squared test of De Long, De Long and Clarke-Pearson to compare the AUROC values [2].

Table 1 depicts the PCC and AUROC values for all classifiers on the test set. The best absolute PCC and AUROC performances per data set are underlined. The MCMC net classifier obtained the best PCC performance on both Bene1 and German credit. For the Bene2 data set, its PCC performance was statistically different from C4.5 at the 5% level but not at the 1% level. The AUROC values of the MCMC classifier were not statistically different from the best AUROC value at the 1% level for all three data sets.

Besides the classification accuracy, we also want to evaluate the Bayesian network classifiers on their complexity. To this end, we will look at the number of nodes and links in the network. For C4.5, we will look at the number of leaf nodes and total number of nodes (leaves and internal nodes) of the tree to assess its complexity. Table 2 displays the complexity of the Bayesian network and C4.5 classifiers. The Naive Bayes and TAN classifiers do not prune any inputs because they all remained in the Markov Blanket of the classification node for all three data sets. However,

[2]http://kdd.ics.uci.edu/

	Bene1		Bene2		German credit	
	PCC Test	AUROC Test	PCC Test	AUROC Test	PCC Test	AUROC Test
Naive Bayes	69.54	<u>76.58</u>	69.09	74.15	71.52	<u>76.64</u>
TAN	68.67	75.63	73.47	<u>77.27</u>	67.88	59.60
MCMC net	<u>71.29</u>	75.39	72.06	75.34	<u>73.03</u>	73.24
C4.5	69.74	71.73	<u>73.75</u>	72.64	71.21	68.25

Table 1. Performance of the Bayesian network classifiers and C4.5.

	Bene1	Bene2	German credit
Naive Bayes	24 nodes and 23 links	29 nodes and 28 links	16 nodes and 15 links
TAN	24 nodes and 45 links	29 nodes and 55 links	16 nodes and 29 links
MCMC net	4 nodes and 5 links	12 nodes and 23 links	6 nodes and 6 links
C4.5	110 leaves and 157 nodes	287 leaves and 423 nodes	37 leaves and 53 nodes

Table 2. Complexity of the Bayesian network classifiers and C4.5.

the Markov Blanket concept allowed the MCMC net classifier to prune a lot of inputs for all data sets. This resulted in parsimonious yet powerful networks for decision making. E.g., for the Bene1 data set, 20 inputs could be pruned leaving only 3 input nodes and the classification node in the network. Note that the decision trees induced by C4.5 are rather complex because of the large number of leaves and nodes. Figure 1 presents the MCMC net that was trained for the German credit data set.

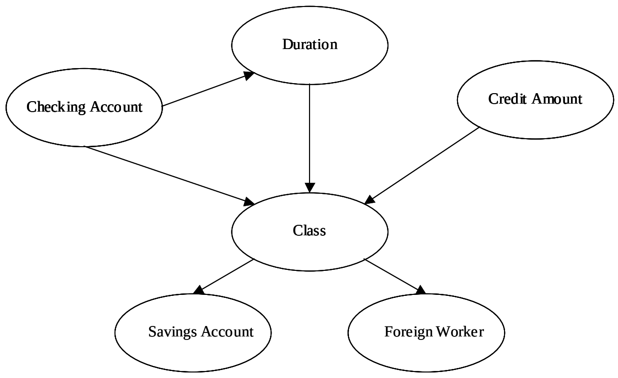

Figure 1. MCMC net for German credit.

4. Conclusions

In this paper, we have evaluated and contrasted several types of Bayesian network classifiers for credit scoring. The evaluation was done by looking at the performance (in terms of classification accuracy and area under the receiver operating characteristic curve) and the complexity of the trained classifiers. We used Markov Chain Monte Carlo (MCMC) search to learn unrestricted Bayesian network classifiers. It was shown that MCMC Bayesian network classifiers have a very good performance and by using the Markov Blanket concept, a natural form of input selection is obtained, which results in parsimonious and powerful models for financial credit scoring.

References

[1] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[2] E. De Long, D. De Long, and D. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, 44:837–845, 1988.

[3] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley, New York, 1973.

[4] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[5] P. Giudici and R. Castelo. Improving markov chain monte carlo model search for data mining. *Machine Learning*, 2001. forthcoming.

[6] D. Heckerman, D. Geiger, and D. Chickering. Learning bayesian networks: The combination of knowledge and statitical data. *Machine Learning*, 20:194–243, 1995.

[7] D. Madigan and J. York. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232, 1995.

[8] J. Pearl. *Probabilistic reasoning in Intelligent Systems: networks for plausible inference*. Morgan Kaufmann, 1988.

[9] J. Swets and R. Pickett. *Evaluation of Diagnostic Systems: Methods from Signal Detection Theory*. Academic Press, New York, 1982.