Learning Gated Bayesian Networks for Algorithmic Trading

Marcus Bendtsen and Jose M. Peña

Department of Computer and Information Science, Linköping University, Sweden
marcus.bendtsen@liu.se, jose.m.pena@liu.se

Abstract. Gated Bayesian networks (GBNs) are a recently introduced extension of Bayesian networks that aims to model dynamical systems consisting of several distinct phases. In this paper, we present an algorithm for semi-automatic learning of GBNs. We use the algorithm to learn GBNs that output buy and sell decisions for use in algorithmic trading systems. We show how using the learnt GBNs can substantially lower risks towards invested capital, while at the same time generating similar or better rewards, compared to the benchmark investment strategy *buy-and-hold*.

Keywords: Probabilistic graphical models, Bayesian networks, algorithmic trading, decision support

1 Introduction

Algorithmic trading can be viewed as a process of actively deciding when to own assets and when to not own assets, so as to get better risk and reward on invested capital compared to holding on to the assets over a long period of time. At the other end of the spectrum is the *buy-and-hold* strategy, where one owns assets continuously over a period of time without making any decisions of selling or buying during the period. This paper introduces a novel algorithm that can be used to learn *gated Bayesian networks* (GBNs, described in Sect. 2) for use as part of an algorithmic trading system. We also present a real-world application of this learning algorithm that shows that, compared to the benchmark *buy-and-hold* strategy, the expected risks and rewards are improved upon.

1.1 Algorithmic Trading

An algorithmic trading system contains several components, some which may be automated by a computer, and others that may be manually executed [1, 2]. A schematic overview of the components of a general algorithmic trading system is shown in Fig. 1.

The type of data used at the research stage varies greatly, e.g. net profit, potential prospects, sentiment analysis, analysis of previous trades, or *technical analysis*, which will be the focus in the enclosed application (described in

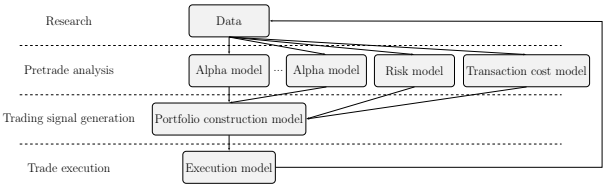

Fig. 1. Components of an algorithmic trading system

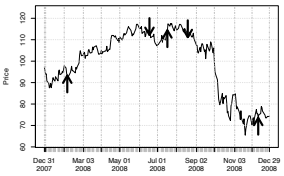

Fig. 2. Buy and sell signals

Sect. 5.1). The analysis of the data is split up into *alpha*, *risk* and *transaction cost* models. The alpha models are responsible for outputting decisions for buying and selling assets based on the data they are given. These decisions are known as buy and sell *signals*, examples of which are depicted in Fig. 2 (an arrow pointing upwards is a buy signal and a downwards facing arrow is a sell signal, the signals are drawn on top of the historical asset price). If followed, these buy and sell signals give rise to certain *risk* and *reward* on the initial investment (which will be described further in Sect. 3). The contribution of this paper is concerned with the use of GBNs as alpha models.

The risk and transaction cost models should be seen as strategies for managing risk and transaction costs in a system that has many alpha models. The output from these three types of models (alpha, risk and transaction) are in their turn the input to the *portfolio construction model* in the trading signal generation stage. Here the output of the previous components are combined to decide which signals to actually execute in order to create a portfolio that is based on a combination of alpha models. The final stage is the actual *execution* of the trading signals, which must be done in a manner that does not affect the price of the asset that is being bought. Although all components are important, we will not be addressing all of them in this paper, our focus will be on the alpha models.

The rest of the paper is organised as follows. We begin by giving a brief introduction to Bayesian networks (BN) and GBNs in Sect. 2, this is important to understand how GBNs can be used as alpha models. We continue by explaining how we can evaluate the performance of an alpha model in Sect. 3. We use this method of evaluation in Sect. 4, where we introduce a novel algorithm that can be used to learn GBNs. In Sect. 5 we make use of the learning algorithm in a real-world application, where we show how learnt GBNs can be used as alpha models. Finally we end the paper in Sect. 6 with a few words regarding our conclusions and future work.

2 Gated Bayesian Networks

BNs can be interpreted as models of causality at the macroscopic level, where unmodeled causes add uncertainty. Cause and effect are modelled using random variables that are placed in a directed acyclic graph (DAG). The causal model implies some probabilistic independencies among the variables, that can easily

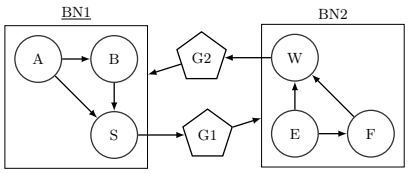

Fig. 3. GBN - two phase example

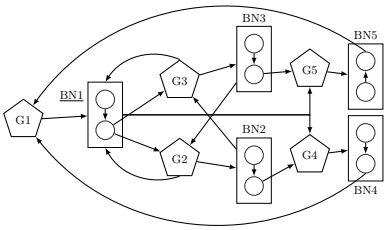

Fig. 4. GBN - multi phase example

be read off the DAG. Therefore, a BN does not only represent a causal model but also an independence model. The qualitative model can be quantified by specifying certain marginal and conditional probability distributions so as to specify a joint probability distribution, which can later be used to answer queries regarding posterior probabilities. The independencies represented in the DAG make it possible to compute these posteriors efficiently. See [3, 4] for more details.

