Title

Beatriz Barros

LiTH - MAT - EX - - 2011/ 12 - - SE

Title

Beatriz Barros

LiTH - MAT - EX - - 2011/ 12 - - SE

To Mariana and Mãe Isabel

Acknowledgments

First of all I would like to express my gratitude to Dr John Noble for presenting me with a very challenging project and for providing me with much needed advice throughout the project.

I would also like to thank my parents, Maria do Carmo e José António, for their unconditional support during all my life and mostly when I was in Sweden. I would also like to thank André and Milton for listening my concerns.

Abstract

This thesis considers the Recursive Autonomy Identification algorithm for structure learning and uses it as the basis for a dynamic structure learning algorithm that may be applied to financial data sets.

The Recursive Autonomy Identification algorithm for structure learning is a constraint based algorithm for constructing an essential graph that describes the independence structure of the variables of a data set; the presence of edges and their orientations are determined by the results of conditional independence tests. A key assumption for the algorithm is that the distribution has a *faithful* graphical representation. The performance of the algorithm is analysed when the conditional independence tests may be unreliable and also when the assumption of faithfulness may not hold. In some circumstances the RAI algorithm seems sensitive to these problems and may produce wrongly oriented edges. Some alterations to the algorithm are proposed to make it more robust under these circumstances.

The algorithm developed in the thesis is designed detect changes in associations and update the network accordingly. The proposed algorithm seems to have some advantages over the algorithms available in the literature. The algorithm identifies changes in the dependence structure which, in the financial context, provides useful insights into the changing associations between the variables.

Contents

Introduction 15

Bayesian Networks and Learning Structures 17

2.1 Introduction 17

2.2 Bayesian Network Background 17

2.3 Structure Learning Algorithms 19

2.4 Constraint Based Algorithms 20

2.4.1 Conditional independence tests 21

2.4.2 Edge Orientation 24

2.5. Fast Adjacency Search 25

2.6 Recursive Autonomy Identification 25

2.7 Recursive Method 27

2.8 RAI, RM and FAST 28

Financial Data set 29

3.1 Introduction 29

3.2 Stock Market Index Data Set 29

3.2 Discretisation 30

3.4 Fading 31

Robustness of the RAI 32

4.1 Introduction 32

4.2 Essential Graph 33

4.2.1 Alteration 1: Essential Graph 34

4.3 Double Arrows 35

4.3.1 Alteration: Compelled Edges 37

4.4 D- Connections 38

4.4.1Alteration 3: D- Connections 40

4.5 Removal of Cycles 41

4.5.1 Alteration 4: Without Cycles 41

4.6 Alarm Network with RAI 42

4.6.1 Alterations 1, 2, 3, 4 44
4.7 The Financial Dataset 46
4.7. 1 Categorical Variables with Fading 46
4.7.2 Alterations 1, 2, 3, 4 48
4.7.3 Continuous Variables with Fading 49
4.7.4 Alterations 1, 2, 3, 4 51
4.7.5 Continuous Variables without Fading 52
4.7.6 Alterations 1, 2, 3, 4 54
4.7.7 Continuous and Categorical Variables with Fading 54
4.7.8 Alterations 1, 2, 3, 4 57
Incremental Learning Structure 58
5.1 Introduction 58
5.3 Incremental Learning Structure Algorithms 59
5.3.1 Buntine Incremental Learning Algorithm 60
5.3.2 Lam and Bacchus Incremental Learning Algorithm 60
5.3.3 Friedman and Goldzmit Incremental Learning Structure Algorithm 61
5.3.4 Roure Incremental Learning Structure Algorithm 62
5.3.5 Nielsen and Nielsen Incremental Learning Structure Algorithm 63
5.3.6 Shi and Tan Incremental Learning Structure Algorithm 64
5.3.7 Comparison 65
5.4 Detection Methods 66
5.4. 1 The CUSUM 67
5.4.2 The Indicator Function 68
5.4. 3 Second Cosine Transform 69
5.4.4 Comparison 70
New Structure Learning Algorithms 71
6.1 Introduction 71
6.2 Incremental Learning Algorithm 1 71
6.3. 1 Detect Change Function 72
6.3.2 Update Function 73
6.3. 3 Conclusion 74
6.3. 1 Experimental Verification 74
6.3 The New Incremental Learning Algorithm 2 78
6.3.1 Detect Change Function 78

6.3.2 Conclusion 79
6.6.3 Experimental Verification 79
6.4 New Incremental Learning Algorithm 3 81
6.4.1 Detect Change Function 81
6.4.2 Updating Function 82
6.4.3 Conclusions 83
6.4.4 Experimental Verification 84
6.5 The Incremental Learning Algorithms Applied to the Financial Data Set 86
Conclusions and Future Work 89
7.1 Conclusions 89
7.2 Future Work 92
Appendix A 97

List of Figures

Figure 1: The orientation rules by Meek, 1995 24
Figure 2: Learning examples: the different learned structure during the learned process. 33
Figure 3: Learning examples: a) the true graph, b) the essential graph of the true graph..... 34
Figure 4: Learning examples: the different rounds in the learned structure after determining the descendant substructure {2, 4, 5}..... 35
Figure 5: Typical situation where there are reversed edges in a stage of RAI algorithm. 36
Figure 6: Example of the influence of order node in the RAI algorithm: a) the reversed edges b) the immorality $5 \rightarrow 3 \leftarrow 2$ is the last added c) the immorality $3 \rightarrow 5 \leftarrow 1$ is the last formed 36
Figure 7: The learned structure in case of considering the conditional independence statement $3 \perp 1 | 4$ wrong. 38
Figure 8: Example that illustrate CI tests of order 2 contradicting CI test of order 0. 39
Figure 9: Example that illustrates CI test of order 3 contradicting CI tests of order 0..... 40
Figure 10: Method for removing cycles: a) cycle composed by three variables b)cycle composed by four variables..... 42
Figure 11: Learned Structure with an alpha of 0.01
Figure 12: Learned Structure with an alpha of 0.05..... 48
Figure 13: Learned Structure with an alpha of 0.01
Figure 14: Learned Structure with an alpha of 0.05..... 51
Figure 15: Learned Structure with an alpha of 0.01
Figure 16: Learned Structure with an alpha of 0.05..... 53
Figure 17: Learned Structure with an alpha of 0.01
Figure 18: Learned Structure with an alpha of 0.05..... 56

List of Tables

Table 1: Result of RAI algorithm with the Alarm Network. 42
Table 2: Result of RAI algorithm with Alteration 1 – Essential Graph with the Alarm Network 45
Table 3: Result of RAI algorithm with Alteration 3 – D-Connections with the Alarm Network 45
Table 4: Result of RAI algorithm with categorical variables of the financial data set. 47
Table 5: Result of RAI with Alteration 1, 3, 4 with categorical variables of the financial data set. 49
Table 6: Result of RAI algorithm with continuous variables of the financial data set. 50
Table 7: Result of RAI with Alteration 1, 4 with continuous variables of the financial data set. 52
Table 8: Result of RAI algorithm with continuous variables without fading of the financial data set. 53
Table 9: Result of RAI with Alteration 1,4 with continuous variables without fading of the financial data set. 54
Table 10: Result of RAI algorithm with categorical and continuous set of variables of the financial data set. 56
Table 11: Result of RAI with Alteration 1,3,4 with categorical and continuous set of variables of the financial data set 57
Table 12 Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving observations from the altered distribution 76
Table 13 Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving learning the change 77
Table 14: Results of the New Incremental Learning Algorithm 1 when the threshold of the CUSUM changes 77
Table 15: Results of the New Incremental Learning Algorithm 2 with Asia Network after receiving observations from the altered distribution 79
Table 16: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving observations from the altered distribution 80
Table 17: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving learning the change 80
Table 18: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving observations from the altered distribution 85
Table 19: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving learning the change 85
Table 20: Results of the New Incremental Learning Algorithm 1, 2, 3,4 with the financial data set. 87

List of Abbreviations

CI	Conditional Independence
CMI	Conditional Mutual Information
DAG	Directed Acyclic Graphical model
Fast	Fast Adjacency Search Learning Structure Algorithm
RAI	Recursive Autonomy Identification Learning Structure Algorithm
RM	Recursive Method Learning Structure Algorithm
SHD	Structural Hamming Distance

Chapter 1

Introduction

Over a period of many decades, the values of different financial variables have been recorded and different methods have been proposed to make inferences and predictions. Consequently, there is now a huge quantity of financial data available. To exploit this data, it is necessary to find methods to deal with very large data sets which can extract relevant information for financial decision makers. This thesis uses the theoretical background of Bayesian Networks to provide a method for understanding how different financial variables are related. Bayesian Networks have been applied to a wide range of fields, giving useful results. For example, Bayesian networks have been an important tool for medical expert systems, medical diagnosis, pattern matching, speech recognition, risk assessment and genetic models, among others (Fenton et al., 2005), (Pourret et al., 2008), (Friedman et al., 2000). They have also been applied to financial problems, for example portfolio selection and return, risk analysis, operational risk and credit rating (Shennoy and Shennoy, 2000) (Demirer, 2005) (Fenton et al., 2005), (Pourret et al., 2008). In this thesis, Bayesian networks are used to find associations between different stock market indices, since the stock markets are of major importance in the global economy. The total daily stock market transactions globally are approximately $200x10^9$ Euros, through the operations of thousand of agents. The stock markets are also a key factor for the modern social an economical history, dictating the quality life style of millions of people (Leon, Pierre, 1982). In the prediction process the first main step consists of the manipulation and transformation of the original data set to facilitate the process of extracting information. The preparation of the data largely involves locating groups of variables which contain the information necessary for forecasting. The stock markets have more than two thousand stocks available and the financial data sets generated by them usually have substantial complexity and they are not in general, stationary. Individual variables may not be very informative and are substantially affected by noise, making the process of deciding which variables are most relevant and influential even more difficult. The scale of the vast quantities of information available that can affect the market to some degree leads to the necessity of machine learning methods for learning the variables that are most appropriate for forecasting.

Bayesian Networks learning techniques can be applied to multivariate probability models with hundreds of variables; the analysis is facilitated through graphical representation of the results. A Bayesian network is the factorisation of a probability distribution over a set of variables along a directed acyclic graph, which encodes a substantial part of the conditional independence structure between the variables. The variables are represented by nodes and the direct dependencies between them by directed edges. There are two main problems in constructing a Bayesian network: learning the structure of the network and learning the conditional probabilities once the structure is known. This thesis considers the first of these and, in the context of financial variables, the search is for a graph that represents the strongest and most significant dependencies between variables.In the first chapter, it is presented the Bayesian Network background and the main concepts of learning structure algorithms, specifically, constraint based ones and three learning structure based constraint algorithms are analyzed. The second chapter presents the financial data set that will be used and the transformations that it is subject to. The third chapter is centered on the Recursive Autonomy Identification learning structure algorithm. Its behavior is analyzed and some improvements are proposed. In the fourth chapter, it is proposed three new incremental learning algorithms for dealing with sequentially received financial data. The correctness of the algorithm is certifying by evaluating its behavior with generated data. The last chapter presents the main conclusions of this thesis and indicates futures directions of research to complement this thesis.

Chapter 2

Bayesian Networks and Learning Structures

2.1 Introduction

A Bayesian network is a directed acyclic graphical model [DAG] that represents the factorisation of a joint probability distribution over a set of variables, which expresses the conditional independence structure between the variables implied by the factorisation. The variables are represented by nodes and the direct dependencies between them by directed edges. Starting from this simple description of Bayesian Network, the first part of this chapter provides the necessary background to Bayesian Networks. The second part of this chapter concentrates on the problem of learning the structure of a Bayesian network, determining the edges and their orientations. The main approaches structure learning are described. The constraint based approach is described in more detail, since it is the basis of the thesis. The third part of this chapter examines recent constraint based structure learning algorithms, pointing out the advantages and drawbacks of each one.

2.2 Bayesian Network Background

The theoretical background for Bayesian networks is taken from "Bayesian Networks: An Introduction" of Timo Koski and John M. Noble (Koski and Noble, ...), from which the most relevant definitions and theorems are taken.

Definition: Graph, Directed Graph and Undirected Graph A graph G consists of a finite node set V = (α_1... α_d) and an edge set E describing relations between the nodes in

V. If $\langle \alpha_j, \alpha_k \rangle \in E$ there is an undirected edge between the nodes α_j, and α_k. If $(\alpha_j, \alpha_k) \in E$ there is a directed edge from α_j to α_k. If all edges in a graph are undirected, it is called an undirected graph, and if al l edges are directed, it is called a directed graph.

Definition: Parent and Child Consider a graph G= (V, E) where V= $\{\alpha_1 ... \alpha_d\}$ and let E=D ∪ U, where D is the set of directed edges and U the set of undirected edges. Let α_j, $\alpha_k \in V$. If $(\alpha_j, \alpha_k) \in D$ then α_k is referred to as a child of α_j and α_j is referred to as parent of α_k.

Definition: Path, Directed Path Let G= (V, E) denote a simple graph, where E= D ∪ U. That is D ∩ U = ϕ, D denotes the directed edges and U denotes the undirected edges. A path of length m from a node α to a node β is a sequence of distinct nodes $(\tau_0,..., \tau_m)$ such that τ_0=α and τ_m = β such tat $(\tau_{i-1}, \tau_i) \in E$ for each i=1,...,m, either $(\tau_{i-1}, \tau_i) \in D$ or, $\langle \tau_{i-1}, \tau_i \rangle \in U$. The path is a directed path if $(\tau_{i-1}, \tau_i) \in D$ for each i=1,...,m. That is, there are no undirected edges along the directed path.

Definition: Cycle Let G= (V, E) be a graph. A m-cycle in G is a sequence of distinct nodes $(\tau_0,..., \tau_{m-1})$ such that $\tau_0,..., \tau_{m-1}, \tau_0$ is a path

Definition: Directed Acyclic Graph A graph G= (V; E) is said to be a directed acyclic graph if each edge is directed (that is G is a simple graph such that for each pair (α, β) ∈ E ⇒ (β, α) ∉ E) and for any node α ∈ V there does not exist any set of distinct nodes $\tau_1,..., \tau_m$ such that α ≠ τ_1 for all i=1,..., m and (α, $\tau_1,..., \tau_m$, β) forms a directed path. That is there are no m-cycles in G for any m≥ 1.

Definition: Bayesian Network Let p be a probability distribution over the, per example a discrete random variables $X_1,...,X_d$, each having a finite number of possible outcomes. Let G = (V , E) be a directed acyclic graph where the node set V represents the random variables and the edge set E describes the relations between them. Let $X_{\prod(i)}$denote the set of parents of the node X_i. The graph is constructed such that p factorizes along G:

$$pX_1, \dots, X_d(x_1, \dots, x_d) = \prod_{i=1}^{d} PX_i | X_{\prod(i)}(x_i | x_{\prod(i)}) \qquad (1.1)$$

Definition: Chain, Collider and Fork A chain connection is formed by a set of three nodes if they follow the structure: $X_1 \rightarrow X_2 \rightarrow X_3$ or $X_1 \leftarrow X_2 \leftarrow X_3$, being X_2 called a chain node. A fork connection is formed by a set of three nodes if they follow the structure: $X_1 \leftarrow X_2 \rightarrow X_3$, being X_2 called a fork node. A collider connection is formed by a set of three nodes if they follow the structure: $X_1 \rightarrow X_2 \leftarrow X_3$, being X_2 called a collider node.

Definition: Blocked Trail A trail τ between two different nodes X and Y in a graph G = (V, E) is blocked by a set of nodes S ⊆ V\ {X,Y} if at least one of the two following

conditions holds: 1. There is a node W ∈ S in τ that is not a collider node. 2. W is a collider node in τ and neither W nor any of its descendants belongs to S.

Theorem: D-Separation Implies Conditional Independence Let G = (V;E) be a directed acyclic graph and let p be a probability distribution that factorises along G. Then for any three disjoint subsets A, B, S ⊆ V, it holds that A ⊥ B| S (A and B are independent given S) if A and B are d-separated by S.

Theorem: Markov Equivalence Two DAGs are Markov equivalent if and only if they have the same skeleton and the same immoralities.

The proofs of the above theorems can be found in the book "Bayesian Networks: An Introduction" (Koski and Noble, 2009).

Definition: Skeleton, Immoralities The skeleton of a graph G= (V, E) is the graph obtained by making the graph undirected. That is, the skeleton of G is the graph G = (V,E) where , $\langle \alpha, \beta \rangle \in E \Leftrightarrow (\alpha, \beta) \in D$ or $(\beta, \alpha) \in D$ or $(\alpha, \beta) \in U$. Let G= (V,E) be a graph. Let E= D ∪ U, where D contains directed edges, U contains undirected edges and D ∩ U= ϕ. An immorality in a graph is a triple (α, β, γ) such that (α, β) ∈ D and (γ, β) ∈ D but (α, γ) ∉ D and $\langle \alpha, \gamma \rangle \notin U$.

Definition: Essential Graph Let G be a Directed Acyclic Graph. The essential graph G* associated with G is the graph with the same skeleton as G, but where an edge is directed in G* if and only if it occurs as a directed edge with the same orientation in every DAG that is Markov equivalent to G. The directed edges of G* are the essential edges of G.

Definition: Faithfulness A probability distribution p and a directed acyclic graph G = (V ; E) are faithful to each other if :

$$X \perp Y \mid S \Leftrightarrow X \perp Y \|_G S \qquad (1.2)$$

for all ∈ V , Y ∈ V and S⊆ V , where the variables X , Y and those in S are disjoint. The faithful graph is not necessarily unique. There might exist more than one DAG faithful the same distribution (markov equiavalence theorem).

2.3 Structure Learning Algorithms

Learning a Bayesian network from a data set by considering all the possible graph structures with *n* nodes is not computationally feasible unless n is small. The number of possible structures grows exponentially with the number of nodes (Robinson, 1977).

Even when the parent sets of variables are restricted to at most k parents, the structure learning problem is NP-Complete (Chickering, 1996). Consequently, heuristic approaches have been developed to restrict the problem or to identify one DAG or a set of equivalent DAG structures from data under certain circumstances. Two main approaches to learn the structure of a Bayesian Network have been proposed.

The first class of methods is search and score (Chickering, 2002; Cooper and Herskovitz, 1992, Heckerman et al 1995). A score is attributed to each candidate structure based on how well the structure describes the data set and the Bayesian Network whose factorization of the joint distribution gives the highest score is chosen. The score function is usually a penalized likelihood such as BIC or BDeu (Heckerman et al., 1995).

The second class is constraint based which test for conditional independence between variables. When a conditional independence is detected, this is added to the set of constraints that the structure must satisfy. Conditional independence is decided by performing a statistical or information-theory based test. These tests determine the skeleton and the immoralities. After this, the compelled edges are determined based on Meek's orientation rules (Spirites et al., 2000).

Each approach has its advantages and its disadvantages. There also hybrid algorithms that combine techniques from the constraint base approach and search and score approach. The search and score algorithms are more flexible and learn graphs with higher likelihood, but they are not able to deal with hundred thousand variables, which can be dealt with by constraint based methods. On other hand, constraint based methods make categorical decisions in the first phases, relying on statistical tests with an arbitrary level of significance which can lead to a cascading effect of errors. This thesis deals exclusively with constraint based methods for learning the structure of Bayesian networks.

2.4 Constraint Based Algorithms

Constraint based methods usually rely on the assumption of faithfulness of the distribution which means that there exists a directed acyclic graph which is faithful to the probability function. The assumption is that for any three disjoint subsets of variables A, B and S the conditional independent statements are equal to the d-separations statements of the three variables (section 2.2).

Instead of searching for a fully directed acyclic graph corresponding to the distribution, constraint based algorithms usually learn the essential graph. The essential graph of G is

the graph with the same skeleton as G, but where an edge is directed in if and only if it occurs as a directed edge with the same orientation in every DAG that is Markov equivalent (section 2.2).

2.4.1 Conditional independence tests

The determination of conditional independence [CI] statements between variables, X,Y|Z, plays a major role for learning the graph structure in a constraint based algorithm. In this section the conditional independence tests are described.

