Delft University of Technology
Faculty of Information Technology and Systems

Masters Thesis

Bayesian Networks in Credit Rating

Samuel GERSSEN

March 12, 2004.
Supervising Committee:

Prof. Dr. H. KOPPELAAR
Drs. Dr. L.J.M. ROTHKRANTZ
Ir. W. DE JONG
Dr. D. FOKKEMA (ABN AMRO)
Dr. Ir. M.J. DRUZDZEL (University of Pittsburgh)

Abstract

Risk assessment of credit portfolios is essential in banking. The bank with the most accurate view on its credit risk, will be most profitable. In order to calculate the risk, each client's 'probability of default' needs to be estimated. The probability of default is defined as the probability that a client can not meet its repayment obligations toward the bank anytime in the next twelve months. Credit rating models assign a probability of default to each client, based on a set of input variables. In this research, the best practice modeling method, logistic regression, is benchmarked with Bayesian networks. A Bayesian network is a graphical representation of a probabilistic model. Different Bayesian network structures return different results, ranging from bad to good. Some structures require advanced learning techniques. Opportunities to improve these techniques are proposed. This research was performed at ABN AMRO Group Risk Management and in the Decision Systems Laboratory of the University of Pittsburgh.

Acknowledgments

This report is the summary of my graduation project, from March 2003 to January 2004. It was a very educational experience, and I am proud of the results. However, the quality of the project and this report was heavily boosted by the support of the following people. Hereby, I would like to express my sincere gratitude to those who helped me in many different ways:

- Maud van der Put
- Leon Rothkrantz
- Diederik Fokkema
- Marek Druzdzel
- Prof. H. Koppelaar
- Kees van der Meer
- Bram Warmenhoven
- Ronan Quere
- Adam Zagorecki
- Anne Gerssen

Furthermore, my stay at the University of Pittsburgh from September 2003 to December 2003 would not have been possible without the financial support of:

- My Parents, Ton & Tammina Gerssen
- Fundatie Vrijvrouwe van Renswoude
- Stimuleringsfonds Internationale Universitaire Samenwerkingsrelaties (STIR), Delft University of Technology
- Koninklijk Instituut voor Ingenieurs (KIVI)

Contents

Chapter 1

Background

1.1 The New Basel Capital Accord

The Basel Committee on Banking Supervision is a committee of representatives from central banks of ten countries all over the world. The committee formulates broad supervisory standards and guidelines and recommends statements of best practice in the expectation that individual authorities will take steps to implement them through detailed arrangements - statutory or otherwise - which are best suited to their own national systems. Accordingly, the Dutch Central Bank (DNB) imposes rules and guidelines to the Dutch banks.

In January 2001 the Basel Committee on Banking Supervision issued a proposal for a new Basel Capital Accord that, once finalized, will replace the current 1988 Capital Accord. One of the topics it focuses on is minimum capital requirements.

The capital ratio of a bank is defined as the banks regulatory capital divided by its risk-weighted assets. It is a percentage of the credit portfolio a bank should always hold back to reduce the risk of bank insolvency. According to the 1988 Capital Accord, a minimum capital ratio of 8% is required for banks. Basel II intends to improve safety by placing more emphasis on flexibility and risk sensitivity. Thus the capital ratio is calculated as follows:

$$CapitalRatio = \frac{TotalCapital}{CreditRisk + MarketRisk + OperationalRisk}$$

Basel II describes, aside from the standardized approach, a new approach to calculate credit risk, the Internal Risk Based (IRB) approach. According to the IRB approach, banks will be allowed to use their internal estimates of borrower creditworthiness to assess credit risk in their portfolios, subject to strict methodological and disclosure standards. Distinct analytical frameworks will be provided for different types of loan exposures, for example corporate and retail lending, whose loss characteristics are different.

The credit risk is directly related to:

- Probability of Default (PD). The probability that a client will default next year;
- Exposure at Default (EAD). The total exposure to the borrower at the time of default;
- Loss Given Default (LGD). The percentage of the exposure, which the bank will be unable to recover.

Basel II does not prescribe a calculation method for the PD, EAD and LGD, therefore banks are permitted to use their own best practice methods.

1.2 Economic Capital

ABN AMRO goes beyond Basel II, it wishes to manage its credit risk even safer than Basel II prescribes, and therefore, it calculates the Economic Capital (EC), which is a quantile (e.g., 99,97 %) of the loss distribution curve. It is the capital a bank requires to cover its unexpected losses (in the case of 99,97 %) in 9.997 out of 10.000 years. Statistically, a bank with an EC of 99,97 will not be able to cover its losses in three out of 10.000 years, and therefore be in default itself. Aside from Basel II, there are two reasons for ABN AMRO to calculate its EC.

ABN AMRO wishes to introduce risk-adjusted rates on its credit portfolios. Pricing will depend on the EL (Expected Loss) and the EC of the portfolio. Best possible estimates are of course needed. Additionally, for budgeting purposes, it is important to know the exact EC per portfolio.

To calculate and manage EC, one needs to know PD, EAD and LGD. To calculate EL, the following formula is used:

$$EL = PD \cdot EAD \cdot LGD$$

Therefore, best possible estimates of PD, EAD and LGD are needed. As mentioned before, these loss characteristics are different for the distinct credit portfolios. Consequently, every credit portfolio needs to be modeled separately.

In March 2003, most credit portfolios in the bank already had a PD-model. However, some portfolios were not covered yet.

Chapter 2

The Assignment

This research was performed for a masters thesis at Delft University of Technology. Leon Rothkrantz from the department of Knowledge Based Systems, run by professor Koppelaar, was the main supervisor of the project. The first six months of the project were completed at ABN AMRO bank, the last three months at the University of Pittsburgh.

ABN AMRO N.V. is the largest Dutch bank. The department Quantitative Consultancy, in Group Risk Management, provides high level quantitative solutions to departments anywhere in the bank. This particular project was performed for the Credit Portfolio Management department of Business Unit the Netherlands. The highly professional standard, along with the extensive quantitative experience, provided an excellent environment for a masters research.

Decision Systems Laboratory is a research group within the Department of Information Science and Telecommunications and the Intelligent Systems Program at the University of Pittsburgh. Bayesian networks is one of the core research areas of this group, many publications originate from DSL. The most advanced software tool for probabilistic reasoning models, GeNIe, is a product of DSL. This group was the perfect environment for an in-depth study on Bayesian networks, and provided inspiration for the development of new learning methods.

The assignment was twofold:

- Provide a validated model for PD estimation for one of the ABN AMRO credit portfolios. Logistic regression and Bayesian Networks are the techniques to be used. The best possible modeling method should be applied.
- Explore Bayesian network modeling and make improvements to parameter learning techniques.

The first part was performed at ABN AMRO, the second part mainly at the University of Pittsburgh.

The general layout of the project is as follows:

1. Literature survey
2. Application of ABN AMRO data
   - Define the portfolio
   - Select and gather the data
   - Select variables based on performance tests and expert interviews
   - Select performance measures
   - Develop and validate a logistic regression model
3. Bayesian networks research
   - Study modeling techniques
   - Improve learning methods
   - Search for new learning methods
4. Application of Bayesian networks on ABN AMRO data
   - Develop Bayesian network models
   - Analyze results
5. Conclusions & recommendations

The literature survey is described in chapter 3. The three main topics are credit rating models, Bayesian networks and rare event modeling.

Part I of this report contains the application of the credit rating data and logistic regression modeling. This is basically a summary of the work done at ABN AMRO. Portfolio definition and construction, and variable selection were very time consuming tasks, which are only briefly discussed in this report, since most of the material is confidential.

In part II, some Bayesian networks theory and a couple of interesting BN modeling techniques are introduced. These techniques are the basis for the new developments and credit portfolio tests.

Part III contains an overview of new developments, and tools built for this project. A new noisy-MAX learning algorithm is discussed, some sampling algorithms and a robust prior assessment and posterior learning method is introduced. Also the tools in which they are implemented are briefly discussed.

The application of Bayesian network models for credit rating is discussed in part IV. All modeling techniques discussed in the previous parts are tested on the credit data, and the results are presented and explained.

Chapter 16 contains the conclusions and recommendations.

Chapter 3

Literature

3.1 Credit Rating

Credit rating can roughly be subdivided into:

1. corporate risk (including bond credit risk, correlating portfolio risk, etc.)
2. retail risk

Corporate risk is based on large amounts of financial data for an individual client. Default of such a client will highly affect a bank. Consequently, it has been a topic of constant attention for banks. Retail risk, on the other hand, does not rely heavily on financial data, since it is not available.

The retail exposures are so small relatively, that it is hardly profitable to spend time on individual rating. However, assessing groups of clients with similar risk characteristics can be very useful. These characteristics are the basis of any retail credit risk model.

Multiple techniques are available for default prediction. One of the first quantitative methods, which became a standard in the 1970s, was developed by Altman [3]. The method, based on multiple discriminant analysis (MDA), uses weighted financial ratios to discriminate between healthy companies and companies likely to default. His selection of ratios was not accepted as widely as the method itself.

The first logistic regression (logit) and probit models [1] were developed in the 1980s, providing more accurate predictions than MDA. Logit and probit models are still the most popular techniques for modeling customer default. Similar to MDA, logit and probit are regression models, but instead of a linear relation between the response variable and the predictive variables, a logistic function or an inverted normal distribution is used. For a more detailed description of logistic regression, see [6, 16] or turn to chapter 6.

Besides MDA, logit, and probit models, several types of neural networks have been applied for default prediction [30, 32]. [32] finds that neural networks outperform the discriminant analysis models, and also [30] shows that neural networks at least attain the same level of performance as multiple discriminant analysis. They argue that neural networks are also easier to use than MDA.

[4] provides a good overview of neural network studies in bankruptcy analysis. Many studies cited in this paper in fact show that neural networks outperform MDA, but definitely not all. The explanation is that a nonlinear model such as a neural network is better able to fit the data than a linear model. Logistic regression is not a linear model, especially not if discrete variables are used. At the end of the day, the performance of logistic regression versus neural networks depend on which scoring measure is used, which input variables are used and which assumptions are made. Resultwise, logistic regression and neural networks are both interesting techniques, but somehow, neural networks have not been accepted by the financial world so much, because the drawback of neural networks is the inability to interpret nodes and parameters. Therefore causality between variables is hidden, resulting in a 'black box' model.

Some researchers focus more on explaining the reasons behind defaults. Cybinski [7] outlines previous research on bankruptcy prediction models, including recent work using neural networks, chaos theory and dynamic models, and argues that the lack of a clear dichotomy between failure and non failure is a major problem. Examines models which compare failed firms with themselves over time (i.e. from survival to failure) and reports a study of 60 US service industry firms which failed between 1971 and 1991. Applies the author's family of models to data from the four years before failure, discusses the results and suggests that this approach (i.e. looking for signals of deterioration) may be better at explaining failure than the 'holy grail' of seeking to predict which firms will and will not fail.

Kaski [19] uses self-organizing maps based on the Fisher metric for default prediction and explanation.

An overview of the methods, best practices and issues for credit risk modeling in retail markets can be found in [2]. They describe the moving trend in the retail loan market from relation lending to and survey credit rating models, such as Fair, Isaac, SMEloan, KMV, Moodys RiskCalc and Credit Risk Plus.

3.2 Bayesian Networks

A Bayesian network [26, 29] is a directed acyclic graph with each node representing a variable and each arc representing a causal relation between two variables. Variables are characterized by a probability distribution for each value. The probability distribution of each node is influenced by the states (for discrete nodes) or values (for continuous nodes) of the parent node. The conditional probabilities of a node are stored in a conditional probability table (CPT).

This CPT is needed to calculate any conditional probability in the model. This process is called inference and was first addressed by Pearl in 1986 [25]. An efficient inference algorithm was developed in 1988 by Lauritzen & Spiegelhalter, called the clustering algorithm [21]. The clustering algorithm constructs a join tree of cliques of nodes, which makes the calculations of marginals very efficient.

Bayesian networks provide the model developer with a unique advantage. A combination of expert knowledge and statistical data can be used to obtain the structure and parameters of a model. A procedure for deriving as much useful knowledge from experts is described by Druzdzel in [8]. Expert knowledge is not always presented in a way that it can be incorporated in a model straight away. Sometimes experts are reluctant to give the exact conditional probabilities. The paper presents a framework for expert knowledge extraction, translation, and combination with statistical data.

Furthermore, both the structure and the parameters can be learned from the data [14]. Structural learning is based on scoring possible networks, and optimizing local parts of the network. Scoring metrics such as likelihood maximization are used. Conditional probability parameters can be obtained by calculation the expectation of the probability distribution of the probability. This approach (probability of a probability) is the Bayesian approach.

A particular type of Bayesian networks, naive Bayes, performs very well in classification models. Naive Bayes assesses each probability of an input variable, given the state of the response variable. Using Bayesian probability calculation, the conditional probability of the response variable can be derived. [11] describes an extension to naive Bayes, called tree augmented naive Bayes (TAN). TAN's yield even better results than regular naive Bayes.

Another competitive classifier is the noisy-OR model, or its extension, noisy-MAX. The noisy-OR classifier is discussed in [31]. Especially for many input variables, the noisy-OR model can greatly reduce the necessary parameters. Noisy-OR learning is not very straightforward, although expectation maximization is the common approach.

3.3 Rare Event Modeling

A data set of observations allows for deriving an estimate for the probability distribution of a response variable, conditional on the values of predictive variables. A large number of random observations will result in a more accurate estimate than just a small sample. Therefore, large data sets usually provide good, stable models.

Unfortunately many real-world data sets are imbalanced; some states of the dependent variable may be dozens to thousands of times less likely than other states. This is the case in, for example, customer bankruptcies in banks, international armed conflicts, or epidemiological infections. So the effect of having a large data set available is canceled by the fact that it contains only a small number of 'interesting' records. Therefore, a model built from such data may be severely biased or highly unstable.

This problem occurs in generalized linear models, such as logistic regression, but also in models learned from data, such as neural networks and Bayesian networks. Especially in the case of generalized linear models, techniques such as prior correction and weighting have been developed to discard most of the 'uninteresting' part of the data set without much performance loss [20]. Additionally there are ways of reducing bias and variance [27]. In Bayesian networks, small sample problems are usually solved by setting a good prior distribution [5, 22], based on expert knowledge. However, if this expert knowledge is not available, we need to set an acceptable prior based on the data. [23] propose Noisy-OR gates to tackle the problem. Also in this case the parameters need to be obtained from the data, since expert knowledge is limited.

Part I

Credit Portfolio & Modeling

Chapter 4

The Data

In this chapter, the construction of the data set and the variable selection process are described. In practice, this was a very time consuming task. Since some of the material is confidential, this chapter is but a summary of the work that has been done to come up with the input data for the models. Additional information about the data and the input variables can be found in an ABN AMRO confidential report (sg-0012).

The construction of the data set was performed with a logistic regression model in mind. As described in the assignment, a logistic regression model was required by ABN AMRO, because it is the best modeling technique from the literature and their own experience. In this research, it is the benchmark model. Please note that some decisions concerning the data set construction clearly favor logistic regression above other models. The impact of these decisions are described in section 4.5.

The goal of this assignment was to develop probability of default (PD) models. Therefore, a proper definition of default is needed. If a customer fails to meet his obligations toward the bank concerning repayment or position of the balance he is said to be in default. What those obligations are, is irrelevant for the modeling, therefore the definition of 'the default state' is not included in this report. Much more important is how the *probability* of default is defined.

Definition 1 (Probability of Default)
The Probability of Default at time $t = 0$ is defined as the probability that a client will ***enter*** *the default state at time $0 < t \leq 1$ (year).*

This means that only the clients who are not in default at $t = 0$ get a PD assigned to them. Furthermore, only the entry of the default state matters, so a client who enters default but is not in default any more at the end of the year is still marked as a defaulted client.

The definition leads to a portfolio with approximately 50.000 clients (50.000 records). The default rate is approximately 0.02 (or 2%). In tables and graphs, the default rate is always represented by a number between 0 and 1 (e.g., 0.02).

The objective of the data selection part is to get a data set with a set of predictive variables of each client at $t = 0$, and a variable indicating if the client went into default at time $0 < t \leq 1$.

This definition assumes default to be a binary event. A client is either in default, or not. Obviously, only two states do not allow a precise description of the client state. Therefore, one could argue the use of more than two states, e.g. different 'degrees' of default. There are a couple of reasons why this is not an option:

- The most important reason is that the rules for calculation of economic capital ($EL = PD \cdot EAD \cdot LGD$) requires a binary default state. The default state is not some assumption for this particular modeling project, it is a bank wide used standard, not only in ABN AMRO, but every Dutch or foreign bank.
- If there are multiple default states, EAD and LGD also need multiple states. This assignment was to build only a PD model for a portfolio, so the EAD and LGD models are not compatible with multiple default states.
- There is no data available about 'almost defaults'. As long as a client meets its obligations, there is no reason for the bank to further investigate the client, so it is assumed to be fine. Therefore, 'almost defaults' are often undetected. When this data is available, it may become very beneficial to include more default states.

Unlike a classification model, this model is supposed to return a probability of default, instead of just a 0 or a 1. This means that the actual state of the client can be expressed along a continuous range from 0 to 1, where high values indicate a higher probability of default. This adds a lot of precision to the resulting value, compared to a 0 or 1.

The calculation of the PD is based on client characteristics, represented by variables. In the systems of the bank, heaps of information can be retrieved for each client, such as age, or marital status, but also account behavior, such as current balance, and balance history. Ultimately, the set of available variables is limited by the systems from which they are extracted. Some information is just not available, such as spending habits of a customer.