Although BNs have successfully been used in many domains, our interest is to model the process of buying and selling assets, and in this particular situation the BN model is not enough. This is the main motivation for us introducing GBNs [5], and the current paper builds upon this previous contribution. When trying to model the process of buying and selling assets, we want to model the continuous flow between looking for opportunities to buy and opportunities to sell. The model can be seen as being in one of two distinct phases: either looking for an opportunity to buy into the market, or an opportunity to sell and exit the market. These two phases can be very different and the random variables included in the BNs modelling them are not necessarily the same.

Switching between phases is done using so called *gates*. These gates are encoded with predefined logical expressions regarding posterior probabilities of random variables in the BNs. This allows activation and deactivation of BNs based on posterior probabilities. A GBN that uses two different BNs (BN1 and BN2) is shown in Fig. 3, follows does a brief explanation of this GBN and how it is used (for the full details we refer the reader to our previous publication [5]).

- A GBN consists of BNs and gates. BNs can be *active* or *inactive*. The label of BN1 is underlined, indicating that it is active at the initial state of the GBN. The BNs supply posterior probabilities to the gates via so called *trigger nodes*. The node S is a trigger node for gate G1 and W is a trigger node for G2. A gate can utilise more than one trigger node.
- Each gate is encoded with a predefined logical expression regarding its trigger nodes' posterior probability of a certain state, e.g. G1 may be encoded with $p(S = s1|\mathbf{e}) > 0.7$. This expression is known as the *trigger logic* for gate G1.
- When evidence is supplied to the GBN an evidence handling algorithm updates posterior probabilities and checks if any of the logical statements in the gates are satisfied. If the trigger logic is satisfied for a gate it is said

to *trigger*. A BN that is inactive never supplies any posterior probabilities, hence G2 will never trigger as long as BN2 is inactive.
– When a gate triggers it deactivates all of its parent BNs and activates its child BNs (as defined by the direction of the edges between gates and BNs). In our example, if G1 was to trigger it would deactivate BN1 and activate BN2, this implies that the model has switched phase.

For the user of the GBN, the knowledge that one or more of the gates have triggered (i.e. the state of the GBN has changed), may be useful in a decision making process. As an example, if the GBN was used as an alpha model, knowing that the GBN has found a buying opportunity and has started modelling selling opportunities would suggest that a buy signal has been generated. Looking again at Fig. 2, each buy and sell signal is generated by the fact that the GBN switched back and forth between its states.

GBNs can consist of many phases, and the phases themselves can have subphases that are made up of several BNs. An example of a GBN with multiple phases is shown in Fig. 4.

3 Evaluating Alpha Models

Regression models can be evaluated by how well they minimise some error function or by their log predictive scores. For classification, the accuracy and precision of a model may be of greatest interest. Alpha models may rely on regression and classification, but can not be evaluated as either. An alpha model's performance needs to be based on its generated signals over a period of time, and the performance must be measured by the *risk* and *reward* of the model. This is known as *backtesting*.

3.1 Backtesting

The process of evaluating an alpha model on historic data is known as *backtesting*, and its penultimate goal is to produce metrics that describe the behaviour of a specific alpha model. These metrics can then be used for comparison between alpha models [6, 7]. A time range, price data for assets traded and a set of signals are used as input. The backtester steps through the time range and executes signals that are associated with the current time (using the supplied price data) and computes an *equity curve* (which will be explained in Sect. 3.2). From the equity curve it is possible to compute metrics of risk and reward. To simulate potential transaction costs, often referred to as *commission*, every trade executed is usually charged a small percentage of the total value (0.06% is a common commission charge used in the enclosed application).

Alpha models are backtested separately from the other components of the algorithmic trading system, as the backtesting results are input to the other components. Therefore, we execute every signal from an alpha model during backtesting, whereas in a full algorithmic trading system we would have a portfolio construction model that would combine several alpha models and decide how to build a portfolio from their signals.

3.2 Alpha Model Metrics

What constitutes risk and reward is not necessarily the same for every investor, and investors may have their own personal preferences. However, there are a few that are common and often taken into consideration [7]. Here we will introduce a few metrics that we will use to evaluate the performance of our alpha models.

Equity Curve Although not a metric on its own, the *equity curve* needs to be defined in order to define the following metrics. The equity curve represents the total value of a trading account at a given point in time. If a daily timescale is used, then it is created by plotting the value of the trading account day by day. If no assets are bought, then the equity curve will be flat at the same level as the initial investment. If assets are bought that increase in value, then the equity curve will rise. If the assets are sold at this higher value then the equity curve will again go flat at this new level. The equity curve summarises the value of the trading account including cash holdings and the value of all assets. We will use $\mathcal{E}_t$ to reference the value of the equity curve at point t.

Metric 1 (Return). The *return* of an investment is defined as the percentage difference between two points on the equity curve. If the timescale of the equity curve is daily, then $r_t = (\mathcal{E}_t - \mathcal{E}_{t-1})/|\mathcal{E}_{t-1}|$ would be the *daily* return between day t and $t-1$. We will use $\bar{r}$ and σ_r to denote the mean and standard deviation of a set of returns.

Metric 2 (Sharpe Ratio). One of the most well known metrics used is the so called *Sharpe ratio*. Named after its inventor Nobel laureate William F. Sharpe, this ratio is defined as: $(\bar{r} - \text{risk free rate})/\sigma_r$. The *risk free rate* is usually set to be a "safe" investment such as government bonds or the current interest rate, but is also sometimes removed from the equation [7]. The intuition behind the Sharpe ratio is that one would prefer a model that gives consistent returns (returns around the mean), rather than one that fluctuates. This is important since investors tend to trade *on margin* (borrowing money to take larger positions), and it is then more important to get consistent returns than returns that sometimes are large and sometimes small. This is why the Sharpe ratio is used as a reward metric rather than the return.