Definition: Independence Two random variables X and Y are independent if their joint probability distribution factorizes as:

$$P_{X,Y}(x,y) = P_X(x)P_Y(y) \qquad (1.1)$$

Definition: Conditional Independence Let X , Y and Z be random variables, X and Y are conditionally independent given a random vector Z if:

$$P_{X,Y,Z}(x,y,z) = P_{X|Z}(x|z)P_{Y|Z}(y|z)P_Z(z) \qquad (1.3)$$

This is written $X \perp Y|Z$.

To verify conditional independence statements $X \perp Y|Z$, statistical tests are needed. There are many conditional independence tests available, either based on statistical measures or in information theory. These tests have the following framework:

- Define the null hypothesis: the two variables are conditionally independent given a set of other variables which size can vary.

$$H_0 = X \perp Y|Z \qquad (1.4)$$

- Define the alternative hypothesis: the two variables are conditionally dependent:

$$H_1 = X \not\perp Y|Z \qquad (1.5)$$

- Compute the value of interest and compare it to a threshold: if the value is lower than the threshold the null hypothesis is reject and if the value is higher the null hypothesis is not reject.

The probability of making the decision to reject the null hypothesis when it should not be rejected is called a *type I* error and has probability α, the significance level, which is specified in advance. The probability of not rejecting the null hypothesis when the alternative hypothesis is true is called error *type II* and has probability β. The probability of rejecting the null hypothesis when it is false is called the *power of the test* and is (1-β). In statistics, not rejecting the null hypothesis does not imply accepting it. However, in the constraint based methods, when the null hypothesis is not rejected, the default decision is to assume that the null hypothesis is true and hence to removing the corresponding edge. This misleading application of the conditional independence can lead to errors in the subsequent phases of the algorithm. An error of *type I* in a the context of a constraint based structure learning algorithm may lead to a false positive edge (keeping the edge) provided it is not later removed by other CI tests and an error of *type II* always leads to wrongly removing an edge (false negative). As a result, there are more false negative errors (wrongly removing an edge) than false positives (wrongly retaining an edge) in constraint based algorithms for learning the structure of a Bayesian Network.

The reliability of a test depends on two main factors: how well the data represents the underlying probability distribution and the *power of the test*. The power depends how far the alternative hypothesis is from the null hypothesis. It is usually difficult to calculate the power exactly, since it involves an infinite power series expansion. Improving the power implies a lower level of significance. The only way to improve the power while maintaining the same significance is to increase the quantity of data.

For categorical data, the CI is usually tested by the chi-square test, likelihood ratio test or Conditional Mutual Information [CMI] test from theory information. For continuous variables, Fisher's test is typically used. This measures the strength of linear associations between the variables. The first group of tests also detects non linear associations between variables. The first group of tests is less reliable when the size of the conditioning set S increases while the robustness of the Fisher's test is not affected by the size of the conditioning test.
In the thesis, for categorical data either the Likelihood Ratio test or CMI test is used and for continuous variables the Fisher test is used. These tests are described more precisely in the next section.

Likelihood Ratio conditional Independence Test

The Likelihood Ratio conditional independence test computes the statistic G^2 from a contingency table containing counts of the variable values that occur in the data. The statistic is shown in the equation above where n(i,j,k) is the number of times simultaneously $X = x_i$, $Y = y_j$ and $Z = z_k$ in the sample, that is, the value of the cell (i,j,k) in the contingency table. By the central limit theorem, G follows approximately a

distribution of chi-squared with $(j_x-1)(j_y-1)j_z$ degrees of freedom, where j_x denotes the number of states for the variable X, j_y denotes the number of states for the variable Y and j_z denotes the number of states for the variable Z. The degrees of freedom indicate the number of parameters that can be varied in the model and fewer parameters more accurate will be the estimations, that is more data can be use to estimate each parameter.

$$G = 2\sum_{i=1}^{m}\ \sum_{j=1}^{p}\ \sum_{k=1}^{q} n(i,j,k)\ln\left(\frac{n(i,j,k)n(.,.,k)}{n(i,.,k)n(.,j,k)}\right) \qquad (1.6)$$

Conditional Mutual Information

Based on information theory, the Conditional Mutual Information test measures dependence degree between the variables X and Y given Z, by measuring the information that one variable provides about the another one. Given a probability distribution P defined over the sets of variables X, Y and Z, the conditional mutual information is:

$$\sum_{i=1}^{m}\ \sum_{j=1}^{p}\ \sum_{k=1}^{q} P(x_i,y_j,z_k)\frac{P(x_i,y_j|z_k)}{P(x_i,|z_k)P(y_j|z_k)} \qquad (1.7)$$

As with the other conditional independence tests, the conditional mutual information is compared to a threshold, but in this case it must be computed. The threshold varies with the size of sample and with the size of the test (Cheng et al., 2002). Some heuristics have been proposed to find the optimal threshold. In this thesis, the approach described by Yehezkel and Lerner will be used, which uses a BDeu score to evaluate the different learned structures with different threshold and chooses the threshold which leads to a highest score (Yehezkel and Lerner, 2009).

Fisher's Test

The Fisher test verifies whether the partial correlation coefficient between the variables X, Y and Z is zero. The null hypothesis is the partial correlation coefficient is zero, ρ, against the alternative hypothesis of the partial correlation coefficient of X and Y given S, R. It uses the following equation, where M is the size of the sample:

$$Z = \frac{1}{2}\sqrt{M-|S|-3}\left(\ln\frac{1+R}{1-R}\right) \qquad (1.8)$$

If ξ is defined as:

$$\xi = \frac{1}{2}\sqrt{M-|S|-3}\left(\ln\frac{1+\rho}{1-\rho}\right) \qquad (1.9)$$

then the distribution of |Z-ξ| is approximately normal and it is possible to compare the value |Z-ξ| with the p-value.

2.4.2 Edge Orientation

The other main step in a constraint based method is the orientation of the edges. This is carried out by determining the immoralities and then by adding the compelled edges. The first rules of edge orientation were formulated by Meek (Meek, 1995) and they consisted on one rule for the v-structures and three more for adding the compelled edges.

1. If A is adjacent to B in the graph G and if C is not in the separation set of A and B, then the edges A→C and B←C are added.

2. The rules of compelled edges are shown in the figure above in which A, B, C and D represent variables in the graph and the arrows represent directed edges.

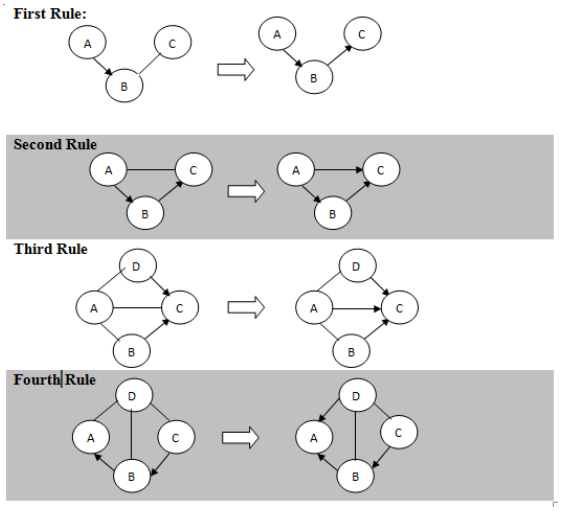


Figure 1: The orientation rules by Meek, 1995

In the thesis, the orientation rules of Spirites, Glymour, and Scheines (Spirites et al., 2000) are applied. These are:

1. For each triple of vertices X, Y, Z such that the pair X, Y and the pair Y, Z are each adjacent in the skeleton but XZ are not adjacent, orient X → Y ← Z as X ⊥Y |Z if and only if Y is not in the conditioning set separating X and Z

2. If A → B, B and C are adjacent, A and C are not adjacent, and there is no arrowhead at B, then orient B – C as B → C.

3. If there is a directed path from A to B, and an edge between A and B, then orient A – B as A → B.

4.

The rules are applied by order (1 to 3) and are proved to give the correct graph when the algorithm determines the correct skeleton and the correct conditioning sets. However when constraint based algorithms are performed with real data, there are usually errors in the skeleton (missing edges or extra edges) and often the wrong conditioning sets, which can lead to inconsistencies when the orientation rules are applied.

2.5. Fast Adjacency Search

The Fast Adjacency Search [Fast] algorithm proposed by Andrew Fast (Fast, 2010) is a structure learning constraint based method. It underlies on the main assumption, as other constraint based algorithms, of faithfulness of the distribution. It was derived from the PC algorithm (Spirites et al, 2000) and consists of two main steps. Starting with a complete undirected graph, it tests conditional independence for all pairs of variables before proceeding for a conditional set of higher order. When a conditional independence structure is obtained, the edge between the pair of variables is removed and for this pair of variables it will not perform any other conditional independence tests. The second step is based on the orientation rules (Spirtes et al. 200) in which the v-structures are formed and the compelled edges are added.

The Fast algorithm was designed for categorical data which means the conditional independence tests are unreliable for higher order of conditioning sets. The Fast algorithm does not perform tests with large conditioning sets. By running conditioning independence tests for all pairs of variables, the Fast algorithm is very robust but not as efficient as other algorithms. The Fast algorithm can incorporate improvements and corrections. As a constraint-based method, most of the errors are false negatives (edges wrongly removed). In the second step, the Fast algorithm may also present some wrong errors of orientations due to the fact that the orientations rules (Spirites et al., 2000) give contradictory results, either because the distribution is not faithful or because a CI test gives an result. This topic is discussed further on chapter 5.

2.6 Recursive Autonomy Identification

As a constraint based method the Recursive Autonomy Identification [RAI] (Yehezkel and Lerner, 2009) has as main assumption the existence of a faithful graph and aims to

locate the essential graph (the essential graph of G is the graph with the same skeleton as G, but where an edge is directed in if and only if it occurs as a directed edge with the same orientation in every DAG that is Markov equivalent). The algorithm decomposes the graph recursively, increasing the order of CI test in each stage. The algorithm relies on two definitions (Yehezkel and Lener, 2009):

Definition: Exogenous Cause A node Y in G= (V,E) is an exogenous cause to G'=(V', E'), where V'$\subset$ V and E'$\subset$ E, if Y $\notin$ V' and X $\underline{\vee}$ $\in$ V', Y $\in$ Parents (X,G) or Y $\notin$ Adjacent(X,G)

Definition: Autonomous Substructure: In a Dag G(V,E), a substructure $G^A(V^A,E^A)$ such $V^A \subset$
V and E^A is said to be autonomous in G given a set $V_{ex} \subset V$ of exogeneous causes to G^A if X $\underline{\vee}$ V^A, Parent (X, G$\subset$ { $V^A \cup V_{ex}$ }. If V_{ex} is empty, it is said the substructure is autonomous.

The algorithm starts with a complete undirected graph and has 4 main steps:

- In this step the algorithm performs CI tests for conditioning subsets of size *n* between the variables in question and the exogenous variables, removing the edge if the variables are conditional independent. It proceeds by looking for the v-structures and adding the compelled edges, according to the orientation rules.
- The second step is similar to the first, but it performs CI test only between the variables in question. After directing the edges, it groups the nodes that have the lowest topological order into autonomous descendant sub-structures and define the ancestor sub-structures.
- It performs recursively the code with the ancestor sub structures, increasing the size of conditional subset (*n*+1).
- In this step it defines the ancestor variables are set as the exogenous variables to the descendant sub-structure and it runs recursively again with the variables in question being the descendant sub-structure, increasing also the size of conditioning subset (*n*+1).

It has an exit condition which prevents the performance of CI test of a *n* order when the number of parents of variables is lower than *n*+1.

The RAI was designed for categorical variables and it uses the CMI test to determine conditional independence. The algorithm seeks to reduce the number of CI tests of higher order which are less reliable and slower to perform. It reduces the number of CI test during the process of structure learning and also the average size of conditioning sets in CI tests due to the combination of the skeleton construction and edge direction along the learning procedure. It only runs CI test for the relevant conditioning subsets

with *n* variables, before increasing the size of the conditioning set. A false negative error in RAI may lead to the detection a wrong v-structure which may lead to a wrong decomposition of the descendant and ancestor sub-structures. The RAI may also give some errors due to the application of the orientation rules, either when there does not exist a faithful graph, or when a CI test returns an incorrect result. Compared with PC algorithm, the sensitivity to the order of the nodes is reduced in RAI.

2.7 Recursive Method

The Recursive Method [RM] algorithm (Xie and Cheng, 2008) is also a constraint based method that assumes faithfulness. The structure of the algorithm is summarised in two main steps:

- From the full data set, an undirected independence graph is constructed and is recursively decomposed into smaller subgraphs, using a binary tree. These subgraphs are then pruned to obtain local undirected independence graphs. When it is not possible to decompose further, the IC/PC algorithm is used to obtain the skeletons.
- The skeletons are combined from the leaf nodes to the root node to form the global skeleton. If an edge appears in one skeleton but has been deleted in the other, then it is deleted. The v-structures are applied and finally the orientation rules are implemented.

The recursive method was designed for categorical and Gaussian variables. In the first type of variables it performs an order partial correlation to locate the independence graph and in the second type it searches for the Markov blanket by using likelihood ratio tests. During the decomposition, the algorithm uses a binary tree which may not have a unique decomposition. The algorithm uses a sub-optimal method to construct it, the maximum cardinality search. Although some errors may be produced in this phase, the authors state that the sub-optimal method leads to very efficient decompositions. The recursive decomposition presents better results than one step decomposition, particularly when there are many v –structures. Although the RM algorithm assumes faithfulness, this assumption is only required for the last phase of the step 1 for defining the skeletons. It may perform reasonably well when the assumption of faithfulness is not guaranteed, particularly if the last step is not implemented.

2.8 RAI, RM and FAST

The Fast and RAI algorithms present different solutions for the trade-off between computational efficiency and accuracy. Fast opts for an increased accuracy while increasing the complexity. The RAI, on the other hand, gives priority to computational efficiency over accuracy, reducing the number of conditional independence tests required. Both algorithms may have inconsistencies with the orientation of the edges due to the lack of faithfulness in the distribution or wrong results of CI tests. In the case of Fast, the directions are added in the end of the algorithm which does not lead to skeleton errors. The RAI algorithms by combining the process of learning the skeleton and orienting the edges reduces the number of conditional independence structures. This strategy however, in cases of real data where the faithfulness assumption may not guaranteed and there is not an "oracle" to determine the conditional independence statements, is more unstable. That is, a wrong orientation in an early phase can produce wrong substructures and cause skeleton errors. Both algorithms were also designed for categorical data where it is necessary to keep the conditioning sets small and also to look for associations that are not linear.

The Recursive Method was proposed for both discrete and continuous variables. In the second case, the algorithm is less reliable because the independence graph is determined by searching the Markov Blanket of each variable, which requires the conditioning set to be all the others variables. In contrast to Fast and RAI algorithm, RM only needs the faithfulness assumption in a later stage, specifically when interpreting the results of the CI tests on the final cliques.

Since the Recursive Autonomy Identification seems more reliable than the Recursive Method and has smaller complexity compared with the Fast algorithm, the RAI was chosen for the financial dataset in this thesis.

Chapter 3

Financial Data set

3.1 Introduction

The first stock market and goods market was established in Amsterdam in the XIV century. It was in the Amsterdam stock market that the first company with a fixed capital stock appeared, the Dutch East India Company. Since then, the stock markets have had great importance in the economy and consequently in the life of citizens. In 1929, there was a crash on the Wall Street Stock Market, a deep devaluation of the shares, caused by the extreme speculation of stock market agents, which led to one of the greatest worldwide economic crises, comparable only to the current economical crisis. This profound crisis affected millions of people, triggering a high unemployment rate in some countries and leading to dictatorial political regimes in other countries (Leon, 1982). Nowadays, there is a stock market and equity market in almost every country. The importance of the stock market to the development of modern societies is one reason for choosing this data set. This chapter aims to present the financial data set which will be analysed in the thesis and describes two techniques which will be used to analyse this data set.

3.2 Stock Market Index Data Set

The stock markets consist of company shares that can be exchanged and aims to provide the companies with access to capital. Each stock market can have one or several stock market indices. A market index is an indicator of the performance of the stock market, representing an average performance of the companies included. These stock market

indices can simply represent an average of the net changes in the prices of the company shares or they can have different weights to reflect the market capitalisation of the companies. In this thesis the selected data set is composed by the close values of 18 stock market indices from 1st January 2005 to 1st January 2011.

1. Amsterdam Stock Index
2. Austrian Traded Index
3. Brussels Stock Index
4. Bombay Stock Exchange Index
5. S&P 500 Index
6. Hang Seng Index – Hong Kong
7. Madrid Stock Index
8. Jakarta Stock Index
9. Koerea Composite Stock Price Index
10. MERVAL -Mercado de Valores de Buenos Aires
11. Mexican Sotck Exchange Index
12. NIKKEI 225 Tokyo
13. NASDAQ -100
14. New Zealand Exchange Stock
15. OMX Stockolm
16. Portugal Stock Index 20
17.American Stock Exchange Index
18. FTSE – Footsie

3.2 Discretisation

The associations between variables can be linear or non-linear and in the financial area there is a prevalence of non-linear associations. The dataset of the diary values of the stock market indices are taken on a continuous scale and it is necessary to discretise them to learn non linear dependencies between the different stock market indices. The discretisation process, however, leads to loss of information. In this thesis, the variables were discretised in the follow manner: there were three states with the borders minus and plus one standard deviation from the statistical mean of each variable.

3.4 Fading

The financial markets are dynamic and the associations between variables change constantly. The older observations should therefore have a lower weight when running a learning algorithm. For the discrete case, the weights were given by the above equation, where N is the number of observations. In the contingency table for the conditional independence test the number one which appears four days ago and seven days ago, will be translate to the row (1,1,1) with a weight of s^4+s^7.

$$N = \sum_{j=0}^{\infty} s^j = \frac{1}{1-s} \quad (2.1)$$

For continuous variables, weights are applied in the computation of the statistical covariance, where represents each observation in the variable 1, represents each observation in the variable 2, and u_1 and u_2 are the mean for the variable 1 and variable 2, respectively.

$$covariance(1,2) = E((x_1 - u_1)(x_2 - u_2)) \quad (2.2)$$

A factor, *f*, where *N* represents the total number of observations and *i* the position of the observation in time, is applied to each observation and the calculation of the mean takes also into account these weights.

$$f_i = \frac{i}{N} \quad (2.3)$$

Chapter 4

Robustness of the RAI

4.1 Introduction

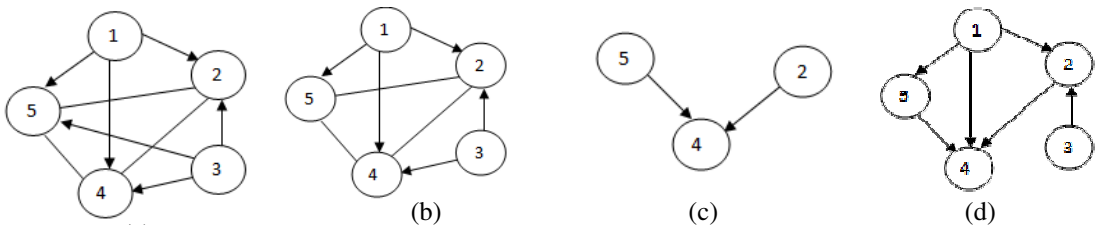
The authors give a full proof that the Recursive Autonomy Identification algorithm returns the correct essential graph when there is a graph faithful to the probability distribution and all the CI tests return the correct answer. However, the CI tests are not entirely reliable. Furthermore, for many real data sets, the assumption of faithfulness does not hold.