Some variables in the resulting data set (such as sex) are a direct copy of a variable from the systems, while other variables (such as trend, volatility, age) are derived by means of calculating a value from one or more variables from the systems. The exact process of the variable construction process can be found in document sg-0012.

Variables can be subdivided into two categories: discrete and continuous variables. The difference is the number of possible states. A continuous variable can have a value anywhere in a certain range, while a discrete variable has a limited number of states.

Among the available variables some are discrete, such as sex (male/female), while others are continuous, such as age or balance. In the modeling data set, all variables are discretized. There are two reasons why this is done:

- It enhances the performance of a logistic regression model. During tests it appeared that discrete variables perform better. This is not surprising, because the number of degrees of freedom is increased when using discretized variables (only in logistic regression models). This allows a better fit in the model.
- Most variables are already discrete by nature (e.g. sex), so at the end of the day an entire set of continuous variables will not be possible anyway. When building a logistic regression model with a mixed set, the fit of the discrete variables will be favored at the cost of the continuous variables. Therefore, a data set with both types of variables is not preferable.

Obviously not all the variables will be used in the models, only the best. Four criteria are used to measure the quality of the variables:

- Predictive power
- Expert opinion
- Missing values
- Correlation

4.1 Predictive Power

The most important criterion for a variable is that it should have a certain predictive power. In other words, there should be a direct or indirect relation between the variable and the observed default rate. Variables with a direct influence on the default rate can be visualized by a univariate plot. Variables with a combined effect can be detected using multi-variate plots, but these plots are hard to interpret by the naked eye. Therefore, in this section, only an exploring data analysis is performed

A way to visualize a direct relationship is the univariate plot. A univariate plot is a bar chart that represents the default rate for each value of the (discretized if necessary) variable.

From the example univariate plot in figure 4.1 it appears that men seem to default more frequently than women. So in this example gender seems to be a variable with predictive power. One important issue to address is to make sure that the population of each category is representative. Suppose we have a group of 10000 men and 10 women. Now suppose we observe 1000 men that defaulted and 1 woman. Statistically we can safely assume that the default rate for men is around 10%, however it is not right to assume the default rate for women is 10% since we only have 10 women. So the real PD for women in general might as well be 30% or 1%. Therefore, a second series is added to the plot, indicating the percentage of occurrences, visualised by the white bars (Figure 4.2).

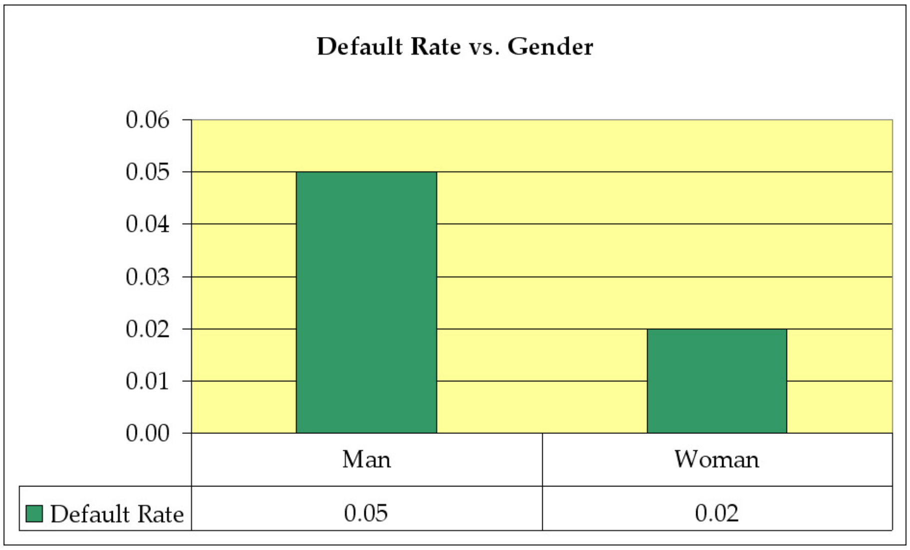


Figure 4.1: Univariate plot

A particularly good example of a variable with strong predictive power is X_2 (Figure 4.2). Notice the large populations in the categories 1, 2 and 3 with a low default rate. Also notice the high default rate in the lower populated, but not insignificant groups 4, 5 and 6.

Univariate plots provide a very convenient way to visualize direct correlation between the predictive variable and the response variable, but it does not show combined correlation. Stepwise logistic regression is a method capable of assessing the predictive power of a single variable when used together with other variables. In chapter 6, the concepts and results of stepwise logistic regression are described.

4.2 Expert Opinion

Each variable has its own 'story'. Some variables seem to have predictive power, while in fact that variable and the default rate are influenced by a common cause. Other variables are somehow dependent on the default rate instead of vice versa. Also some variables can not be used due to operational issues (e.g., discontinued maintenance).

Expert meetings were organized (for each portfolio), to verify the available variables and to obtain new ideas and underlying causes for defaults. At those meetings, people from the branch network and the restructuring unit were present. The comments on the variables were very useful. Unfortunately the general comments of the experts could not be applied in the models because the required data is too detailed or too subjective. This means, possible causes of interest described by the experts could not be related to the available data.

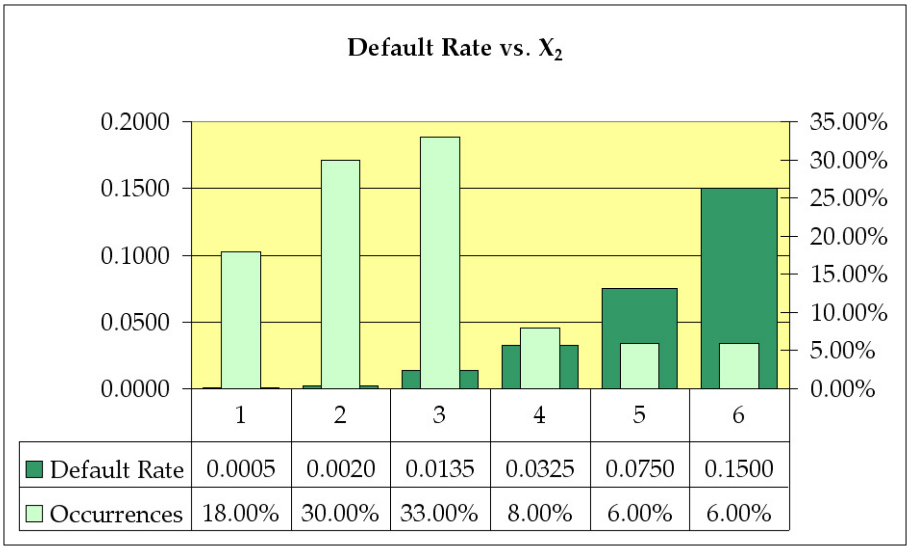


Figure 4.2: Univariate plot of X_2

The experts also commented on the univariate plots. Their comments on the variables are included and sometimes explained in the summary tables of the variables later in the correlation section.

4.3 Missing Values

Unfortunately some variables in the data set contain missing values. There are many techniques for dealing with missing values in input variables, such as presented in [10]. The majority of these methods fill in the blanks with a most likely value using expectation maximization, maximum likelihood, etc. However, there is one very important requirement which should be met when applying such a method. The cause of the missing value should be independent with the response variable.

Especially in this data set, the fact that a record contains a missing value absolutely affects the probability of default. Figure 4.3 shows a variable for which the default rate of clients with missing value is 10 times higher than the average portfolio default rate.

Obviously the missing values do not appear randomly in the data set. Usually the cause of the missing value indicates the randomness. Clients unwilling to give their phone number to the bank are definitely not a random sample of the entire portfolio. As a matter of fact, the value being missing is a very useful piece of information when assessing the probability of default. Now suppose a missing value generation method is applied to ‘retrieve’ the missing values, then this information will be lost. Therefore, no value generating algorithm was applied on this dataset.

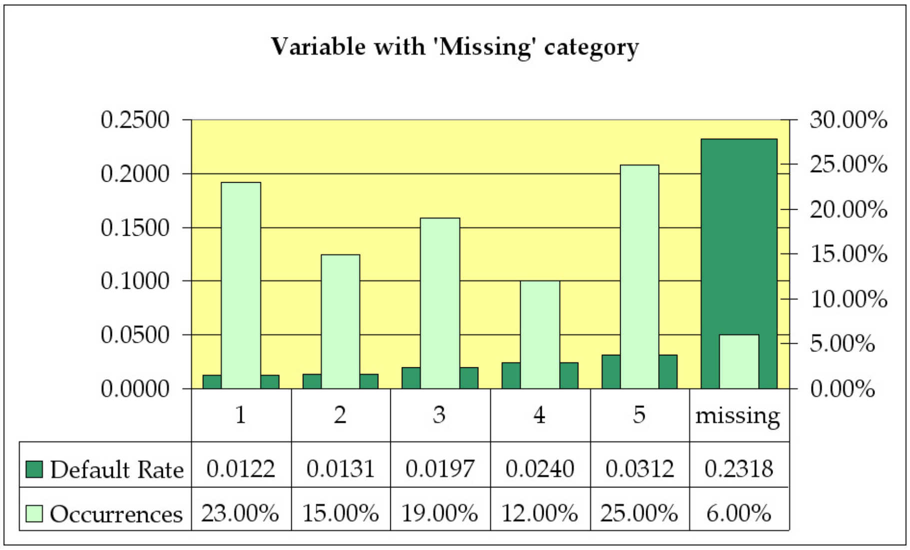


Figure 4.3: Variable with non-random missing values

There are two ways to deal with a missing value in a model:

- Add a 'missing' state to the variable.
- Build a separate model for the clients with a missing value.

Since all variables are discrete, adding a state is always possible. When adding a 'missing' state to a variable, the effects of the other variables on the PD are assumed to be independent of the value being missing. In logistic regression this means that the parameters of the other variables are not affected by one missing variable. However, if the clients with missing values clearly have default characteristics, this assumption is not correct. In that case a separate model should be developed for the clients with a missing value.

In the case of this portfolio it is safe to assume that a complete record and a record with a missing value have identical default characteristics. For confidentiality reasons, the context of this assumption will not be discussed in this report. Every variable with missing values will get an extra state, since the missing values indicate a higher probability of default.

4.4 Correlation

Strictly speaking, the word association (instead of correlation) should be used, since some variables are nominal and therefore it is impossible to calculate correlation (e.g., product-moment correlation). But, this being said, the word 'correlation' will be used for convenience reasons.

For logistic regression models, variables should have as little correlation as possible. Correlated variables jeopardize the stability of the model. Some variables are correlated by definition and, therefore, mutually exclusive.

For the calculation of correlation, different statistics have to be used. All variables are discretized, but there is a difference between ordinal and nominal variables. Ordinal variables have a natural order in the categories, such as age ('20-29' is lower than '30-39'), while nominal variables do not, e.g., sex (male/female).

For correlation between ordinal values, the Gamma measure of association is used. The estimator of Gamma is based only on the number of concordant and discordant pairs of observations. It ignores tied pairs. Somers' D is a measure which takes ties into account, but it does not guarantee a value between -1 and 1, while $-1 \leq \gamma \leq 0$. Assume two observations with ordinal variables a and b. Now suppose that in the second observation a is higher than in the first. If b is also higher, we call the pair concordant. If b is lower in the second observation we call the pair discordant. With an equal a or b, it is a tie. Gamma is obtained by:

$$\gamma = \frac{P - Q}{P + Q}$$

where P is the number of concordant pairs and Q is the number of discordant pairs. Gamma is appropriate only when both variables are ordinal.

Correlation between any other combination of two discrete variables is measured by Cramer's V statistic. It is a measure of association derived from the χ^2 statistic, which involves the differences between the observed and expected frequencies. Cramer's V is designed in such a way that the range is $0 \leq V \leq 1$.

Suppose a two-way cross-tabulation of variables X and Y where n_{ij} denotes the number of occurrences of $[x_i, y_j]$ and n is the total number of observations. Now let

$$m_{ij} = \frac{\sum_k n_{ik} \cdot \sum_k n_{kj}}{n}$$

so $\sum_k n_{ik}$ is the row total and $\sum_k n_{kj}$ is the column total. Then the Pearson χ^2 statistic is defined as:

$$Q_P = \sum_i \sum_j \frac{(n_{ij} - m_{ij})^2}{m_{ij}}$$

Cramer's V can now be calculated as follows:

$$V = \sqrt{\frac{Q_P/n}{min(s_X - 1, s_Y - 1)}}$$

where Q_P is the chi-square statistic and s_X and s_Y are the number of possible values (categories) of variables of interest X and Y.

Table 4.1: Evaluation of variables

	Predictive power	**Expert opinion**	**Missing**	**ME with:**
X_1	++	++	0%	
X_2	++	++	1.08%	
X_3	++	++	0%	X_4, X_5
X_4	++	+	0%	X_3, X_5
X_5	+	+	0%	X_3, X_4
X_6	+	+	0%	
X_7	+	+	2.97%	
X_8	+	+	4.86%	X_{10}
X_9	+	-	0%	
X_{10}	+	-	3.03%	X_8
X_{11}	+	-	4.86%	
X_{12}	-	-	0%	
X_{13}	-	-	4.37%	
X_{14}	- -	-	4.28%	
X_{15}	- -	-	29.96%	
X_{16}	++	- -	5.29%	
X_{17}	-	- -	0%	

Cramer's V has range $0 \leq V \leq 1$ and Gamma has the range $-1 \leq \gamma \leq 1$. Values close to zero indicate little correlation, while values around -1 and 1 indicate strong correlation.

Before the correlations are calculated, the useless variables will be discarded because they will unnecessarily distort the correlation tables. Which variables are useful and which are not can be derived from Table 4.1, where the variables (discretized versions) and quality criteria are given. Mutually exclusive variables are also given.

The variables are sorted from best to worst, according to the quality criteria. The first 11 variables are possibly interesting for the model. Although X_5 and X_{10} belong to those 11, they are discarded anyway due to mutually exclusiveness with other better variables. Since X_3 and X_4 are equally predictive, the decision which one to drop will be made later, depending on their performance in the model. This leaves 9 variables in the data set.

X_{16} and X_{17} have to be discarded because X_{16} is not updated anymore; basically this variable is nonexistent. X_{17} is not a cause for default, rather, this

	X_1	X_2	X_3	X_4	X_6	X_7	X_8	X_9	X_{11}
X_1		-0.2717	-0.1673	0.1849	0.0598	0.0582	-0.0136	0.0589	0.0264
X_2			0.7566	-0.5456	0.1740	0.0692	0.0324	0.1678	-0.1727
X_3				0.9882	0.0714	0.0364	0.0551	0.0856	-0.3391
X_4					0.2382	0.1364	-0.0581	0.1552	0.2317
X_6						0.1286	-0.0250	0.2810	-0.0169
X_7							0.1461	0.1742	0.0567
X_8								0.0633	-0.0341
X_9									0.1039
X_{11}									

[0, 0.05]
[0.10, 0.20]
[0.05, 0.10]
[0.20, 0.5]
[0.5, 1]

Figure 4.4: Correlation table

variable is changed after a default occurs and therefore cannot be of predictive value.

Figure 4.4 shows the correlations for the remaining variables. As mentioned before, the Gamma ($-1 \leq \gamma \leq 1$) and Cramer's V ($0 \leq V \leq 1$) statistics can be interpreted the same way (close to 0 means no correlation, close to -1 or 1 means strong correlation) and, therefore, these different statistics will be used in the same table.

From the remaining 9 variables, 7 variables in the model are ordinal and 2 are nominal (X_7 and X_9), so the first two columns in the table are Cramer's V values and the other columns are Gamma measures of association.

This table confirms the assumption that X_3 and X_4 are in fact mutually exclusive. Therefore, depending on which variable performs best in the model, one of them will be dropped. Apparently X_2 is correlated with X_3 and X_4. This is not surprising, since X_2 is somewhat correlated to those variables by definition. However, X_3 and X_4 will not be discarded both, since they contribute to the performance of the model, when used separately.

The definite data set will contain six variables ($X_1, X_2, X_4, X_6, X_7, X_{11}$), but the final selection will be discussed in chapter 6. These six variables will be used for every model.

4.5 Impact on Models

Ultimately, the impact on the results of the data selection process might be that for different modeling techniques, a different set of variables is optimal. This data selection process clearly favors logistic regression in a couple of ways:

- Stepwise logistic regression is used as the selection criteria to measure predictive power. Although stepwise logistic regression is a very convenient way to analyze the added predictive value of one variable to a set of variables, the underlying model is assumed to be a regression type of model.
- Correlation should be avoided in regression models, since it jeopardizes the stability of the model. Other modeling techniques, such as Bayesian networks, are very well able to handle correlated variables. In fact, an extra variable in a model can add performance, even if it is correlated with another variable.
- For logistic regression, a very limited amount of expert knowledge is needed. Basically, the exact meaning of the variables is enough. For Bayesian networks on the other hand, expert knowledge is the key to a well performing model. The knowledge of the experts appeared to be very limited, and by far not enough for Bayesian networks. Unfortunately, there was no opportunity to 'further examine' the experts.

This might seem insuperable, the bottom line is that the data set contains six powerful predictive variables and a response variable, fit for any modeling technique.

Chapter 5

Model Performance

5.1 Cross-validation

Aside from a development data set, an additional test data set is needed for the assessment of the model performance. Since the defaults are rare in these portfolios, a separation of the observations into a development sample and a test sample is not encouraged. Instead the portfolio is cross-validated by testing a model built on 80% of the portfolio on the remaining 20%, and this is repeated four times. The selection of the 20% samples happens at random, but they remain mutually exclusive, so each observation is in a test sample exactly once.

The result after applying the five models is a data set which contains for each observation an estimated probability of default. For the calculation of the GINI coefficient (Section 5.2), the estimated PD's will be compared with the observed defaults.

