Bayesian Optimisation of Gated Bayesian Networks for Algorithmic Trading

Marcus Bendtsen
marcus.bendtsen@liu.se
Department of Computer and Information Science
Linköping University, Sweden

Abstract

Gated Bayesian networks (GBNs) are an extension of Bayesian networks that aim to model systems that have distinct phases. In this paper, we aim to use GBNs to output buy and sell decisions for use in algorithmic trading systems. These systems may have several parameters that require tuning, and assessing the performance of these systems as a function of their parameters cannot be expressed in closed form, and thus requires simulation. Bayesian optimisation has grown in popularity as a means of global optimisation of parameters where the objective function may be costly or a black box. We show how algorithmic trading using GBNs, supported by Bayesian optimisation, can lower risk towards invested capital, while at the same time generating similar or better rewards, compared to the benchmark investment strategy *buy-and-hold*.

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 the assets over a long period of time. On 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. An algorithmic trading system consists of several components, some which may be automated by a computer, and others that may be manually executed [1, 2, 3]. At the heart of an algorithmic trading system are the *alpha models*. They are responsible for outputting decisions for buying and selling assets based on the data they are given. These decisions are commonly referred to as *signals*. The data which is supplied to the alpha models varies greatly, e.g. potential prospects, sentiment analysis, previous trades, or *technical analysis*, which will be the focus of the included application. If the signals are followed, then they give rise to certain risk and reward on the initial investment, which will be described further in Section 3.2. Further down the line in algorithmic trading systems are components that combine signals from several alpha models, and other so called risk models, to combine a portfolio of assets. We will not address these later components in this paper, our focus will be on the alpha models.

In Figure 1, the price of an asset is plotted along with buy signals (upward arrows) and sell signals (downward arrows). We view the time spent between these signals as two different phases: before a buy signal, our intention is to have a model that identifies good opportunities to buy the asset, once such an opportunity has been identified and a buy signal has been generated, we move into a different phase. In this second phase, we intend to model the identification of good opportunities to sell the asset. Once a sell signal is generated, we move back to the original phase, once again using a model to generate buy signals. This particular situation was the main motivation for the introduction of *gated Bayesian networks* (GBNs) [4, 5, 6], which we will describe in Section 2.

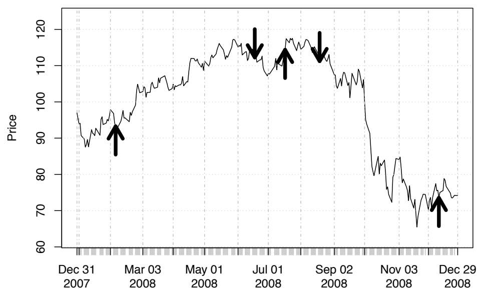

Figure 1: Buy and Sell Signals

Alpha models normally take a set of parameters, allowing them to be tuned to the input data. Naturally, two different sets of parameters may yield two different sets of signals. Therefore, it is imperative to assess how *good* a set of signals are, so that different parameter sets may be compared.

This is usually done by *backtesting*, a type of simulation that calculates certain scores of the signals, e.g. how much the return on the initial investment would have been. Backtesting cannot be written as a function of the alpha model's parameters in closed form, thus it is not possible to analytically find the optimal parameters. Instead, backtesting must be considered a black box function that should be optimised.

Bayesian optimisation has grown in popularity in the machine learning community as an intuitive way of maximising either black box objective functions and/or very costly objective functions (costly in the sense of both time and resources) [7]. Utilising a prior over objective functions, and then sparingly evaluating the objective function at certain points (guided by the posterior), Bayesian optimisation attempts to find the global maximum of the objective function within a predefined grid.

Our intention in this paper is to combine the use of GBNs as alpha models and optimising the parameters of these GBNs using Bayesian optimisation.

The rest of the paper is organised as follows. We begin by giving a brief introduction to GBNs in Section 2, this will illuminate how GBNs can be used as alpha models. We continue by explaining by which metrics alpha models can be evaluated in Section 3, and give slightly more details regarding backtesting. In Section 4 we will describe the components of Bayesian optimisation, including the use of Gaussian processes as priors, as well as kernel and acquisition functions. In Section 5 we will account for the procedure we will use to evaluate the expected performance of using Bayesian optimisation over the parameters of GBNs. Once the procedure has been described, we will in Section 6 account for a real-world application where we show how GBNs can be used as alpha models with support from Bayesian optimisation. Finally, in Section 7 we will offer a few words regarding our conclusions and future work.