The first part of this chapter describes several problems that can arise if the above assumptions do not hold, by considering theoretical examples. Several modifications to the RAI algorithm are proposed to deal with these problems. The second part of this chapter analyses the behaviour of the RAI algorithm when applied to simulated data generated by the Alarm network and the behaviour of the modified algorithm applied to the same data set. The third part evaluates the RAI algorithm when it is performed with a real data set, concretely the financial data set, where the assumption of faithfulness is not assured. The evaluation is also made when the RAI algorithm is modified with the alterations proposed in the first part. In the financial data set, these two analyses are made when the variables are a categorical and subject to a fading process, when they are continuous and also subject to a fading process, when they are continuous but are not submitted to a fading process and using both data sets: categorical variables and continuous variables with a fading process.

4.2 Essential Graph

The RAI algorithm aims to locate the essential graph from the data set. However, in some situations it does not produce the essential graph because some edges are directed that should not be. This situation occurs when, in a previous stage of the algorithm, an edge has been directed due to a v-structure and in the following stage the other edge of the v-structure is eliminated, while the edge is maintained with its direction. This is illustrated by a theoretical example.

Figure 3 shows the true structure and the essential graph of the structure. Figure 2 shows the most relevant stages of the learning process of the essential graph. After stage 0, the exit condition is not met and the stage 1 is skip (there is no exogenous variables). In stage 2 the algorithm finds the relation $1 \perp 3 | \Theta$ and directs the edges according to the v-structures: $1 \rightarrow 2 \leftarrow 3$; $1 \rightarrow 4 \leftarrow 3$; $1 \rightarrow 5 \leftarrow 3$ (figure 2a). It identifies as ancestor substructures {1} and {3} and the descendant sub-substructures {2, 4, 5}. It proceeds to stage 3 in which the size conditional set is increased to 1. The recursive call is returned immediately for both ancestor sub-structures because the exit condition is met. Moving to stage 4, the analysis of descendant substructure, it looks for conditional independence structures between the descendant substructure and the exogenous variables (the ancestor sub-structures) with conditional subsets of size 1. It determines $3 \perp 5 | 4$, removing the edge 3-5 in stage 1. In stage 2 it does not identify any pair of variables in the descendant sub-structure that are conditionally independent with a conditioning set of size n=1 (figure 2b). It moves on to determine the lowest topological order, the descendant sub-structure {2, 4, 5} and the ancestor substructure, the empty set {}. In the stage 3 the exit condition is satisfied for the empty ancestor set with n=2. It calls the descendant sub-structure with n=2 and it does not find any new conditional independence in stage 1 but it obtains, in the stage 2, the $52|4$ statement. It performs the orientation of the edges: $2 \rightarrow 4 \leftarrow 5$ (figure 2c). The ancestor substructures are {2} and {5} and descendant-substructure is {4}. It calls recursively each ancestor substructure but are returned immediately with a n=3. In the stage 4, for the descendant substructure it performs conditional independence tests with n=3, obtaining in stage 1 a conditional independence structure $3 \perp 4 | \{1, 2, 5\}$ (figure 2d). No immoralities are added. It forms the ancestor and descendant substructures {} and {4}, respectively. Both exit conditions are satisfied and the RAI returns the graph in figure 2.



(b) (c) (d)

Figure 2: Learning examples: the different learned structure during the learned process



(a) (b)

Figure 3: Learning examples: a) the true graph, b) the essential graph of the true graph.

The graph obtained by the RAI algorithm is not the essential graph since the edge 1→ 5 is directed. During the stage 2, with the descendant sub-structure {2, 4, 5} and with a n=1 the algorithm deletes the edge 3→5of the v-structure. However, the directed edge 1→5 is retained.

4.2.1 Alteration 1: Essential Graph

This section proposes an alteration to the RAI algorithm that returns the essential graph in the situation described above. This code differs from that of Raana Yehezkel and Boaz Lerner only in stage 1 and 2. Before performing the orientation rules, the directed edges from the previous stage are undirected.

In the figure 3 the main steps of the RAI with the proposal alteration are shown when applied to the same example. The alteration only has an effect after stage 4, with the descendant sub-structure {2, 4, 5} and n=1 (figure 4a). After, identifying the constraint 35|4, the algorithm undirects the graph and proceeds to apply the orientations rules. It detects 3 v-structures: 1 → 2 ← 3; 3 → 2 ← 5; 1 → 4 ← 3 (figure 4b). Comparing this same stage with the original RAI, in this case the edge 1-5 is not directed and the edge 5 – 2 is directed. In the next task, it determines as ancestor substructure, the empty set {}, and as descendant substructure the set {2, 5, 4}. The ancestor meets the exit condition. It runs the stage 1, with n=2 but does not identify any constraint between the exogenous variables {1, 3} and {2, 4, 5}. In stage 2, it obtains the constraint 5⊥2|{1, 3} and after undirected the graph, the v-structures are added 1 → 2 ← 3; 2 → 4 ← 5; 1 → 4 ← 3 (figure 3c and figure 3d, respectively). It also determines {2}, {5} as ancestors and {4} as descendant. The algorithm proceeds for each ancestor until the exit condition is met. For descendant {4}, it removes the edge between the variable 3 an 4 due to the constraint 3⊥4|{1, 2, 5} in the stage 1. After undirecting the graph the immoralities are formed 1 → 2 ← 3; 2 → 4 ← 5 (figure 4e). In the stage 2 nothing is altered and it forms the ancestor and descendant substructures {} and {4}, respectively. Both exit

conditions are met. The algorithm performs the orientation rules and the rule 1 is applied: 1 → 4 (figure 4f) is directed. With this alteration, the RAI produces the essential graph. This alteration proposed gives only a minor increase in the computational complexity.

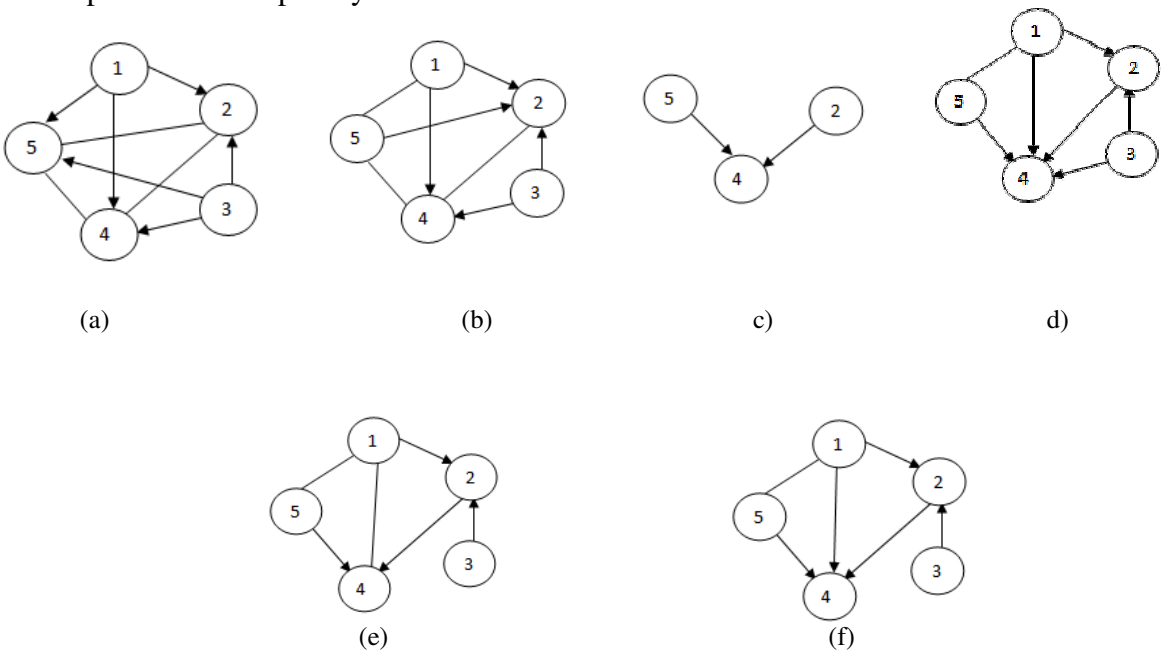


Figure 4: Learning examples: the different rounds in the learned structure after determining the descendant substructure $\{2, 4, 5\}$.

4.3 Double Arrows

The RAI algorithm orients edges by adding in the corresponding immoralities after every CI test that removes an edge. On the one hand, the algorithm requires fewer CI tests than the FAST algorithm, but on the other hand it can produce a larger number of "double arrows", where the edge is oriented in one direction due to one immorality and also oriented in the other direction due to another immorality in the same stage of the algorithm. These "doubles arrows", or reverse edges, can lead in a extreme case to a wrong skeleton and wrong edge orientations and to a learned structure that does not satisfy all the constraints or even a structure that is not a valid Bayesian network. The order of nodes can affect the orientation of the edges. An example of common situation which this situation occurs in the RAI algorithm either with simulated data or real data is given later.

In the beginning of the RAI algorithm, for a DAG with five variables, the exit condition is met and stage A is skip. In stage B, the algorithm determines two conditional independence structures, for the empty set: 1 and 2; 3 and 4. The following step is the

formation of the immoralities, according to the orientation rules. The algorithm forms the immoralities $1 \rightarrow 3 \leftarrow 2$; $1 \rightarrow 4 \leftarrow 2$; $1 \rightarrow 5 \leftarrow 2$; $3 \rightarrow 1 \leftarrow 4$; $3 \rightarrow 2 \leftarrow 4$; $3 \rightarrow 5 \leftarrow 4$ which means that the edges 1 – 3, 2 – 3, 1 – 4 and 2 – 4 are reversed, double arrows as shown in the figure 5:

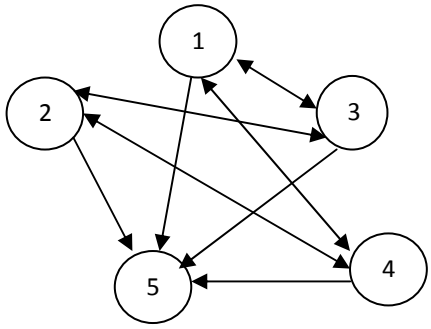


Figure 5: Typical situation where there are reversed edges in a stage of RAI algorithm.

The fact that the orientation of the edges is dependent of the order node, with different node orders can lead to formation of different substructures, descendant and ancestors.

An example is now given to illustrate the claim that the order of nodes can influence the learned structure. The RAI learns the structure of a graph with five variables. In the first round with empty conditioning set no edge is removed. In the second round, in the stage B, the algorithm identifies the conditional independence structures: $3 \perp 1 | 4$, $4 \perp 2 | 1$, $\perp | 1$. When the immoralities are formed ($3 \rightarrow 5 \leftarrow 1$, $4 \rightarrow 3 \leftarrow 2$, $5 \rightarrow 3 \leftarrow 2$) there are two reversed edges (figure). If by the order of nodes the immorality $3 \rightarrow 5 \leftarrow 1$ is the last added, the ancestor substructure will contain all the variables and the descendant set will be the empty set. However, if by the order of the nodes the immorality $5 \rightarrow 3 \leftarrow 2$ is formed in the end, the ancestor substructure will be the variable $\{3\}$ and the descendant substructure the set $\{1, 2, 4, 5\}$.

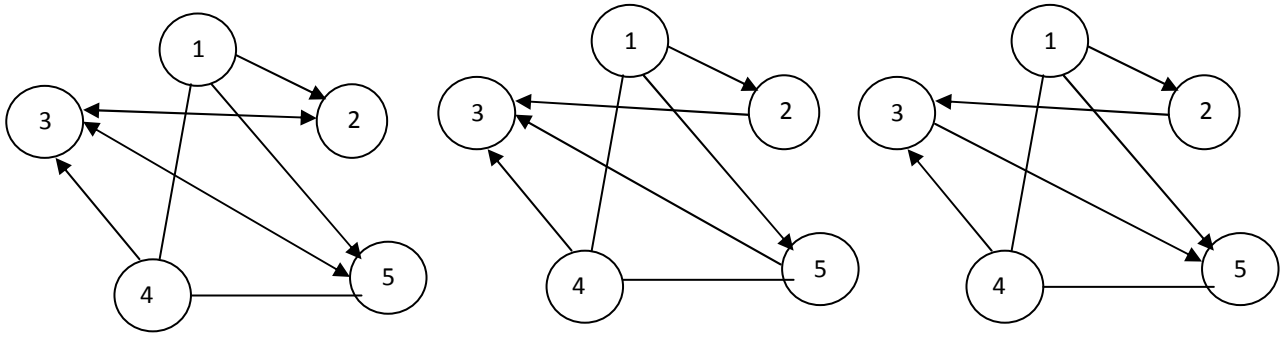


Figure 6: Example of the influence of order node in the RAI algorithm: a) the reversed edges b) the immorality $5 \rightarrow 3 \leftarrow 2$ is the last added c) the immorality $3 \rightarrow 5 \leftarrow 1$ is the last formed

4.3.1 Alteration: Compelled Edges

A solution to the problem of reversed edges requires an understanding of the root of the problem. Two scenarios are illustrated in the following example (figure 6). The first scenario is that the independence structures are both correct in the current round of the algorithm which means that there is no `correct' graph. In the following section 4.6 it is shown that the RAI with alarm network produces many reversed edges during the first round but, fortunately for this data set, most of the double arrows are eliminated in rounds with larger conditioning sets.

The second scenario is that one of the conditional independence structures is wrong due to a wrong conclusion of an independence test, which leads to the production of false negative errors. Constraint base methods are more likely to produce false negative errors since not rejecting the null hypothesis does not mean it is true. The false negatives edges can be more dangerous in the RAI algorithm because these errors cannot be repaired as with false positives, which can be removed in a further stage with higher order of CI tests.

The false negative errors correspond to type I errors and a simple way to reduce them is by increasing the significance level, α. However, this will be accompanied by a rise of false positive edges. Other approaches have been proposed but they all deal with this same trade-off, as for example the “rule of thumb”. (Afast, 2010). Andrew Fast even proposes an algorithm to maintain stable the value of β during the performance of all CI tests (Afast, 2010). However, this approach increases substantially the computational complexity, which is related with the non centrality parameter of the X^2 distribution used to approximate the G statistic is somewhat arbitrary. Other approaches to reduce the false negatives rates through decreasing the size of the conditioning sets are also present in the literature as, for example, the necessary path or the weak correction (Abellan et al., 2006) (Afast, 2010). In the thesis it was established that the RAI algorithm would not perform CI tests with higher conditioning sets of 4 and so these two approaches do not work with the RAI algorithm.

Instead of following these approaches, an edge orientation optimization could be used, adding one of the edges that were removed and hence reducing the number of reversed edges. In this example, the reversed edges are eliminated entirely. In the case of more than two conditional independence structures, the criteria to choose the edge to add would be the edge that gives the largest reduction in the number of reversed edges. This approach has two drawbacks. The first is that when a graph has many variables and many double arrows at some stage of the algorithm, this would imply adding too many edges and substantially increasing the false positive errors. The second drawback would be that the added edge could change the formation of the substructures, the descendant and ancestor sub-structure, which could lead to a different learned structure. Using the

previous example, if the edge 3 – 1 is added to eliminate the double arrows (figure 7), the ancestor substructure would be the entire set of variables, instead of being only variable 3.

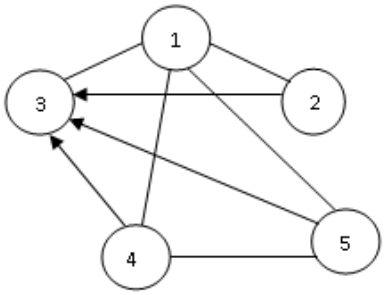


Figure 7: The learned structure in case of considering the conditional independence statement $3 \perp 1 | 4$ wrong.

Andrew Fast suggests another method to deal with this problem, a constraint optimization algorithm which searches over possible orientations to find the orientation which satisfies the maximum number of constraints (Afast, 2010). The implementation of this algorithm to the RAI code will not work if each direction for an edges satisfies the same number of constraints, which happens in the first example, or if it opts for an orientation for an edge which is removed in a further stage. There are also another methods suggested in literature for revising the learned structured based on the inconsistencies such as a greedy search, a set weights along with constraints in order to maximize the sum of weights, but all of them are ad hoc.

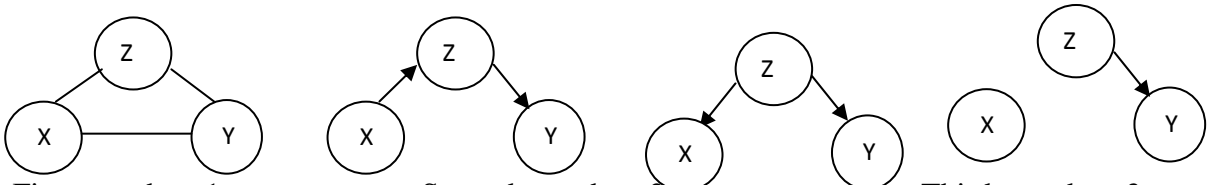
The RAI algorithm with real and synthetic data will usually produce double arrows. If the compelled edges are added during the rounds, they can produce more errors because they can be the result of double arrow or even a result from an edge that is directed in that stage but in a further one will not be. In fact, with the Alarm Network and with the financial data set, the algorithm only adds the compelled edge at the end (section 4.6). To prevent these possible errors and to decrease the steps of the RAI, the proposed alteration – Compelled Edges - is that the compelled edges should only be directed at the end of the algorithm.

4.4 D- Connections

Conditional independence statements are decided in the light of statistical tests or information theory measures. Particularly, in the case of categorical variables these tests are less and less reliable as the size of the conditioning set increases. That is, the result of CI test performed with an empty conditioning set is more robust than a CI test

performed with a conditioning set of a one, and so on. The RAI and Fast algorithms can develop inconsistencies due this nature of the CI tests.

There are two situations where these inconsistencies arise. The RAI only performs CI tests with conditioning sets of size at most four. The first situation occurs when the algorithm for the variable X and Y states that they are conditionally dependent given the empty separation set, so that the final graph should have a path between the two variables. In the second round, the conditional independence structure X⊥Y|Z is identified which means that there is a directed path between X and Y which has a chain or a fork node. In the third round, the inconsistency may arises: the RAI removes the edge between Z and X due to the CI tests that identifies the statement Z⊥X|{A, B} contradicting the first CI test which stated that there is a path between X and Y, a chain or a fork.

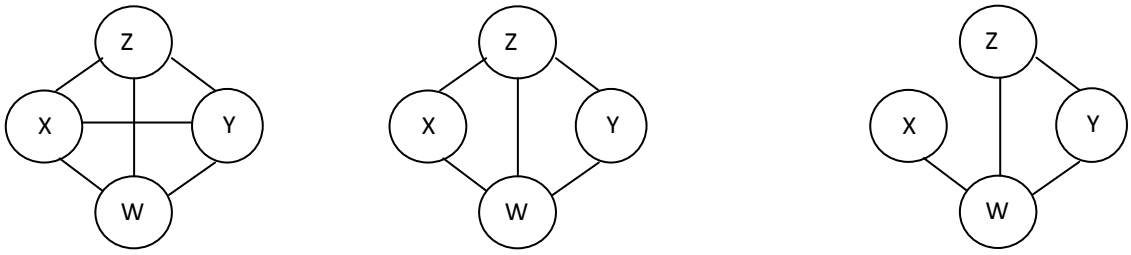


First round: n=1 Second round: n=2, Third round: n=3

X⊥Y|Ø X⊥Y|Z X⊥Z|{A,B}

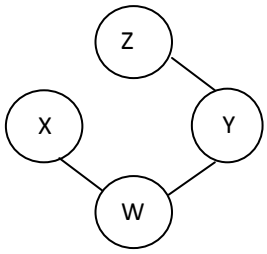
Figure 8: Example that illustrates CI tests of order 2 contradicting CI test of order 0.

The second situation occurs when the conditioning separation set is size zero and X and Y are verified to be dependent. This means that the null hypothesis was rejected when the CI test was made. In the second round, the edge between the variables X and Y is kept. However, in the third round with a conditioning set of size two the algorithm finds the conditional independence structure: X⊥Y|{Z, W} which means that neither the collider connection $X \rightarrow Z \leftarrow Y$ nor $X \rightarrow W \leftarrow Y$ are present. When the algorithm performs the CI tests in the fourth round with a conditioning set of size three it eliminates the edge between X and Z (or for example, X and W , or Y and W or Y and Z). If there is no edge between W and Z due to other CI tests, the graph contradicts the dependency between X and Y determined in the first round, when they are not conditioned on any variable.