The selection of the development and test samples happens completely at random. However, stratified sampling is used to make sure that the samples have the same default rate in general. This means that instead of taking 80% of all observations, 80% of the non-defaulted clients are picked at random and joined with a random 80% of the defaulted clients. Now the test sample has the same default rate as the whole data set. Stratified sampling is used because of the low default rate, and test sets with a completely different default rate than the average should be avoided.

The number of cross-validation samples is arbitrary. For this case, five test data sets of each 20% were used, but theoretically any number of test data sets could be used, with a maximum n, where n is the total number of records. When the number of test data sets is exactly n, each test data set contains only one record. This method is called leave-one-out. It is the most proper way of cross-validation, since the development data sets contain $n - 1$ records, which is almost as many records as the entire data set. Unfortunately, it requires the development of n different models. With a data set of approximately 50.000 records, leave-one-out is computationally too expensive.

It should be noted that this type of testing is sufficient for a performance test, but a real validation of a model should be done with an out-of-time sample (the same data set at a different point in time). Unfortunately out-of-time data was not yet available at the time of the development of the models.

5.2 The GINI Index

Regardless what modeling technique is used, the result will be a probability of default assigned to each client in the portfolio. The objective is to develop the best possible model. Therefore, scoring measures are needed to quantify the performance of each model.

There are many scoring measures, all with different properties. The right measure to use depends on the application of the model. In classification problems, typically measures such as precision, sensitivity, and specificity are used [12]. These measures focus on ratios of false and true positives (or negatives). Essentially this case is a classification problem with two classes, default and non default. However, the relative classification (client A is more likely to default than client B) is more important, and so is the value of the assigned PD per client.

From the point of view of a bank, the most important issue is to calculate the risk for a particular client or a group of clients as accurately as possible. This will allow a detailed risk profile for the portfolio and maximize the risk-adjusted returns on capital.

The GINI index [33] is commonly seen as the most powerful measure to assess the overall ability of a model to discriminate between clients likely-to-default and clients unlikely-to-default. If a model is able to rank clients according to their probability of default (PD) then most of the likely-to-default clients should receive a high PD. The opposite should be true for unlikely-to-default clients.

The GINI index is illustrated by a graph where the cumulative default percentage of clients is plotted against the total percentage of clients when they are sorted according to their PD (low PD's on the left, high PD's on the right). This curve is known as the Lorentz curve. An example of a Lorentz curve is given in Figure 5.1.

The diagonal line is the 'random model' line. On the x-axis, the clients are sorted according to their assigned probability of default. We see that 80% of the observed defaults occur in the last 20% of the clients (the 20% with the highest PD). This means that the model evidently is able to make a good separation. So the bigger the area between the diagonal and the curve, the higher the performance of the model. Defaults among clients with an equal PD are assumed to be distributed randomly over those clients.

Definition 2 (The GINI index)
The GINI index is the area between the diagonal and the Lorentz curve divided by the total area under the diagonal.

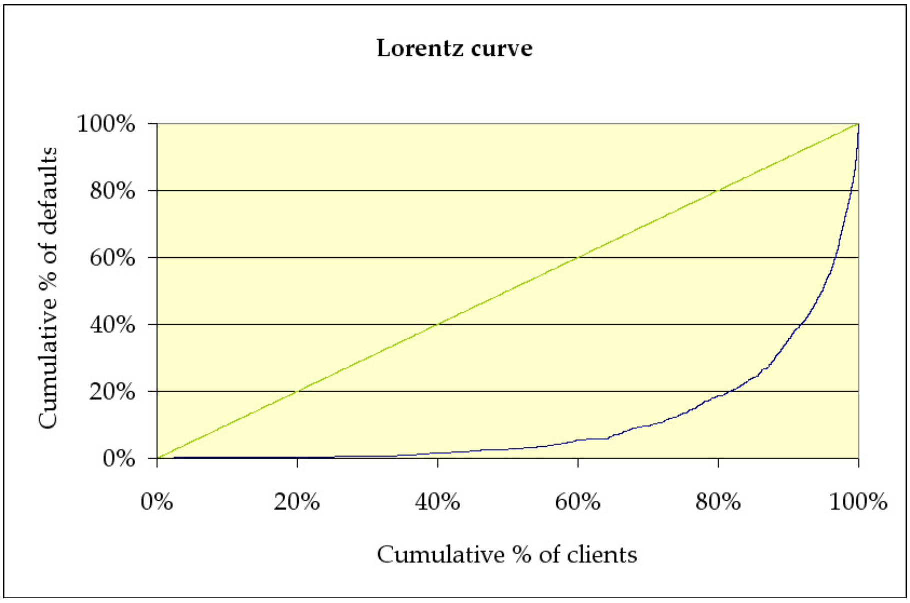


Figure 5.1: Lorentz curve (blue) and diagonal (light green)

The range of the GINI index is $0 \leq G \leq 1$ and the models with the highest discriminative power, i.e. the models which make a perfect sort between default clients and non-default clients, will get the highest coefficients. In practice, it depends on the data what score a good model will get. For application scoring (rating for new clients) a GINI index of 55% is already very high while for behavioural scoring (rating for existing clients) a GINI index is typically above 70%.

The perfect model would perfectly distinguish the default clients from the non default clients without exceptions and therefore would show a straight line along the x-axis from 0 until the % of non-defaults. From there, it will go straight to the point $[100\%, 100\%]$ (every increment in total clients is an increment in total defaults). Notice that a GINI index of 100% can not be achieved if the number of defaults is greater than 0.

The drawback of only a GINI index is the inability to evaluate the absolute values of the calculated PD's. In order to calculate the GINI index, the clients are sorted according to their PD, therefore only the relative PD has an effect on the GINI index. Suppose 0.5 is added to each PD, then the GINI index will remain the same because the order was not changed. The predicted risk profile, however, changes completely.

5.3 The χ^2-score

For this model not only discriminative power is important. Ultimately a bank wishes to gain insight in the risk profile of its credit portfolio. In retail this

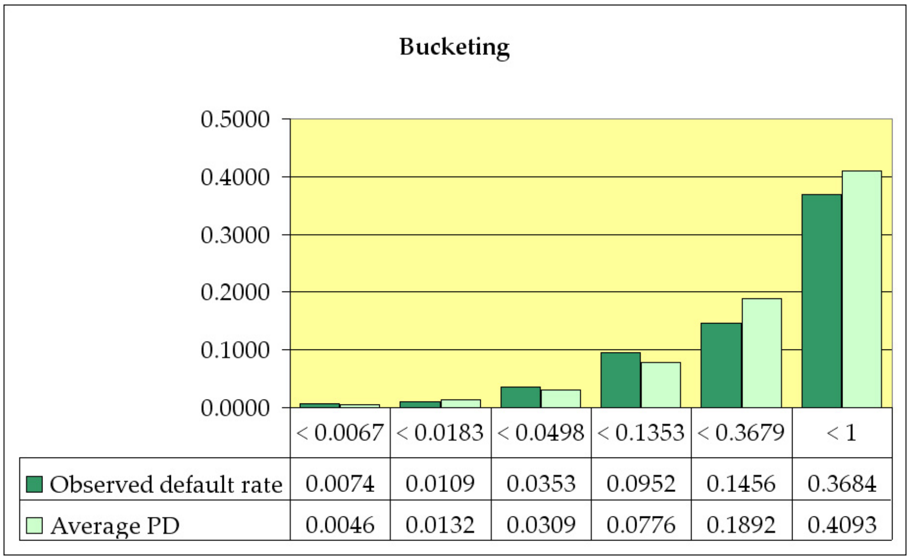


Figure 5.2: Graph of bucketing

means it wishes to estimate a PD as accurately as possible for groups of clients. Hence clients are grouped into buckets according to their PD. In theory, the limits of these buckets can be set randomly, however it is useful to make sure that there is a reasonable number of clients in each bucket. For this portfolio the PD's follow the pattern of an exponential distribution, therefore the limits for each bucket are set exponentially. If the number of buckets is N, the upper limit for each bucket is given as:

$$l_i = \frac{\exp(i)}{\exp(N)} \quad , \quad i = 1..N$$

For each bucket, the average PD and the actual default rate are calculated. In Table 5.1 an example of the bucketing is given for a portfolio of 1000 clients with 64 defaults separated into six buckets. In Figure 5.2 a graph is plotted for the average PD and observed default rate for each bucket.

Table 5.1: Example of bucketing

Bucket #	**1**	**2**	**3**	**4**	**5**	**6**
Upper limit	$< .0067$	$< .0183$	$< .0498$	$< .1353$	$< .3679$	< 1
# of clients	136	274	283	147	103	57
# of defaults	1	3	10	14	15	21
Default rate	0.0074	0.0109	0.0353	0.0952	0.1456	0.3684
Average PD	0.0046	0.0132	0.0309	0.0776	0.1892	0.4093

The average PD (provided by the model) should match the observed default rate for each bucket as close as possible. In other words, the difference between the observed default rate and average PD should be as small as possible. Instead of looking at the sum of the absolute differences, weighting will be applied to boost the effect of differences in the buckets with a low default rate. One way to accomplish weighting is by using the χ^2-measure. For N buckets, each with average PD a_i (prediction) and observed default rate r_i (real values), the χ^2-measure is given by:

$$\chi^2 = \sum_N \frac{(a_i - r_i)^2}{a_i}$$

Obviously small values of the χ^2-measure indicate a close (good) fit and bad models will yield a high χ^2 value. There is no need to normalize the χ^2-measure, since it will only be used to compare two models.

The χ^2 measure can also be applied to every single client, to measure the difference between the prediction and the real event. The disadvantage of this 'error minimizing' is that it is not very stable for the entire portfolio. Risk managers want an overview of the risk characteristics of the entire portfolio, so on average, the PD's assigned to groups of clients, should be accurate. Bucketed χ^2 exactly measures this.

There is another reason why not the squared error of each client is measured. When the β parameters are assessed, logistic regression minimizes the mean square error, which means that given a set of input variables, according to the χ^2 measure, there will never be a better model than logistic regression, if each individual error is measured. Even for a bucketed χ^2 score, logistic regression is by its nature bound to yield a good score.

As mentioned before, χ^2 measures the quality of the assigned PD's. The disadvantage is that it does not measure discriminative power. Assume a model which assigns each client a PD value which exactly matches the portfolio default rate. All clients will end up in the same bucket, where the difference between average PD and observed default rate will be low and therefore the obtained χ^2-measure will be low too. However in practice this model will be useless since it is not able to distinguish the good clients from the bad clients. Notice that the value of the GINI index affects the usability of the χ^2-measure.

Consequently, the χ^2-measure is extremely useful for this case when used simultaneously with the GINI index. In practice, the focus will be on the GINI index and the χ^2-measure will be used to verify that the PD's actually make sense. However if the matter arises when two models show similar GINI results, the χ^2-measure can make the difference.

The number of buckets should not be too low (e.g., one or two), because bucketing will wipe out small changes. A large number of buckets measures the fit very precisely, but has a disadvantage that it is not very stable. The samples in the buckets will be too small to get a reliable observed default rate. In the results presented in the next chapters, a χ^2-measure over six buckets is used. Additionally, a graph such as Figure 5.2 can give extra information about the individual buckets.

Chapter 6

Logistic Regression

6.1 Concepts of Logistic Regression

A simple and straightforward technique for predicting a binary variable (default entry) based on multiple continuous or Boolean variables is the binary logistic regression model. The probability p of a client entering the default (D) status is related to the variables ($X_1 \dots X_n$) as shown in the following equation.

$$ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n$$

The β's represent the parameters of the model. β_0 is called the intercept and the other β's the slope parameters. Such a model has the advantage of producing a probability between zero and unity, regardless of the input values. Rewriting the previous equation and using vectors to represent the parameters of the model as well as the input variables, the probability of default can be rewritten in a form that appears more explicit, as shown below.

$$p = \frac{\exp(\beta \cdot \mathbf{X})}{1 + \exp(\beta \cdot \mathbf{X})}$$

When providing the input variables (the vector $\mathbf{X}$) for a client to the model, its probability of default, p, should be obtained. Thus the fixed parameters of the model need to be evaluated, namely the values that make up vector β.

A discrete variable can not be parameterized by one β. For every possible state but one, a dummy Boolean variable has to be introduced instead. So a variable with four possible states will be replaced by three dummy variables, one for each state and the fourth state will be represented by the three variables set to false. In that case, only the intercept will have an effect. The number of degrees of freedom of a variable is the number of possible states minus one.

The event of entering the default status is considered to be a random variable, D, which can only be in two states ('1' for entry into default and '0' for non-entry). The probability of the event $d = 1$ to occur was defined earlier: it follows a logistic function, f, as defined in the equations of the previous section. Therefore:

$$
\begin{aligned}
P(d=1) = &\quad f(\beta \cdot \mathbf{X}) \\
P(d=0) = &\quad 1 - f(\beta \cdot \mathbf{X})
\end{aligned}
$$

Each observation in the sample used to build the model is assumed to be a single draw from a Bernoulli distribution with probability of success $f(\beta \cdot \mathbf{X})$. Because all observations are assumed to be independent in the data set, the joint probability that all observations occur can easily be calculated. Assuming N clients in the data set:

$$P(D_1 = d_1, D_2 = d_2, .., D_N = d_N) = \prod_{i=1}^{N} P(D_i = d_i)$$

This joint default probability, also called likelihood function (L) in the sense that it depends on the β's, can be rewritten as follows:

$$L = \prod_i \left[f(\beta \cdot \mathbf{X_i})\right]^{d_i} \cdot \left[1 - f(\beta \cdot \mathbf{X_i})\right]^{1-d_i}$$

The goal is to find the vector β that maximizes the likelihood function. In other words, those parameters of the model need to be found that maximize the probability of defaulted clients being recognized as such and non-defaulted clients being attributed a small probability of default. Using, for example, Newton's method with an appropriate software package, the vector β of those parameters that maximize the joint probability that all trials realize is found within a few iterations.

As mentioned before, the data described in the previous chapter will be used to develop a logistic regression model. The software tool used to calculate the parameters and the performance of the variables is SAS version 8.2. For further reading on logistic regression, [16] is strongly recommended.

6.2 Results

As mentioned before, variable performance in a logistic regression model can be measured at the time of modeling. The most common way to measure the performance of variable X_i in a logistic regression model is the residual χ^2 score. This χ^2 score statistic will test the hypothesis $H_0 : \beta_i = 0$ (the Wald test). In other words, the residual χ^2 score statistic represents the probability that variable X_i has no influence on the model. Consequently, values close to zero indicate high performance.

To get a first impression of the variable performance in a model, a 'stepwise logistic regression' is performed, which means that a model with one variable is built and one by one the other variables are included in the model while constantly monitoring the performance of the other variables. Based on the Wald test, the variable is either left in the model or removed automatically. The results are given in Table 6.1.

Table 6.1: Results of stepwise logistic regression

	DF	**Wald** χ^2	$P > \chi^2$
X_6	4	42.3728	$< .0001$
X_9	3	38.2647	$< .0001$
X_7	5	218.3	$< .0001$
X_2	7	546.394	$< .0001$
X_1	6	370.481	$< .0001$
X_4	6	23.0845	0.0008
X_3	3	52.4052	$< .0001$
X_{11}	10	62.1638	$< .0001$
X_8	4	10.9833	0.0268

In the table, DF denotes the number of degrees of freedom, which is the number of possible values of a discrete variable, minus one. Wald χ^2 is the χ^2 score and $P > \chi^2$ is the χ^2 score statistic derived from the χ^2 score and the number of degrees of freedom. Notice that X_8 is the worst variable, although all variables seem to perform well (none of the variables was removed automatically).

A couple of models were built and tested. Based on the results listed Table 6.2, it can be concluded that the optimal set of variables is $X_1, X_2, X_4, X_6, X_7, X_{11}$.

Table 6.2: The best logistic regression models

	X_2	X_1	X_3	X_4	X_6	X_7	X_9	X_{11}	**GINI**
Model 1	x	x	x	x	x	x	x	x	78.3%
Model 2	x	x	x		x	x	x	x	77.9%
Model 3	x	x		x	x	x	x	x	78.3%
Model 4	x	x			x	x	x	x	77.5%
Model 5	x	x		x	x	x		x	78.3%

These models have the best performance. Leaving other variables out causes the GINI coefficient to decrease. X_3 and X_4 are mutually exclusive and therefore the first model is not preferred. Fortunately, the GINI indexes of the first, third and fifth model are equally good, so either of them can be picked. The model with the least number of variables (Model 5) is preferred, due to stability.

The χ^2 score is 0.0023, which is very low (compared to other models). Figure 6.1 shows that the model fits the actual loss curve very well.

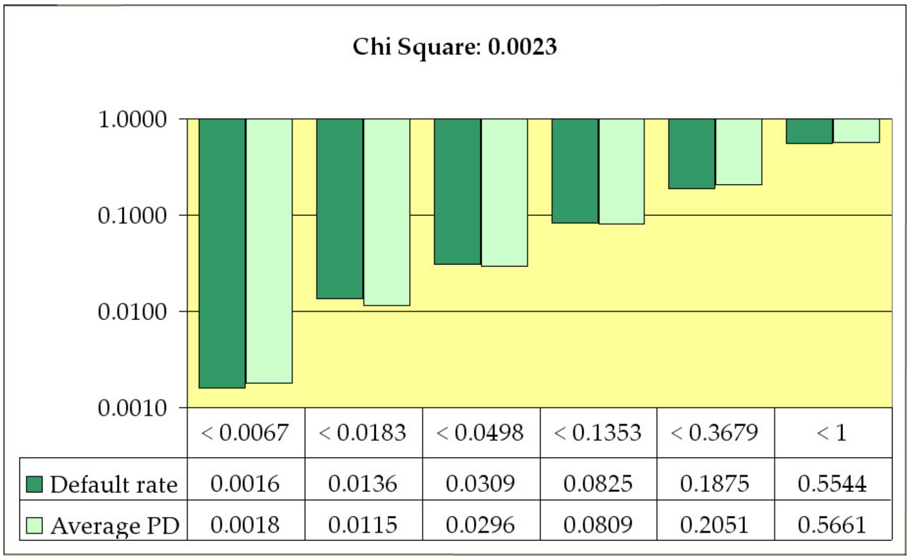


Figure 6.1: χ^2 for six categories: 0.0023

Apparently a logistic regression model behaves extremely well for this data set. Chapter 8 contains a more detailed discussion about logistic regression results.