Drawdown Risks Using the Sharpe ratio as a metric will ensure that the alpha models are evaluated on their risk adjusted return, however, there are other important alpha model behaviours that need to be measured. A family of these, that we will call *drawdown risks*, are presented here (please see Fig. 5 for examples of an equity curve and these metrics).

Metric 3 (Maximum Drawdown (MDD)). The percentage between the highest peak and the lowest trough of the equity curve during backtesting. The peak must come before the trough in time. The MDD is important from both

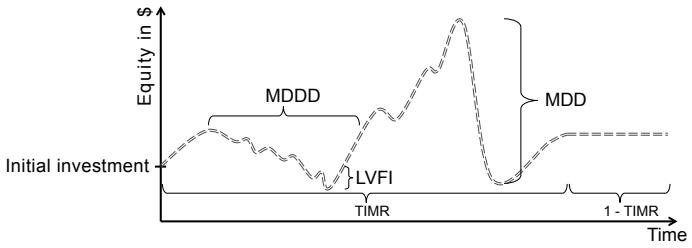

Fig. 5. Example of equity curve with drawdown risks

a technical and psychological regard. It can be seen as a measure of the maximum risk that the investment will live through. Investors that use their existing investments that have gained in value as safety for new investments may be put in a situation where they are forced to sell everything. Other risk management models may automatically sell investments that are loosing value sharply. For the individual who is not actively trading but rather placing money in a fund, the MDD is psychologically frustrating to the point where the individual may withdraw their investment at a loss in fear of loosing more money.

Metric 4 (Maximum Drawdown Duration (MDDD)). The longest it has taken from one peak of the equity curve to recover to the same value as that peak. Despite its unfortunate name it is not the duration of the MDD, but rather then longest drawdown period. There is an old adage amongst investors to "cut your losses early". In essence it means that it is better to take a loss straight away than to sit on an investments for months or years, hoping that it will come back to positive returns. During this time one could have re-invested the money elsewhere, rather then breaking-even much later (or taking a larger loss much later). Models that have long periods of drawdown lock resources when they could have been used better elsewhere.

Metric 5 (Lowest Value From Investment (LVFI)). The percentage between the initial investment and the lowest value of the equity curve. This is one of the most important metrics, and has a significant impact on technical and psychological factors. For investors trading on margin, a high LVFI will cause the lender to ask the investor for more safety capital (known as a *margin call*). This can be potentially devastating, as the investor may not have the capital required, and is then forced to sell the investment. The investor will then never enjoy the return the model could have produced. Individuals who are not investing actively, but instead are choosing between funds that invest in their place, should be aware of the LVFI as it is the worst case scenario if they need to retract their equity prematurely.

Metric 6 (Time In Market Ratio (TIMR)). The percentage of time of the investment period where the alpha model owned assets. This metric may seem odd to place within the same family as the other drawdown risks, however it fits

naturally in this space. We can assume that the days the alpha model does not own any assets the drawdown risk is zero. If we are not invested, then there is no risk of loss. In fact, we can further assume that our equity is growing according to the risk free rate, as it is not bound in assets.

3.3 Buy and Hold Benchmark

At first the buy-and-hold strategy may seem naïve, however it has been shown that deciding when to own and not own assets requires consistent high accuracy of predictions in order to gain higher returns than the buy-and-hold strategy [8]. The buy-and-hold strategy has become a standard benchmark, not only because of the required accuracy, but also because it requires very little effort to execute (no complex computations and/or experts needed).

Now consider the family of metrics that we called drawdown risks. The buy-and-hold strategy holds assets over the entire backtesting period and so will be subject to the full force of these metrics. For instance, as an asset will be held throughout the period, the lowest point of the assets value will coincide with LVFI. Furthermore, the initial investment will always be locked in assets, not being able to make money from risk free rates during periods of decreasing value. These are serious risks of using buy-and-hold that algorithmic trading could improve upon, which we will explore in the enclosed application in Sect. 5.

4 Learning Algorithm

The algorithm proposed in this paper for semi-automatically learning the structure of a GBN consists of two parts: a *GBN template* and a novel combination of k-fold cross-validation and time series cross-validation (time series cross-validation is sometimes known as *rolling origin* [9] or *walk forward analysis* [6]).

4.1 Gated Bayesian Network Templates

A GBN template is a representation of the modelled phases, including the possible transitions between them. The template defines where BNs and gates can be placed. For each slot where a BN can be placed, there is a library of BNs to choose from, similarly so for gates (gates differ in their trigger logic, e.g. the thresholds may vary between them). A template with four slots and corresponding libraries is depicted in Fig. 6.

The only restrictions on the BNs and gates are the ones they place on each other, e.g. if the gates placed in G2 expect a particular node as trigger node, then the BNs placed in BN2 must contain that node. Except for these restrictions, the BNs and gates can be configured freely.

Selecting a BN and a gate from the libraries for each slot in the template creates a GBN (e.g. Fig. 3), we call this a *candidate* of the template. We use $\mathcal{C}_i$ to denote GBN candidate i of a GBN template.