First round: n=1 Third round: n=3, a chain or a fork Fourth round: n=3

X⊥Y|Ø X⊥Y|{Z,W} X⊥Z|{A,B,C}



Fourth round: n=3

X⊥Z|{A,B,C}

Figure 9: Example that illustrates CI test of order 3 contradicting CI tests of order 0.

4.4.1 Alteration 3: D- Connections

To prevent the first situation, that is, removing the edge between X and Z, or Y and Z the follow alteration was added to the RAI code during the performance of CI tests:

> ***Algorithm Alteration 1****: The inputs are a list of pair of variables, List(X,Y), a significance level of CI tests, α, and a type of CI tests, CI*
>
> ---
>
> 1: Alteration 1(List(X,Y), α, CI)
> 2: Loop i
> 3: List(i,i)
> 4: If n=2
> 5: If Z C Sep {X, all the variables\{Z}} } or X C Sep{Z, all the variables\{Z}} }
> 6: CI Test not performed
> 7: Else
> 8: CI Test (α, CI)
> 9: End
> 9: Else n<2
> 10: CI Test (α, CI)
> 11: End
> 12: End

For the second case, the alteration proposed is that the CI test for an edge between Z and X (or X and W, or Y and W or Y and Z) is only carried out with a conditioning set of three variables if there is the edge between Z and W (per example):

Algorithm Alteration 2*: The inputs are a list of pair of variables, List(X,Y), a significance level of CI tests, α, and a type of CI tests, CI*

```
1: Alteration 1(List(X,Y), α, CI )
2: Loop i
3:     List(i,i)
4:         If  n=3
5:             If  Z C Sep {X, all the variables\{Z}} or X C Sep{Z, all
the variables\{Z}}
6:              (Sep {X, all the variables\{Z}}={Z,W})
7:                 If not edge between Z and W
8:                                 CI Test not performed
9:                  Else
10:                 CI Test (α, CI)
11:             End
12:        Else  n<3
13:          CI Test (α, CI)
14:      End
15: End
```

4.5 Removal of Cycles

If a probability distribution does not have a faithful graph, or if there are errors in the learned CI statements, the RAI may produce a graph with cycles.

4.5.1 Alteration 4: Without Cycles

There are two approaches to eliminate the cycles. The first is for each set of three variables that has directed edges without any collider, the edges are undirected. For each cycle of length equal or greater than 4 that is not triangulated, a chord is added and the edges are undirected, so that the chain component is triangulated.

Definition: Triangulation An undirected graph is said to be triangulated if every cycle of length 4 has a chord.

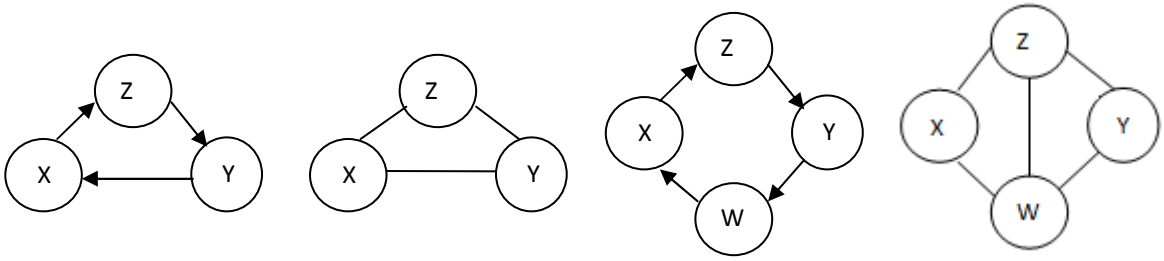


Figure 10: Method for removing cycles: a) cycle composed by three variables b) cycle composed by four variables.

The other approach is to use a heuristic algorithm to remove edges from cycles. In the thesis his second method is adopted, because the first approach adds edges. This goes against the general aim, which is to learn the strongest dependencies between the variables. The heuristic used is based on the conditional mutual information between the variables.

4.6 Alarm Network with RAI

In this section, the behaviour of the RAI algorithms is analysed with the Alarm Network which has thirty seven variables and forty eight edges. This data set is available in the software BNT_SLP for the Matlab. The proposed corrections in the previous section will be also implemented and compared with results without altering the RAI code. The CMI test will be used with a threshold of 0.003 for 10 000 observations and of 0.0045 for 5 000 observations. Yehezkel and Lerner presented the SHD, which sums the five structural errors (extra direction, missing direction, reversed direction, extra edge and missing edge) for the Alarm Network with 10 000 observations and the results are much better compared to the other algorithms used in the article (Yehezkel and Lerner, 2009). In the above table is shown the SHD results and the number of reversed edges during the learning process and in the final graph:

Table 1: Result of RAI algorithm with the Alarm Network.

RAI	Missing edge directions	Extra edge directions	Reversed edge directions	Extra edge directions	Missing edge directions:	SHD	Reversed Edges during the learning process	Reversed Edges in the final graph
10000 observations	0	0	1	0	1	2	69	8
5000 observations	2	1	1	0	1	5	98	9

Although the RAI algorithm learned the Alarm Network almost perfectly, it is interesting to understand its behaviour. In the first round for the Alarm Network with 10 000 generated instantiations, the algorithm identifies 401 pairs of independent variables (that is, conditionally independent given the empty set). The next step aims to learn the immoralities: 960 are added. During this learning process, 69 edges are reversed ("double arrows") – the chosen direction only depends on the order in which the nodes are taken. In the next phase, the ancestor substructure consists of all the nodes with the exception of the node 37 which is the descendant substructure. The size of conditioning set is increased by one and CI tests are performed in the ancestor substructure: 187 edges are removed, corresponding to the conditional independence statements detected. The removal of the edges leads to the elimination of 46 of the edges that were reversed in the previous stage. The algorithm proceeds to locate the new v-structures: only one is added, not leading to any reversed edge. The ancestors substructure are {1, 2, 5, 6, 7, 8, 11, 12, 13, 15, 16, 18, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32}, {3, 9, 14, 17, 19, 20, 22} {4} and{10} and descendant substructure is {33, 34, 35, 36}. Any conditional independence pair of variables is identified between the exogenous variables and the variables that compose the first ancestor, but five conditional independence tests do not reject the null hypothesis for a conditioning set of two variables. No other immorality is formed and the ancestor set is divided into the ancestors {1, 2, 6, 7, 8, 12, 13, 15, 18, 21, 23, 25, 26, 27, 28, 30}, {5} and {11, 16} and descendants {24, 29, 31, 32}. For a conditioning set of size three variables no results are produced and the first ancestor mentioned in the previous round is split into ancestor {1, 26, 7, 8, 12, 13, 15, 18, 21, 23, 25, 26, 27, 28} and descendant {30}. For both substructures, the exit condition is met and for the ancestors {5} and {11, 6} it is also met. It follows that the next conditional independence structure is only determined for the descendant {24, 29, 31, 32}, given three nodes. The ancestor and descendant are determined, being respectively, {} and {24, 29, 31, 32}. The exit condition is achieved for the ancestor and any conditional independence structure with a conditioning set of size four is formed. Again the same ancestor and substructure is formed from the descendant and any results are obtained. The algorithm proceeds to a previous ancestor substructure {3, 9, 14, 17, 19, 20, 22} and one more conditional independence structure is obtained for a conditioning set of size two. The substructure is divided in {3} and {9, 17} for the ancestors and {14, 20, 22} for the descendant. The exit condition is met for the ancestors and the descendant substructure is tested with a conditioning set of three variables, but no edge is removed. Again, an ancestor set is formed, the empty set, and the descendant set {14, 20, 22}. The exit condition is met for both.

The algorithm goes back to the other ancestor {4} and {10}, and the exit condition is achieved for n equal to 3. In the same level in the descendant substructure {33, 34, 35, 36}, one conditional independence statement is located. This descendant is subdivided in an empty ancestor and a descendant containing all the variables and the algorithm achieves the exit condition for both. For the descendant sub-structure {37}, conditional independence tests are performed between this variable and the exogenous variables, the

other 36 variables. 19 conditional independence statements conditioned on one variable are identified. In this step, 17 from the 19 edges are removed. These all correspond to reversed edges. When the size of the conditioning set is increased to two, five more edges are removed, four of which correspond to reversed edges. The last operation of the algorithm is to apply the orientation rules: it directs five edges using orientation rule 1. The learned structure has 48 edges; eight of them "double arrows". The wrongly oriented edge is due to a v-structure added in the first round that does not exist after later rounds, since the other edge that formed the v-structure was removed.

In the case of the Alarm Network with only 5 000 observations available, the behaviour of the RAI is similar. In the first round, there are 406 independent pairs of variables and the corresponding immoralities are formed, a total of 962. During this process, 98 edges are reversed. In the further round more edges are removed: 19 due to conditioning sets of size 1; ten due to conditioning sets of size two; and three due to conditioning sets of size 3. Only one more v-structure is added and the orientation rules for compelled edges are applied only at the end of the algorithm. From the first rule, five compelled edges are added. In the end, the learned network has 45 edges and a total of nine edges that were reversed during the learning process. The wrongly oriented edge is the same as in the example presented above, but in this case the other edge of the v-structures is maintained (there are two v-structure that have this same edge).

With the ALARM network, the algorithm produces many reversed edges in an early phase, but the overwhelming majority of these are removed from the skeleton due to conditional independence tests with larger conditioning sets. Only nine and eight reversed edges remain at the end of the algorithm when there are 10 000 observations and 5 000 observations respectively. Although the direction of these nodes is due to the order of the nodes, the direction of these edges is correctly learned. If the variables were in a different order, the algorithm could have more directional errors.

Another relevant aspect is that the wrongly oriented edge for the data set with 10 000 instantiations is due to an immorality formed towards the beginning of the algorithm, with the other edge removed later.

4.6.1 Alterations 1, 2, 3, 4

The four alterations to the RAI code proposed in the above section were illustrated with theoretical examples and their performance tested using an artificial data set, the ALARM network. As stated previously, the second alteration – Compelled Edges leaving the orientation of the compelled edges to the end - aims to decrease the number

of errors in the direction of the edges. In the example above if a compelled edge was added during the first round due to the "double arrows" it could lead to some errors. The alteration of only adding the compelled edges at the end does not alter the final outcome for the ALARM network, but it can be useful in case of real data set, besides decreasing the run time. If the Alteration 1: Essential Graph proposed, which undirects the edges after the conditional independence tests are performed, is added to the RAI code when it is learning the Alarm Network the following results, in terms of SHD, are obtained:

Table 2: Result of RAI algorithm with Alteration 1 – Essential Graph with the Alarm Network

RAI	Missing edge	Extra edge	Reversed edge directions	Extra edge directions	Missing edge directions:	SHD
10000 observations	0	0	0	0	4	4
5000 observations	2	1	1	0	3	7

The results are only slightly worse than the RAI algorithm without alteration, but in terms of the total number of edges in the learned graph, the difference is insignificant. In the case of 10 000 observations, this procedure does not alter the rate of false negatives or false positive errors but it has four edges undirected that should have been directed. The learned structure has two immoralities: 2 → 13 ← 12; 3→ 14 ← 9. In the case of 5 000 observations, the proposed alteration does not increase the numbers of missing edges and extra edges, there are two additional unoriented edges that should be directed. The algorithm identifies three v-structures: 13 → 12 ← 18; 2 → 13 ← 12; 3 → 14 ← 17, leading to only one double arrow in the end (edge 12 – 13).

If the algorithm alterations 1 and algorithms 2 suggested in the section 4.5.1 D-connections are applied, the number of errors is slightly worse, but again the difference is insignificant. Due to the fact that some conditional independence tests are not performed, there is an increasing in the number of false positives (extra edges), three more edges with 10 000 observations and one more with 5 000. There are no changes in the directional errors with these proposed alterations.

Table 3: Result of RAI algorithm with Alteration 3 – D-Connections with the Alarm Network

RAI	Missing edge	Extra edge	Reversed edge directions	Extra edge directions	Missing edge directions:
10000 observations	0	3	1	0	1
5000 observations	2	2	1	0	1

In the case of alarm network for 10 000 observations and 5 000 observations no cycles are produced and so the suggested cycle removing algorithm proposed is not applied.

4.7 The Financial Dataset

The aim of this section is to give an empirical analysis of the behaviour of the RAI algorithm when it learns a network from the financial dataset. Four cases will be tested: categorical variables with fading using the likelihood ratio conditional independence, continuous variables with fading using the Fisher conditional independence test, continuous variables without fading using the Fisher conditional independence test and using both continuous and categorical variables with the respective tests. The CI tests obtain a p-value that is compared to a threshold, α. This threshold called the significance level, α, means the probability of the CI tests rejecting the null hypothesis, the conditional independence, when it should not be rejected. It was chosen for the threshold the value 0.01 and 0.05 because these values are the most widely used in the literature. When α assumes the value 0.01 it is more difficult to reject the null hypothesis than when the value is 0.05, leading to a learned Bayesian network with lower number of edges.

4.7. 1 Categorical Variables with Fading

The algorithm starts with the undirected graph, which has 153 edges. For the first CI tests, performed with an empty conditioning set, the null hypothesis of independence is reject for all pairs of variables. The algorithm defines as ancestor substructure the empty set and as descendant all the variables. For the ancestor substructure, the exit condition is met. For the descendant substructure, during the phase of thinning the graph with a conditioning set of size 1, 18 edges are removed. The algorithm then locates the immoralities. There are 208 of them, producing 72 reversed edges (edges directed both ways). The ancestor substructure is defined as the set of all variables and the descendant substructure in this round is the empty set. Recursively, the RAI identifies 86 conditional independence statements given a conditioning set of two variables. The removal of edges decreases the number of reversed edges; 42 are eliminated. The algorithm proceeds, but does not add any new v-structures. It decomposes the ancestor in a new ancestor, a smaller set {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 17} and a descendant set {11, 13, 18}. In the ancestor set, six more conditional independence

structures are learned with three variable conditioning sets and again no additional edges are directed, reducing by five the reversed edges. Recursively, the algorithm forms the ancestor {1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 16, 17} for which the exit condition is met and the descendant substructure {7, 14, 17} for which also the exit condition is achieved. The algorithm goes back to the descendant {11, 13, 18} and detects two conditional independence statements for the variable 13 and 18 with a three variable conditioning set, eliminating one more reversed edge. Finally, the algorithm directs three edges using the first rule of compelled edge orientation. The learned network has 41 directed edges, no undirected edges and 20 reversed edges. The learned structure is shown in the figure 11 and it is not acyclic. The figure 12 represents the learned model obtained by using a threshold of 0.05.

The follow table shows the number of directed, undirected and total number of edges in the learned network, the edges that were reversed during the learned process and the number of these edges that remained in the learned network for a threshold 0.01 and 0.05. It shows also the edges that are in both learned structure.

Table 4: Result of RAI algorithm with categorical variables of the financial data set.

	Directed Edges	Undirected Edges	Total	Reversed Edges during the learning process	Reversed edges in the final graph
α= 0.01	41	0	41	72	20
α= 0.05	40	0	40	60	18
Number of same edges	31	-	-	-	-

With the simulated data from the ALARM network, it is possible to verify whether or not the results of RAI are correct. With the financial dataset, the underlying distribution is unknown. Consequently, the performance of the algorithm cannot be evaluated in direct way, but the reversed edges give some clues. The reversed edges in first place indicate that the there are contradictory immoralities. If the reversed edges were formed due to an immorality that resulted from a CI test of higher order that contradicted another immorality resulting from a CI test of lower order, it would possible to take some measures. However, in these cases and also in the ALARM network, these reversed edges are formed in same round: the direction of the edge is obtained randomly; it follows from the ordering of the variables. In these two cases and in the ALARM network, the number of reversed edges in the beginning is high and in the end they are reduced (in the financial dataset more than thirty percent). However, these reversed edges account for almost fifty percent of the total number of edges in the learned financial network as opposed to around twenty percent in the learned ALARM network. The main reason is that the algorithm locates a substantially lower frequency of v-structures for the ALARM network than for the financial data set. The large

number of immoralities learned, from a distribution that possibly does not have a faithful graph, leads to more reversed edges. These are the principal reasons that there are more reversed edges in the final graph. It is important to point out that some of these reversed edges may not be caused directly by immoralities, but rather by the later application of Meek's rules.

The results are similar for the two thresholds, 0.05 and 0.01. Although the number of similar edges in both cases are high, one is not a subset of the other, since different results on the CI tests at an earlier stage can lead to a different learned graph, with different orientation of the edges, consequently different ancestor and descendant substructure, which means that different conditional independence statements are tested

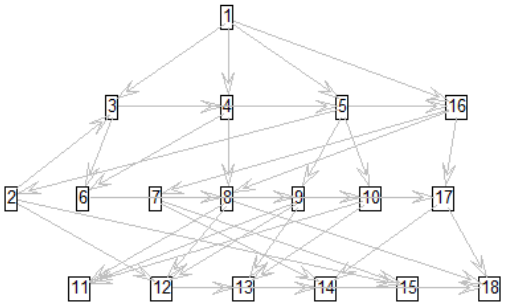


Figure 11: Learned Structure with an alpha of 0.01

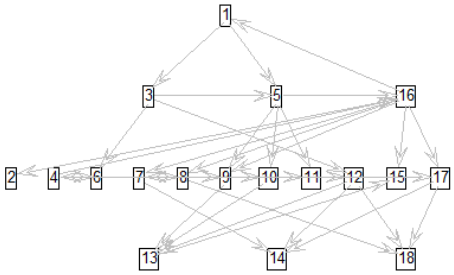


Figure 12: Learned Structure with an alpha of 0.05

4.7.2 Alterations 1, 2, 3, 4

The three proposed alterations to the structure of RAI are analysed empirically with the financial dataset in this section. The first alteration, Essential Graph, aims to decrease the risk that the learned structure may not be an essential graph by removing the directions of the edges before applying the orientation rules at each stage. The table shows an increasing number of the reversed edges when this solution is implemented, 4 and 8 for each case, and a decreasing on the number of total edges for both cases. In this graph, the reversed edges account for more than half the total edges which are not so much attractive. The second alteration, Compelled Edges, where the compelled edges are only directed in final step of the code, does not produce different results when it is applied to this dataset in either case, with 0.05 or 0.01 threshold. The application of the compelled edge orientation rules during the learning process leads to an unnecessary increasing run time. The third alteration, D-connections, aims to eliminate the possibility of the algorithm removing edges during CI tests of higher order that

contradict dependency relations verified with CI tests of lower order. This modification to the code leads to a rise on the number of edges that are kept which is also accompanied by an increase in the number of reversed edges. The proportion of the reversed edges and total number of edges is lower compared to the results obtained using the RAI without this alteration. Without making any modification to the RAI code, the learned networks for a threshold 0.01 and for 0.05 are not acyclic. The implementation of the alteration Without Cycles decreases the total number of edges and the number of reversed edges and the decrease is substantially more when the threshold of 0.05 is used:

Table 5: Result of RAI with Alteration 1, 3, 4 with categorical variables of the financial data set.

	Threshold	Directed Edges	Undirected Edges	Total	Number of same edges	Reversed edges in the learned network
Alteration 1 - Essential Graph	α= 0.01	39	0	39	28	24
	α= 0.05	39	0	39		26
Alteration 3 - D-Connections	α= 0.01	51	0	51	24	29
	α= 0.05	51	0	51		24
Alteration 4 - Without Cycles	α= 0.01	40	0	40	26	19
	α= 0.05	32	0	32		12

4.7.3 Continuous Variables with Fading

The variables for the financial data set are continuous. As explained above, in the section 2.3, the observations are submitted to a fading process since associations are considered to be changing and the old observations are less important than the new ones for determining a graph that describes the current situation. The original code of the RAI algorithm proposed by Yehezkel and Lerner, 2009, only runs with categorical, but it was altered to perform conditional independence tests for continuous variables, by testing for CI using Fisher's test with thresholds of 0.01 and 0.05.