6.3 Validation

Before a logistic regression model is accepted, two very important criteria have to be met. Firstly, for each discrete variable, every state except one has its own β parameter. The obtained β's should be compared with the univariate plot and should somehow expose a 'clear structure'. An example of such a comparison is given in Table 6.3.

Table 6.3: Parameters β do not make sense

	x_1^0	x_1^1	x_1^2	x_1^3	x_1^4	x_1^5
Default Rate	0.002	0.009	0.016	0.023	0.031	0.078
β_1^i	0.953	-0.482	-0.411	0.078	0.091	n.a.

Assume variable X_1 has 6 states, $[x_1^0, x_1^1, x_1^2, x_1^3, x_1^4, x_1^5]$. Every state but one has a parameter β_1^i. Now assume that the obtained β's and the default rates for each category are given. Notice that the pattern of the β's absolutely does not follow the default rate pattern in any way. This means the variable does not add extra power to the model. For this model the parameters might give good results, but for the next data set they will definitely not make sense, since they are based on coincidence. Therefore, such a pattern is an indication for an unstable model.

And secondly, the Wald test, performed on the intercept, should return a satisfying number (preferably > 0.001). Sometimes, one of the parameter estimates shows an extreme value. To correct for this, the intercept is also completely off. Typically, the Wald test then performed on the intercept returns very high values (sometimes close to 1). When this occurs the model is very unstable.

When both of the criteria above are met, there is no obvious sign for the model to be unstable.

Part II

Bayesian Networks Theory

Chapter 7

Bayesian Networks

7.1 Context

Many techniques for learning, such as logistic regression or neural networks, are very quantitative. In contrast, the knowledge contained in expert systems is usually based on qualitative linguistic knowledge. Bayesian networks (BN), also called belief networks, allow both to be incorporated in a single model. The representation allows to learn new knowledge by combining expert domain knowledge with statistical data.

Bayesian networks have proved to be very useful for diagnosis. Especially for medicine and troubleshooting, symptoms can be very accurately related to diseases or failures. In case of doubt, an observation of an extra feature can be suggested. One of the most famous BN applications is the Office Assistant in Microsoft Office.

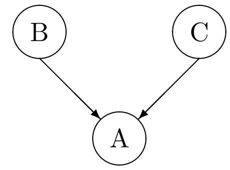
Bayesian networks have a couple of very convenient properties:

- Nodes in the model actually represent real life variables. Therefore, the structure of a Bayesian network is easy to understand, unlike a neural network. Especially in critical environments, acceptance of the model is very important.
- Calculation provides not only an expected value for a variable, but also the entire probability density function, which makes it possible to calculate standard deviation.

7.2 Concepts of Bayesian Networks

A BN is a directed acyclic graph (DAG) which represents a set of variables. Every node in the graph corresponds to a variable in the model. A direct causal relation between two variables is represented by an arc in the graph. Such a graph is easy to interpret, since arcs indicate conditional dependence and the absence of arcs mean conditional independence. In this thesis only discrete variables will be considered. Therefore variables will have states, not values.

$P(b) = 0.2$



$P(c) = 0.6$

$$P(a \mid b, c) = 0.3$$
$$P(a \mid \bar{b}, c) = 0.1$$
$$P(a \mid b, \bar{c}) = 0.8$$
$$P(a \mid \bar{b}, \bar{c}) = 0.5$$

Figure 7.1: Example of a BN

Figure 7.1 is a representation of a BN that models three binary variables: A, B, and C. The model assumes that A is dependent on B and C. B and C are assumed to be independent, since no arc is drawn between the two nodes.

Each state of a node has a specific probability of being observed, which is conditional on the states of the parent nodes. The conditional probabilities of a node are stored in a conditional probability table (CPT), as can be seen in Figure 7.1. The size of the CPT depends on the number of states of the node (s), the number of states of the parent nodes (s_p), and the number of parents (p) in the following way:

$$size\_CPT = s \cdot (s_p)^p$$

It is obvious that for a large number of parents the CPT will expand very rapidly. Learning techniques can be used to obtain the values of a CPT.

7.3 Inference

Given the CPT's of every node in the network, any conditional probability in the network can be calculated. This is called inference and is based on Bayes' Theorem. The theorem for two dependent variables states:

$$P(x \mid y) = \frac{P(y \mid x) \cdot P(x)}{P(y)}$$

The second theorem needed for inference is the expansion rule. In the equations below, the expansion rule is given for binary variables ($P(\bar{x}) = 1 - P(x)$):

$$\begin{aligned} P(x) =& P(x \mid y) \cdot P(y) + P(x \mid \bar{y}) \cdot P(\bar{y}) \\ P(x) =& P(x \mid yz) \cdot P(yz) + P(x \mid \bar{y}z) \cdot P(\bar{y}z) + \\ & P(x \mid y\bar{z}) \cdot P(y\bar{z}) + P(x \mid \bar{y}\bar{z}) \cdot P(\bar{y}\bar{z}) \\ =& \sum_Y \sum_Z P(x \mid yz) \cdot P(yz) \\ & etc.. \end{aligned}$$

Suppose in the example from Figure 7.1 the probability that C is true given that A is true needs to be calculated. The way this is done is:

$$
\begin{aligned}
P(c \mid a) &= \frac{P(a \mid c) \cdot P(c)}{P(a)} \\
&= \frac{(P(a \mid bc) \cdot P(b) + P(a \mid \bar{b}c) \cdot P(\bar{b})) \cdot P(c)}{\sum_B \sum_C (P(a \mid bc) \cdot P(bc))} \\
&= \frac{(P(a \mid bc) \cdot P(b) + P(a \mid \bar{b}c) \cdot P(\bar{b})) \cdot P(c)}{\sum_B \sum_C (P(a \mid bc) \cdot P(b) \cdot P(c))}
\end{aligned}
$$

First, Bayes' Theorem is applied. Then the marginal probabilities $P(a)$ and $P(a \mid c)$ are expanded using the expansion rule, taking into account the dependencies of the model. Finally, all possible cases of $P(bc)$ can be split in $P(b) \cdot P(c)$ due to independence. Now, all the values in the equation can be derived from the CPT's. Thus it will yield a probability of $\frac{3}{11}$.

Probabilistic inference was first addressed by Pearl in 1986, [25]. In 1988, Lauritzen and Spiegelhalter [21] proposed the well-known clustering algorithm. This algorithm constructs a join tree of cliques nodes. For each node in the join tree, the marginal can be calculated and from these marginals, the conditional probabilities can be derived.

7.4 Conditional Independence

Obviously, calculating the probabilities for an environment with n dependent variables will require 2^n calculations. For an environment with five variables, this is of course feasible, but for an environment with 40 variables it is a different story.

However, many variables are in fact conditionally independent, such as B and C in the example network of Figure 7.1. Therefore $P(b \mid ac) = P(b \mid a)$. Assume an environment with N variables, but each variable depends on at most r variables. Calculating the probabilities then only costs $N \cdot 2^r$, due to conditional independence. Note that this is only polynomial in N instead of exponential.

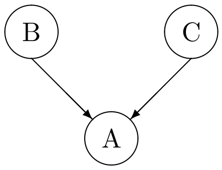


Figure 7.2: Converging connection

Variables with no direct connection by a vertex are not necessarily independent. In the example of Figure 7.1, B and C are independent if A is known, but if A is not known, they are dependent. Variables in a BN are independent if they are d-separated. D-separation can best be explained by distinguishing between three different types of connection.

Nodes B and C have a converging connection if they both have a vertex pointing to A, such as in Figure 7.2.

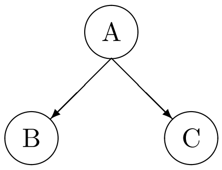


Figure 7.3: Diverging connection

Nodes B and C have a diverging connection if they both have a vertex pointing at them from A, such as in Figure 7.3.

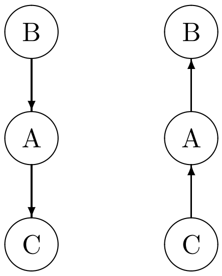


Figure 7.4: Serial connections, up and down

Nodes B and C have a serial connection if there is a path from B to C or from C to B through A, such as in Figure 7.4.

Definition 3 (D-separation)
Two nodes B and C are d-separated iff for every path between them, there is an intermediate variable A such that:

- *The connection is serial or diverging and A is known.*
- *The connection is converging and neither A nor any descendant of A is known.*

Definition 4 (D-connection)
Two nodes B and C are d-connected iff they are not d-separated.

Two variables in a Bayesian network are independent, if they are d-separated. This means, $P(c \mid ab) = P(c \mid a)$ in both the left and the right networks of Figure 7.4. Independent variables can reduce the computational complexity drastically.

Chapter 8

Simple Bayesian Networks: CPT Approach

8.1 Single Layer CPT Networks

The most straightforward way to apply a Bayesian network (BN) is by making the variable default, represented by node D, directly conditional on all predictive variables (a single layer BN), illustrated in Figure 8.1.

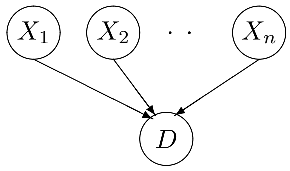


Figure 8.1: General type single layer BN

As the states of the parent nodes are always given, the interesting part of such a network is the conditional probability table (CPT) of variable D, hence the name CPT approach. The size of this CPT is dependent on the number of states of the parent nodes, and the number of parent nodes. For every possible combination of parent states, a probability distribution for D is given in the CPT.

When building a model, the objective is to obtain the CPT entries from the data. It is easy to see that for large CPT's a vast amount of data is needed to obtain accurate estimates. In this particular data set, there are six parent nodes, each with approximately seven possible states. In this case, the CPT will be very large (132.000 entries). Since the data set 'only' contains 50.000 records, the model will probably not be very stable. Most entries in the CPT will not even be observed once, and therefore contain very unreliable values.

The only way to shrink the CPT without reducing the number of parent variables is reducing the number of states of the parent variables. As the number of states is a decision of the model developer, this can be done. The first reduction in the data set resulted in an average number of six states for each parent. The CPT size was reduced drastically (only 32.000 entries), but it is still too big when developing a model with 50.000 observations.

Therefore, the number of parent states was again reduced (to an average of 3), yielding a CPT of only 672 entries. Now almost every entry in the CPT is observed at least once, but some values are still very inaccurate. Every entry in the CPT is a degree of freedom in the model. A CPT with 672 entries implies 672 degrees of freedom in the resulting model. This is a considerable amount for a development data set of 50.000 records.

Reducing the number of states of the parent nodes improves the performance initially, as will be shown later in this chapter. The trade-off is that the predictor variables are less accurate when the number of states is reduced. Taking this in consideration, the number of parent states will not be reduced to less than 672 states in total. Too much valuable information will be discarded if the number of states is reduced beyond that number.

GeNIe learning assigns Dirichlet (Beta) priors to every CPT entry and, after counting the number of occurrences, sets the posteriors accordingly. The distributions are discarded, and only the expected values of the probabilities are stored in the CPT.

8.2 Multi-Layer Networks

Although limiting the number of parent nodes will reduce the number of degrees of freedom, discarding variables is not an option. For many real world problems, the corresponding Bayesian Networks contain multiple layers. Having multiple layers reduces the maximum number of parents per node and, therefore, also the size of the CPT's.

Unfortunately, in the case of the customer default data, it is pointless to apply a multi-layer BN, which is illustrated by this section. Assume one of the predictive variables (X_k) is not directly connected to node D, but to nodes X_1 and X_2, which are also predictive variables. The corresponding graph structure is shown in Figure 8.2.

All the input variables are known. Now the PD is calculated regardless of X_k, since X_1 and X_2 are already known (=fixed). Therefore this network is essentially identical to the network in 8.1. For the calculation of the PD, the only variables that matter are the ones directly connected to it by a vertex. Vertexes between two input nodes, as shown in Figure 8.3, have no influence on the PD calculation and might therefore be left out too.

The other option to put extra layers in the model, is by creating artificial nodes (see Figure 8.4). This is a variable which is not on the input data set and therefore unknown.

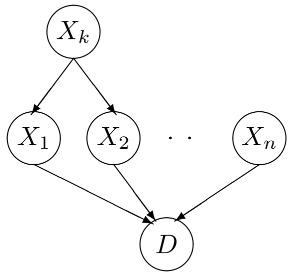


Figure 8.2: Multi-layer BN

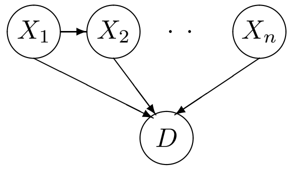


Figure 8.3: Bayesian Network with extra vertex

The problem with this network is training. Since there is no data available on the additional node it is not possible to learn from observations. In some particular cases there might be sufficient expert knowledge available to create reasonable conditional probabilities. Unfortunately for a PD model this was not possible. The knowledge we derived from experts did not add a lot to the available variables. Somehow they were not able to clarify matters on the available data, but they had very interesting comments about information which is not available in the systems, like someone's private financial statements or the reputation of a companies accountant.

8.3 Best Theoretical Model

For this specific modeling task, the requirements to be met are clearly defined by the input data set, which is constructed with a logistic regression model

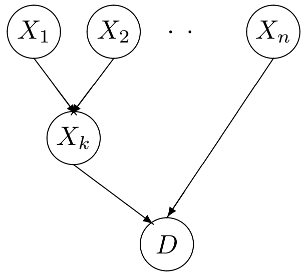


Figure 8.4: BBN with an artificial node representing a PD model

in mind. This section is a short glance at what might have been an optimal theoretical situation for building a BN for the probability of default model.

It is clear that a CPT for a node with many parents is extremely hard to calculate, based just on a data set. It is much more accurate to compute probabilities of underlying causes, given the observed predictive variables. Figure 8.5 illustrates the optimal BN approach for this problem.

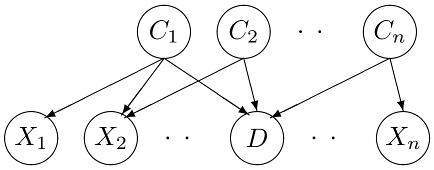


Figure 8.5: Optimal BN approach

In this graph, $X_1, \ldots, X_n$ are the input variables available in the data set. These variables can indicate a couple of underlying causes for default ($C_1, \ldots, C_n$). The causes have to be derived from experts, and of course the parameters (CPT entries) for the relations between the causes and the observed effects. Finally, the relations between the causes for default ($C_1, \ldots, C_n$), and the actual default event (D) need to be obtained from experts.

Apparently, the experts have to provide a lot of knowledge for the optimal Bayesian Network. As mentioned before, the opposite was the case. The experts provided whatever knowledge they had, but it could not be related to any data. Furthermore, estimates of numbers by the experts were very unreliable, since default rates are typically very small percentages, and the default events occur rarely.

At this stage it is very clear, that this problem (big data set, no expert knowledge) clearly can not make use of some of the BN advantages. However, in the next chapters a couple of modeling techniques using BN will be discussed, which can approximate the logistic regression results very closely.

Chapter 9

The Noisy-Max Approach

9.1 Noisy-OR

The key element in a noisy-OR model is, partly revealed by the name, a deterministic OR gate. Suppose a model with n binary input variables X_i, and a binary response variable Y. Binary variables have two states, $[0, 1]$. Presence of a variable X_i indicates an observation where $x_i = 1$, absence is defined as $x_i = 0$. The properties of a deterministic OR gate for these variables are:

$$P(y = 1 \mid x_1 \dots x_n) = \begin{cases} 0 & \text{if } \forall_i x_i = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Notice that this equation defines the CPT entries for node Y. The noisy-OR model follows the concept of a deterministic OR, but it assumes that presence of an input variable X_i can cause presence of Y with a probability c_i, modeled by an inhibitor node Z_i. Figure 9.1 illustrates the concept of a noisy-OR model.

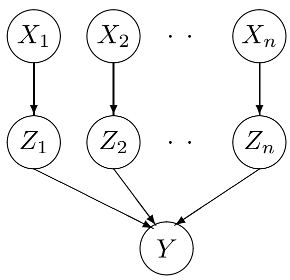


Figure 9.1: Noisy-OR BN

The CPT for node Y is identical to any deterministic OR table (Table 9.1). The CPT's of the inhibitor nodes $Z_1 \dots Z_n$ will resemble Table 9.2. Presence of X_i will cause presence of Z_i with probability c_i, which implies that absence of Z_i has probability $1 - c_i$. Absence of X_i will always imply absence of Z_i, hence the 0 and 1 in the right column of the CPT.