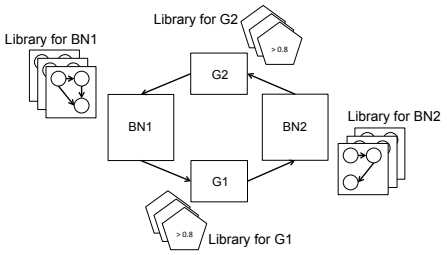

Fig. 6. GBN template

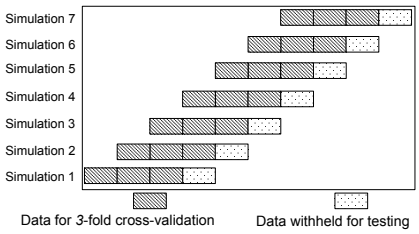

Fig. 7. Combined k-fold cross-validation and time series cross-validation using $n = 10$ blocks and $k = 3$ folds

4.2 K-Fold and Time Series Cross-Validation

In this section we will discuss how a GBN candidate, from a GBN template, is evaluated. In the domain of algorithmic trading it is natural for the test data to always come after the training data in temporal order. This is to ensure that we are not training on data that may carry information not available during the time the testing data was produced.

Splitting the Data A data set $\mathcal{D}$ of consecutive evidence sets, e.g. observations over all or some of the random variables in the GBN, is divided into n equally sized blocks $(\mathcal{D}_1, ..., \mathcal{D}_n)$ such that they are mutually exclusive and exhaustive. Each block contains consecutive evidence sets and all evidence sets in block $\mathcal{D}_i$ come before all evidence sets in $\mathcal{D}_j$ for all $i < j$.

Depending on the amount of available data, k is chosen as the number of blocks used for training. These blocks will be used for k-fold cross-validation. Starting from index 1, blocks $1, .., k$ are used for training and $k + 1$ for testing, thus ensuring that the evidence sets in the testing data occurs after the training data (as in time series cross-validation). The procedure is then repeated starting from index 2 (i.e. blocks $2, .., k + 1$ are used for training and $k + 2$ for testing). By doing this we create repeated simulations, moving the testing data one block forward each time. An illustration of this procedure when $n = 10$ and $k = 3$ is show in Fig. 7.

4.3 Algorithm

Let $\mathcal{J}(\mathcal{C}_i, \mathcal{D}_j, \{\mathcal{D}\}_l^m)$ be the score, e.g. Sharpe ratio, LVFI, etc., for GBN candidate i when block j has been used for testing and the blocks $\mathcal{D}_l, ..., \mathcal{D}_m$ have been used for training. The algorithm then works in three steps (with an optional fourth):

1. For each simulation t, where (as discussed previously) $\mathcal{D}_{t+k}$ is the testing data and $\mathcal{D}_t ... \mathcal{D}_{t+k-1}$ is the training data, find C^t that satisfies (1). This corresponds to finding the GBN candidate with the maximum mean score of

the k evaluations performed during k-fold cross-validation over the training data. This is done by taking into consideration every possible candidate, thus exhausting the search space.

$$\mathcal{C}^t = \arg\max_{\mathcal{C}_i} \ \frac{1}{k} \Sigma_{j=t}^{t+k-1} \mathcal{J}(\mathcal{C}_i, \mathcal{D}_j, \{\mathcal{D}\}_t^{t+k-1} \backslash \mathcal{D}_j) \ . \tag{1}$$

2. For each $\mathcal{C}^t$ calculate its score $\rho_{\mathcal{J}}^t$ on the testing set with respect to the scoring function $\mathcal{J}$ according to (2). This corresponds to training the found GBN candidate from (1) using all training data and evaluating the performance on the data withheld for testing.

$$\rho_{\mathcal{J}}^t = \mathcal{J}(\mathcal{C}^t, \mathcal{D}_{t+k}, \{\mathcal{D}\}_t^{t+k-1}) \ . \tag{2}$$

3. The expected performance $\bar{\rho}_{\mathcal{J}}$ of the algorithm, with respect to the score function $\mathcal{J}$, is then given by the average of the scores $\rho_{\mathcal{J}}^t$ (3).

$$\bar{\rho}_{\mathcal{J}} = \frac{1}{n-k} \Sigma_{t=1}^{n-k} \rho_{\mathcal{J}}^t \ . \tag{3}$$

4. (Optional) If the objective is to find the candidate to be used on future unseen data (i.e. block $\mathcal{D}_{n+1}$) then (1) is used once more to find $\mathcal{C}^{n-k+1}$. This candidate can then be used on $\mathcal{D}_{n+1}$ with an expected performance $\bar{\rho}_{\mathcal{J}}$.

It may seem unorthodox to use k-fold cross-validation with unordered data in step 1, i.e. the testing block may come before some training blocks. However, this step is only used to select a model to evaluate in step 2. The data used in step 2 is always ordered, i.e. the test block is always the immediate successor of the training blocks. This does give a fair evaluated performance on the testing data. Step 1 attempts to use the training data to its maximum, allowing for each candidate to be assessed on several data sets before selecting the one to move forward with.

In the description of the algorithm, one scoring function $\mathcal{J}$ has been used both for choosing a candidate in (1) and for evaluating the expected performance of the algorithm in (2). In Sect. 3.2 we have defined several metrics used to evaluate alpha models. The scoring function $\mathcal{J}$ used in (1) could internally use many of these metrics to come up with one score to compare the different candidates with. However, it is natural in the current setting to expose the actual values of these metrics during step 2, and so several scoring functions $\mathcal{J}$ can be used to get a vector of scores $[\rho_{\mathcal{J}_1}^t, ..., \rho_{\mathcal{J}_m}^t]$ and use a vector of means as the performance of the algorithm $[\bar{\rho}_{\mathcal{J}_1}, ..., \bar{\rho}_{\mathcal{J}_m}]$.