2 GATED BAYESIAN NETWORKS

Bayesian networks (BNs) can be interpreted as models of causality at the macroscopic level, where unmodelled 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 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 posterior distribution, which can later be used to answer queries regarding posterior probabilities, interventions, counterfactuals, etc. The independencies represented in the DAG make it possible to compute these queries efficiently. Furthermore, they reduce the number of parameters needed to represent the joint probability distribution, thus making it easier to elicit the probability parameters needed from experts or from data. See [8, 9, 10] for more details.

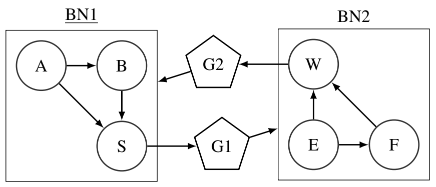

Figure 2: Two Phased GBN

Despite their popularity and advantages, there are situations where a BN is not enough. For instance, when trying to model the process of buying and selling assets, we wanted to model the constant flow between identifying buying opportunities and then, once such have been found, identifying selling opportunities, as is required by an alpha model. These two phases can be very different and the variables included in the BNs modelling them are not necessarily the same. The need to switch between two different BNs was the foundation for the introduction of GBNs.

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 for the activation and deactivation of BNs based on posterior probabilities. A GBN that uses two different BNs (BN1 and BN2) is shown in Figure 2. Here, we will give a brief explanation of GBNs in general, and the GBN in Figure 2 in particular (for the full definition of GBNs see [4, 6]):

- 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.

If the GBN was used as an alpha model, then when the GBN identifies a buying opportunity, and moves to the sell phase, a buy signal is generated. Looking again at Figure 1, each buy and sell signal was generated as the GBN switched back and forth between its phases.

For the purpose of discussing GBN parameter optimisation in general, we will say that a GBN is parameterised by three disjoint parameter sets Θ, Λ and Γ. The parameters in Θ are the parameters of the marginal and conditional probability distributions of the variables in the contained BNs. All other free parameters are contained in Λ, while any fixed parameters are contained in Γ. For instance, in a setting where the only unknowns are the thresholds in the trigger logic of the gates, we say that the thresholds are in Λ and all other parameters are fixed in Γ. This notation allows a bit of convenience when discussing the evaluation of the optimisation procedure in Section 5 and the application in Section 6.

3 EVALUATION OF ALPHA MODELS

As we alluded in Section 1, and as we shall see in Section 6, it is necessary to assess how *good* a set of signals are, thereby assessing the performance of an alpha model. 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 cannot 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 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 [11, 12]. 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 Section 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 included 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 metrics that are common and often taken into consideration [12]. Here we will introduce the metrics that we will use to evaluate the performance of our alpha models.

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 [12]. 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.

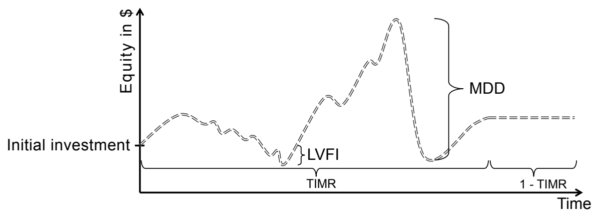

Figure 3: Equity Curve with Drawdown Risks

Furthermore, under certain assumptions it can be shown that there exists an optimal allocation of equity between alpha models (in the portfolio construction model), such that the long-term growth rate of equity is maximised [12]. This growth rate turns out to be $g = r + S^2/2$, where r is the risk free rate and S is the Sharpe ratio. Thus, a high Sharpe ratio is not only an indication of good risk adjusted return, but holding the risk free rate constant, the optimal growth rate is an increasing function of the Sharpe ratio.

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 are known as *drawdown risks*, are presented here (see Figure 3 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 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 (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 investment 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 investment prematurely.

Metric 5 (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.

4 BAYESIAN OPTIMISATION

Our intention is to use GBNs as alpha models and to optimise the free parameters Λ with respect to the metrics given in Section 3.2. In order to do so we must backtest the signals that a GBN produces, and thus we cannot analytically solve the optimisation problem, as backtesting as a function of Λ has no general closed form expression. At the same time, backtesting is relatively costly, as one must create the model, prepare data, estimate parameters, generate signals and walk through the time range to simulate the trading. For this reason, it is not feasible to exhaustively sweep a large grid of parameters. However, Bayesian optimisation allows us to prioritise the points on the grid to evaluate, thus reducing the number of evaluations, while still finding the global maximum of a potentially costly and black box objective function.