The RAI algorithm, with a threshold of 0.01, starts by performing CI tests with order 0 but does not find any conditional independence statements. The ancestor set is the empty set and the descendant set is the set of all variables. With CI tests of order one,

73 conditional independence statements are obtained. These give 238 v-structures with 64 edges directed in both directions (reversed edges). The descendant substructure is now the empty set and the ancestor is the set of the 18 variables. The algorithm with CI tests with conditioning set of size two, identifies 29 conditional independence statements, but there are no new immoralities. Again the ancestor is the substructure with all the variables, being the descendant substructure an empty set. The RAI determines 5 conditional independences statements with a conditioning size three and no new immoralities are obtained. The descendant substructure is formed by the variable 14 and the ancestor substructure is the remainder of the variables. The CI tests do not locate any more conditional independence structures and the algorithm does not orient any compelled edge. From the 46 edges in the final learned network, 33 were reversed edges from the initial stage. The learned structure is not acyclic and it is presented in the figure 13.

Table 6: Result of RAI algorithm with continuous variables of the financial data set.

	Directed Edges	Undirected Edges	Total	Reversed Edges during the learning process	Reversed edges in the learned network
α= 0.01	46	0	46	64	33
α= 0.05	49	0	49	74	44
Number of same edges	42	-	-	-	-

The learned structures with thresholds 0.01 and 0.05 do not have any undirected edges since all the edges belonged to an immorality at some stage of the learning process. The number of edges that are the same in the learned structure indicates that the edge set produced when a threshold of 0.01 is used is a subset of the edge set produced with a threshold of 0.05 (more edges are removed with a higher threshold) and also that the substructures obtained in both learned structures are similar (that is, the CI tests were performed on similar sets of variables in both cases). The number of reversed edges in the learning process is similar to that produced when the algorithm is run on discretised categorical variables. The number of reversed edges remaining is rather high, indicating that there may not be a faithful DAG corresponding to the underlying probability distribution. Unfortunately, with the real data sets it is no possible to determine whether or not the independence statements come from distribution that has a faithful DAG.

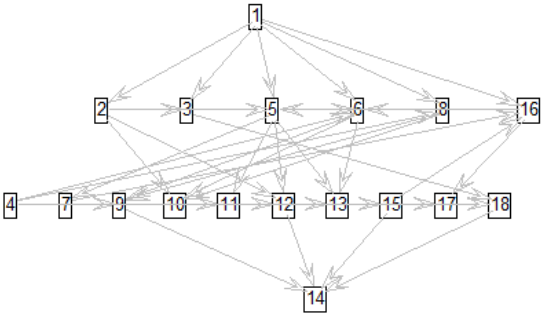


Figure 13: Learned Structure with an alpha of 0.01

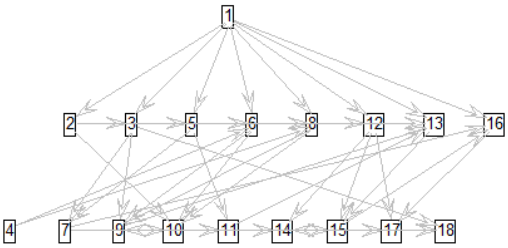


Figure 14: Learned Structure with an alpha of 0.05

4.7.4 Alterations 1, 2, 3, 4

Although the graph produced contains contradictions that indicate that the graph by no means represents the complete independence structure, the outcome does indicate some of the key associations between the variables. To reduce the number of contradictions, some of the alterations proposed in the section were applied to the continuous variables with a fading process. The first alteration proposed implies the addition of the compelled edges in the end of the code. In both cases, there are no compelled edges and so this alteration does not produce a different result. The alteration Essential Graph undirects the edges before applying the orientation rules at each stage of the RAI code. There is no reason to apply the alteration D-Connections when continuous variables are used, since the CI tests of higher order are not less reliable than the CI tests of lower order. The fact that the above learned structures were not acyclic suggests the alteration Without Cycles. The table shows the results for the alteration Essential Graph and Without Cycles. Unlike with categorical variables, the alteration Essential Graph does not alter substantially the total number of edges and the number of reversed edges in the learned network for either threshold used: 0.01 and 0.05. This is because the variables in the ancestor and descendant structures are the same in most stages of the RAI, with and without the alteration. This means that the pairs of variables on which the CI tests are performed are the same. The alteration Without Cycles removes in each case one edge but does not decrease the number of reversed edges, because the edge eliminated was not a reversed edge.

Table 7: Result of RAI with Alteration 1, 4 with continuous variables of the financial data set.

		Directed Edges	Undirected Edges	Total	Number of same edges	Reversed edges in the learned network
Alteration 1 - Essential Graph	α= 0.01	46	0	46	43	33
	α= 0.05	50	0	50		44
Alteration 4 - Without Cycles	α= 0.01	45	0	45	41	33
	α= 0.05	48	0	48		44

The RAI algorithm with the financial data set with continuous variables and a fading process possibly indicates that the faithfulness assumption is not present. However, if the continuous variables are not submitted to a fading process the results are much better. The next section presents the behaviour of the RAI code with the financial continuous variables, without fading, using a threshold of 0.01 and 0.05.

4.7.5 Continuous Variables without Fading

The behaviour of RAI described below is for a threshold of 0.01. In the first round, for CI tests of order zero, five conditional independence structures are identified, which leads to the formation of 60 immoralities. The ancestor substructures are obtained, {1, 2, 3, 5, 14} and {18}, and the descendant substructure is the variables not contained in the ancestor, {4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17}. For the first ancestor, the CI tests of order 1 are run and one pair of variables is identified as conditionally independent. Two v-structures are added to the list of v-structures and the ancestor {1, 2, 3, 5, 14} is formed and the descendant is also formed, {}. CI tests are performed and three conditional independence statements are identified with a conditional set of size three (no conditional independence statement is determined for a conditional set of size two). No immoralities are formed and the ancestor substructures are {1, 2, 5, 14} and {3}.The exit condition is met for both ancestor and for the ancestor of round 2, {18}. The algorithm proceeds to the descendant substructure {4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17} where it finds 63 conditional independence structures, leading to the formation of 58 more immoralities. There are 22 reversed edges. The ancestor substructures are identified as {4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16} set and {17} set. CI tests are performed for an order of one and 26 conditional independence statements are obtained. Any immorality is added and the algorithm obtains the ancestor {4, 6, 7, 8, 9, 10, 16}

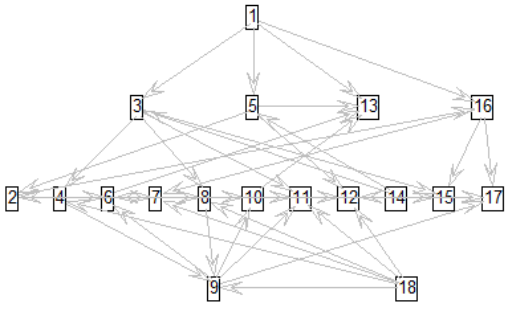
and the descendant {11, 12, 13, 15}. For the ancestor, two CI statements are identified with a conditioning size of three and for the descendant five CI statements are identified with a conditioning size of two. In the end, the algorithm adds two compelled edges and the final learned structure is not acyclic.

The table shows the final results for the learned structure with a threshold of 0.01 and 0.05.

Table 8: Result of RAI algorithm with continuous variables without fading of the financial data set.

	Directed Edges	Undirected Edges	Total	Reversed Edges during the learning process	Reversed edges in the learned network
α= 0.01	48	0	48	22	12
α= 0.05	50	2	52	24	14
Number of same edges	45				

The edges retained in the learned structure for an alpha of 0.01 are subset of the edges retained in the learned structure for an alpha of 0.05. This last one presents two unoriented edges. The final number of reversed edges is lower, compared with the continuous variables with fading process and it is similar to the proportion present in the ALARM network, indicating that the algorithm is running better with this data set. The above figures show the learned structure for each iteration with thresholds of 0.01 and 0.05 respectively.



: Figure 15: Learned Structure with an alpha of 0.01

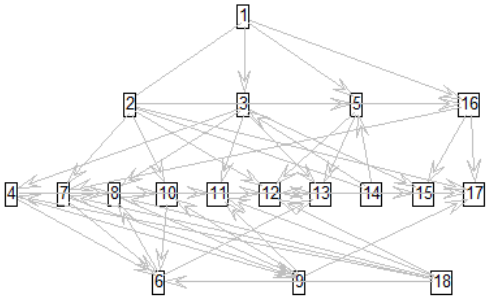


Figure 16: Learned Structure with an alpha of 0.05

4.7.6 Alterations 1, 2, 3, 4

The alterations presented in the section 4.2.1, 4.3.1, 4.4.1 and 4.5.1 were also applied to this dataset. The alteration where compelled edges are only directed at the end of the algorithm does also not alter the learned model in this case, but decreases the run time. The second alteration, Essential Graph, which undirects the edges before directing the immoralities again, leads to a higher total number of edges, eight edges for both threshold values. The ancestor and descendant substructures are formed are differently. The alteration also leads to a substantial increase in the number of reversed edges, almost four times as many. The alteration Without Cycles removes for both case two edges without altering the number of reversed edges.

Table 9: Result of RAI with Alteration 1,4 with continuous variables without fading of the financial data set.

		Directed Edges	Undirected Edges	Total	Number of same edges	Reversed edges in the learned network
Alteration 1 - Essential Graph	α= 0.01	56	0	56	47	49
	α= 0.05	58	0	58		50
Alteration 4 - Without Cycles	α= 0.01	46	0	46	43	12
	α= 0.05	48	0	48		14

4.7.7 Continuous and Categorical Variables with Fading

Financial researchers are interested in learning whether there are relationships among the variables, but, at first glance, they want a graph that indicates whether or not there is a relationship; they do not want to restrict attention to relationships of a linear nature or non-linear nature. This section presents an alteration to RAI algorithm in order to learn the associations between the variables that can be from different types: linear or not linear. The algorithm runs two CI independences tests: one for the continuous variables using the Fisher test based on correlations that detect linear relationships and the other for the discretised categorical variables. The running time of the algorithm is proportional to the number of statistical calls made, references to the data set to get the

information to perform a CI test. The following code is used to decrease the number of CI tests performed and hence decrease the running time.

For each pair of variables in the analysis, a CI test is performed on the continuous data. If the null hypothesis is rejected, which means that the dependency is accept, the algorithm proceeds to the new pair of variables. If CI test does not reject the null hypothesis, the algorithm proceeds to tests for CI with the discretised variables. The algorithm only removes the edge between the variables if both CI tests obtain a conditional independence statement.

Learning Skeleton: Continuous and Categorical Variables: The algorithm has as input the a list of pair of variables to test the CI, List, a threshold, α, a continuous data set, CoD, a categorical data se, CaD, and a the conditioning set, S

```
1: Learning Skeleton (List, α, CoD, CaD, S)
2: Loop i
3:      Result_Co =CI test (List(i), CoD, α)
4:      If Result_Co =1
5:        Result_Ca=CI test (List(i), CaD, α)
6:      End
7:      If Result_Ca=1
8:        Remove the edges between List(i)
9:      S← S U {Lsit(i)}
10:     End
11: End
```

Two data sets are used: the continuous data set with the fading process and the categorical data set (the continuous data set discretised) with also a fading process. The tests are performed both with a alpha of 0.01 and 0.05

With a threshold of 0.01 the algorithm starts to perform both CI test for a conditioning set of size 0. However, no conditional independence structure is obtained for either data set. The ancestor substructure is the set of all variables and the descendant is the empty set. For CI tests of order one, 78 conditional independence statements are obtained and consequently, 219 immoralities are formed, with 56 reversed edges. The ancestor substructure is the set of all variables and descendant is the empty set. With a conditioning set of size two, 39 conditional independence statements are identified, removing 29 reversed edges. No immorality is formed and the exit conditioning for the new ancestor and descendant substructures is met. In the figure 17, the learned model

with the 0.01 threshold is described. Figure 18 shows the learned model with a 0.05 threshold. The table presents the number of total edges, the number of directed and undirected edges, the number of edges that are the same and the number of reversed edges during the learning process and at the end.

Table 10: Result of RAI algorithm with categorical and continuous set of variables of the financial data set.

	Directed Edges	Undirected Edges	Total	Reversed Edges during the learning process	Reversed edges in the learned network
α= 0.01	36	0	36	56	27
α= 0.05	39	0	39	70	35
Number of same edges	33	-	-	-	-

The algorithm behaves in a similar way with both the threshold values, of 0.01 and 0.05 there is a high number of edges that are the same in both learned models. The learned structure with a the threshold of 0.05 is acyclic but is not acyclic for the threshold 0.01.The number of reversed edges produced during the learned process is similar to other experiments (for example, with the ALARM network), but in the end the number of reversed edges is higher. The number of reversed edges is higher when a value for alpha of 0.01 is used, because the number of reversed edges in the beginning is also high, although the number of conditional independence statements identified with a conditioning set of size of two is larger than the ones learned with an alpha value of 0.05.

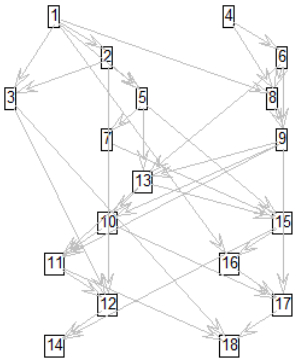


Figure 17: Learned Structure with an alpha of 0.01

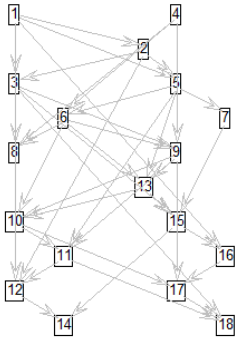


Figure 18: Learned Structure with an alpha of 0.05

4.7.8 Alterations 1, 2, 3, 4

The RAI algorithm is altered according the alterations 1, 2, 3 and 4 proposed. As with other data sets, the results of the RAI code which only directs of the compelled edges at the end of the algorithm does not alter the learned structure, it only decreases the run time. The alteration Essential Graph presents the best results. The number of reversed edges decreases sharply in both learned structures. For an alpha value of 0.01, the total number of edges also decreases.

The alteration D-connection prevents the algorithm from assuming conditional independence statements when two variables have been determined as dependent in a previous step of the algorithm. The number of total edges is larger than with the other alterations since the learned Bayesian network assumes more dependencies which are missing in the other learned networks. The number of reversed edges in the learned network also increased substantially.

The alteration Without Cycles only modifies the learned model with an alpha value of 0.01, but does not alter the number of reversed edges.

Table 11: Result of RAI with Alteration 1,3,4 with categorical and continuous set of variables of the financial data set.

		Directed Edges	Undirected Edges	Total	Number of same edges	Reversed edges in the learned network
Alteration 1 - Essential Graph	α= 0.01	29	4	33	28	11
	α= 0.05	36	0	36		23
Alteration 3 - D-Connections	α= 0.01	42	0	42	34	37
	α= 0.05	44	0	44		53
Alteration 4 - Without Cycles	α= 0.01	34	0	34	32	27
	α= 0.05	39	0	39		35

Chapter 5

Incremental Learning Structure

5.1 Introduction

Incremental learning algorithms are used when the observations of a data set follow a time series. Thousands of financial data are generated daily, which makes Incremental Learning Algorithms a very useful tool for learning dynamically the associations between the variables. There are many incremental structure learning algorithms available, each having different assumptions and different trade-offs between accuracy, computational complexity and memory requirements. Most of these algorithms are only able to deal with new observations that come from the same underlying distribution as the old observations and are not able to deal with a change in the domain of the variables. In the first part of the chapter the proposed incremental structure learning algorithms are described. The second part of this chapter presents the theoretical background needed for the third part of this chapter, specifically, detection methods and Bayesian network score functions. In the third part three new incremental learning algorithms are proposed, which are able to deal with observations that do not come from a stationary domain, but rather a piecewise stationary domain. After the description of each new incremental learning algorithm, experiments are carried out with simulated data to test the ability of the algorithm to detect a change and to incorporate that change into the learned model. It was decided to use categorical variables in the incremental learning algorithms, since the discretised data seems to produce more satisfactory Bayesian networks. The fourth part of the chapter applies the three incremental learning algorithms developed in the thesis to the financial data set for categorical variables submitted to a fading process.

5.2 Score Functions for Learning Bayesian Networks

There are two approaches for scoring a Bayesian Network in terms of how well it fits the data (Campos, 2006). The first is to use a Bayesian score function, where the posterior probability distribution given the dataset, P(B|D) is computed. When a uniform prior is used over all Bayesian networks, so that P(D) is equal for each structure D, the logarithm prior probability log(P(B,D)) gives the same result and may be easier to compute. Examples where this approach is used are K2 (Cooper and E. Herskovits, 2002) or BDeu (Heckerman and Chickering, 1995). The second approach is based on information theory, which encodes the minimum number of elements that are necessary to represent a message; in this case, the minimum number of edges required to represent the data. Some score functions based on information theory add a factor of penalization depending on the complexity of the network as the AIC function (Akaike, 1974).

In this thesis, the Minimum Length Description [MDL] will be used. MDL is composed by two terms: the first measures how well a given network fits the data and the second is a penalty term based on the complexity (over fitting):

$$\sum_{i=1}^{n} \sum_{J=1}^{q} \sum_{k=1}^{r} N_{ijk} log\left(\frac{N_{ijk}}{N_{jk}}\right) - \frac{1}{2} C(G) \log(N) \qquad (4.1)$$

C(G) is the complexity measure, which is proportional to the number of free parameter of the factorised joint probability distribution. The BIC function criterion (Scwarsz, 1978) is similar to the MDL (Campos, 2006). The MDL was chosen for this thesis because it does not require prior parameters and it has been used extensively as metric of comparison.

5.3 Incremental Learning Structure Algorithms

Incremental structure learning algorithms aim to learn the Bayesian network from data that is received sequentially. The algorithm starts by learning the structure of the network using the available data. When a new observation is incorporated, an update function is triggered. When a new instantiation is received, part of the learned structure, or even the entire learned structure, can be updated. Different algorithms have been proposed which give different answers to the questions of how to decide when the

updating function should be called, what part of the structure should be updated and how it should be updated (that is, which structure learning algorithm should be used). Six main approaches have been proposed (Buntine 1991, Lam and Bacchus 1994, Friedman and Goldszmidt, 1997, Roure 2004 and Nielsen and Nielsen 2006, Shi and Tan, 2007). A brief summary of these approaches to the problems is described below.

5.3.1 Buntine Incremental Learning Algorithm

The incremental structure learning algorithm proposed by Buntine, 1991 is an extension of his batch structure learning algorithm, based on a search and score principle which ranges from a greedy search to a beam search (a search algorithm that explores a graph by considering the most promising node in a limited set). After learning the structure when one or several new observations are received, the algorithm can take one of two approaches depending on the computational power available. The first only updates the posterior probabilities in the light of the new data. In the second approach after each update, two sets of information are stored: the parent sets that present a better score and the corresponding learned parameters of the structure. After receiving a new instantiation, the learned structure is revised by updating the posterior probabilities of the parent sets – alive list, deciding on the best set of parents for the variables in the analysis (that is, searching in the neighbourhood of the current structure: a variable is added to the parent set if this improves score of the structure) and identifying others variables that should be also updated – open list (Buntine, 1991).

5.3.2 Lam and Bacchus Incremental Learning Algorithm

The incremental structure learning algorithm proposed by Lam and Bacchus is based on the assumption that the structure learned so far is fairly accurate, given the data available (Lam and Bacchus, 1994). When there is new information available, it calculates the Minimum Length Description (section 5.2) for the new data with different proposed partial networks. It chooses the partial network that has the lowest MDL. To learn the structure of the partial network, it uses their batch version algorithm which is also based on a principle of search and score. Having decided on the best partial network, the complete learned structure of the old data set is refined using the partial network in such a way that the new learned structure has a lower total description length. This method supposes that a substantial number of instantiations are received, at

least sufficiently many to represent the underlying probability distribution so that the algorithm can compute the partial network.