5 Application

In this section we show a real-world application where our proposed algorithm has been used to learn GBNs for use as alpha models, using backtesting to

evaluate their performance. Following the discussion in Sect. 3.3, the aim is to generate buy and sell signals such that the drawdown risks defined in Sect. 3.2 are mitigated as compared to the buy-and-hold strategy, while at the same time maintaining similar or better rewards.

5.1 Methodology

The variables used in the BNs of our GBNs are all based on so called *technical analysis*. One of the major tenets in technical analysis is that the movement of the price of an asset repeats itself in recognisable patterns. *Indicators* are computations of price and volume that support the identification and confirmation of patterns used for forecasting. Many classical indicators exists, such as the *moving average* (MA), which is the average price over time, and the *relative strength index* (RSI) which compares the size of recent gains to the size of recent losses. Technical analysis is a topic that is being actively developed and researched [10]. In this application we will be using three indicators: the MA, the RSI and the relative difference between two MAs (MADIFF). Please see [11] for the full definition and calculations of these indicators.

GBN Template A template with one BN per phase was created (see Fig. 6), along with eight BNs per BN slot (see Fig. 8) and four gates per gate slot, giving a total of 1024 candidates. The eight BNs used for BN1 are identical to those used in BN2, however the gates' trigger logic are different. The trigger logic for G1 asks for the posterior probability of a good buying opportunity (i.e. a predicted positive future climate) while the trigger logic for G2 asks for the posterior probability of a good selling opportunity (i.e. a predicted negative future climate).

The random variables in the BNs are discretizations of technical analysis indicators (RSI, MA and MADIFF) and their corresponding first and second order 1 and 5 day backward finite differences ($\nabla_1^1, \nabla_5^1, \nabla_1^2$ and ∇_5^2) which approximate the first and second order derivatives. The parameters used in the indicators are standard 14 day period for RSI [11] (written as RSI(14)), 20 day period for MA, representing 20 trading days in a month (written as MA(20)), and 5 and 20 day period for MADIFF, where 5 days represent the 5 trading days in a week (written as MADIFF(5,20) and calculated as $\frac{MA(5)-MA(20)}{MA(20)}$). We also consider the previous indicators but with an offset of 5 days in the past and 5 days into the future. The random variables that are offset into the future represent the future economical climate, one of which was involved in the trigger logic of the gates. The true values for these future random variables were naturally not part of the testing data sets. The BNs used for the BN slots are presented in Fig. 8. The node named S was used as the trigger node for all gates. The GBN generated trading signals as it transitioned between its two phases (as described in Sect. 2).

Data Sets A set of actively traded stock shares where chosen for the evaluation of our learning algorithm: Apple Inc. (AAPL), Amazon.com Inc. (AMZN), Inter-

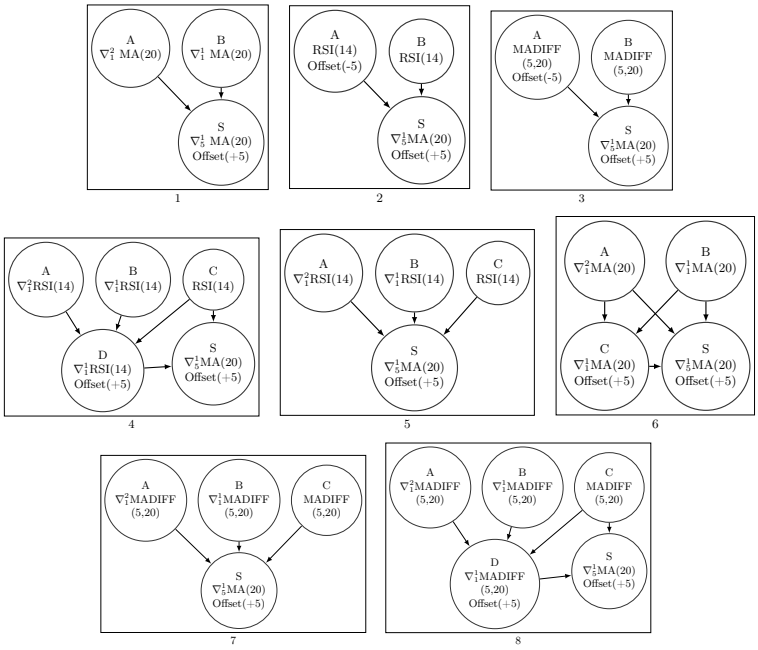

Fig. 8. BNs in GBN template libraries

national Business Machines Corporation (IBM), Microsoft Corporation (MSFT), NVIDIA Corporation (NVDA), General Electric Company (GE), Red Hat Inc. (RHT). The daily adjusted closing prices for these stocks between 2003-01-01 and 2012-12-31 were downloaded from Yahoo! Finance$^{\text{TM}}$. This gave a total of 10 years of price data for each stock, where each year was allocated to a block, and thus n = 10. For the learning algorithm, k was chosen to be 3, giving 7 simulations from which to calculate $[\bar{\rho}_{\mathcal{J}_1}, ..., \bar{\rho}_{\mathcal{J}_m}]$. The split of the data is visualised in Fig. 7.

Scoring Functions The signals generated were backtested (see Sect. 3) in order to calculate the relevant metrics. For step 1 in the learning algorithm we used the Sharpe ratio. This choice was made as it combines both risk and reward into one score, which can then easily be compared between candidates. For step 2 we used the return and drawdown risks as described in Sect. 3.2 to create a score vector. For the buy-and-hold strategy the same metrics as in step 2 were calculated for the 7 simulations.