4.1 GAUSSIAN PROCESS AS SURROGATE FUNCTION

Essentially, we would like to find the parameters $\Lambda^* \in \mathbf{\Lambda}$ that maximises an unknown function f. We place a prior, $p(f)$, over the possible functions f, and compute the posterior over f using observations $\{\Lambda_{1:i}, f_{1:i}\}$, where $f_j = f(\Lambda_j)$. Hence, we compute $p(f|\{\Lambda_{1:i}, f_{1:i}\}) \propto p(\{\Lambda_{1:i}, f_{1:i}\}|f)p(f)$. We can then use this posterior distribution over objective functions as an estimate of our objective function. This is sometimes known as using the posterior as a *surrogate function* to the true objective function.

In Bayesian optimisation it is common to use a Gaussian process (GP) as the surrogate function [7]. It is defined as a multivariate normal distribution of infinite dimension, where each dimension is a point along some grid. A finite set of these dimensions will form a Gaussian distribution, thus allowing a GP to be defined completely by a mean function μ and a kernel function κ. The GP over the grid $\mathbf{\Lambda}$ is then defined as $\mathcal{N}(\mu(\Lambda), \kappa(\Lambda, \Lambda'))$ for all $\Lambda, \Lambda' \in \mathbf{\Lambda}$. Commonly, the prior $\mu(\Lambda)$ is assumed to be zero for all $\Lambda \in \mathbf{\Lambda}$, although this is by no means necessary if prior information is available to suggest otherwise. The more involved task is to define the kernel function κ. With κ we can express our prior belief about the objective function that we wish to maximise. Although we do not know the form of the objective function, we often assume that points close to each other on the grid give similar results, thus we assume the objective function to possess at least some smoothness. These assumptions can be articulated in κ, for instance by the rational quadratic kernel in Equation 1, where c is a tuning constant for how smooth we believe the objective function to be. For points close to each other, Equation 1 will result in values close to 1, while points further away will be given values closer to 0. The GP prior will obtain the same smoothness properties, as the covariance matrix is completely defined by κ. To visualise the smoothness achieved by tuning c, Figure 4 shows the decreasing covariance as distance grows with three different settings of c (1, 5 and 10). As can be seen, as c increases the decrease is slower, thus more smoothness is assumed.

$$\kappa(\Lambda, \Lambda') = 1 - \frac{||\Lambda - \Lambda'||^2}{||\Lambda - \Lambda'||^2 + c} \tag{1}$$

Assuming that we have observed $\{\Lambda_{1:i}, f_{1:i}\}$, and that we wish to calculate the posterior predictive distribution for an unobserved point Λ_{i+1}, a closed form expression exists for this calculation as described in Equation 2. Thus, it is possible to efficiently calculate the posterior distribution of an unobserved point where both the prior smoothness and observed data have been considered. For more on GPs, please see [13].

$$\begin{aligned}
&\begin{bmatrix} f_{1:i} \\ f_{i+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_*^{\mathbf{T}} \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix}\right) \\
&\mathbf{K} = \begin{bmatrix} \kappa(\Lambda_1, \Lambda_1) & \cdots & \kappa(\Lambda_1, \Lambda_i) \\ \vdots & \ddots & \vdots \\ \kappa(\Lambda_i, \Lambda_1) & \cdots & \kappa(\Lambda_i, \Lambda_i) \end{bmatrix} \\
&\mathbf{K}_* = \begin{bmatrix} \kappa(\Lambda_{i+1}, \Lambda_1) & \cdots & \kappa(\Lambda_{i+1}, \Lambda_i) \end{bmatrix} \\
&\mathbf{K}_{**} = \kappa(\Lambda_{i+1}, \Lambda_{i+1}) \\
&p(f_{i+1} | \{\Lambda_{1:i}, f_{1:i}\}) = \mathcal{N}(\mu_i(\Lambda_{i+1}), \sigma_i^2(\Lambda_{i+1})) \\
&\mu_i(\Lambda_{i+1}) = \mathbf{K}_* \mathbf{K}^{-1} f_{1:i} \\
&\sigma_i^2(\Lambda_{i+1}) = \mathbf{K}_{**} - \mathbf{K}_* \mathbf{K}^{-1} \mathbf{K}_*{}^T
\end{aligned} \tag{2}$$

4.2 ACQUISTION FUNCTIONS AND BAYESIAN OPTIMISATION

Using a GP as a surrogate to the objective function allows us to encode prior beliefs about the unknown objective function, and sampling the objective function allows us to update the posterior of the surrogate. What is left to do is to decide where to sample the objective function.