5.3.3 Friedman and Goldzmit Incremental Learning Structure Algorithm

Friedman and Goldzmit present three approaches for Incremental Learning that range from assuring a better robustness or assuring that information warehoused is lower (Friedman and Goldzmit ,1997). For assuring a better accuracy, they proposed an algorithm that each time an observation is received the batch algorithm is run again, leading to a huge increasing in the computational complexity; a new structure is learned each time the full data set is kept in the memory at each stage. The approach in the other extreme is similar to Butane algorithm but only records the Bayesian network learned for the available data. Searching for the network that has the maximum a-posterior probability, that is, the network that is most probable given the available data, when a new instance is obtained, the posterior probability is update by recurring to this network, with the appropriate sample size, as a prior. The third approach is a better trade-off between quality and information kept. It is only kept the information that is statistical sufficient to evaluate each candidate networks in the next decision, by using heuristics approaches to pruning the information. The algorithm relies on maintaining a set of candidate structures that are on the frontier of the process that is, keeping the networks that seem more promising in the current time. After evaluating the candidates on the frontier, the changes are applied, by adding, deleting or reversing an edge from the current one, as observations are received. The next step is to call the search procedure to determine the next frontier and the information that is recorded (Friedman and M. Goldszmidt, 1997). The function for updating is called after receiving a pre-specified number of instantiation. Because only the statistical information is kept, the process of choosing the candidate structure is dominated by the most recent observations (Tien et al, 2001) (Friedman and M. Goldszmidt, 1997). A lower number of instantiations incorporated can lead to a decrease of the robustness of the algorithm and a larger number of instantiations can lead to a decrease on the efficiency (Tien et al, 2001). Contrary to Buntine's algorithm, this algorithm can consider as candidates structures that were not considered promising in previous stages.

5.3.4 Roure Incremental Learning Structure Algorithm

Roure proposes another incremental structure learning algorithm that relies on the batch hill-climbing algorithm, but which can also be applied to other search and score algorithms as K2, Cl and B (Roure, 2004). To construct an incremental algorithm, two heuristic are used. The aim of the first is to decide when the updating function should be triggered. This is achieved by revising the learning path used to learn the current network, the transversal operators, ordered in decreasing contribution of quality. A transversal operator, in the Bayesian network domain, is a set of operations (for example, adding or removing an edge) for an argument, which is the variable that will be added or removed. That is, a transversal operator given the argument and the actual model represents a new model which takes into account the old model and the alterations represent by the operators and the arguments. A learning path is, literally, the sequence of pairs of operators and arguments that were used to build the sequence of intermediate models. This sequence of models (pairs of operator and pairs) is ruled by the contribution of quality which means that intermediate model has a lower score than the posterior and higher than the previous one. If there is a change in the order, it means that the Kullback Leibler distance is larger than a pre-specified threshold and so the current Bayesian network no longer fits the data well when the new observations are incorporated.

The second heuristic is used to reduce the search space by eliminating the candidate parents that had a low score in the previous stage by using a greedy hill-climbing search. Rouse also proposes two heuristics to deal with the problem of the time spent going through the entire database and the amount of space consumed by database. These heuristics present a method of pruning the information, storing only the sufficient statistics. Roure showed experimentally that his approaches to incremental learning can obtain a learned structure of similar quality to the one learned by the batch algorithm, mitigating the fact the algorithm can get stuck at local maxima. While decreasing the learning time, he showed empirically that the algorithm has low sensitivity to the order that the instantiations are received. These two characteristics answer to the two main problems of learning incremental when the observations come from the same domain (Roure, 2004).

5.3.5 Nielsen and Nielsen Incremental Learning Structure Algorithm

Nielsen and Nielsen present a different approach for incremental learning (Nielsen and Nielsen, 2006). Instead of assuming that the new instances comes from the same distribution and that therefore more information will give a more robust result as with the previous algorithms, they assume that the underlying probability distribution of the domain may change over time. The changes in the domain are assumed to be piece-wise, which means that the domain is stationary before and after the change. These changes are also assumed to take place in a small subset of the learned structure. That is, from the previous interval of the domain to the current one, only associations between variables in a small subset of the total variable set have changed. The handling of the drift concept (that is, the changes in the domains or the sets of neighbours for the variables) is made by measuring the scores of each variable when new data is received in the light of the current structure learned. If the second cosine transform of the most recent 2k scores for each variable is greater than a pre-specified threshold, it indicates that there has been a change in the set of neighbours of the variable and so the updating function must be called.

This approach means that a drift is detect with a delay of k instantiations. This algorithm differs from the previous algorithms by using a constrained based batch approach which assumes that there is a faithful graph for the distribution. The update function learns the new associations of the set of variables whose neighborhood have changed. This is obtained by performing conditional independence tests in three different phases. The first consist on determining the Markov Blanket, the set of parents, children and parent's children of the variable currently being examined, through the use of a decision tree. The second aims to determine which variables of the Markov blanket are adjacent (parents and children) to the variable currently being examined. The third set of CI tests identifies the orientation of edges, which means determining which variables are the parents and which are the children of the variable currently being examined. Through experimental verification, Nielsen and Nielsen argue that this method outperforms the results of the Friend algorithm with non-stationary domains, being more stable in the sense that it does not update the network so often and it updates fewer variables. However, the experiments also shown that when they are close to stationary domains the approach of Friedman can be better (Nielsen and Nielsen, 2006).

5.3.6 Shi and Tan Incremental Learning Structure Algorithm

Shi and Tan have proposed an incremental structure learning algorithm for financial data modelling that runs a hybrid updating function instead of using a search and score approach. The incremental learning algorithm is composed of three main parts. The first is a function which is responsible for indicating if the current Bayesian network should or should not be updated in the light of new instantiations. This indicator function uses the ratio between the MDL score of the network given the dataset before receiving the new information and the number of observation and the ratio between the MDL score of the network after the new information is received, which is compared with a given threshold value. This indicator function will be explained more fully in the section 5.4.2, but it is important to point out that, as in Nielsen and Nielsen, it indicates which variables should be relearned.

The second step is the updating function, which is partly similar to the constrain-based algorithm Three-Phase Dependency Analysis, TPDA which has a main assumption: a monotone DAG-faithful underlying distribution (Cheng et al., 2002). It uses the first two steps of the TPDA to learn the undirected Bayesian network: the first step consist of testing whether the mutual information between each pair of variable is greater than a threshold and adding an edge between the two variables when there is a positive answer; the second step determines if the variables in the pair are conditionally independent conditioned on some relevant cut-set, removing the edge when this conditional independence is established (Cheng et al., 2002). The skeleton learned by the two steps described above allows the candidate parent set to be built for each variable, the variables that have an undirected edge between them, corresponding to the third step of the updating function.

By using the heuristic minimal-redundancy-maximal-relevance, mRMR (Peng et al., 2005), different parents sets are built from the candidate parent set of each variables, where the first set has only one parent and the last set m parents. Using the score of the each parents set as the criterion, the set with highest score is selected. The construction of the parent set for each node is not yet finished; for each node it is tested whether retaining or removing each of its pre-selected parents will give a higher score. At this point of the algorithm, the authors have added a constraint that is necessary for real applications, the maximum number of parents that each variable is permitted to have. When the number of variables rises, the two steps described above require a substantial amount of time because of the increased complexity, which is reduced by the proposed constraint. Although the constraint has the advantage that it reduces complexity, it leads to cycles and to undirected edges that should be directed. In the article, a greedy search to remove the cycles is suggested: in each cycle the score of the network is tested to determine whether each edge should be removed or redirected and the alteration that gives the greatest improvement to the score is kept.

The v-structures are formed using the value of the conditional mutual information, but instead of comparing this value to a threshold, the value of conditional mutual information of the two variables given the empty set is compared to the conditional mutual information of the two variables given a third variable (if it is lower, the algorithm forms the immorality). The location of the children variables is through a similar method of the immoralities: if the value of conditional mutual information of the two variables X and Y given the empty set is lower than the conditional mutual information of the two variables given a third variable Z and Y is the parent of Z, there is no other adjacency path between the two variables and the edge between X and Z is not directed, then X is child of Z. After having directed the edges, the algorithm runs the third step: by removing the nodes in a random order, the algorithm tries to identify the best network corresponding to the order of removal.

5.3.7 Comparison

The main focus of the first five algorithms is to reduce the information that is stored and to use a search and score approach to update the function. In contrast, the two last algorithms use a constraint based approach to learn the network (Shi and Tan use a hybrid method). The Buntine and Friedman approaches try to restrict the search space for the candidate structures and updates the function when a few new instantiations are received. On the other hand, Lam updates the learned model after receiving enough information to learn the partial network. With the algorithm of Roure and the algorithm of Shi and Tan, the current network is revised when it no longer fit the data set, assuming that the observations are independent instantiations from the same distribution. The Nielsen and Nielsen approach assumes that the process is piece-wise stationary.

The Roure, Lam and Bacchus and one of the versions of Friedman and Goldszmidt algorithms trigger the update function each time a new instantiation is received, which is very time consuming. Instead, Nielsen and Nielsen and Shi and Tan use a function to detect a change in the domain which calls the update function when necessary, decreasing substantially the run time. It is also seems to give more accuracy than other Friedman and Goldszmidt, which updates the current network when a pre-specified number of new instantiations have been received. For these reasons, the performance of the incremental learning algorithms, proposed here, are compared with the Nielsen and Nielsen and Shi and Tan algorithms.

5.4 Detection Methods

Incremental structure learning algorithms can either assume that the observations received come from a single underlying distribution or, on the on the other hand, there are shifts in the domain. This thesis aims to proposed incremental learning algorithms that are able to cope with datasets that have different distributions and respectively parameters that change with time due to the fact that the associations between the financial variables change with time. To trigger the updating function, the algorithm should be able to detect the change in the associations. The change is considered as a change in the independence structure or in the parameters which occurs rapidly in the sampling period, it is stationary both before and after the change (Basseville and Nikiforov, 1993).

In the literature, many methods for detection have been proposed and to built the right one some questions should be taken into account: which variables are most useful for detecting the change; which information is available, for example is the magnitude of the change known and what will be the rule to the trigger the updating function in the light of the variables used. The detection algorithm should be also evaluated by the probabilities associated with the occurrence of error *type I* (performing a false trigger) and with the occurrence of error *type II* (failing to trigger the update function when there is a change). Contrary to detection off-line that aims to detect a change or non-existence of change in the whole data set, the detection on-line tries to detect a change with only the information available up to the moment, t_o. This type of detection is concerned with activating the updating function as early as possible while not producing false alarms. Broadly speaking this type of detection uses a rule of stopping: when the function analysed overtakes a predetermined value, the frontier, the updating function is called.

Given the aim of the incremental structure learning algorithms, an on-line detection method should be used and the time the method takes to learn that a shift occurred should be evaluated. The number of false detections (*error I*) and the number true changes not detected (*error II*) should be counted. In the thesis three methods of detecting change will be analysed: the CuSum (Page, 1954), the Second Cosine Transform and the Indicator Function mention by Shi and Tan (Shi and Tan, 2007).

5.4. 1 The CUSUM

The CUSUM algorithm is a method of detecting changes in the parameters of a distribution, usually the mean. The cumulative sum of a likelihood ratio is compared with a threshold value and an alarm is triggered if the cumulative sum crosses the threshold. Let y_i $l \leq i \leq n$, denote a sequence observations from independent random variables, each with a probability function p_θ (y_i) that depends on a parameter θ. Suppose that there is a change in the parameter value at observation t; for observations { y_1, ..., y_t } the value of the parameter is θ_0 and for observations { y_{t+1},....} the parameter value is θ_1. Suppose that only two parameter values, θ_0 and θ_1 are being compared. For observation i, the likelihood ratio is:

$$s_i = \ln \frac{p_{\theta 0}(y_i)}{p_{\theta 1}(y_i)} \qquad (4.2)$$

If the observation y_i is more likely with a parameter value θ_0, then s_i is positive; if it is more likely with a parameter value \theta_1, then s_i is negative. The observations { y_1, ..., y_t } should give positive s_i, supporting parameter value θ_0, while the observations after observation t should support a parameter value of θ_1. The behaviour of the cumulative sum should indicate when a change-over has taken place.

The parameter values θ_0 and theta_1 are required for the calculation of the likelihood ratio. The initial value, θ_0, is estimated from a predetermined number of observations that come from a stationary domain. The value of $\theta_1$is computed by deciding the magnitude of difference from θ_0 that constitutes a change in the parameter value. That is, how much should the parameter θ_1 differ from θ_0, to consider that the domain had changed. The parameter θ_1 may be rewritten as $\theta_1 = \theta_0 + v$, where v represents the predefined value of magnitude that constitutes a change. This method of comparing the value of θ to θ_1 and to θ_0 is equivalent to making a hypothesis test and deciding whether or not it is possible to reject a null hypothesis and accepting the alternative hypothesis or not given the value of a test statistic. The statistic is the cumulative sum of the likelihood ratio starting from the first sample just received after the last update, i, until the last observation received at the moment, k.

$$H_0: \theta = \theta_0$$

$$H_1: \theta = \theta_1$$

$$S_k = \sum_{i=j}^{k} s_i \qquad (4.3)$$

When the null hypothesis is true, the data should give negative values for the likelihood ratio and the cumulative sums, S_k, will, on average, decrease. When null hypothesis is no longer true, which means that a change occurred (the value of θ is closer to θ_1), the data should give positive values for the likelihood ratio and the cumulative sum should start, on average, to increase. The change in the parameter θ can be detected when the cumulative sum S_k has increased sufficiently. The minimum value of the cumulative sum should represent the precise point when there was a shift in the parameter. The CUSUM method for detecting this minimum uses a decision rule: if the difference between the actual value of S_k and the recorded minimum value of cumulative sums until the moment is larger than a threshold value, there had been a change:

$$g_k = S_k - m_k \geq h \qquad (4.4)$$

Where

$$m_k = mim_{j \leq l \leq k} S_k \qquad (4.5)$$

The alarm is triggered, indicating that there was a shift, when the g_k is larger than the threshold. This threshold, although partly predefined, it is adaptive. It is able to adapt in real time, because S_k is compared with m_k+ h where m_k is changing. When the cumulative sums are decreasing, it follows the movement but when the minimum cumulative sum is achieved, it keeps that same value.

5.4.2 The Indicator Function

Shi and Tan in the article `Incremental Learning Bayesian Networks for Financial Data Modeling' proposed a method to indicate that the update procedure should be called (Shi and Tan, 2007). That is, they developed a specific method for evaluating the goodness of fit for a Bayesian Network to a data set through the use of scores. Before receiving a new instantiation, a Minimum Length Description value, MDL(B,D), is assigned to the network with N samples, and after new information is received, a new score, MDL(B,D'), is computed for the same network using all the data together, the previous data set and the new information, N'. The method calculates the angle, theta, made by the two legs of the right angle with the value of MDL (B,D), and the number of samples N and the angle made by MDL(B,D'), and N'.

$$\Theta = \left| atan \frac{MDL(B,D)}{N} - \text{atan} \frac{\text{MDL}(\text{B,D}')}{\text{N}'} \right| \qquad (4.6)$$

According to the theorem proved in the article, if the learned network is a perfect representation of both the old data set and the new dataset, the angle will be the same. This does not occur in practice and so angle made by the MDL score of the dataset with more observations will be larger than the first one. If the difference between the two angles is larger than a pre-specified threshold, then the Bayesian network is longer the best model to describe the dataset with the new observations. The threshold can be adapted according to the noise in the dataset. (Shi and Tan, 2007).

5.4. 3 Second Cosine Transform

Nielsen and Nielsen in the article Adapting Bayes Network Structures to Non-stationary Domains suggested a different method to detect a shift in the domain (Nielsen and Nielsen, 2008). Instead of using the cumulative sums of the likelihood ratio as statistic in the CUSUM method, the second discrete cosine transform of the last 2k samples is used and compared with a threshold value. This method is widely use in the field of data compression and image processing. As describe in Press et al., the negative second discrete cosine transform component can be calculated using the following formula (Press, et al. 2002):

$$\mathrm{F_k} = \sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathrm{f_j} \cos \frac{\pi k(j+0.5)}{N} \qquad (4.7)$$

With the inverse

$$\mathrm{f_j} = \frac{2}{N} \sum_{k=0}^{N-1} F_k cos \frac{\pi k(j+\frac{1}{2})}{N} \qquad (4.8)$$

The auxiliary function used is

$$y_j = \frac{1}{2}\left(f_j + f_{N-j-1}\right) + sin\frac{\pi\left(j+\frac{1}{2}\right)}{N}\left(f_j - f_{N-j-1}\right) \qquad (4.9)$$

j=0,...,N-1

The even components are calculated from the formula 4.9 and the odd components are determined by recurring downwards, through the formula from k=N/2-1, starting from the formula:

$$F_{2k} = cos\frac{\pi k}{N} R_k - sin\frac{\pi k}{N} I_k \text{ a} \qquad (4.10)$$

$$F_{2k-1} = sin\frac{\pi k}{N} R_k + cos\frac{\pi k}{N} I_k + F_{2k+1} \text{b} \qquad (4.11)$$

$$F_{N-1} = \frac{1}{2} R_{N/2}\mathrm{c} \qquad (4.12)$$

In the Appendix A the code for determining the second discrete cosine transform of the 2k samples is described.

5.4.4 Comparison

The CUSUM has been used widely in diverse areas of detection, giving good results (Gustafsson, 1997). The CUSUM has low computational complexity due to the statistic that it uses and the fact that only keeps in memory the last cumulative sum. In contrast with other method such as control charts, this method uses all the observations received after the last updating. In common with all methods applied to real data where there is noise, this method can only indicate approximately when the change point occurred.

The Indicator Function was developed for data sets where each observation comes from the same underlying distribution as the other observations and so it is not able to deal with changes in the domain of the data set. The experimental results of the Indicator Function proved that it needed different thresholds for synthetic data and for real data to be able to trigger the update function. Real data has noise and so the threshold needs to be adapted for each data set.

The Second Cosine Transform is worse in terms of computational complexity, but has the advantage that, unlike the CUSUM, it does not require a distribution to be specified in order to calculate the statistic. The article did not indicate the extent to which the performance of the algorithm was sensitive to noise in the data sets. Taking into account the advantages and drawbacks of each method, the CUSUM method and the Second Cosine Transform were chosen to detect the change point.

Chapter 6

New Structure Learning Algorithms

6.1 Introduction

This chapter discusses several new incremental learning algorithms for situations where the dependency structure between variables is changing. The various algorithms for structure learning and techniques for change point detection are combined and developed into an algorithm that can learn a structure for financial data sets, detect changes in associations and update the structure accordingly.

6.2 Incremental Learning Algorithm 1

The algorithm proposed in this section is composed of two main functions: a change detection function and an update function. The change detection function aims to indicate when the learned structure should be relearned and the update function aims to learn the network for the new domain. Before proceeding to a detailed description of the algorithm, the main assumptions of this algorithm are stated. The update function uses a constraint based approach, which therefore assumes that there is a faithful graphical model to describe the probability distribution. The change detection presupposes that the domain is piece-wise stationary and that for each stationary interval there are enough instantiations to perform robust conditional independence tests. Another assumption is that there are no missing or hidden values in the datasets. Algorithm 1 relaxes the assumption that when there is a shift of the domain, only the associations of a subset of variables are different; it is able to deal with changes in associations between all variables. Contrary to this relaxation, the Nielsen and Nielsen algorithm is only able

to deal with a change in a subset of the probabilistic relationships among the variables when there is a shift (Nielsen and Nielsen, 2006).

This algorithm can be run with datasets composed with only categorical variables or only continuous variables or with the combination of information from the continuous values of the data set and the categorical values obtained by the discretisation of the continuous variables. The algorithm is also able to deal with data sets where the old instantiations have lower relatively importance compared with new instantiations, through the use of a fading procedure or the elimination of old observations for processes where only new information is relevant.