Table 9.1: CPT of node Y

$P(y \mid z_1 \dots z_n)$	$z_1 \dots z_n = 1$	$\dots$	$z_1 = 1, z_2 \dots z_n = 0$	$z_1 \dots z_n = 0$
$y = 1$	1	$\dots$	1	0
$y = 0$	0	$\dots$	0	1

Table 9.2: CPT of nodes $Z_1 \dots Z_n$

$P(z_i \mid x_i)$	$x_i = 1$	$x_i = 0$
$z_i = 1$	c_i	0
$z_i = 0$	$1 - c_i$	1

Basically, a noisy-OR model assumes the presence of any variable X_i to be sufficient to cause Y to be present. At the same time it guarantees absence of Y if all input variables are absent.

Inference in a noisy-OR model is performed as follows:

$$P(y = 1 \mid x_1 \dots x_n) = 1 - \prod_i (1 - c_i) \cdot x_i$$

Using this formula, the CPT for Y can be obtained straight from $X_1 \dots X_n$ and parameters $c_1 \dots c_n$. Table 9.3 shows how the conditional probabilities are calculated.

Table 9.3: CPT of node Y

$P(y \mid x_1 \dots x_n)$	$x_1 \dots x_n = 1$	$\dots$	$x_1 = 1, x_2 \dots x_n = 0$	$x_1 \dots x_n = 0$
$y = 1$	$1 - \prod_i (1 - c_i)$	$\dots$	c_1	0
$y = 0$	$\prod_i (1 - c_i)$	$\dots$	$1 - c_1$	1

9.2 Leaky Noisy-OR

A noisy-OR model is correct only if it satisfies the requirement that the possible causes are collectively exhaustive. This means that, aside from the causes included in the model, there should absolutely be no other possible causes for presence of Y (because $P(y = 1 \mid x_1 \dots x_n = 0) = 0$). For many real world situations, there are a multiple possible causes for presence of Y, some of which are impossible to include in the model. This illustrates the need for a parameter which models the probability of Y present, given the absence of $X_1 \dots X_n$.

Leaky noisy-OR is an extension to the noisy-OR model. In a leaky noisy-OR model, Y is assumed to have a certain probability c_L to be present, given the absence of all input variables X_i. An extra node Z_L is added to the structure of the network to incorporate this extra probability $P(y = 1 \mid x_1 \ldots x_n = 0)$. The leaky noisy-OR structure is illustrated in Figure 9.2.

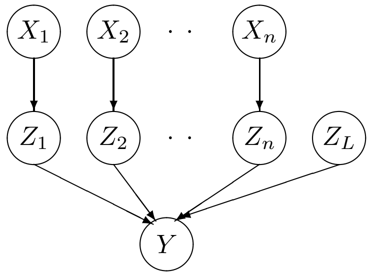


Figure 9.2: Leaky Noisy-OR BN

Table 9.4: CPT of node Y

$P(y \mid x_1 \ldots x_n)$	$x_1 \ldots x_n = 1$	$\ldots$	$x_1 = 1, x_2 \ldots x_n = 0$	$x_1 \ldots x_n = 0$
$y = 1$	$1 - (1 - c_L) \cdot \prod_i (1 - c_i)$	$\ldots$	$c_1 + (1 - c_1) \cdot c_L$	c_L
$y = 0$	$(1 - c_L) \cdot \prod_i (1 - c_i)$	$\ldots$	$(1 - c_1) \cdot (1 - c_L)$	$1 - c_L$

The CPT for node Y in a leaky noisy-OR model is given in Table 9.4. Notice that $P(y = 1 \mid x_1 = 1, x_2 \ldots x_n = 0) = c_1 + (1 - c_1) \cdot c_L$. This is not consistent with the way c_i was defined in a noisy-OR model without leak. In a leaky noisy-OR model, it is certainly not a convenient parameter to obtain from experts. Therefore, leaky noisy-OR parameters p_i are sometimes defined as:

$$p_i = c_i + (1 - c_i) \cdot c_L$$

9.3 Noisy-MAX

Up to this point, the input and response variable were always assumed to be binary. A fairly easy extension of this model can be obtained by changing the OR properties into MAX properties. A noisy-MAX model is based on a deterministic MAX gate. The output variable of a MAX gate is defined as:

$$y = \texttt{max}(x_1 \ldots x_n)$$

In a MAX gate, variables can have more than two states, which allows non-binary variables to be used, but there are a couple of requirements:

1. The variables should be ordinal, not nominal.
2. The variables should have an equal amount of states.

Obviously, the data set for this problem does not meet the second requirement. The response variable only has two states, while the number of states of the input variables is approximately seven. This inconvenience can be solved by making the number of states of inhibitor nodes $Z_1 \dots Z_n$ equal to the number of states of the response variable D (i.e. 2).

Technically, noisy-MAX is not applicable for this data set. In fact, the observed states of the input variables influence the presence of the inhibitor nodes, but beyond that point, the presence of Y (D in this case) is obtained by a deterministic OR. However, the number of states of the input variables are more than two, so the term noisy-OR is not appropriate. Therefore, this method will be referred to as noisy-MAX.

Table 9.5: Example of Noisy-MAX CPT for node Z_i

$P(z_i \mid x_j)$	$x_j = 2$	$x_j = 1$	$x_j = 0$
$z_i = 1$	0.7	0.4	0
$z_i = 0$	0.3	0.6	1

The noisy-MAX network is identical to the noisy-OR network. An example of a CPT for Z_i is shown in Table 9.5. This example is given to illustrate the requirements for a noisy-MAX CPT:

- The lowest value of x_i is always assumed to be the 'absent' state, therefore, Z_i is always absent if X_j is absent. The right column in such a CPT should contain a 1 at the bottom row, implying all other values in that column should be zero.
- $P(z_i = 1 \mid x_j = k) > P(z_i = 1 \mid x_j = l)$ if $k > l$. High values of X_j will more likely result in presence of Z than low values.

Instead of just one parameter c_i for each node, the CPT's contain multiple parameters. The number of parameters depends of course on the number of states of the input variable and the response variable.

Chapter 10

Naive Bayes

10.1 Concept of Naive Bayes

The credit rating problem is essentially similar to a binary classification problem. Many studies have focused on Bayesian network classifiers, e.g., [9, 10, 11, 24]. One particularly interesting conclusion is that naive Bayes models perform very well as a classifier.

The naive Bayes approach basically assumes that the observed input variables are dependent on the response variable. In other words: the cause variables and the result variable are switched. From a causal reasoning point of view this approach does not make sense. Nonetheless, the value of D still depends on the values of X_i.

The main criticism against usage of naive Bayes, is that it makes the very unrealistic assumption that the states of the predictive variables are independent given the state of the result variable. Despite this comment, naive Bayes performs surprisingly well in practice. Which may lead to the conclusion that the relevance of the assumption is overestimated.

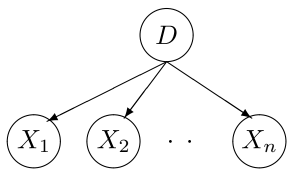


Figure 10.1: Naive Bayes network

Figure 10.1 shows a basic naive Bayes model. In this model the variables $X_1, \ldots, X_n$ are dependent on the state of D. Since D only has two states (1-Default and 0-Non Default), the CPT size of X_i is the number of states of X_i times two, which for most variables will be 14. Notice that the total number of degrees of freedom is the sum of the CPT size of each variable X_i: approximately 84.

Performing inference in a CPT network is very straightforward, since $P(d \mid x_1, \ldots, x_n)$ is exactly listed in the CPT of node D. In naive Bayes this is not the case, however, it can be calculated as follows:

$$
\begin{aligned}
P(d \mid x_1, \ldots, x_n) &= \frac{P(x_1, \ldots, x_n \mid d) \cdot P(d)}{P(x_1, \ldots, x_n)} \\
&= \frac{\prod_i P(x_i \mid d) \cdot P(d)}{\prod_i P(x_i)} \\
&= \frac{\prod_i P(x_i \mid d) \cdot P(d)}{\prod_i \left(P(x_i \mid d) \cdot P(d) + P(x_1 \mid \bar{d}) \cdot P(\bar{d})\right)}
\end{aligned}
$$

These values can all be obtained from the CPT's of the naive Bayes network. It is surprising how much similarity the equation listed above shows compared to the logistic regression calculation of $P(d)$:

$$
\begin{aligned}
P(d) &= \frac{\exp(\beta_0 + \beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n)}{1 + \exp(\beta_0 + \beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n)} \\
&= \frac{\exp(\beta_0) \cdot \exp(\beta_1 \cdot x_1) \cdot \ldots \cdot \exp(\beta_n \cdot x_n)}{1 + \exp(\beta_0) \cdot \exp(\beta_1 \cdot x_1) \cdot \ldots \cdot \exp(\beta_n \cdot x_n)} \\
&= \frac{\prod_j \exp(\beta_j \cdot x_j) \cdot \exp(\beta_0)}{1 + \prod_j \exp(\beta_j \cdot x_j) \cdot \exp(\beta_0)}
\end{aligned}
$$

Although both models have a different number of degrees of freedom (naive Bayes 84 and logistic regression 38), and the parameters are used differently (straight in naive Bayes, and as a power in logistic regression), the structure of the equation is identical. Both methods calculate the $P(d)$ as a product of all input parameters.

10.2 Tree Augmented Naive Bayes

Naive Bayes makes the assumption of conditional independence as mentioned before. In this portfolio this assumption is not realistic. This section describes an extension of naive Bayes, the tree augmented naive Bayes (TAN). For some classification problems, TAN outperforms naive Bayes [11].

Figure 10.2 illustrates the structure of a TAN. The predictive variables are still dependent on the response variable, however, there are also dependencies among the predictive variables. This eliminates the assumption of conditional independence. The additional vertices in the graph indicate correlations.

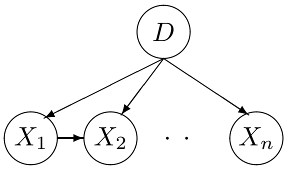


Figure 10.2: Tree augmented naive Bayes network

There are some restrictions to the additional vertices. Since a Bayesian network is defined as a directed acyclic graph (DAG), the addition of an extra correlation vertex should not jeopardize this property.

	X_1	X_2	X_4	X_6	X_7	X_{11}
X_1		-0.2717	0.1849	0.0598	0.0582	0.0264
X_2			-0.5456	0.1740	0.0692	-0.1727
X_4				0.2382	0.1364	0.2317
X_6					0.1286	-0.0169
X_7						0.0567
X_{11}						

[0, 0.05]
[0.10, 0.20]
[0.05, 0.10]
[0.20, 0.5]
[0.5, 1]

Figure 10.3: Correlation table

As mentioned before, the extra vertices represent correlation between input variables. From Figure 10.3 it follows that correlation exists between X_2 and X_4. Therefore, starting with a normal naive Bayes network, an additional vertex is added between X_2 and X_4.

From one correlation value, it is hard to determine in which direction the new vertex should be pointing. Therefore two TAN networks were built, TAN 1 (Figure 10.4), and TAN 2 (Figure 10.5).

In this study, no method was developed to search the space of all possible TAN's and find the optimal network, because is would extend the scope of the

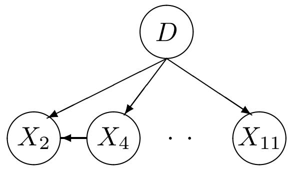


Figure 10.4: TAN 1: X_2 is dependent on X_4

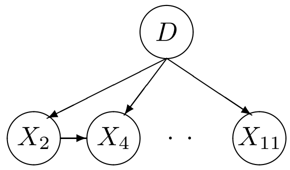


Figure 10.5: TAN 2: X_4 is dependent on X_2

project far too much. However, it is possible that some interesting results might be acquired with this approach.

Chapter 11

Overview

This chapter provides a brief overview with advantages and disadvantages to the different modeling techniques, illustrated in Table 11.1.

Table 11.1: Overview of modeling techniques

MODEL	Adv./Disadv.
Logistic Regression	+ Low number of degrees of freedom + Minimizes mean squared error - No correlation between input variables - Assumption of summation over input variables
CPT	+ Very accurate on individual observations + Well able to handle correlated input variables - Very high number of degrees of freedom - Needs a lot of input data
Noisy-MAX	+ Low number of degrees of freedom - Learning techniques are not straightforward - MAX assumption sometimes not correct
Naive-Bayes	+ Low number of degrees of freedom + TAN allows correlation between input variables - Assumption of conditional independence

Part III

Trends and developments in BN research

Chapter 12

Learning

Learning is a very useful property of Bayesian networks. Based on a data set of observations, the optimal model can be derived using learning techniques. Learning can provide the complete structure of the network, optimize a given network structure, or set the parameters for a given network structure.

12.1 Structural Learning

Structural learning techniques provide the best possible network structure given a data set of observations, O. Scoring functions evaluate the goodness of fit of a graph G. A scoring function assigns a score $S(G)$ to a graph. The goal is to find the structure with the best score, given the dataset. Typically, the scoring functions are the maximum likelihood of the structure or the posterior probability of the structure given the data.

Since the value of a node is only influenced by its direct parents, the local scoring functions of the nodes can be multiplied to obtain the total score.

$$S(G \mid O) = \prod_i S_L(X_i, Pa_i \mid O)$$

Changing one vertex at each move and calculating the gain can eventually result in the optimal network structure. Changing a vertex can imply addition, deletion, or reversion. If a vertex to X_i is added or deleted, only $S_L(X_i, Pa_i \mid O)$ needs to be revised. If a vertex between X_i and X_j is reversed, $S_L(X_i, Pa_i \mid O)$ and $S_L(X_j, Pa_j \mid O)$ need to be re-evaluated.

The task is to search the neighboring structures for the structure with the best $S(G \mid O)$. The number of possible network structures grows super-exponentially with the number of nodes. Finding the optimal structure is NP-hard. Therefore heuristics are typically used.

Greedy hill-climbing is a searching method, which only accepts a change in the structure if it increases the score. The problem with this approach is that a solution can get stuck in local optima.

Simulated annealing avoids local optima by accepting changes that increase the score and accepting changes that decrease the score with a certain probability. Simulated annealing does not guarantee an optimal structure, due to computational limits.

The methods described in this section are the common approach to structural learning. Many improvements on a scoring function or search method have been developed to gain computational advantage. This research does not focus on structural learning. Recommended papers on structural learning are [14, 15].

12.2 Parameter Learning

Each node in a BN represents a variable, and therefore contains a list with conditional probabilities, just like the example in the figure. The exact probability (physical probability) θ is not known, so the belief about θ is represented by a probability density function. Suppose there is a discrete variable, represented by a node with a probability 0.5, then Figure 12.1 could be the prior probability density curve $p(\theta)$.

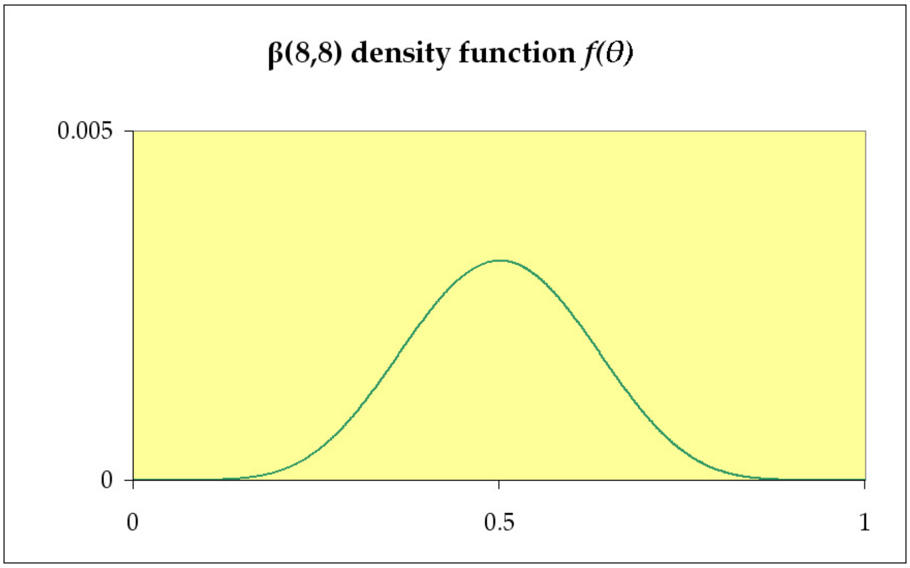


Figure 12.1: $P(\theta) = \beta(8,8)$

Suppose that an occurrence of 1 is observed. Then the probability is updated with $p(\theta|x=1) = c \cdot \theta \cdot p(\theta)$. The obtained new probability is given in Figure 12.2.

Notice that the posterior is shifted slightly to the right and is slightly narrower. Of course if a 0 is observed, the probability is updated by $p(\theta|x=1) = c \cdot (1-\theta) \cdot p(\theta)$. For six occurrences of 0 and two occurrences of 1, and for 60 occurrences of 0 and 20 occurrences of 1, the posterior probability density curves will respectively look like Figure 12.3 and 12.4.