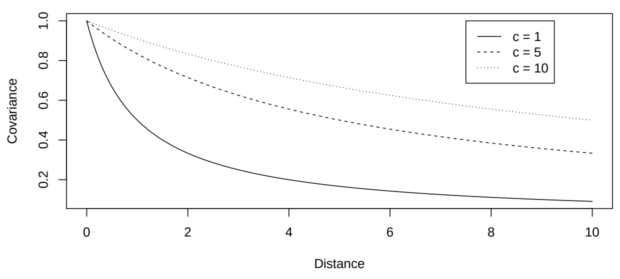

Figure 4: Covariance Decrease by Distance

In Bayesian optimisation we make use of a so called *acquisition function*. Several acquisition functions have been suggested, however the goal is to trade off exploring the grid where the posterior uncertainty is high, while exploiting points that have a high posterior mean. We will use the *upper confidence bound criterion*, which is expressed as $UCB(\Lambda) = \mu(\Lambda) + \eta\sigma(\Lambda)$, where $\mu(\Lambda)$ and $\sigma(\Lambda)$ represent the mean and standard deviation at the point Λ of the GP, and η is a tuning parameter to allow for more exploration (as η is increased) or more exploitation (as η is decreased).

Succinctly, define a GP over a grid with some kernel function, then randomly sample a point and evaluate the objective function at this point. Calculate the posterior of the GP given this new observation and find Λ' that maximises the acquisition function. Then Λ' is the next point where to evaluate the objective function. Iterate these steps for a predefined number of iterations. Once all iterations have passed, the Λ with the highest posterior mean is the set of parameters that maximises the objective function.

5 EVALUATION PROCEDURE

In Section 6 we will account for a real-world application of GBNs as alpha models supported by Bayesian optimisation. However, in this section we will introduce the optimisation procedure used, as well as the method used to evaluate the performance of the optimisation, which is essentially the same method used in [5].

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 optimisation. Starting from index 1, blocks 1,...,k are used for optimisation and $k + 1$ for testing, thus ensuring that the evidence sets in the testing data occurs after the optimisation data. The procedure is then repeated starting from index 2 (i.e. blocks 2, ..., k+1

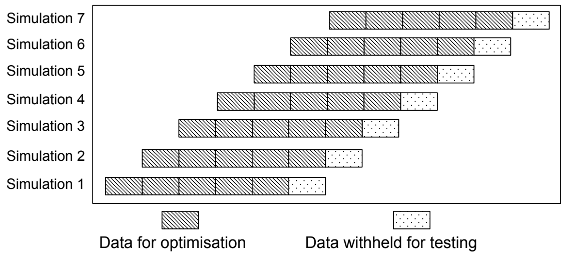

Figure 5: Data Split For Optimisation and Testing

are used for optimisation and $k + 2$ for testing). By doing so we create t repeated simulations, moving the testing data one block forward each time. An illustration of this procedure when $n = 12$, $k = 5$ and $t = 7$ is shown in Figure 5.

During Bayesian optimisation, when the objective function is evaluated for some acquired Λ, a cross-validation estimate is calculated for the k blocks used. Here, $k-1$ blocks are used to estimate the parameters Θ of the contained BNs and the held out block is used as validation data to calculate a score ρ. The value of the objective function, given parameters Λ, is thus the average of all ρ when each block in the optimisation data has been held out.

In order to formalise the procedure used to evaluate the optimisation, recall from Section 2 that Λ is used to represent the free parameters of a GBN and Γ is used to represent all fixed parameters. Let $\mathcal{J}$ be a score function such that $\mathcal{J}(\Lambda, \mathcal{D}_j, \{\mathcal{D}\}_l^m | \Gamma)$ is the score for a GBN under some parameterisation Λ and Γ when block j has been used for either testing or validation and the blocks $\mathcal{D}_l, ..., \mathcal{D}_m$ have been used to estimate Θ of the BNs in the GBN (under the parameters Λ and Γ).