5.2 Results and Discussion

To visualise the backtesting that was done for each simulation, Fig. 9 gives two examples of stock price, generated signals (an upward arrow indicates a

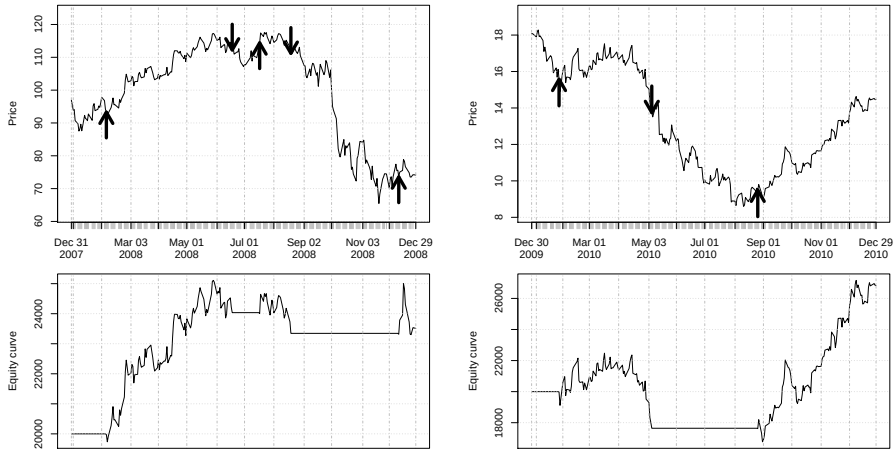

Fig. 9. Price, signals and GBN equity curve for IBM 2008 (left) and NVDA 2010 (right)

buy signal and a downward arrow indicates a sell signal) and resulting equity curve (with an initial investment of $20,000 USD) for the evaluated GBN. The equity curve is the one achieved by executing the signals from the GBN, the corresponding equity curve for the buy-and-hold strategy would follow the stock price exactly, as it holds shares over the entire period. The GBN equity curve grows in a more monotonic fashion, which is desirable because this decreases the drawdown risks, while at the same time generating positive returns. The buy-and-hold strategy would have made a loss in both these examples, because the final price is lower than the initial one, furthermore it would have displayed bad intermediate behaviour, reflected by the high drawdown risk values that would have been incurred. These are declining years for the shares, however the GBN does its best to get as much value as possible from the price movements.

Table 1 presents the score vectors from the learning algorithm versus the score vector of the buy-and-hold strategy over the 7 simulations. Rows named *min*, *max*, *mean* and *sd* (standard deviation) are based on (2) where *mean* corresponds to (3). As each block used by the learning algorithm had an approximate length of one year, the Sharpe ratio that is given by dividing the mean with the sd of the return column is a yearly Sharpe ratio based on seven years (where the risk-free rate has not been included). All values are ratios except for MDDD which is measured in number of days.

Analysis of Results The Sharpe ratio is our measure of reward, premiered above the raw return for reasons discussed in Sect. 3.2. Our first concern is to ensure that the learnt GBNs are producing similar or better Sharpe ratios than the buy-and-hold strategy over the testing period. As can be seen in Table 1, this is the case except for NVDA and RHT. As we have previously discussed, it requires a very high accuracy of predictions to consistently beat the Sharpe ratio of buy-and-hold.

Table 1. Metric values comparing GBN with buy-and-hold

		GBN					Buy-and-hold				
		Return	MDD	MDDD	LVFI	TIMR	Return	MDD	MDDD	LVFI	TIMR
AAPL	min	-0.000	0.122	35.0	0.001	0.520	-0.559	0.129	28.0	0.001	1.000
	max	0.851	0.331	164.0	0.184	0.944	1.419	0.589	250.0	0.590	1.000
	mean	0.347	**0.206**	**95.0**	**0.055**	**0.723**	0.489	0.274	116.0	0.162	1.000
	sd	0.334	0.076	50.3	0.061	0.155	0.707	0.168	82.7	0.218	0.000
	Sharpe	**1.041**					0.691				
AMZN	min	-0.204	0.134	56.0	0.042	0.510	-0.466	0.157	45.0	0.001	1.000
	max	0.784	0.306	142.0	0.245	0.768	1.740	0.634	249.0	0.620	1.000
	mean	0.271	**0.218**	**101.7**	**0.109**	**0.630**	0.463	0.317	118.6	0.215	1.000
	sd	0.374	0.060	32.8	0.088	0.091	0.829	0.171	89.9	0.234	0.000
	Sharpe	**0.725**					0.559				
IBM	min	-0.022	0.062	53.0	0.013	0.494	-0.210	0.088	28.0	0.001	1.000
	max	0.238	0.176	176.0	0.121	0.944	0.596	0.442	190.0	0.302	1.000
	mean	0.125	**0.117**	112.3	**0.044**	**0.712**	0.170	0.174	**106.4**	0.086	1.000
	sd	0.094	0.042	45.4	0.042	0.173	0.245	0.120	59.7	0.101	0.000
	Sharpe	**1.332**					0.694				
MSFT	min	-0.256	0.099	88.0	0.001	0.365	-0.457	0.141	74.0	0.001	1.000
	max	0.381	0.305	197.0	0.279	0.741	0.659	0.498	250.0	0.498	1.000
	mean	0.056	**0.168**	**143.3**	**0.114**	**0.557**	0.069	0.249	168.6	0.200	1.000
	sd	0.202	0.068	41.9	0.091	0.156	0.338	0.119	67.8	0.155	0.000
	Sharpe	**0.278**					0.204				
NVDA	min	-0.420	0.182	64.0	0.032	0.241	-0.765	0.253	67.0	0.077	1.000
	max	0.342	0.541	227.0	0.467	0.700	1.230	0.820	249.0	0.821	1.000
	mean	0.016	**0.284**	**148.1**	**0.209**	**0.516**	0.202	0.458	172.3	0.311	1.000
	sd	0.284	0.120	62.1	0.140	0.171	0.701	0.195	76.6	0.268	0.000
	Sharpe	0.057					**0.288**				
GE	min	-0.302	0.049	60.0	0.015	0.404	-0.555	0.089	69.0	0.001	1.000
	max	0.461	0.465	217.0	0.438	0.570	0.222	0.657	217.00	0.642	1.000
	mean	0.040	**0.169**	**144.3**	**0.119**	**0.488**	-0.001	0.314	157.0	0.236	1.000
	sd	0.235	0.142	69.7	0.150	0.062	0.257	0.228	53.7	0.257	0.000
	Sharpe	**0.169**					-0.005				
RHT	min	-0.222	0.096	87.0	0.001	0.433	-0.370	0.143	40.0	0.001	1.000
	max	0.436	0.428	221.0	0.348	0.784	1.341	0.676	221.0	0.617	1.000
	mean	0.038	**0.254**	156.9	**0.136**	**0.613**	0.201	0.338	**133.0**	0.243	1.000
	sd	0.259	0.103	45.6	0.123	0.136	0.579	0.197	61.6	0.234	0.000
	Sharpe	0.145					**0.346**				