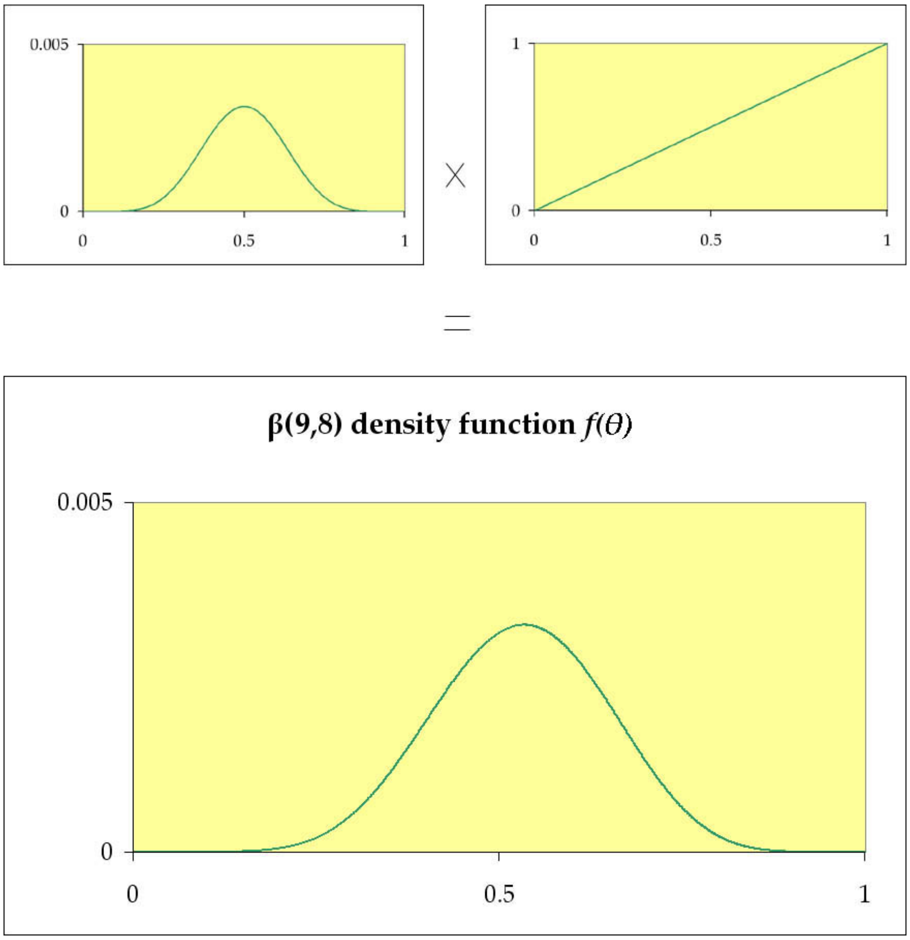


Figure 12.2: $P(\theta) = \beta(9, 8)$

Instead of updating the probability density function for each observation, it would be convenient to have a distribution where the parameters are in fact the number of events and nonevents. The beta distribution (for binomial variables) with parameters α and β has exactly this property.

$$P(\theta|\alpha\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(1-\theta)^{\beta-1}\theta^{\alpha-1}$$

$$\mu = \frac{\alpha}{\alpha+\beta} \quad , \quad \sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

Where α is the number of events + 1 and β is the number of nonevents +

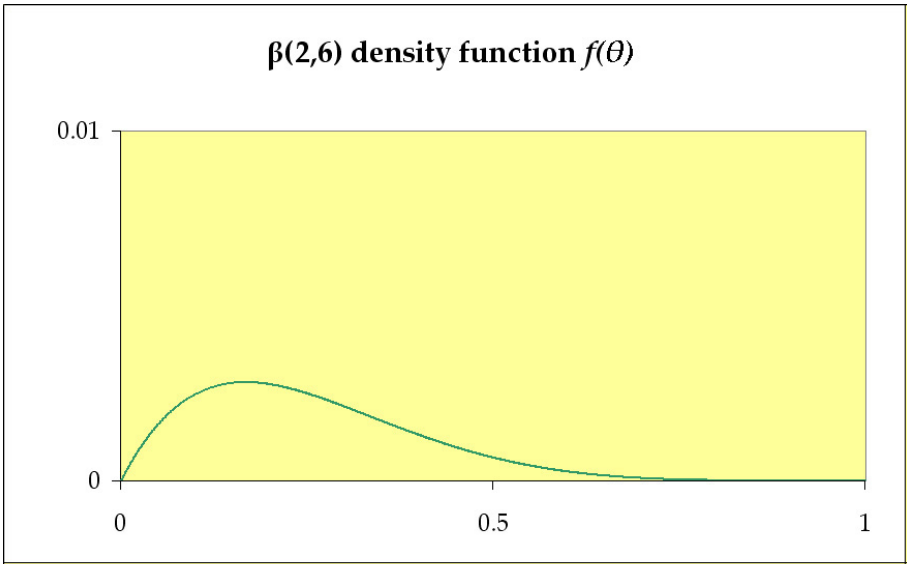


Figure 12.3: $P(\theta) = \beta(2,6)$

1. Also a prior can be introduced by setting the variables α and β before the learning process starts.

For multinomial variables, the Dirichlet distribution is used. For a variable with k states, the number of occurrences of state i is represented by α_i and the corresponding probability by θ_i. The Dirichlet distribution is defined as:

$$P(\theta_1\theta_2..\theta_k|\alpha_1\alpha_2..\alpha_k) = \frac{\Gamma\left(\sum_k \alpha_i\right)}{\prod_k \Gamma\left(\alpha_i\right)} \prod_k \theta_i^{\alpha_i - 1} \qquad \sum \theta_i = 1$$

So the learning process now comes down to counting the number of observations of each state. Then the mean and variance can be calculated using the formulas above.

The goal is to increase the confidence about probabilities by feeding data to the network. Updating of the probability distributions is called learning.

Updating can be performed for each node, so the network is able to converge to a very accurate model. It should be stated however that the better the prior estimate, the better the results.

12.3 Learning Noisy-MAX Parameters

CPT and naive Bayes models make use of the techniques described in the previous section. Noisy-MAX parameter learning is a bit more complex.

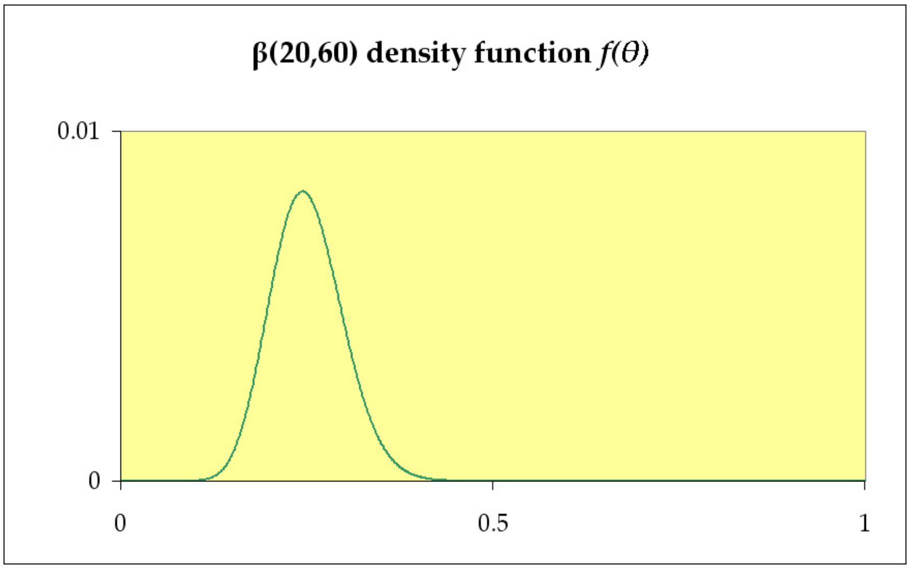


Figure 12.4: $P(\theta) = \beta(20, 60)$

Once the structure of the model is defined, learning the parameters for a noisy-MAX model imposes a problem. For each record, the states of the variables $X_i \ldots X_n$ and D are known. However, it is impossible to retrieve the states of $Z_1 \ldots Z_n$. In other words, it is impossible to determine which inhibitor node caused the default if there is more than one X_i present. For records where all X_i are absent or just one X_i is present, or D is absent, retrieving the states of Z_i is not a problem.

There are many ways to obtain parameters for a noisy-MAX model. The most straightforward method is proposed by Onisko and Druzdzel [23]. They apply parameter updating using all records, except the ones where D is present and more than one X_i is present. For data sets with rare occurrences of D this eventually leads to a very small sample of discarded records, but these records are very interesting. Notice that by discarding these records, the variables are assumed to be conditionally independent. This method will not be tested in this research, since a completely new method was used to learn Noisy-MAX parameters.

The framework of this new method was proposed by Adam Zagorecki and further developed and fine tuned in this research. This method will be referred to as CPT approximation, since it is based on the CPT of a 'normal' BN. A CPT such as in chapter 8 is filled with probabilities learned from the data set. When inference is performed on this CPT, it will yield the value of the corresponding probability. On a Noisy-MAX network, one can also perform inference. When the two values are compared, the difference should be small. CPT approximation is essentially finding the noisy-MAX parameters which will minimize the difference between CPT inference and noisy-MAX inference.

This method is interesting in a couple of ways. Firstly, it reduces the number of degrees of freedom compared to the CPT approach. This will benefit the model definitely. But secondly, it takes the exact properties of the CPT into account when calculating the parameters. This causes the resulting model to resemble the CPT as close as possible, but without the instability.

The resemblance between the CPT and the noisy-MAX parameters can be expressed using two distance measures:

- Euclidean distance
- Kullback-Leibler (KL) divergence

The Euclidean distance between two vectors is defined as the square root over the sum of squared differences between the two values. A definition for the Euclidean distance between two CPT's is given below:

Definition 5 (Euclidean distance)
The Euclidean distance D_E between two CPT's A and B is defined as the sum of the Euclidean distances of all instances of $P_A(Y \mid \mathbf{X})$ and $P_B(Y \mid \mathbf{X})$:

$$D_E(P_A(Y \mid \mathbf{X}), P_B(Y \mid \mathbf{X})) = \sum_i \sum_j \left(P_A\left(y_j \mid \mathbf{x_i}\right) - P_B\left(y_j \mid \mathbf{x_i}\right)\right)^2$$

Euclidean distance is relatively insensitive to small probabilities and tends to favor the larger probabilities. For small probabilities, the Kullback-Leibler (KL) divergence is more appropriate.

Definition 6 (KL distance)
The KL distance D_{KL} between two CPT's A and B is defined as the sum of KL distances of all instances of $P_A(Y \mid \mathbf{X})$ and $P_B(Y \mid \mathbf{X})$:

$$D_{KL}(P_A(Y \mid \mathbf{X}), P_B(Y \mid \mathbf{X})) = \sum_i \sum_j P_A\left(y_j \mid \mathbf{x_i}\right) \ln \frac{P_A\left(y_j \mid \mathbf{x_i}\right)}{P_B\left(y_j \mid \mathbf{x_i}\right)}$$

The KL distance boosts the effect of small probabilities and is therefore very appropriate for this portfolio. The disadvantage of KL is that it can not handle zero values. Since zero values are very common in a CPT, it will become clear that using KL distance provides worse results than Euclidean distance.

In order to tackle the problem of calculating the distance between a value and a zero, cutting was introduced. When an entry in the CPT was obtained from less than c values, it is not taken into account when the distance is calculated. This means that only the frequent observations will determine the fit of the noisy-MAX model. In practice, cutting improves the results when KL distance is used.

Efficient CPT calculation from a noisy-MAX model is implemented for this research in an algorithm which is proved to converge to a global minimum when the Euclidean distance is used. There is no proof yet that the KL distance will converge to a global minimum.

First, θ_{ij}^{CPT} are defined as the original CPT entries, for the i-th combination of parents states and the j-th state of Y. θ_{ij}^{MAX} are the noisy-MAX based CPT entries. The algorithm finds those noisy-MAX parameters that minimize:

$$\sum_{i,j} \left(\theta_{ij}^{CPT} - \theta_{ij}^{MAX}\right)^2$$

Now the entries of a cumulative CPT are defined as:

$$\Theta_{ij} = \begin{cases} \sum_{k=1}^{j} \theta_{ik}^{CPT} & \text{if } j \neq 0, \\ 0 & \text{if } j = 0. \end{cases}$$

Notice that $\theta_{ij} = \Theta_{ij} - \Theta_{i(j-1)}$. Each inhibitor node Z_i in the noisy-MAX network has a CPT with values q_{ijk} where j is the state of X_i and k is the state of Z_i. The cumulative noisy-MAX probability is defined as:

$$Q_{ijk} = \begin{cases} \sum_{l=1}^{k} q_{ijl} & \text{if } k \neq 0, \\ 0 & \text{if } k = 0. \end{cases}$$

An efficient way of calculating Θ_{ij} is:

$$\Theta_{ij} = \prod_{x_p^r \in \mathbf{X}} Q_{prj}$$

Now the θ_{ij}^{CPT} parameters can be calculated using the formulas above:

$$\begin{aligned} \theta_{ij}^{CPT} &= \Theta_{ij} - \Theta_{i(j-1)} \\ &= \prod_{x_p^r \in \mathbf{X}} Q_{prj} - \prod_{x_p^r \in \mathbf{X}} Q_{pr(j-1)} \\ &= \prod_{x_p^r \in \mathbf{X}} \sum_{k=1}^{j} q_{prk} - \prod_{x_p^r \in \mathbf{X}} \sum_{k=1}^{j-1} q_{prk} \end{aligned}$$

The Euclidean distance is now defined as:

$$\sum_{i} \sum_{j} \left(\theta_{ij}^{CPT} - \left(\prod_{x_p^r \in \mathbf{X}} \sum_{k=1}^{j} q_{prk} - \prod_{x_p^r \in \mathbf{X}} \sum_{k=1}^{j-1} q_{prk} \right) \right)^2$$

To minimize the Euclidean distance, hill-climbing is used. Small additions and subtractions to individual noisy-MAX parameters are made (procedure *ChangeMAX*)if the distance is reduced by that change. Sometimes, adjustments to other parameters are necessary to meet the requirements of the rules of probability. The distance is calculated, using the equation above, by the *CalculateDistance* procedure.

Given a CPT with parameters C, the algorithm finds a set of noisy-MAX parameters M^*, such that distance $d^* < \epsilon$.

Algorithm 1 Noisy-MAX parameters from CPT

```
M* ← initialize
step ← initialize
d* ← CalculateDistance(M*, C)
repeat
  M ← M*
  d ← d*
  m* ← NULL
  for all m_i* ϵ M* do
    M ← ChangeMAX(m_I*, M*, step)
    D ← CalculateDistance(M, C)
    if d < d* then
      d* ← d
      m* ← m_i*
      step* ← step
    end if
  end for
  if m* ≠ NULL then
    M* ← ChangeMAX(m*, M*, step*)
  else
    Decrease(step)
  end if
until d* < ϵ
```

The algorithm guarantees a global minimum, if Euclidean distance is used. Obviously, execution time depends on the size of the CPT. Tests with a CPT of 132.000 entries required only minutes on a PC.

There is no proof that the Kullback-Leibler distance converges to a global minimum. Testing confirms this, since the results are sometimes stuck in a local minimum.

Probably the most widely applied method is the expectation maximization (EM) learning method. Given a data set S with observations $O_i = (\mathbf{x_i}, d_i)$, the EM algorithm maximizes

$$\sum_i \log P(d_i \mid \mathbf{x_i})$$

EM usually returns very good results. Using EM for a Noisy-MAX network yields results similar to logistic regression [31]. The EM algorithm was implemented with Adam Zagorecki was used to test this learning approach on this portfolio. The results are surprisingly good compared to the CPT approximation.

Chapter 13

Setting Priors and Learning Posteriors

The CPT models did not perform well due to the instability. This instability is caused by the rare occurrences of many CPT entries. When inference is performed on such a CPT, it will not return a sound value, because the occurrence may never have been observed.

As mentioned before, the common learning procedure in Bayesian networks is updating β-distributions. The observation will transform the prior probability (or probability distribution) into a posterior probability. This updating process can be done at once, or in several stages.

Up to this point, the initial probabilities in a CPT were always assumed to be 0.5 before any observation, and α/β after some observations, where α is the number of observed defaults, and β is the number of non-defaults.

This chapter describes an approach where the prior probabilities of a CPT will be used to gain stability in a model. There are several ways to set priors for a CPT:

- Uniform priors
- Noisy-MAX priors
- Naive Bayes priors

and several ways to merge the priors with the posteriors:

- Cutting the priors
- Weighing the priors

First, the prior assessment methods will be discussed and then the different merging approaches.

13.1 Prior Assessment

There are many ways to invent priors for a CPT, in this research, only a couple were tested. The most straightforward prior is the average default rate of the portfolio. Instead of initializing the CPT with probabilities 0.5, all possible input variable combinations get a probability of 0.02 assigned. Obviously, equal values for all cases does not make sense, but at least it will prevent the model from assigning $P(d = 1 \mid x_1 \dots x_n) = 0.5$ to all instances where $x_1 \dots x_n$ has not been observed before.

The second prior assessment technique is building a Noisy-MAX network and learning the parameters of that network from the data set. Then perform inference for every input combination $P(d = 1 \mid x_1 \dots x_n)$ and assign those values to the CPT entries.

This approach benefits from the low number of degrees of freedom in the noisy-MAX approach, which guarantees a stable network, and then polishes the values by initiating an extra learning cycle for the CPT network. Rare occurrences can rely on the stable prior, and frequent occurrences can be fine tuned according to the data.

The third approach is basically identical to the second, except instead of noisy-MAX priors, naive Bayes priors are used. This is very doubtful, since the causal relations in naive Bayes are completely opposite to a CPT network. However inference on both networks yields exactly the same result: $P(d = 1 \mid x_1 \dots x_n)$.

Since the naive Bayes results are better than the noisy-MAX results, the naive Bayes priors will probably outperform the noisy-MAX priors too. This approach has not been applied before in the literature.

13.2 Merging Priors and Posteriors

Merging of priors and posteriors can be handled in a couple of ways. Two methods will be discussed here, one is based on a minimum cut value, the other on weighing.

Application of a cut value is described in [23]. This approach is referred to as smoothing, but this does not refer to the transition from prior to posterior, and therefore, the name is somewhat confusing. The approach is very straightforward and comes down to this: If an occurrence of $x_1 \dots x_n$ is observed less than a certain number of times w, the prior will remain unchanged. If the number of occurrences of $x_1 \dots x_n$ exceeds w, then the prior will be discarded, and only the probability learned from the data will be stored in the CPT.