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 optimisation data, use Bayesian optimisation to find the parameters Λ^t that satisfies Equation 3.

$$\Lambda^t = \underset{\Lambda \in \mathbf{\Lambda}}{arg\ max} \frac{1}{k} \sum_{j=t}^{t+k-1} \mathcal{J}(\Lambda, \mathcal{D}_j, \{\mathcal{D}\}_t^{t+k-1} \backslash \mathcal{D}_j | \Gamma) \tag{3}$$

2. For each Λ^t calculate the score $\rho_{\mathcal{J}}^t$ on the testing set with respect to the scoring function $\mathcal{J}$ according to Equation 4.

$$\rho_{\mathcal{J}}^t = \mathcal{J}(\Lambda^t, \mathcal{D}_{t+k}, \{\mathcal{D}\}_t^{t+k-1} | \Gamma) \tag{4}$$

3. The expected performance $\bar{\rho}_{\mathcal{J}}$ of the optimisation, with respect to the score function $\mathcal{J}$, is then given by the average of the scores $\rho_{\mathcal{J}}^t$, i.e. $\bar{\rho}_{\mathcal{J}} = \frac{1}{t} \sum_t \rho_{\mathcal{J}}^t$.

Two things to note about this procedure. First, during cross-validation inside the objective function we disregard the natural order of the data, thus allowing a validation block to come before a block used for estimating the parameters Θ. This could potentially induce a bias in the cross-validation estimate as the data used for estimating the parameters would not have been known at the time the data for the validation block was generated. However, as we do not use this scheme when we evaluate the performance of the optimisation, the expected performance of the optimisation is not biased in this way. We simply use this scheme to make the best use of the data during cross-validation. Second, one scoring function $\mathcal{J}$ has been used both during optimisation and for evaluating the expected performance of the optimisation. The scoring function $\mathcal{J}$ could internally use many different metrics to come up with one score to maximise. However, it is natural in the coming setting to expose the actual values of several metrics, thus several scoring functions $\mathcal{J}$ are used to get a vector of mean scores $[\bar{\rho}_{\mathcal{J}_1}, ..., \bar{\rho}_{\mathcal{J}_m}]$.

Another approach to combine Bayesian optimisation with cross-validation is to reduce the number of fold evaluations necessary [14], as certain folds may be closely correlated, however our approach is to reduce the number of parameters that we need to test with cross-validation.

6 APPLICATION

Having established the optimisation procedure, and the method we intend to use to evaluate the performance of the optimisation, we turn our attention to a real-world application. We aim to use GBNs as alpha models to generate buy and sell signals of *stock indices* in such a way that drawdown risks are mitigated, compared to the buy-and-hold strategy, while at the same time maintaining similar or better rewards.

Stock indices are weighted averages of their respective stock components. For instance, the Dow Jones Industrial Average (DJIA) is a weighted average of 30 large companies based in the United States. Indices may have different schemes for how the different components are weighted, however they all aim to give a collective representation of their components.

An *index fund* owns shares of the components of a specific index, proportional to the weights, such that the fund's return is mirrored by the index. These funds are very popular, as they are easy for the investor to comprehend but at the same time trading the individual components of an index requires a lot of effort.

A buy-and-hold strategy on stock indices via index funds may be convenient, however it implies that the equity is put through the full force of drawdown risks described in Section 3.2. 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

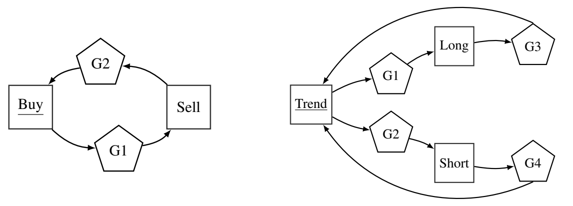

Figure 6: GBN-1 and GBN-2

throughout the period, the lowest point of the assets value will coincide with LVFI. In dwindling stock markets, the index funds will lose value, and equity could be salvaged and possibly be placed in risk-free assets during these periods. Furthermore, utilising certain financial products, it is also possible to increase equity during these times of distress by purchasing *short* positions of the index. Short positions can be thought of as a loan, where the value of the loan increases if the index decreases in value, and it is possible to sell the loan at its higher value (to make the distinction, regular positions are called *long* when short positions are considered).

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 [15]. 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).

6.1 METHODOLOGY

We used two different GBN structures to create alpha models. The first GBN structure (henceforth known as GBN-1) modelled buying and selling long positions only, while the second GBN structure (GBN-2) modelled buying and selling long and short positions. The structures are depicted in Figure 6 (GBN-1 on the left and GBN-2 on the right). The structure for GBN-1 works as described in Section 2. The structure for GBN-2 starts in the $Trend$ phase, from where either $G1$ or $G2$ can trigger. If $G1$ triggers then a long open signal is generated and the $Long$ phase is activated (deactivating the $Trend$ phase). If then gate $G3$ triggers then a long close signal is generated, and the $Trend$ phase is activated again (deactivating the $Long$ phase). However, if before $G1$ triggers $G2$ triggers instead, then a short open position is generated, and the $Short$ phase is activated (deactivating $Trend$). In similar fashion, when $G4$ triggers a short close signal is generated, activating $Trend$ and deactivating $Short$.