From this we can conclude that the GBNs do not get beaten consistently by the buy-and-hold strategy when considering the annual Sharpe ratio, even though it is considered a nearly optimal strategy. Furthermore, we should take into consideration TIMR. The GBNs are spending less time in the market, reducing risk to equity and possibly increasing equity value from risk free investments. Potential gain in equity from risk free rates have not been added to the Sharpe ratios presented in the table. Considering that the learnt GBNs consistently spend considerably less time in the market (shown by the low TIMR values), this could give a significant boost to the Sharpe ratios. An example of this can be seen for NVDA where the Sharpe ratio for GBN is lower than for buy-and-hold, but the GBN only spent on average 51.6% of the time in the market, risk free investments could potentially drive the Sharpe ratio for the GBN above that of the buy-and-hold strategy.

Turning our attention to the drawdown risks (as defined in Sec. 3.2) we first consider the MDD and MDDD. The difference of the MDD values are substantial, the MDD mean and sd are consistently smaller for the GBNs than they are for the buy-and-hold strategy. This signals that the equity we gain from our investments

are at less risk when using the GBNs compared to the buy-and-hold strategy. For MDDD the means differ in favour of either approach, we would not prefer one in front of the other given only this metric.

The LVFI is a major threat to equity (see Sect. 3.2), and it is the one metric where buy-and-hold severely under-performs. Considering the max values we note that for NVDA the buy-and-hold strategy wiped out 82.1% of the equity at worst, while the GBNs did 46.7% at worst for NVDA. Considering the LVFI mean and sd for all stocks we note that they are consistently almost half for the GBNs compared to the buy-and-hold strategy. LVFI is important because it is the risk of the initial investment, loosing much of the initial investment may lead to premature withdrawal of funds and/or force liquidation by margin-calls.

All in all, the results above clearly indicate that GBNs are competitive with buy-and-hold in terms of Sharpe ratio, whereas they induce a more desirable behaviour in terms of MDD, LVFI and TIMR.

Post-Analysis One of the benefits of using BNs is that we can get transparency as to why a particular signal was generated. Our aim here was to look at the non-discretized values of the variables at the time a signal was generated. We combined the signals from all simulations (regardless of which stock was traded) and then grouped the signals by which BN generated them and if they were buy or sell signals. We then did pair-wise combinations of the variables in each BN to create scatter plots with values of the variables along the axes and also added an approximated density using the frequency of signals. These scatter plots show when GBNs are generating signals. Examples of these plots for the BNs that generated the most signals are given in Fig. 10 (using 7 from Fig. 8) and Fig. 11 (using 5 from Fig. 8).

In Fig. 10 the BN is used to look for buying opportunities. In the first plot we see that most signals are generated when both $\nabla_1^1 MADIFF(5,20)$ and $\nabla_1^2 MADIFF(5,20)$ are positive, indicating that the difference between the two MAs is growing and increasing in speed, but not so positive so as to making it impossible to benefit from the trend. The second two plots in Fig. 10 plot $\nabla_1^2 MADIFF(5,20)$ against $MADIFF(5,20)$ and the $\nabla_1^1 MADIFF(5,20)$ against $MADIFF(5,20)$. Both these confirm what we knew about the first and second order difference, but also indicate that $MADIFF(5,20)$ should be positive (so the short period MA should be above the long period MA). From a technical analysis perspective this kind of pattern is common, it indicates a trend change, as the shorter MA is moving above and away from the longer MA. It is noteworthy to mention that we have not set any priors on the BNs that would indicate that these are the kind of patterns we are interested in, so our learning algorithm is able to re-discover these human-like commonly used patterns. An example of selling signals is presented in Fig. 11, here we are using RSI which is bounded between 0 and 100. When RSI moves up towards 100 it indicates that the buying pressure is increasing, and should drive prices higher, the opposite is true when RSI moves towards 0. The first plot indicates that most selling signals are generated when $\nabla_1^1 RSI(14)$ is close to zero or negative (i.e. RSI has