6.3. 1 Detect Change Function

Starting from a learned network obtained from the few observations possible, when a new observation is received, the detect change function attributes a score to the learned network in terms of how well this fits the new data set, the old observations with the incorporation of the new coming data. This procedure is done continuously until the moment that the score given to the learned model indicates that the learned relations between the variables do not represent anymore the actual distribution (the merging of the old data set and the new instances), which means that there was a shift in the domain. After detecting a change in the domain, the detect change function invokes the updating function.

After obtaining a new instance, the detect change function evaluates the fitness of the all learned network to the set of observations received until the moment through the use of an information theory based score, the Minimun Description Length. The MDL encodes the length of describing the Bayesian network, taking into account a penalization factor for the complexity of the network. In the section 5.3, MDL is explained more detailed. A lower MDL score indicates that the learned model fits better the data, but a single high value cannot be used to prove that there was a change because it can be just an outline value. As though, it is needed to analyze the behaviour of the MDL scores. In this algorithm it was opt to use the CuSum method to detect when the new observations received correspond to a new interval of the domain. This method, besides allowing the detection in real time, is simpler computationally, only recording the information relatively the frontier and the last cumulative sum. This method also uses all the observations received after the last updating. The Cusum method calculates a statistic, based on the cumulative sum of likelihood ratio between the probabilities of the new score tending to a new domain or being closer to the same domain, to compare with a frontier value that adapts along the time. The shift in domain is detected by testing the

null hypothesis, there is no change in the domain, against the alternative hypothesis, if the new observations belong to a new interval. The usually behaviour of the cumulative sums is to decrease, not rejecting the null hypothesis, until the moment it starts to increase, which indicates that the mean has suffered a change. As the frontier is adaptive, a threshold plus the minimum value of the cumulative sums, when the values of the cumulative sums start to rise, the detection function triggers the updating function. The method CUSUM is deeper described in the section 5.4.1. The detect change function code is:

Detect Change Function. Takes the imput as an Learned Network, LN, an set of obervations D,an MDL score and a value of frontier, α

```
1:Detect Change Function (LN,D,MDL, α)
2: Loop i
3:      d←Next Case(D')
4:       Fadinf (D U D')
5:      List(i) =MDL score ( LN, D U D')
6:     CuSum (List)
7:    If  CuSum > α
8:        Updating Function
9:    End
```

6.3.2 Update Function

The change detection function calls the update function when a shift in the domain is detected, indicating that the current learned network does not describe the data set well. In this algorithm, the updating function relearns again all the associations between the variables, to assure a better accuracy of the learned model. The constraint based algorithm used in the updating function is the Recursive Autonomy Identification (Yehezkel and Lerner, 2009). The Recursive Autonomy Identification allows a lower computational complexity, while maintaining higher accuracy compared with many other structure learning algorithms. The RAI is discussed in great detail in the section 2.6. With simulated data sets, the algorithm gave better results than TPDA, PC, MMHC and other algorithms (Yehezkel and Lerner, 2009).

The code of the updating function is:

Updating Function. Takes the input as the set of the observations D, a threshold,β and a type of CI test, CI

```
1: Updating Function (D, β, CI)
2: PDAG = RAI (D, β, CI)
3: LN= Removal Cycles (PDAG, D)
4: MDL score ( LN, D)
```

6.3. 3 Conclusion

The performance of Algorithm 1 compares favourably with the other available incremental structure learning algorithms presented in section 5.3. It deals with changes in the domain of the distribution where the domain is stationary both before and after the change, as with the Nielsen and Nielsen algorithm. Compared with the Nielsen and Nielsen algorithm, Algorithm 1 does not assume that only a small number of associations have changed compared with the previous interval and provides a more robust approach, since it re-learns the entire independence structure. This may lead to an increase the computational complexity, but on the other hand it is based on the RAI algorithm, which is a computationally efficient structure learning algorithm. The algorithms of Shi and Tan and of Roure are also heavy in terms of computational complexity, since they use search and score methods to determine the skeleton.

6.3. 1 Experimental Verification

In this section, the performance of Algorithm 1 is evaluated by analysing its behaviour with a known network The Algorithm 1 can face three main types of changes in the domain: edge deletion, an edge addition and a changed joint probability (the probability of each state of one variable is changed). The performance of the algorithm detecting changes with simulated data will indicate how it will perform on a real data set.

For the experimental verification, the Asia Network with categorical variables was used to test the performance of the algorithm. The performance of the algorithm is evaluated

by looking at three criteria: counting the number of false detections (error type I), that is, the number of times that the detect change function evokes the updating function when there is no change in the structure, true changes not detected (error type II) and the delay in detection.

During the experiments, it was defined, in the updating function that the performance of the CI tests is through the use of the Likelihood Ratio test. The Likelihood Ratio test determines the p-value which will be compared to threshold of 0.01. . If the p-value is smaller than 0.01 the null hypothesis is reject and if not it is accepted. This value is chosen because the aim of the thesis is to learn a network that represents the strongest relations between the variables.

The CUSUM also needs a threshold value to decide if there is a shift or not. This threshold represents a trade-off. A high value decreases the number of false alarms that are caused by fluctuations of the cumulative sums, but increases the delay in the detection of a shift of the domain. The value associated with the alternative hypothesis was chosen according to the variance of the MDL scores; a higher value was used when the variance was larger. In this case, the parameter considered by the change detection function is the mean of the MDL scores and the p-value is based on the assumption that the MDL scores follow, at least approximately, a normal distribution.

The Asia network is available in the package BNT SLP and it contains eight variables and six edges. The experiments were made by generating 1 000 observations in which the first 500 observations come from the real distribution and the other 500 come from the same distribution with one of the alterations: a removed edge or an added edge or a change in the joint probabilities. The data set was also submitted to a fading process.

The first table shows the performance of algorithm 1 detecting the fact that the second half of the observations come from a different distribution. The column `True Detection' (yes / no) indicates if the algorithm was able to detect a change in the domain and construct the new domain correctly. The second column `False Detection' indicates the number of times the update function was wrongly invoked. The `Detection Delay' column gives the number of instantiations required by the algorithm to detect the drift on the domain after the shift had taken place. The fourth column indicates the average running time of the Algorithm 1 in each situation.

Table 12 Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving observations from the altered distribution

	True Detection	False Detection	Delay on the detection	Average Running Time (s)
Edge Deletion	Yes	0	47	12.8
Edge Addition	Yes	0	60	12.9
Changed Joint Probability	Yes	0	54	14.3

A fading factor has applied to the data set since in financial data, the older observations are less relevant than the newer and therefore they should have a lower weight. The Asia Network has categorical data set and so the fading factor is applied through the implementation of a weight, in this case 999/1000, to the contingency table. The fading method is deeper explained in the section 3.4. A parameter v is used to model the alternative hypothesis of the CUSUM test. This is adapted according the variance of MDL scores in order to represent the magnitude of change between the null hypothesis and alternative hypothesis.

The Algorithm 1 was able to construct the initial network correctly – that is, all the statistical associations were detected and there were no additional false associations. In terms of average running time, 'Edge Deletion' and 'Edge Addition' present the lower value. After the change point, the algorithm was able to detect a change in a single edge in the structure and it did not give any false detection signals. Had there been more changes in the domain at the change point, the delay would have been smaller. Clearly, detection cannot happen instantaneously; some data from the new distribution is needed to show that a systematic change has taken place that cannot be attributed to random fluctuations.

The follow table presents, for the same the experiments, the number of updates to the learned network after the algorithm had recognizing the artificial changed. The Correct Updating indicates the number of updates made that correspond to a new learned structure. On the other hand the Wrong Updating indicate the number of updates made that the learned structure was equal to the last learned network. In the seven situations, the New Incremental learning Algorithm identified correctly six new learned structures and it only performs one false alarm.

Table 13 Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving learning the change

	Correct Updating	Wrong Updating
Removed Edge	2	0
Added Edge	2	0
Changed Joint Probability	2	1

The delay on the detection or contrarily a false alarm are dependent on the threshold value used in the CUSUM. If this is large, then a larger number of instantiations from the new distribution is needed to reach the threshold and trigger the update function. On the other hand if the threshold is small there is a greater risk of a false signal. The following table shows a brief study of the effects on changing the threshold in the three situations. When the threshold is increased the algorithms need more observations to detect the change, but they detect the change correctly. If the threshold is decreased in the situation of edge deletion and change joint probability, the algorithm detects the correct change earlier. In the case of edge addition, reduction of the threshold leads to a false alarm.

Table 14: Results of the New Incremental Learning Algorithm 1 when the threshold of the CUSUM changes

	Change in 5 points less the threshold	Change in 5 points more the threshold	Change in 10 points more the threshold
Edge Deletion	53	65	70
Edge Addition	43	53	57
Changed Joint Probability	49	59	63

6.3 The New Incremental Learning Algorithm 2

Algorithm 2 has the same updating function as the Algorithm 1, but the detect change function is different. The Algorithm 1 uses the CUSUM as detection method while the Algorithm 2 uses the Second Discrete Cosine Transform. The Algorithm 2 is able to perform in the same situation as the Algorithm 1 which means it is able to deal with piece-wise changes in the domain of the observations even though there is a large number of variables or the total number of variables with their associations changed.

6.3.1 Detect Change Function

In this algorithm, the Second Discrete Cosine Transform is used instead of the CUSUM method to decide if the learned structure should be revised. The Second Discrete Cosine Transform, as its name indicates, computes the negative second discrete cosine transform component from the last 2k MDL scores and compares the value with a pre-define threshold. If the value is larger than the threshold, the detect change function calls the update function.

Detect Change Function. Takes the imput as an Learned Network, LN, an set of obervations D,an MDL score and a value of frontier, α

```
1:Detect Change Function (LN,D,MDL, α)
2: Loop i
3:    d←Next Case(D')
4:      Fadinf (D U D')
5:     List(i) =MDL score ( LN, D U D')
6:    Second Cosine Transform (List)
7:    If  Second Cosine Transform > α
8:       Updating Function
9:    End
```

6.3.2 Conclusion

Algorithm 2 is similar to Algorithm 1, having the almost the same assumptions but using a different detect change function. The properties of Algorithm 2 are very similar to those of Algorithm 1. The detection of the change point is made using the method used in the Nielsen and Nielsen algorithm. The Second Cosine Transform is computationally more complex than the CUSUM method.

6.6.3 Experimental Verification

In this section, the performance of Algorithm 2 is evaluated by comparing its performance with that of Algorithm 1, using the same simulated data. The same criteria are used; true detection, false detection and delay of detection. The update function uses a Likelihood Ratio test as CI test and the p-value is compared to a threshold of 0.01 as before. The same data as before is used; each data set consists of 1 000 instantiations, the first 500 from one distribution and the subsequent 500 from the same distribution, but with a single alteration. The same fading factor is used.

The results for Algorithm 2 are given in table 14. The first column indicates whether the algorithm detected the change in the domain (Yes/No), the second column gives the number of false detections, while the third gives the delay in detecting the change. The fourth column indicates the average running time of the algorithm in seconds.

Table 15: Results of the New Incremental Learning Algorithm 2 with Asia Network after receiving observations from the altered distribution

	True Detection	False Detection	Delay on the detection	Average Running Time (s)
Edge Deletion	Yes	0	113	13.6
Edge Addition	Yes	0	77	14.7
Changed Joint Probability	Yes	0	87	16.9

Algorithm 2 was able to detect all the changes without producing false alarms. The average running time is bigger for the situation 'Changed Joint Probability' and in all the cases is the Algorithm 2 takes longer time than the Algorithm1. The number of observations until the algorithm triggers the update function is bigger than in the Algorithm1. This may be because the threshold used in the Second Cosine Transform. If the threshold of the Second Cosine Transforms is adapted in order to the Algorithm 2 be able to detect the changes earlier, the following results are obtained:

Table 16: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving observations from the altered distribution

	True Detection	False Detection	Delay on the detection
Edge Deletion	Yes	1	130
Edge Addition	Yes	0	65
Changed Joint Probability	Yes	0	47

The adaptation of threshold for the situation 'Changed Joint Probability' and 'Edge Addition' presents goods results: any false alarm and the delay on the detection is similar to the Algorithm 1. In the case of 'Edge Deletion' the result is worse since there is a false alarm which leads to great delay on the detection. The fact that the Algorithm 2 has a longer average running time and the adaptation of the threshold does not present as good result as Algorithm 1 in the three situations, the Algorithm 2 will not be used with financial data set.

The number of updates made after learning the new alteration is presented in the above table for the three possible situations. When there was a modification in the changed joint probability of one of the variables the algorithm did not make one wrong updating as the New Incremental Learning Algorithm 1.

Table 17: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving learning the change

	Correct Updating	Wrong Updating
Removed Edge	2	0
Added Edge	2	0
Changed Joint Probability	2	0

6.4 New Incremental Learning Algorithm 3

As the first two algorithms, Algorithm 3 aims to detect a change in the domain when observations are received sequentially. The difference is that Algorithm 3 attempts to locate the part of the structure that has changed and only modify that part of the structure. As with algorithms 1 and 2, it is composed of two main functions: a detect change function and an update function. The detect change function is more sophisticated; it looks at each variable and, for each variable, identifies when its parent set has changed. It then evokes the update function, which restricts updating to the variables that have been identified by the detect change function. The update function uses a constraint based algorithm.

As with the other two algorithms, Algorithm 3 assumes that the domain is piecewise stationary and that, for each stationary piece the instantiations are observations from independent identically distributed random vectors and that, for each piece, there is a faithful graphical representation of the independence structure of the probability distribution of the random vector. The algorithm also assumes that all instantiations are complete. As with the other two algorithms, it is supposed that the instantiations are from a time series, where older instantiations give less information about current associations and therefore, as with algorithms 1 and 2, a fading factor is used.

6.4.1 Detect Change Function

The detection of a new shift in the domain is achieved by monitoring a Minimum Length Description, MDL, score for the family of each variable. The MDL encodes the information of the learned Bayesian Network in the light of data set. It aims to use the minimum length to describe this information which means that a Bayesian Network with lower score fits better the data set. . Instead of scoring the complete learned model, the detect change function identifies the families of each variable, that is, determines the variables that are connected with the node being analysed in the learned model and attributes a BIC score. (section 5.3). That is, in algorithms 1 and 2, the goodness of fit of the learned network to the data set is computed using all variables together, while in the algorithm 3 the goodness of fit of the learned network to the data set is measure by considering separately each variable – parent configuration, given the observed data.

Since the Second Discrete Cosine Transform increased computational complexity without delivering any real improvement, it was decided to use the CUSUM method. The detect change function computes the sum of the likelihood ratios between the

probability of an observation coming from a different domain and the same domain. If the CUSUM increases significantly, the alarm is triggered. If the domain does not change, the cumulative sum will, on average, decrease; a significant increase indicates that a change has taken place (section 5.4.1). Upon detecting a change, the detect change function calls the update function, giving it as input the variables whose families should be revised. The algorithm of the detect change function is given below:

Detect Change Function. Takes the imput as an Undirected Learned Network, LNU, an set of obervations D, and a value of frontier, α

```
1:Detect Change Function (LNU,D, α)
2: Loop i
3:       d←Next Case(D')
4:          Fading (D U D')
5:          For  X ∈ V
6:                F=Family (X, LNU)
5:                List(X) =MDL score ( F, D U D')
6:                CUSUM (List (X))
7:                If  CuSum > α
8:                      S ← S U {X}
9:                End
10:         End
11:   Updating Function (S)
12: End
```

6.4.2 Updating Function

The update function re-learns the associations between the variables that were identified by the detect change function. Contrary to the algorithm 1 and 2, the final learned model is undirected since in the financial context the orientation of the edges is not very relevant (particularly when applying the learning model to time series analyses). Although the graph presented is undirected during the learning process, the immoralities are recorded for eventual necessity. The update function starts from the learned network obtained by the RAI and proceeds to undirect its edges, while keeping in memory the v-structures in the first call. After undirecting the learned structure, the update function does not use the RAI algorithm to re-learn the whole graph, but instead calls a

modification of the Fast Adjacency Algorithm to revise the families of the variables indicated by the detect change function. The Fast Adjacency Algorithm performs all the relevant CI tests of one order before proceeding to CI tests of a higher order, as with the RAI algorithm. But unlike the RAI algorithm, the FAST does not introduce the orientation of the edges between of CI tests. Instead, it only applies edge orientation at the end of the algorithm (Fast, 2010). When learning an entire network, this means that a larger number of CI tests have to be performed, leading to greater computational complexity, but also to a higher accuracy of the learned network. For Algorithm 3, the assumption is that a domain change will only occur within a small number of variables and that these variables will have reasonably small families. The number of CI tests required is therefore limited so that the increased complexity of the FAST algorithm is not large and is outweighed by the greater accuracy.

The update function applies the first and second steps of the FAST algorithm; the first is the performance of CI tests and the second learns the immoralities. The immoralities are not added to the structure, but they are recorded. The decision to only record the immoralities, but to use the undirected graph, is due to the context of the use of the algorithm. In the financial context, it is most important to learn the skeleton. The algorithm can be slightly change to incorporate the v-structures.

Updating Function. Takes the input as the set of the observations Undirected Learned Network,an set of variables, S, a set of observations, D, a threshold,β and a type of CI test, CI and the set of immoralities, I

1: Updating Function (ULN,I, D, β, CI, S)
2: PDAG= Fast_Edges (U LN, D, β, CI, S)
3: I=Fast_Immoralities (PDAG, I)

6.4.3 Conclusions

Algorithm 3, instead of re-learning the entire model as with Algorithm 1, only relearns the associations of the variables that the detect change function has indicated. This algorithm also relaxes the assumption of Nielsen and Nielsen algorithm in which only a small number of associations among the variables can be altered. In the case of a change in all the variables, the algorithm 3 will be less efficient than the algorithm 1, because the first one applies the FAST algorithm rather than RAI. The comparison is

discussed in section 2.8. All the algorithms presented in the section 5.1 restrict updating to variables detected where a change has taken place, as with Algorithm 3. This decreases computational complexity for the case where a small number of associations of variables had changed.

Algorithm 3 uses the first step of the Fast algorithm to learn part of the skeleton, testing for each order of CI test the all combinations of pair of variables that involve the variables in question (the variables that were detected by the detect change function).

In terms of accuracy this algorithm presents a theoretical improvement over the Nielsen and Nielsen algorithm with categorical variables, since Nielsen and Nielsen use a Markov Blanket heuristic and CI tests for determining the adjacent variables and for implementing the orientation rules. The Markov Blanket heuristic performs CI tests for a variable conditioned on all other variables which means that the CI results are less reliable. For categorical variables CI tests with large conditioning sets decreases the reliability of the results.

6.4.4 Experimental Verification

This section is presented the results obtained by applying Algorithm 3 to the same simulated data used for algorithms 1 and 2. As with Algorithms 1 and 2, the criteria used for assessing Algorithm 3 is the number of wrong calls of the updating function, whether or not it detects the change, and the detection delay for the same three situations used for algorithms 1 and 2.

The CI tests performed by the RAI and FAST algorithms will use a Likelihood Ratio with threshold 0.01. The detect change function will monitor the mean of the MDL scores of each variable and apply the CUSUM method, assuming a normal distribution of theses scores, as with Algorithm 1. The threshold value will be the same as that for Algorithm 1. As described for Algorithm 1, the change to be detected is a difference in the family of one variable, an edge removed or an edge added or a change in the orientation of an edge.

The first table indicates whether or not the algorithm was able to detect the change in domain (yes/no), the number of false detections and the delay, after the domain changed, in detecting the change. The second table shows, after the algorithm had learned the alteration in the domain, how many new correct intervals it detected and the number of false triggers of the update function.

Table 18: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving observations from the altered distribution

	True Detection	False Detection	Delay on the detection	Average Running Time (s)
Edge Deletion	Yes	0	73	9,8
Edge Addition	Yes	0	62	11,1
Joint Probability Change	Yes	1	98	10,5

The algorithm was able in the first update to detect the change in domain (for edge removal or edge addition, but for change of orientation, the algorithm gave a false alarm before the domain had changed). The average running time was less than that for Algorithms 1 and 2.