6.1.1 Variables

The variables used in the GBNs were discretisations of so called *technical analysis indicators*. 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. For the full definition of these indicators, please see [16, 17].

For each phase in the GBNs (Buy, $Sell$, $Trend$, $Long$ and $Short$), we placed a naïve Bayesian classifier over the same technical analysis indicators. However by allowing the parameterisation of one of the technical analysis indicators to vary between the phases, we essentially created different variables in the different phases. The tuning of the technical analysis parameters allowed us to better capture the dynamics of the data, as they may differ between assets as well as between the different phases of trading.

Figure 7 depicts the classifier structure and variables used. The variables are explained below, along with an example in Figure 8.

- S represents the first-order finite backward difference of 5 periods of the MA of ψ periods, shifted 5 periods into the future. To clarify, the first plot in Figure 8 shows the price of an asset along with the MA. If the current time is t, then S represents the slope of the line between what the MA will be at $t + 5$ and what it is at t.
- A represents the same slope as S but at its current value (i.e. between t and $t - 5$).
- B represents the difference between the current value of the MA of ψ periods and the current raw price. This can be seen in Figure 8 as the difference between the two time series in the first plot.
- C represents the current RSI value (at t in the second plot of the figure) using 14 periods.
- D represents what C was 5 time steps in the past (at $t - 5$ in the second plot of the figure).

The choice of 14 periods for RSI is based on the prevailing standard [16, 17], and the choice of 5 periods as the prediction horizon is based on the number of trading days in a week.

Variables A, B, C and D were discretised into six bins, each using equal width binning, and S was discretised into two bins separated by zero. Thus, the states of S represents

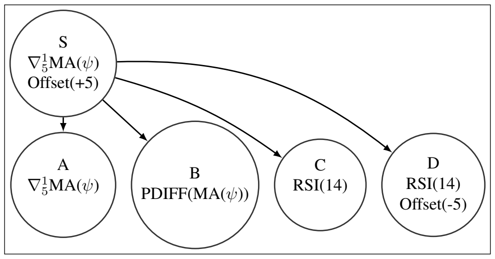

Figure 7: Bayesian Classifier in GBN Phases

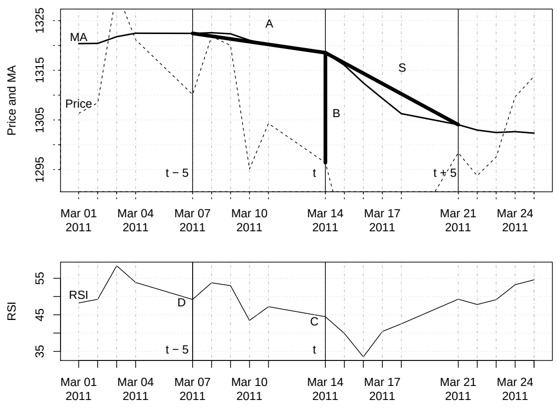

Figure 8: Visualisation of Variables

a predicted positive or negative future value of the modelled asset price (smoothed by the moving average of ψ periods). As S represents a future value, evidence for S was only available during estimation of the parameters Θ, not during the generation of signals.

The gates all defined trigger logic over the posterior distribution of S with some threshold τ. For instances, in GBN-1, the trigger logic for $G1$ was $TL(G1)$: $p(S = positive \mid \mathbf{e}) > \tau_{G1}$, i.e. if the posterior probability of a positive climate is greater than some threshold, then the model should give a buy signal and move to the next phase (the sell phase). Naturally, the trigger logic for $G2$ in the same GBN was $TL(G2) : p(S = negative \mid \mathbf{e}) > \tau_{G2}$, thus giving a sell signal if the posterior probability of a negative future value exceeds some threshold.

6.1.2 Bayesian Optimisation Settings

The previous section implies the following for the two GBNs:

- For GBN-1 the free parameters Λ to be optimised are: $\Lambda = \{\tau_{G1}, \tau_{G2}, \psi_{Buy}, \psi_{Sell}\}$.
- For GBN-2 the free parameters Λ to be optimised are: $\Lambda = \{\tau_{G1}, \tau_{G2}, \tau_{G3}, \tau_{G4}, \psi_{Trend}, \psi_{Long}, \psi_{Short}\}$.
- In both cases all τ were confined to $[60, 90]$ and all ψ to $[10, 40]$.