The results of cut value application depend on the value of w. For high values the model will rely mostly on the priors, which in many cases will be very stable. The model will probably have a better fit for low values of w, but the estimates could be less reliable.

Weighing is the second method for merging priors and posteriors, used in this research. Aside from assigning each conditional probability a value, also a weight is assigned to the priors, which is the same for every entry. The weight indicates the number of observations from which the prior is derived.

Assume a prior probability $P_{rior}(d = 1 \mid x_i) = 0.015$ and a weight (w) of 20. If the data set contains a record (d, x_i) then the posterior probability will be calculated as follows as follows:

$$P_{ost}(d = 1 \mid x_i) = \frac{(w \cdot P_{rior}(d = 1 \mid x_i)) + d}{w + 1}$$

Weighting allows a smooth transition between priors and posteriors. Unlike a cut value, the influence of the posterior will gradually increase when the number of observations increases. Similar to a cut value, large values for w will result in models where the prior is very prominent, while a small w allows more observation influence.

Chapter 14

Tools

This chapter describes the tools used to produce the results. Some existing software packages were used and some tools were developed solely for this research.

For the construction of the data set, the correlation calculations and the variable selection process, SAS 8.2 was used, along with Microsoft Excel. The logistic regressions were also performed using SAS 8.2.

14.1 Bayesian Networks Tool

The first CPT networks were built using a custom tool, built in C++. GeNIe is obviously a better alternative, but it does not provide an official learning functionality yet. Furthermore, programming the basic data structures, inference and learning algorithms gives a clear understanding to Bayesian networks.

CPT networks are as straightforward as possible in terms of inference. The probabilities $P(D = 1 \mid x_1 \ldots x_n)$ can be derived from the CPT right away. Therefore, no inference mechanism is needed for simple CPT networks. However, the tool also provides inference for naive Bayes networks.

Inference in naive Bayes is also straightforward, though not as simple as for CPT. The basic formula for inference in a naive Bayes network (not for a TAN) is:

$$
\begin{aligned}
P(d \mid x_1, \ldots, x_n) &= \frac{P(x_1, \ldots, x_n \mid d) \cdot P(d)}{P(x_1, \ldots, x_n)} \\
&= \frac{\prod_i P(x_i \mid d) \cdot P(d)}{\prod_i P(x_i)} \\
&= \frac{\prod_i P(x_i \mid d) \cdot P(d)}{\prod_i \left(P(x_i \mid d) \cdot P(d) + P(x_1 \mid \bar{d}) \cdot P(\bar{d})\right)}
\end{aligned}
$$

Aside from a probability estimate, the tool also models the actual probability distribution. Therefore, the standard deviation is also a possible output.

Finally, the tool can be used for sampling. The only supported distribution is the β distribution. Sampling from a β distribution is very easy. Assume a β distribution with parameters α and β. A sample θ from this distribution can be acquired as follows:

Algorithm 2 Sampling from a $\beta(\alpha, \beta)$ distribution

generate $\alpha + \beta + 1$ random numbers between 0 and 1
sort the numbers in ascending order
$\theta \Leftarrow$the $[\alpha + 1]$-th number

This algorithm is described and proved in [28]. This research gives an extension to this sampling algorithm for Dirichlet distributions. A Dirichlet distribution is the generalization of a β distribution, because the number of states is not limited to two.

Assume a multinomial discrete variable X with possible values $\{x_1, x_2, ..x_k\}$. A number of occurrences $\{\alpha_1, \alpha_2, ..\alpha_k\}$ of each value is observed. Now assume the probabilities $\pi_i = P\{X = x_i\}$ for $i = 1..k$ are Dirichlet distributed with parameters $\alpha_1, \alpha_2, ..\alpha_k$. Estimates for π_i can be obtained by sampling values θ_i from a Dirichlet distribution.

$$f_{\pi_1,\pi_2,..\pi_k|\alpha_1,\alpha_2,..\alpha_k} = \frac{(\sum\limits_k \alpha_i)!}{\prod\limits_k \alpha_i!} \cdot \prod_k \pi_i^{\alpha_i} \quad , \quad \sum \pi_i = 1$$

First, the Dirichlet sampling algorithm is given, followed by a proof that a sample from a Dirichlet distribution is essentially a number of samples from β distributions.

Algorithm 3 Sampling from a Dirichlet distribution

$i \Leftarrow 1$
repeat
  generate $1 + \sum \alpha_j$ random numbers between 0 and 1
  sort the numbers in ascending order
  $\theta_i \Leftarrow (1 - \sum_{j=1}^{i-1} \theta_j) \cdot$ (the $[\alpha_i + 1]$-th number)
  $i \Leftarrow i + 1$
until $i = k$

The complexity of the algorithm can be reduced by reusing the random numbers for the next iteration. This will not affect the samples, since the remaining random numbers will be completely random over exactly the interval $[(\sum_{j=1}^{i-1} \theta_j), 1]$.

Theorem 1
$\pi_1 = \theta$ with $0 \leq \theta \leq 1$. The distribution $f_{\pi_2,\pi_3,..\pi_k|\alpha_1,\alpha_2,..\alpha_k,\pi_1=\theta}$ is Dirichlet.

Algorithm 4 Simple sampling from a Dirichlet distribution

$i \Leftarrow 1$
generate $1 + \sum \alpha_j$ random numbers between 0 and 1
sort the numbers in ascending order
repeat
  $\theta_i \Leftarrow$(the $[\alpha_i + 1]$-th number) $-(\sum_{j=1}^{i-1} \theta_j)$
  $i \Leftarrow i + 1$
until $i = k$

Proof 1

$$
\begin{aligned}
f_{\pi_2,\pi_3,..\pi_k|\alpha,\pi_1=\theta} &= \frac{(\sum\limits_k \alpha_i)!}{\prod\limits_k \alpha_i!} \cdot \theta^{\alpha_1} \cdot \prod_{2..k} \pi_i^{\alpha_i} \qquad\qquad \sum \pi_i = 1 \\
&= \frac{(\sum\limits_k \alpha_i)!}{\alpha_1! \cdot (\sum\limits_{2..k} \alpha_i)!} \cdot \theta^{\alpha_1} \cdot \frac{(\sum\limits_{2..k} \alpha_i)!}{\prod\limits_{2..k} \alpha_i!} \cdot \prod_{2..k-1} \pi_i^{\alpha_i} \cdot (1 - \theta - \sum_{2..k-1} \pi_i)^{\alpha_k} \\
&= \left[\frac{(\sum\limits_k \alpha_i)!}{\alpha_1! \cdot (\sum\limits_{2..k} \alpha_i)!} \cdot \theta^{\alpha_1} \cdot (1-\theta)^{\sum\limits_{2..k} \alpha_i} \right] \\
&\quad \times \left[\frac{(\sum\limits_{2..k} \alpha_i)!}{\prod\limits_{2..k} \alpha_i!} \cdot \left(\frac{1}{1-\theta}\right)^{\sum\limits_{2..k} \alpha_i} \cdot \prod_{2..k-1} \pi_i^{\alpha_i} \cdot (1 - \theta - \sum_{2..k-1} \pi_i)^{\alpha_k} \right] \\
&= \left[\frac{(\sum\limits_k \alpha_i)!}{\alpha_1! \cdot (\sum\limits_{2..k} \alpha_i)!} \cdot \theta^{\alpha_1} \cdot (1-\theta)^{\sum\limits_{2..k} \alpha_i} \right] \\
&\quad \times \left[\frac{(\sum\limits_{2..k} \alpha_i)!}{\prod\limits_{2..k} \alpha_i!} \cdot \prod_{2..k-1} \left(\frac{\pi_i}{1-\theta}\right)^{\alpha_i} \cdot \left(\frac{1 - \theta - \sum\limits_{2..k-1} \pi_i}{1-\theta}\right)^{\alpha_k} \right] \\
&= \left[\frac{(\sum\limits_k \alpha_i)!}{\alpha_1! \cdot (\sum\limits_{2..k} \alpha_i)!} \cdot \theta^{\alpha_1} \cdot (1-\theta)^{\sum\limits_{2..k} \alpha_i} \right] \\
&\quad \times \left[\frac{(\sum\limits_{2..k} \alpha_i)!}{\prod\limits_{2..k} \alpha_i!} \cdot \prod_{2..k-1} \left(\frac{\pi_i}{1-\theta}\right)^{\alpha_i} \cdot \left(1 - \sum_{2..k-1} \left(\frac{\pi_i}{1-\theta}\right)\right)^{\alpha_k} \right] \\
&= f_{\pi_1=\theta} \cdot f_Z
\end{aligned}
$$

$f_{\pi_1=\theta}$ is a β distribution with parameters $\alpha = \alpha_1$ and $\beta = \sum_{2..k} \alpha_i$ and f_Z is a Dirichlet distribution with $k-1$ parameters, where $Z = \pi_i/(1-\theta)$. ■

Therefore, when sampling from a Dirichlet distribution, each π_i can be sampled from a $\beta(\alpha_i, \sum_{i+1..k} \alpha_j)$ distribution. The remaining parameters are Dirichlet distributed over the interval $[0, (1 - \sum_{j=1}^{i-1} \theta_j)]$.

The basic functionality of the tool is illustrated in Figure 14.1. The network structure is not among the input, since it can only be coded (and therefore fixed) in the program.

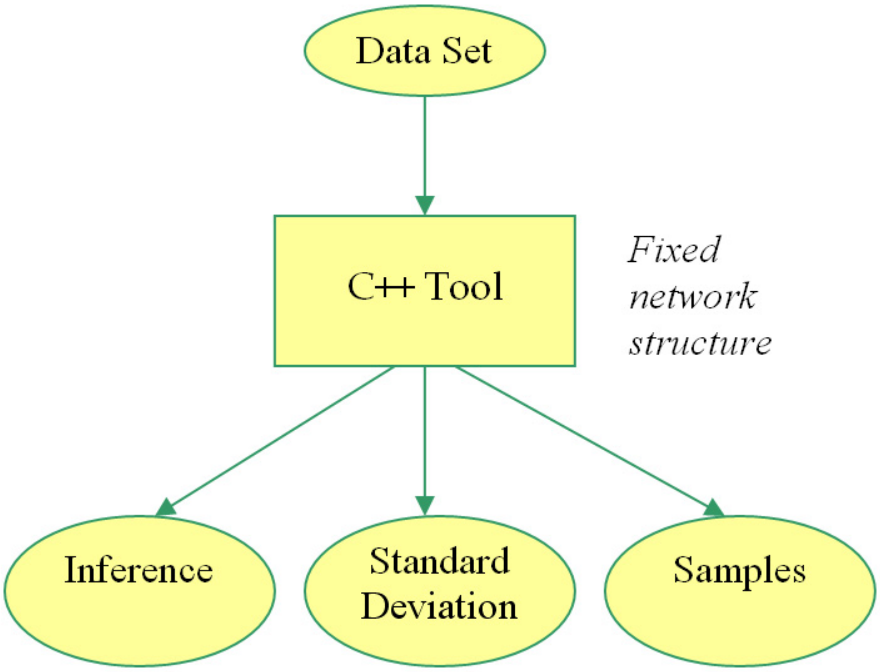


Figure 14.1: Structure of the BN C++ tool

Not only straight parameter learning is possible with this tool, but also the prior/posterior learning techniques, since GeNIe does not include this facility. Suppose $\theta_{x_1 \ldots x_N}$ is the number of occurrences $(y, x_1 \ldots x_n)$ divided by the number of occurrences $(x_1 \ldots x_n)$, or $\sum_Y (y, x_1 \ldots x_n)$. In general, if for a variable$X$, with states $x_1 \ldots x_N$, $\mathbf{p_i}$ is the i-th combination of parent states, then θ_{ij} denotes the probability of x_j given $\mathbf{p_i}$. Let $o(\mathbf{p_i}, x_j)$ be the number of observations $(\mathbf{p_i}, x_j)$ in the data set. θ_{ij} can now be calculated as follows:

$$\theta_{ij} = \frac{o(\mathbf{p_i}, x_j)}{\sum_{k=1}^{N} o(\mathbf{p_i}, x_k)}$$

In this research, a gentle transition is proposed, by weighting the priors. The priors will receive an integer value, weight w, which is equivalent to a number of observations. The resulting CPT entries will be a weighted average of the priors and the posteriors. Let the prior for θ_{ij} be p_{ij}, the posterior is then calculated as:

$$\theta_{ij} = \frac{(w \cdot p_{ij}) + o(\mathbf{p_i}, x_j)}{w + \sum_{k=1}^{j} o(\mathbf{p_i}, x_k)}$$

The priors (p_{ij}) are obtained by inference in a naive Bayes network. The formula for calculation of $p(y \mid \mathbf{p_i})$ in a network with predictive variables $X_1 \dots X_N$ and response variable Y with states $y_1 \dots y_M$ is given:

$$\begin{aligned} p_{ij} = p(y \mid \mathbf{p_i}) &= \frac{p(\mathbf{p_i} \mid y) \cdot p(y)}{p(\mathbf{p_i})} \\ &= \frac{\prod\limits_i p(x_i \mid y) \cdot p(y)}{\prod\limits_i p(x_i)} \\ &= \frac{\prod\limits_i p(x_i \mid y) \cdot p(y)}{\prod\limits_i \sum\limits_j (p(x_i \mid y_j) \cdot p(y_j))} \end{aligned}$$

Combining these two equation yields:

$$\theta_{ij} = \frac{\left(w \cdot \frac{\prod_i p(x_i|y) \cdot p(y)}{\prod_i \sum_j (p(x_i|y_j) \cdot p(y_j))} \right) + o(\mathbf{p_i}, x_j)}{w + \sum\limits_{k=1}^{j} o(\mathbf{p_i}, x_k)}$$

This type of learning is also implemented in the tool.

14.2 Cross-Validation and Scoring Tool

The second piece of software built for this research is a cross-validation and scoring tool. This is also a C++ program, which uses SMILE libraries. The functionality of this tool is to perform cross validation on an indexed data set, given a model, and calculate the GINI index and χ^2-measure. Indexed means each record contains an additional number indicating in which sample the record belongs. This was to make sure that for every model, exactly the same cross-validation was applied. The functionality comes in two flavors:

- The required input is only a data set and the structure of a BN model, no parameters. The program will perform the learning, create 5 models, cross validate the data set, and output the scores.
- The input is a data set, and 5 BN models, with the right parameters. Now the program will only perform cross validation on the data set and output the scores.

For some structures, the learning could be performed using GeNIe. However, GeNIe learning does not offer prior/posterior learning. Therefore the functionality was later extended to handle learning too.

The file format of the BN model is the standard GeNIe/SMILE format, a .dsl-file, to allow the usage of SMILE inference algorithms. The structure of this cross-validation program is illustrated in Figure 14.2.

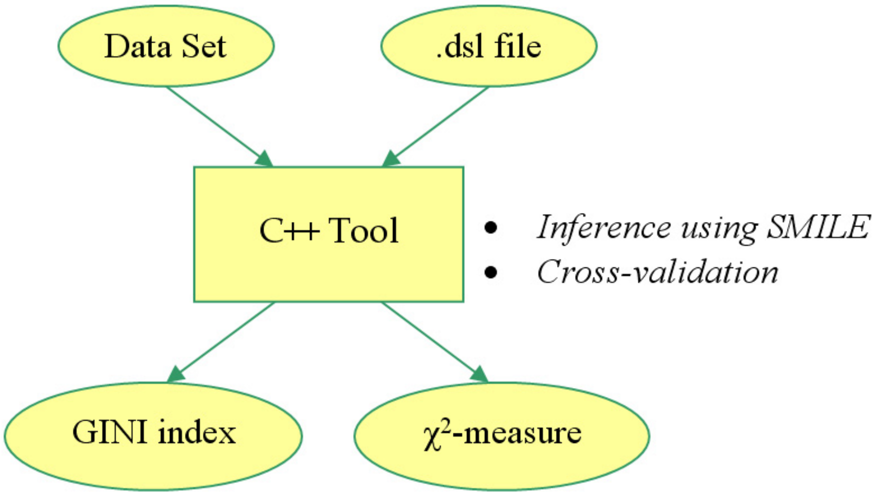


Figure 14.2: Structure of the cross-validation tool

14.3 GeNIe and SMILE

The Decision Systems Laboratory (DSL) in Pittsburgh (where part of this research was performed) has developed one of the most advanced software packages for construction and interaction with probabilistic models. Especially for Bayesian networks, these tools are more advanced than many commercial packages.

SMILE (Structural Modeling, Inference, and Learning Engine) is a fully platform independent library of C++ classes implementing graphical probabilistic and decision-theoretic models, such as Bayesian networks, influence diagrams, and structural equation models. SMILE provides a very convenient set of classes for probabilistic models and a wide variety of functionality, such as learning, inference and sampling algorithms. Its libraries are easily accessible to other C++ applications and is has been successfully used for many years.

GeNIe (Graphical Network Interface) provides a very convenient interface to SMILE, where graphical models can be constructed in a simple drag-and-drop interface. Models constructed in GeNIe can be accessed via the SMILE libraries.

GeNIe does not provide a learning tool yet. However, for this research an incomplete version of a beta release was used in some cases to do parameter learning. For CPT models and naive Bayes, this resulted in good parameters. Noisy-MAX and prior/posterior learning is not possible with this module, therefore own code was used to do the job.

Figure 14.3 shows a screen shot of GeNIe 2.0, a very advanced modeling tool.