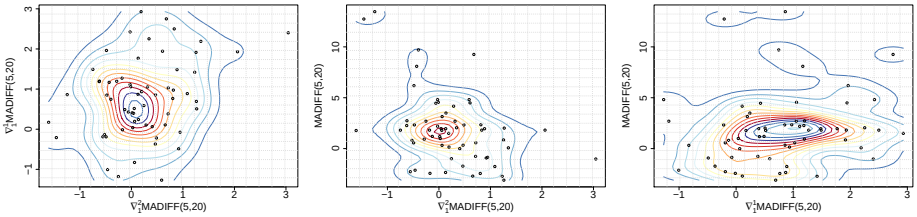

Fig. 10. Buy decisions using 7 from Fig. 8

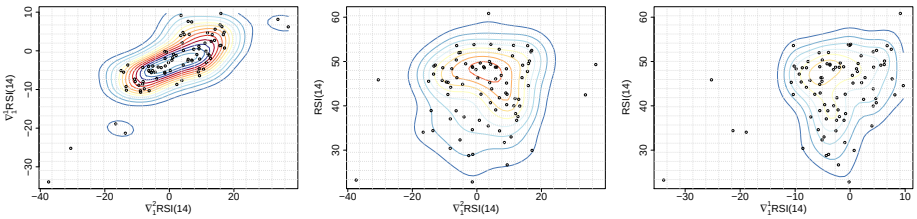

Fig. 11. Sell decisions using 5 from Fig. 8

started to decrease) and $\nabla_1^2 RSI(14)$ is bounded around ± 10. The two other plots in Fig. 11 represent $\nabla_1^2 RSI(14)$ against $RSI(14)$ and $\nabla_1^1 RSI(14)$ against $RSI(14)$. These last two figures confirm our findings in the first figure, but also indicates that the $RSI(14)$ should be below 50 (but not too much below 50 so as to miss the selling opportunity). This seems reasonable from a technical analysis perspective, as RSI goes below 50 and decreases, the selling pressure increases, indicating that the price will go lower, and so a selling signal is generated. We reemphasise that we did not set any prior in the BNs that would suggest that these are the type of signals we should be looking for.

Modelling Different Phases The GBNs used herein do not attempt to switch between BNs to adapt to changes in non-stationary data, but instead they change when the decision being made has changed (i.e. first we are looking to buy, then to sell). GBNs in general model different phases in a process, albeit that data may be non-stationary in some or all phases. This makes GBNs different from formalisms that switch between models to adjust for shifts in non-stationary data, where it is common to take into consideration the performance of the models as part of the weighting or switching probability [12].

6 Conclusions and Future Work

We have introduced a novel algorithm for semi-automatic learning of GBNs, and shown how this algorithm can be used to learn GBNs for use as alpha models in algorithmic trading systems. We have applied the algorithm to evaluate

the expected performance of the learnt GBNs as alpha models compared to using the benchmark buy-and-hold strategy. The results show that learnt GBNs consistently reduce risk with similar or better rewards and do so while at the same time staying out of the market for considerable amounts of time, during these non-invested days the equity is at zero risk and can gain value from risk free assets.

Our future work will include developing the learning algorithm to become more automatic, avoiding having to create a GBN template and rather allow the algorithm to place the phases, BNs and gates in such a way that it optimises some score. We are also interested in combining GBNs with utility and decisions nodes, as are used in influence diagrams. This would allow us to trigger gates depending on the utility of some decision, and this utility could be subject to risk adjustment by using concave utility functions. Furthermore, we have very preliminary ideas on using GBNs to give explanations to models induced by chain graphs and vice versa [13].

Acknowledgments The second author is funded by the Center for Industrial Information Technology (CENIIT) and a so-called career contract at Linköping University, and by the Swedish Research Council (ref. 2010-4808).

References

1. Treleaven, P., Galas, M., Lalchand, V.: Algorithmic Trading Review. Commun. ACM. 56, 76-85 (2013)
2. Nuti, G., Mirghaemi, M., Treleaven, P., Yingsaeree, C.: Algorithmic Trading. Computer. 44, 61-69 (2011)
3. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, (1988)
4. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs. Springer, (2007)
5. Bendtsen, M., Peña, J.M.,: Gated Bayesian Networks. In: 12th Scandinavian Conference on Artificial Intelligence, pp. 35-44. IOS Press, (2013)
6. Pardo, R.: The Evaluation and Optimization of Trading Strategies. John Wiley & Sons, (2008)
7. Chan, E.P.: Quantitative Trading: How to Build Your Own Algorithmic Trading Business. John Wiley & Sons, (2009)
8. Sharpe, W.F.: Likely Gains from Market Timing. Financial Analysts Journal, 31, 60-69, (1975)
9. Tashman, L.J.: Out-of-Sample Tests of Forecasting Accuracy: An Analysis and Review. International Journal of Forecasting, 11, 437-450, (2000)
10. Journal of Technical Analysis, http://www.mta.org
11. Murphy, J.J.: Technical Analysis of the Financial Markets. New York Institute of Finance, (1999)
12. Liehr, S., Pawelzik, K., Kohlmorgen, J., Lemm, S., and Müller, K.-R.: Hidden Markov Gating for Prediction of Change Points in Switching Dynamical Systems. ESANN, 405-410, (1999)
13. Peña, J.M.: Every LWF and AMP Chain Graph Originates From a Set of Causal Models. ArXiv e-prints, (2013)