We used the upper confidence bound acquisition function (as described in Section 4.2) with $\eta = 5$, which allowed for abundant exploration, as our objective function was not extremely expensive to evaluate. We used the rational quadratic kernel as described in Equation 1 with $c = 1$.

For GBN-1 we ran the Bayesian optimisation for 1,600 iterations, and for GBN-2 we ran 12,800 iterations.

6.1.3 Data Sets

We used four indices in this study, DJIA and NASDAQ which are both based on companies in the United States, FTSE100 which is based on companies in the United Kingdom and DAX which is based on companies located in Germany. We ran our experiments on daily adjusted closing prices for these indices ranging from 2001-01-01 to 2012-12-28 (data downloaded from Yahoo! Finance$^{\text{TM}}$). This gave a total of 12 years of price data for each index, where each year was allocated to a block, thus $n = 12$. For the cross-validation step we used $k = 5$ giving $t = 7$ simulations from which to calculate $[\bar{\rho}_{\mathcal{J}_1}, ..., \bar{\rho}_{\mathcal{J}_m}]$ (the data split is depicted in Figure 5).

6.1.4 Scoring Functions

The signals generated were backtested in order to calculate relevant metrics. During optimisation (i.e. step 1 in Section 5) the objective function used the Sharpe ratio. The choice was made as it combines both risk and reward into one score, for which a cross-validation estimate could be returned by the objective function. For evaluating the performance of the optimisation (step 2 in Section 5), we used the return and the drawdown risks described in Section 3.2 to create a score vector $[\bar{\rho}_{\mathcal{J}_1}, ..., \bar{\rho}_{\mathcal{J}_m}]$. The same metrics were calculated for the buy-and-hold strategy.

6.2 RESULTS AND DISCUSSION

The score vectors from the evaluation of the optimisation versus the the score vector for the buy-and-hold strategy over the seven simulations are shown in Table 1. The annual Sharpe presented in the table is the mean return divided by the standard deviation of returns over the seven simulation, and since each block was allocated one year of data it becomes the *annual* Sharpe ratio.

Will will first turn our attention to GBN-1. We use the Sharpe ratio as our measure of reward, prioritised above the raw return for reasons discussed in Section 3.2. Therefore, we must first ensure that the Sharpe ratio of our algorithmic trading system produces similar or better Sharpe ratios than the buy-and-hold strategy. As can be seen, this was the case for DJIA, NASDAQ and DAX, but not for

Table 1: Metric Values for GBNs and Buy-and-Hold

DJIA	Score	GBN-1	GBN-2	BaH
	Annual Sharpe	0.289	**0.330**	0.157
	Return	0.019	**0.032**	0.028
	MDD	**0.085**	0.116	0.167
	LVFI	**0.058**	0.062	0.119
	TIMR	**0.628**	0.91	1.0
NASDAQ	Score	GBN-1	GBN-2	BaH
	Annual Sharpe	**0.308**	0.081	0.254
	Return	0.033	0.012	**0.067**
	MDD	**0.101**	0.164	0.207
	LVFI	**0.062**	0.099	0.146
	TIMR	**0.554**	0.94	1.0
FTSE100	Score	GBN-1	GBN-2	BaH
	Annual Sharpe	-0.057	-0.64	**0.127**
	Return	-0.006	-0.074	**0.022**
	MDD	**0.098**	0.167	0.188
	LVFI	**0.074**	0.121	0.142
	TIMR	**0.649**	0.962	1.0
DAX	Score	GBN-1	GBN-2	BaH
	Annual Sharpe	**0.778**	0.589	0.278
	Return	**0.081**	0.062	0.069
	MDD	**0.107**	0.171	0.213
	LVFI	**0.056**	0.059	0.154
	TIMR	**0.610**	0.926	1.0

FTSE100. Secondly, we must take into consideration the TIMR. For GBN-1, we were invested only slightly above half of the time compared to buy-and-hold, reducing risk to equity considerably. Meanwhile, the rest of the time the equity could have gained in value from interest rates (or other risk-free assets), this potential gain was not considered in these results. Risk to equity from MDD was half its counterpart from the buy-and-hold strategy for all indices. The LVFI is a major threat to equity (as discussed in Section 3.2), and one where buy-and-hold severely underperforms. For DAX the LVFI was only a third of the buy-and-hold LVFI, and for the other three indices it was half.