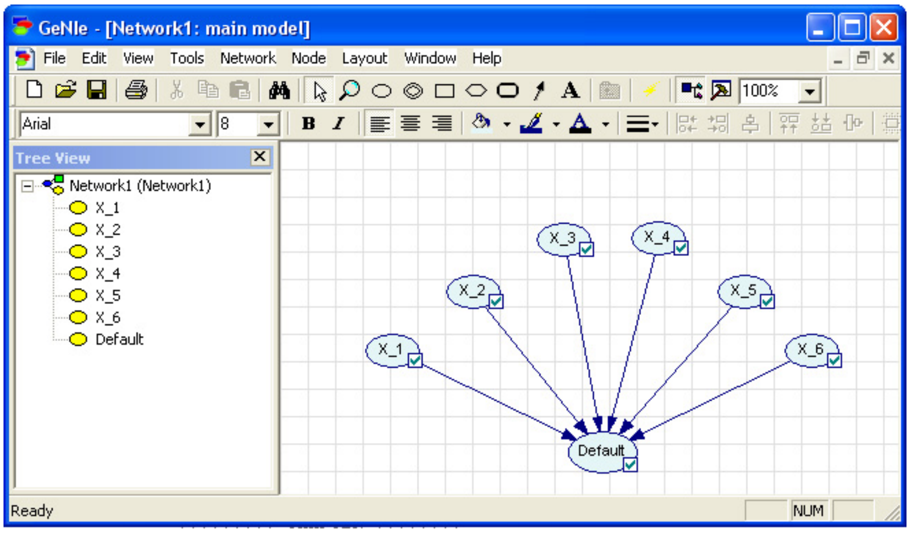


Figure 14.3: Screenshot of GeNIe 2.0

Part IV

Bayesian Networks in Credit Rating

Chapter 15

BN Results on Credit Data

15.1 CPT Networks

The performance of the CPT models is listed in Table 15.1. As expected, the CPT model performance increases as the number of degrees of freedom is reduced. But even with 672 degrees of freedom, the results are still very distant from the logistic regression results.

Table 15.1: Performance of CPT models

MODEL	**GINI**	χ^2 **for 6 cat.**
132000 input combinations		
Logistic Regression	78.3%	0.0023
CPT	55.6%	7.93
32000 input combinations		
CPT	56.6%	1.17
672 input combinations		
CPT	68.6%	0.080

Apparently, a model with a CPT approach has too many degrees of freedom. It allows a very nice fit if the relation between the variables is very complex. However, in most real-world data sets, this is not the case. The logistic regression model only has 38 degrees of freedom, but it assumes that the behavior response variable is a combination (the sum) of the behavior of the predictor variables:

$$P(d) = \frac{\exp(\beta_0 + \beta_1 \cdot x_1 + \ldots + \beta_{38} \cdot x_{38})}{1 + \exp(\beta_0 + \beta_1 \cdot x_1 + \ldots + \beta_{38} \cdot x_{38})}$$

Therefore, many possible outcomes of $P(d = 1)$ are possible, but only based on 38 parameters. Accurate estimates for those parameters can be obtained when the data set contains 50.000 records. This will enhance the stability and performance of the resulting model.

15.2 Noisy-MAX

Noisy-MAX is an interesting modeling method for credit rating, because it always focuses on the 'worst' input variable. Suppose one input variable indicates a very high probability of default, but the others all indicate a low PD. The noisy-MAX model will assign a high probability of default to that client, while logistic regression, which is a combination of all input variables, might estimate the PD to be average. It is hard to reason which model best fits the real world, which makes the comparison between Noisy-MAX and logistic regression very interesting.

As mentioned before the learning process for noisy-MAX networks is non-trivial and, therefore, multiple methods have been implemented and tested. GeNIe does not provide these methods yet, so an extended version of SMILE was used for a C++ program.

Table 15.2 contains the noisy-MAX results. For most of these models, the reduced states data set was used.

The CPT approximation with Euclidean distance (Noisy-MAX Eucl) performs not bad, better than the results from the CPT approach. This is not very surprising, the number of degrees of freedom is in a noisy-MAX model is much lower than for a CPT model. Nevertheless, a GINI index of almost 72% is still far from the 78.3% from logistic regression. This difference can be due to the assumptions that both models make about the portfolio. The CPT approximation method tries to make a close fit to the CPT, while logistic regression minimizes the error on each individual client.

The KL distance without cut performs very bad, this is mainly due to its incapability to handle zero values. When a cut of 5 is established, many of these zero values have disappeared from the CPT already, however, there are still many zero values left. Therefore, a GINI index of only 61.2% is attainable with the KL distance.

The EM algorithm shows the best noisy-MAX results. In fact, it is not very distant from the logistic regression score. The χ^2-measure for logistic regression still seems unbeatable.

Table 15.2: Performance of noisy-MAX models

MODEL	**GINI**	χ^2 **for 6 cat.**
132000 input combinations		
Logistic Regression	78.3%	0.0023
CPT	55.6%	7.93
Noisy-MAX Eucl	71.9%	0.0191*
32000 input combinations		
CPT	56.6%	1.17
Noisy-MAX Eucl	71.7%	0.0067*
Noisy-MAX KL no cut	33.4%	0.575*****
Noisy-MAX KL cut=5	61.2%	1.91****
Noisy-MAX EM	75.4%	0.0123(*)
672 input combinations		
CPT	68.6%	0.080

* = no values for 1 category

** = no values for 2 categories

...

15.3 Naive Bayes

Table 15.3 contains the results for naive Bayes and the TAN's. Exactly as expected, naive Bayes performs very well. The GINI index is almost as high as the logistic regression GINI. Unfortunately, the χ^2-measure of naive Bayes is fairly large compared to logistic regression. This is probably due to the usage of the logistic function in logistic regression. The logistic function handles small values very well and is very accurate.

15.4 Setting Priors, Learning Posteriors

Table 15.3 shows the results for the prior/posterior learning techniques. In fact, many experiments have been conducted with different weights, but these are not all listed in this table. For an overview of all results, turn to appendix A.

Compared to existing methods, such as noisy-MAX priors, naive Bayes priors perform better on the test data set. Unfortunately, the best performing model on this data set was naive Bayes without any posterior learning. Therefore, more data sets to test these methods on are required for a definite statement. These initial results show that this is a very promising learning technique for rare events data..

Table 15.3: Performance of Prior/Posterior models

MODEL	**GINI**	χ^2 **for 6 cat.**
132000 input combinations		
Logistic Regression	78.3%	0.0023
CPT	55.6%	7.93
Naive Bayes	77.2%	0.3610
Noisy-MAX Eucl	71.9%	0.0191*
TAN 1	76.9%	0.1893
32000 input combinations		
CPT	56.6%	1.17
Naive Bayes	77.5%	0.3610
TAN 1	77.5%	0.1893
TAN 2	77.5%	0.1844
Noisy-MAX Eucl	71.7%	0.0067*
Noisy-MAX KL no cut	33.4%	0.575*****
Noisy-MAX KL cut=5	61.2%	1.91****
Noisy-MAX EM	75.4%	0.0123(*)
Uniform Priors W=40	64.3%	0.151**
Naive Priors W=40	77.3%	0.3590
Noisy-MAX Priors W=10	71.4%	0.7870
Smoothing Naive & CPT W=20	76.6%	0.3620
Smoothing Noisy-MAX & CPT W=20	71.4%	0.0077
672 input combinations		
CPT	68.6%	0.080
Naive Bayes	76.7%	0.3460

Chapter 16

Conclusions & Recommendations

16.1 Logistic Regression

Logistic regression is the best practice modeling method in banks, despite many attempts to use other models. The effect variable is assumed to be the sum of the parameterized causes. Input variables are assumed to be conditionally independent and uncorrelated. These assumptions may seem very unrealistic, and in fact, for most real world data sets, they do not hold. However, the calculation method appears to be very accurate and stable.

Logistic regression performs well because apparently it makes exactly the right assumptions for this data set. For this data set it has exactly the right number of degrees of freedom. It is a very powerful modeling tool, because parameter assessment is done by minimizing the mean squared error. Therefore, logistic regression is by its nature bound to return very low (=good) χ^2-measures.

16.2 Bayesian Networks

Bayesian networks provide two major advantages. First, transparency of the model makes Bayesian networks easy to interpret. Second, incorporation of expert knowledge, a feature not supported by most other modeling techniques, can be very powerful. The drawback of Bayesian networks also concerns expert knowledge. Although there are methods available for learning the structure of a network from data, the construction of a BN requires some knowledge about the variables and their relations. Additionally, good prior parameters can increase performance, but again expert knowledge is required.

Several approaches for Bayesian networks can be used to model credit risk, such as CPT learning, or naive Bayes and its extension, tree augmented naive Bayes. Noisy-MAX is also interesting, but there are some issues to the noisy-MAX parameter learning. This research focused on a completely new noisy-MAX learning algorithm and a learning method for rare events data. These techniques were implemented in GeNIe and C++, and tested on the credit data. The results were compared to the logistic regression results.

16.3 Comparison

The appendix contains a complete list of the results. It is clear that for this data set, logistic regression yields the best model. This is possibly caused by the variable selection process, which was very much focused on logistic regression. The strength of Bayesian networks, combining expert knowledge with data, could not fully be utilized due to the lack of expert knowledge. Therefore, the BN models were basically solely built on the data set.

The best BN model is tree augmented naive Bayes. This approach is powerful because it combines the good classification properties of naive Bayes with dependencies, otherwise restricted in naive Bayes. Noisy-MAX almost attains the naive Bayes and TAN results. This is not surprising, since in both models, the conditional probabilities are assumed to be a sum or product of the input variables. A new Noisy-MAX learning algorithm is tested, CPT approximation, but it can not outperform EM learning. Apparently, the Euclidean and KL distances to the CPT are only partly indicators of the fit of a model. A normal CPT model returns very bad results, but this is not surprising if the number of degrees of freedom is higher than the number of records in the data set.

The construction of the data set favored logistic regression, because stepwise logistic regression was used for variable selection. There are structural learning techniques available for Bayesian networks, from which it should be possible to derive a 'stepwise BN construction'. This research did not focus on structural learning, but incorporation in the data selection procedure might improve the results. Stepwise BN construction and variable selection is definitely an interesting research topic.

Although the application of Bayesian networks could not give a direct boost to the credit rating models, a couple of new modeling techniques have been developed, which give interesting results.

16.4 Prior and Posterior Learning

A very promising modeling technique is setting priors and learning posteriors. Learning for rare event data can be avoided by setting good priors. Two improvements to learning for rare event data were suggested in this research. Firstly, weighted merging instead of cutting, which allows a more gentle balance between a prior and a posterior. Secondly, inference in a naive Bayes models can provide excellent priors for a CPT in a 'normal' Bayesian network.

Compared to existing methods, such as noisy-MAX priors, naive Bayes priors perform better on the test data set. Unfortunately, the best performing BN model on this data set was naive Bayes without any posterior learning.

Therefore, more data sets to test these methods on are required for a definite statement. These initial results are very promising.

Appendix A

Table of Results

MODEL	**GINI**	**χ^2 for 6 cat.**
132000 input combinations		
Logistic Regression	78.3%	0.0023
CPT	55.6%	7.93
Naive Bayes	77.2%	0.3610
Noisy-MAX Eucl	71.9%	0.0191*
TAN 1	76.9%	0.1893
32000 input combinations		
CPT	56.6%	1.17
Naive Bayes	77.5%	0.3610
TAN 1	77.5%	0.1893
TAN 2	77.5%	0.1844
Noisy-MAX Eucl	71.7%	0.0067*
Noisy-MAX KL no cut	33.4%	0.575*****
Noisy-MAX KL cut=5	61.2%	1.91****
Noisy-MAX EM	75.4%	0.0123$^{(}*)$
Uniform Priors W=10	64.3%	0.0268*
Uniform Priors W=20	64.3%	0.23*
Uniform Priors W=40	64.3%	0.151**
Uniform Priors W=80	64.3%	0.669**
Uniform Priors W=160	64.3%	0.68**

...continued on next page.

MODEL	GINI	χ^2 for 6 cat.
Naive Priors W=1	76.4%	0.3530
Naive Priors W=5	77.0%	0.3510
Naive Priors W=10	77.1%	0.3520
Naive Priors W=20	77.2%	0.3570
Naive Priors W=40	77.3%	0.3590
Naive Priors W=80	77.3%	0.3590
Naive Priors W=160	77.4%	0.3600
Noisy-MAX Priors W=10	71.4%	0.7870
Smoothing Naive & CPT W=10	75.4%	0.3640
Smoothing Naive & CPT W=20	76.6%	0.3620
Smoothing Naive & CPT W=40	77.0%	0.3620
Smoothing Noisy-MAX & CPT W=20	71.4%	0.0077
672 input combinations		
CPT	68.6%	0.080
Naive Bayes	76.7%	0.3460

* = no values for 1 category
** = no values for 2 categories
...

Bibliography

[1] Aldrich, J.H. & F.D. Nelson. *Linear Probability, logit and probit models.* Sage Publications, 1986.

[2] Allen, L., G. DeLong & A. Saunders. 'Issues in the Credit Risk Modeling of Retail Markets.' Working paper, February 2003.

[3] Altman, E.I. 'Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy.' *Journal of Finance* 23:589-609, 1968.

[4] Atiya, A.F. 'Bankruptcy Prediction for Credit Risk Using Neural Networks: A Survey and New Results' *IEEE Transactions on Neural Networks* 12(4):929-935, July 2001.

[5] Beck, N., G. King & L. Zeng. 'The Problem with Quantitative Studies of International Conflict.' http://web.polmeth.ufl.edu/papers/98/beck98.zip.

[6] Cramer, J.S. *The Logit Model - An introduction for economists.* London, Edward Arnold, 1991.

[7] Cybinski, P. 'The Evolution of Bankruptcy Studies: from prediction to explanation.' School of Economics, Griffith University, Brisbane, February 2003.

[8] Druzdzel, M.J. & L.C. van der Gaag. 'Elicitation of Probabilities for Belief Networks: Combining Qualitative and Quantitative Information.' *Uncertainty in Artificial Intelligence* 95:141-148, 1995.

[9] Elkan, C. 'Boosting and Naive Bayesian Learning.' Technical Report No.CS97-557, UCSD, 1997.

[10] Friedman, N. 'Learning belief networks in the presence of missing values and hidden variables.' *Proc. 14th International Conference on Machine Learning*, Morgan Kaufmann, 125-133, 1997.

[11] Friedman, N., D. Geiger & M. Goldszmidt. 'Bayesian Network Classifiers.' *Machine Learning* 29(2-3):131-163, 1997.

[12] Han, J. & M. Kamber. *Data Mining.* Morgan Kaufmann, 2001.

[13] Hatcher, L. & E.J. Stepanski. *A Step-by-Step Approach to using the SAS System for Univariate and Multivariate Statistics.* Cary, NC: SAS Institute Inc., 1994.

[14] Heckerman, D. 'A Tutorial on Learning Bayesian Networks.' Technical Report, Microsoft Research, 1995.

[15] Heckerman, D., D. Geiger & D.M. Chickering. 'Learning Bayesian Networks: The Combination of Knowledge and Statistical Data.' Technical Report, Microsoft Research, 1995.

[16] Hosmer, D.W. & S. Lemeshow. *Applied Logistic Regression.* John Wiley & Sons, 1989.

[17] Jensen, F.V., K.G. Olesen & S.K. Andersen. 'An algebra of Bayesian belief universes for knowledge-based systems.' *Networks* 20(5):637-659, 1990.

[18] Jensen, F.V., S.L. Lauritzen & K.G. Olesen. 'Bayesian updating in causal probabilistic networks by local computation.' *Computational Statistics Quarterly* 4:269-282, 1990.

[19] Kaski, S., J. Sinkkonen & J. Peltonen. 'Bankruptcy Analysis with Self-Organizing Maps in Learning Metrics.' *IEEE Transactions on Neural Networks* 12:936-947, 2001.

[20] King, G. & L. Zeng. 'Logistic Regression in Rare Events Data.' *Political Analysis* 9(2):137-163, 2001.

[21] Lauritzen, S.L. & D.J. Spiegelhalter. 'Local computations with probabilities on graphical structures and their application to expert systems (with discussion).' *Journal of Royal Statistical Society, Series B* 50(2):157-224, 1988.

[22] Neal, R.M. *Bayesian Learning for Neural Networks.* Springer-Verlag, 1996.

[23] Onisko, A., M.J. Druzdzel & H. Wasyluk. 'Learning Bayesian network parameters from small data sets: application of Noisy-OR gates.' *International Journal of Approximate Reasoning* 27(2):165-182, 2001.

[24] Pazzani, M. J. 'Searching for dependencies in Bayesian classifiers.' *Proceedings on the Fifth International Workshop on Artificial Intelligence and Statistics*, 239-248, Springer-Verlag, 1996.

[25] Pearl, J. 'Fusion, propagation and structuring in belief networks.' *Artificial Intelligence* 29:241-288, 1986.

[26] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[27] Ripley, B.D. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[28] Ross, M.S. *Introduction to Probability Models.* Academic Press, 1985.

[29] Spirtes, P., C. Glymour & R. Scheines *Causation, Prediction and Search.* Lecture Notes in Statistics 81, Springer Verlag, 1993.

[30] Udo, G. 'Neural Network Performance on the Bankruptcy Classification Problem.' *Computers and Industrial Engineering* 25:377-380, 1993.

[31] Vomlel, J. 'Noisy-OR classifier.' *Proceedings of the 6th Workshop on Uncertainty Processing*, 291-302, 2003.

[32] Wilson, R.L. & R. Sharda. 'Bankruptcy Prediction using Neural Networks.' *Decision Support Systems* 11(5):545-557, 1994.

[33] Xu, K. 'How has the literature on Gini's Index evolved in the past 80 years?' Working paper, 2003.