Table 19: Results of the New Incremental Learning Algorithm 1 with Asia Network after receiving learning the change

	Correct Updating	Wrong Updating
Removed Edge	3	0
Added Edge	3	0
Changed Joint Probability	2	0

The set of variables pass to the updating function was always of size one. The algorithm updates each time the family of one variable which leads to a more number of updates than the Algorithm 1 and Algorithm 2 because for Algorithm 1 and 2 the same iteration may alter many variables in the way of the new domain.

6.5 The Incremental Learning Algorithms Applied to the Financial Data Set

The Financial Data set is analysed using two of the three algorithms proposed in the previous chapter. The Algorithm 1 and 3 are applied to the financial data. The Algorithm 2 is not used in this section since it only differs from Algorithm 1 in the detect change function by applying the Second Cosine Transform. The CUSUM, besides being computationally less expensive, presented better results in the simulated data set, the alarm network (section 6.2 and 6.3). For each, the update function used a Likelihood Ratio test with an alpha value of 0.01 and the observations are submitted to a fading process. The fading process submitted is deeply explained in the section 3.4, precisely for categorical variables.

The update function used the RAI algorithm with the alteration `Remove Cycles' (section 4.5.1), to return a directed acyclic graph. For Algorithms 1 and 3, the same threshold value was used in the detect change function as for the experiments with the Asia Network. The parameter *v*, for the alternative hypothesis in the CUSUM method was refined according to the statistical variance of the MDL scores. This parameter means the magnitude of change between the null hypothesis, the mean of the MDL values before the change and the mean of the MDL values after the change ($\theta_1 = \theta_0 + v$). That is, if the variance of the MDL scores is high, the parameter *v* should be higher. On the other hand, if there is a low variance in the MDL scores the parameter *v* should be low. With simulated data from the Asia Network it is possible to confirm whether or not the algorithm has detected a real change and the delay, because we know the distributions that generated the data. In the financial data set, the underlying distribution is not known and therefore such a comparison is not possible. The results are therefore assessed according to the number of triggers where the update function produced a new model, the number of triggers where the updated model was the same, the average number of observations between the updates and the average running time for each proposed algorithm. The average running time is not totally precise since the code of both algorithm can be optimised slightly to decrease the running time. The values presented for algorithm 3 correspond to the number of times the detect change function calls the updating function, and not to the number of variables updated.

Table 20: Results of the New Incremental Learning Algorithm 1, 2, 3,4 with the financial data set.

	Number of Triggers leading to an Update	Number of False triggers	Average Number of observations between updates	Average Running Time (s)
New Incremental Learning Algorithm 1	14	0	73.1	305
New Incremental Learning Algorithm 3	13	0	73,4	97

Algorithm 1 updated the network 14 times, each time producing a different learned network. For this reason, there were no false triggers. There were, on average, 73.1 observations between updates. The Algorithm 3 called the update function 13 times and the learned networks were all different. For each call of the update function, only the family of a single variable was altered. This algorithm did not give any false triggers. The number of observations between the updates was, on average, 73.4. The average number of observations between the updates is very similar in both algorithms as is the number of triggers. The threshold of the CUSUM was adapted to have the same delay on both algorithms. Algorithm 3 only updates one variable. It changes the family of the variable, which can imply changes in other families. Despite this, the average running time is much greater for Algorithm 1, which has to re-learned the entire structure with each iteration, than for Algorithm 3, which only updates the family of one variable with each trigger of the update function. The codes of the algorithms have not been optimised to reduce the running time. In the best case, the running time would not be altered substantially and the differences in the average running time for both algorithms shows that the Algorithm 3 is computationally more efficient for this data set.

The two algorithms indicate that there is a change of domain approximately every 2,5 months. This seems reasonable in the context of the financial data set. The financial data set is generated by a very complex system, which involves interactions of large number of agents (millions, in some cases). These agents, the investors, act in a independent manner, basing their decisions on personal motives and analyses, in a different and often contradictory way. In terms of behaviour, the system has a virtual unlimited number of possible “states”.

Besides this, the behaviour of the stock markets and their associations with each other change with time; stock market behaviour is volatile (Lawrence et al., 1996), (Zang and Hutchinson, 1994). This non-stationary behaviour is due to a series of factor such as alteration in the behaviour of the agents, in the way the stock markets are operated, in the political environment, in the economical context, in the companies that perform in each stock market, in the actual technologies, in the social cultural context and environmental events. The financial data set comes from the closing values of 18 stock market indices which are affected by the global environment and also highly dependent

on the specific factors of the country where they are located. All this causes the relationships between these 18 stock market indices to change with time and a rate of around 2,5 months between updates seems consistent with reality and financial news.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The aim of the thesis was to analyse the performance of the Recursive Autonomy Identification Algorithm, to propose improvements to use it as the basis for an incremental learning algorithm to detect changing associations in non-stationary domains. The analysis of the RAI algorithm and the construction of the incremental learning algorithms were directed towards the analysis of a specific financial data set containing the closing values of 18 stock markets indices during a period of five years.

In the first chapter, three constraint based structure learning algorithms were analysed, the Recursive Method, the Fast Adjacency Algorithm and the RAI. The RAI presents substantial advantages over the RM when the variables are categorical. The RAI and the Fast have different trade-offs between the accuracy and computational complexity. While Fast focuses on the accuracy, leading to a higher computational complexity, the RAI, by combining CI tests with orientation rules rather than directing the compelled edges at the end, decreases the complexity substantially, but can lose out with accuracy.

The second chapter of this thesis considers the RAI algorithm and evaluates the performance of the RAI algorithm when CI test are not perfect and when the faithfulness assumption does not hold. The RAI algorithm may produce a learned network that may not be an essential graph simply because too many edges are oriented, or it may produce a structure with reversed edges, or a structure with cycles. The first case was dealt with by a simple alteration to the RAI algorithm to produce an essential graph. For the second, more serious case, a heuristic approach was adopted to decide which edges should be removed in order to eliminate the cycles. The fact that the RAI combines the process of testing for conditional independence with edge orientation can lead to the formation of contradictory immoralities and hence reversed edges. The

reversed edges clearly have a negative impact on the accuracy of the learned network, because they influence the formation of the ancestor and descendant substructures and hence the set of variables where the CI tests are performed and also the orientation of the compelled edges. Reversed edges can appear during the running of the algorithm even when the distribution has a faithful DAG. There are several approaches to deal with this problem in the literature, but none of them give substantial improvement when a stage of the algorithm produces a large number of reversed edges. The alteration proposed in the thesis is simply to orientation of the compelled edges to the end of the algorithm since, without the alteration, edges can be wrongly oriented as a result of compelling an edge in an early phase of the algorithm. Another inconsistency was detected in the RAI algorithm: the results of CI tests of higher order may contradict conditional dependence statements established by CI tests of a lower order. An alteration is proposed to ensure that this does not happen.

The performance of the RAI is evaluated using simulated data from the Alarm network. The algorithm presents a very accurate learned model. However, during a early phase the algorithm adds many reversed edges that, in this case, are removed in a later phase by CI tests of higher order. The direction of these edges is determined randomly, by the order in which the nodes are labelled.

When the RAI is applied to the financial data set where the variables have been discretised and a fading procedure applied, the number of reversed edges in the final graph is substantially larger than for the Alarm Network. This number could be due simply to the fact that the simulated data is better behaved and because the maximum parent set size is smaller for the Alarm Network than for the financial data set.

In the financial context, it is important to learn whether variables are associated with each other; it does not matter whether the association is linear or not. Following this idea, the RAI algorithm was altered to learn a structure that indicates both types of relations, through the Likelihood Ratio test with discretised variables for the non-linear associations and Fisher's test applied to the continuous variables for the linear relationships at same time. This is because Fisher's test remains robust even though the conditioning set is rather large; if Fisher's test detects an association, then there is an association present. The alteration D-connections ensures that dependencies that have been established are not later removed as a result of less reliable tests. The alterations Cycle Removal, Compelled Edges and Essential Graph all improve accuracy, although they may lead to more reversed edges appearing; these reversed edges represent associations present in the data; the reversal indicates that there is not a faithful graphical model to describe the independence structure.

Incremental structure learning, as opposed to batch algorithms that learn the Bayesian network in a single "shot", are designed to revise the structure learned if the instantiations are presented as a time series, where associations may be changing. The

challenge is to re-evaluate the current structure in the light of the new data and update the structure if necessary, within reasonable constraints of time and memory. Algorithms that can do this are of great value for financial data.

Chapter six addresses the problem of incremental structure learning algorithms, proposing three new incremental learning algorithms. These algorithms suppose that the domain is piece-wise stationary. These algorithms differ from the algorithm of (Nielsen and Nielsen and Shi and Tan) in that they are based on the RAI algorithm for structure learning. The algorithm presented by Nielsen and Nielsen, which is able to deal with piece-wise changes in the domain for a small subset of variables was only tested for networks with ten variables.

Algorithm 1 uses a detect change function based on MDL scores and the CUSUM method with and adaptive threshold and an updating function similar to the RAI algorithm that revises the entire learned structure. The learned model has greater accuracy, because the entire network is re-learned, but this makes the algorithm computationally more expensive. Algorithm 2 is very similar to Algorithm 1; the only difference is that it uses The Second Cosine Transform, as in Nielsen and Nielsen, rather than the CUSUM. This alteration increases the run time given the number of cycles need to do. Algorithm 3 opts for a different trade-off between accuracy and complexity. As with Nielsen and Nielsen, and Shi and Tan (Shi and Tan, 200.), the algorithm uses a detect change function to detect the set of variables that should be re-learned. The detection function attributes MDL scores to the family of each variable and then uses the CUSUM method for detecting a change in the scores. This algorithm learns the initial structure using the RAI algorithm, un-directing the learned structure and saving the immoralities. When the updating function is triggered, it uses the first step of the Fast algorithm to relearn the associations of the variables marked by the detect change function and the keeps a record of the immoralities. Using the Fast algorithm to relearn part of network increases accuracy, without substantial increase of the computational complexity. The algorithm obtains an undirected learned model and the list of immoralities.

The performance of the proposed algorithms was verified using data simulated from the Asia network with various changes in the domain: a removed edge, an added edge and a variable with a different joint probability. With simulated data, Algorithm 1 updates correctly the three situations. Depending on the threshold chosen for the CUSUM, the algorithm can have a larger delay in detection or, when it is decreased substantially, an increase in the number of false alarms. Algorithm 2 was also able to detect all the changes, but required a a larger number of observations than Algorithm 1. In terms of running time, Algorithm 1 is quicker than Algorithm 2. Algorithm 3 only updates the family of a single variable with each trigger. This algorithm gives a false alarm in the situation changed joint probability. Algorithm 3 gives substantial improvement in the running time.

Algorithms 1 and 3 were applied to financial data set with discretised variables and with a fading procedure. None of the algorithms gave a false trigger; every time `update' was triggered, the re-learned network was different from the previous one. Both algorithms triggered almost the same number of updates for the learned structure with almost the same average number of observations between updates. The two algorithms update, on average, almost five times per year, a value that seems consistent with the nature of the non-stationary behaviour of the financial markets. Algorithm 3 updates he family of only one variable each time, which leads to a substantial smaller average running time than the Algorithm 1. The Algorithm 3 is better suited to this data set, since the update function only needs to update a family of one variable each time. Algorithm 1 would be better for a data set where a relatively large subset of variables has to be updated with each trigger.

7.2 Future Work

The topics presented in this thesis can be developed and extended in several directions. In the financial context, a key issue would be to develop a conditional independence test to able to deal with continuous variables that are not Gaussian, since most financial variables are not. The tests in the RM algorithm test for conditional correlation, which is a test for linear associations. In financial research, there are many important non-linear associations between variables and it would be useful to investigate structure learning algorithms that were able to incorporate information about both linear associations, derived from Fisher tests, and non linear associations based on discretised variables, in the same learned network. An approach to this was outlined in the thesis; further investigation could produce useful results.

The assumption of faithfulness may be questionable for many `real world' data sets and constraint based approaches, with a larger class of graphical models (e.g. Chain graphs) could provide more satisfactory learning algorithms. An algorithm similar in nature to the Recursive Autonomy Identification algorithm, aiming to learn a chain graph, could reduce the number of reversed edges and inconsistencies.

There is also substantial room for improvement in dealing with contradictory immoralities. The sensitivity of the RAI algorithm to the initial ordering of the nodes when the CI test results are not perfect could also bring interesting results.

For incremental structure learning, there are many directions to develop. Firstly, more experimental justification for each of the algorithms is required. Also, the investigation of scores that can be compared directly with Bayesian Networks that come from different distributions would improve the algorithms. It would also be pertinent to create

a score function which used information from both the discretised data set and the continuous data set in the financial context. Algorithm 3 uses the first step of Fast to learn the skeleton. It could be improved by using another algorithm, based on the RAI, to decrease the complexity without too much cost in terms of accuracy.

Bibliography

Abellan, J., Gomez-Olmedo M., Moral, S., et al. (2006) Some variations on the PC algorithm. In Proceedings of the 3rd European Workshop on Probabilistic Graphical.

Akaike, H. (1974).A new look at the statistical model Identification. IEEE Transactions on Automatic Control, pp.716–723.

Buntine, W. (1991) Theory Refinement on Bayesian Networks. In UAI 91, pages 52–60. Morgan Kaufmann.

B., Michele and Nikiforov, I. (1993) Detection of Abrupt Changes: Theory and Applications. Prentice-Hall.

Campos, M. de (2006) A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests, Journal of Machine Learning Research 7, pp. 2149-2187.

Cheng, J., Greiner, R., Kelly, J., Bell, D., Liu, W., et al. (2002) Learning Bayesian networks from data: An information-theory based approach. Artificial Intelligence, pp. 43-90.

Chickering, M. (1996) Learning Bayesian networks is NP-complete. Learning from Data: Artificial Intelligence and Statistics V, pp. 121-130.

Cooper, G. and Herskovits, E. (1992) A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 9:309–348.

Friedman, N., and Goldszmid, M. (1997) Sequential up-date of Bayesian network structure. In UAI 97, pp. 165–174. Morgan Kaufmann.

Friedman, N., Linial, M., Nachaman, Iftach, Pe'er, D. (2000) Using Bayesian Networks to Analyze Expression Data, Journal of Computational Biology, Volume 7, Number ¾

Gustafsson, F. (1997) A comparative study on change detection for some autmotive applications. European Control Conference, Brussels, IEEE Control Systems.

Heckerman, D., Geiger, D., Chickering, D., et al. (1995) Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, pp. 197-243.

Lawrence, S., Tsoi, A., Gilles, C, et al. (1996) Noyse Yime Series prediction using Symbolic representation and Recurrent Neural Networks Grammatical inference. Technical report UMIACS- TR-96-27 and CS-TR-3625, Institute for Advanced Computer Studies, University of Maryland.

Leon, Pierre (1982) Historia económica e social do mundo: guerras e crises, Sá da costa Editora.

Meek, C. (1995) Causal inference and causal explanation with background know ledge Proceedings of the Eleventh Conference on Uncertainty in Articial Intellegence, pp 403 – 410.

Neil,M., Fenton, N., Tailor, M., et al. (2005) Using Bayesian Networks to Model Expected and Unexpected Operational Losses Risk Analysis, Vol. 25, No. 4, Demirer,R., Mau, R., Shenoy, C., et al. (2006) Bayesian Networks: A Decision Tool to Improve Portfolio Risk Analyzes. Journal of Applied Finance, Volume: 16, Issue: 2, pp. 106-119.

Nielsen, S. H. and Nielsen, T. D., (2008) Adapting Bayes Network Structures to Non-stationary Domains. International Journal of Approximate Reasoning.Vol. 49, No. 2. pp. 379-397.

Page, E.S. (1954) Continuous inspection schemes. Biometrika, 41:100–115.

Peng, H.C., Long, F.H. and Ding,C., et al.(2005) Feature Selection Based on Mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. Pattern Analysis and Machine Intelligence: Vol. 27, pp 1226-1238.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., et al. (2002) Numerical Recipes in C++: The Art of Scientific Computing .Cambridge University Press, 2nd edition.

Pourret, O., Naim, P., Marcto, B. (2008) Bayesian Networks A pratical Guide to Applications. Statistics in Practice, Willey, USA.

Robinson, R.W. (1977) Counting Unlabelled Acyclic Digraphs. Springer Lecture Notes in Mathematics: Combinatorial Mathematics V, C.H.C. Little (ed.) pp. 28 – 43
Schwarz , G. (1978) Estimating the dimension of a model. Annals of Statistics, 6:461–464.

Roure, J. 2004. Incremental hill-climbing search applied to Bayesian network structure learning. In Proceedings of the 15th European Conference on Machine Learning.

Shenoy, C. and Shenoy, P. (2000) Bayesian Networks of Portfolio Risk and Return. The MIT Press.

Shi, D. and Tan, S., (2007) Incremental Learning Bayesian Networks for Financial Data Modeling, 22nd IEEE International Symposium on Intelligent Control, IEEE Multi-Conference on Systems and Control Singapore.

Spirtes, P., Glymour, C., & Scheines, R. (2000). Causation, prediction, and search. The MIT Press, second edition.

Tian, F., Zhang, H., Lu, Y., Shi, C., et al. (2001) Incremental Learning of Bayesian Networks with Hidden Variables Proceedings of the 2001 IEEE International Conference on Data Mining.

Yehezkel, R., & Lerner, B. (2009). Bayesi Network Structure Learning by Recursive Autonomy Identification. Journal of Machine Learning Research, pp. 1527-1570.

Zang, X. And Hutchinson, E. (1994) Simple Architectures on Fast Machine: Practical issues in nonlinear time series prediction. In Weigend, A. E Geshenfeld, N. 8ed): Time Series prediction: Forecasting the Future and Understanding the Pat, Addison- Wesley.

Appendix A

The follow code calculates the second cosine transform (Press, et al. 2002).

For an inverse transform the output array should be multiplied by 2/n.

```
If isign=1
wr=1
n=length(Y)
theta=0.5*pi()/n
wr1=cos(theta)
wi1=sin(theta)
wpr=-2*wi1*wi1
wpi=sin(2*theta)
for i=1:n/2, i++
y1=0.5*(Y[i]+Y[n-i-1]);
y2=wi1*(Y[i]-Y[n-i-1]);
Y[i]=y1+y2
Y[n-i-1]=y1-y2
wr1=(wtemp=wr1)*wpr-wi1*wpi+wr1
wi1=wi1*wpr+wtemp*wpi+wi1
end
for i=2:n-1, i++
 wr=(wtemp=wr)*wpr-wi*wpi+wr
wi=wi*wpr+wtemp*wpi+wi
y1=Y[i]*wr-Y[i+1]*wi
y2=Y[ i]*wr+Y[i+1]*wi
Y[i]=y1
Y[i+1]=y2
end
sum=0.5*Y[1]
for i=n-1:0, i++
sum1=sum
sum=Y[i]
Y[i]=sum1
end
Else if isign==-1
```

```
Ytemp=y[n-1]
For i=n-1:-2:<n
Y[i]=y[i-2]-y[i]
Y[i]=2*ytemp
Wr=(wtemp=wr)*wrp-wi+wr
Wi=wi*wpr+wtemp*wpi+wi
Y1=y[i]*wrp+y(i+1]*wi
Y2=y[i+1]*wrp-y(i]*wi
Y[i]=y1
Y[i+1]=y2
End
Realft (y,-1)
For i=0;i<n/2, i++
Y1=Y[i]+Y[n-1-i]
Y2=0.5/wi1*(Y[i]-Y[n-1-i]
Y[i]=0.5*(y1+y2)
Y[n-1-i]=0.5*(y1-y2)
wr1=(wtemp=wr1)*wpr-wi1+wri
wi1=wi1Pwpr+wtemp*wpi+wi1
End
End
End
```