All in all, the results clearly indicate that GBN-1 was competitive with the buy-and-hold strategy for three of the indices, as Sharpe ratios were improved upon and risk to equity was decreased significantly. Furthermore, these results were achieved while at the same time only having equity invested half of the investment period. It is also clear that we cannot expect the same GBN to be useful for all indices, as the reward was not improved upon for FTSE100. Some of the parameters that were fixed in Γ may have to be tuned in order to accommodate the dynamics of FTSE100, such as the technical analysis indicators used, or the fixed parameters of the ones used currently.

Moving on to GBN-2, we can see that allowing the GBN to open short positions changes the results dramatically. For DJIA, we improved upon the Sharpe ratio, at the cost of the drawdown risks. Both MDD and LVFI were increased marginally, yet still lower than buy-and-hold. The TIMR was also increased to such a degree that we were invested almost the entire investment period. There is potential gain in reward from using GBN-2 for DJIA, however the increased risk must be considered.

For NASDAQ, FTSE100 and DAX there is no improvement over GBN-1. Instead, Sharpe ratios are decreased, as well as an increase in drawdown risks. There could be several reasons for this that are worth investigating, however our immediate response is that we have either overfitted the model due to several more parameters being optimised over the same amount of data, or the fact that a bad short position is doubly bad on equity as we will lose out of the profit from a long position during the same time.

7 CONCLUSIONS AND FUTURE WORK

Our results show that it is feasible to use GBNs as alpha models, and to use Bayesian optimisation to tune them in order to beat the buy-and-hold benchmark, with respect to certain risk and reward metrics. Some of the design decisions made before optimisation may however have reduced the performance of the GBNs on some of the used data sets. Short positions are optimally taken during times of distress, and due to increased volatility, markets move very differently compared to stable increasing markets. We decided to lock in the forward and backward horizons to 5 time steps, and the RSI period of 14, which may have made it impossible to capture the more volatile dynamics. Furthermore, stock indices generally increase in value over long periods of time, thus short selling will always be in the opposite of the long term trend, which in general is ill-advised.

Nevertheless, we are encouraged to see the included positive results and are at the same time motivated to address the problems we faced with GBN-2. We would not expect the exact same model to perform well on all given data sets, and so further work is needed to improve upon the results on FTSE100 to make them in par with the other three indices. For instance, there is room to make the objective function even more expensive by not only estimating BN parameters, but also performing variable selection and structure learning during cross-validation.

Acknowledgements

BN inference in our implementation is based on the SMILE reasoning engine, contributed to the community by the Decision Systems Laboratory of the University of Pittsburgh and available at https://dslpitt.org/genie/.

References

[1] P. Treleaven, M. Galas, and V. Lalchand, "Algorithmic trading review," *Communications of the ACM*, vol. 56, no. 11, pp. 76–85, 2013.

[2] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, "Algorithmic trading," *Computer*, vol. 44, no. 11, pp. 61–69, 2011.

[3] R. K. Narang, *Inside the black box*. John Wiley & Sons, 2013.

[4] M. Bendtsen and J. M. Peña, "Gated Bayesian networks," in *Proceedings of the Twelfth Scandinavian Conference on Artificial Intelligence*, pp. 35–44, 2013.

[5] M. Bendtsen and J. M. Peña, "Learning gated Bayesian networks for algorithmic trading," in *Proceedings of the Seventh European Workshop on Probabilistic Graphical Models*, pp. 49–64, 2014.

[6] M. Bendtsen and J. M. Peña, "Gated Bayesian networks for algorithmic trading," *International Journal of Approximate Reasoning*, 2015, submitted.

[7] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," Tech. Rep. UBC TR-2009-023 and arXiv:1012.2599, 2009.

[8] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers, 1988.

[9] F. V. Jensen and T. D. Nielsen, *Bayesian networks and decision graphs*. Springer, 2007.

[10] K. B. Korb and A. E. Nicholson, *Bayesian artificial intelligence*. Taylor and Francis Group, 2011.

[11] R. Pardo, *The evaluation and optimization of trading strategies*. John Wiley & Sons, 2008.

[12] E. P. Chan, *Quantitative trading*. John Wiley & Sons, 2009.

[13] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[14] K. Swersky, J. Snoek, and R. P. Adams, "Multi-task Bayesian optimization," in *Advances in Neural Information Processing Systems 26*, pp. 2004–2012, 2013.

[15] W. F. Sharpe, "Likely gains from market timing," *Financial Analysts Journal*, vol. 31, no. 2, pp. 60–69, 1975.

[16] J. J. Murphy, *Technical analysis of the financial markets*. New York Institute of Finance, 1999.

[17] M. J. Pring, *Technical analysis explained*. McGraw-Hill, 2002.