DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Bayesian Neural Networks for Financial Asset Forecasting

ALEXANDER BACK

WILLIAM KEITH

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Bayesian Neural Networks for Financial Asset Forecasting

ALEXANDER BACK

WILLIAM KEITH

Degree Projects in Financial Mathematics (30 ECTS credits)
Master's Programme in Industrial Engineering and Management (120 credits)
KTH Royal Institute of Technology year 2019
Supervisor at Lynx Asset Management:
Max Nyström Winsa, Hannes Meier, Jesper Welander
Supervisor at KTH: Timo Koski
Examiner at KTH: Timo Koski

TRITA-SCI-GRU 2019:095
MAT-E 2019:51

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Bayesian Neural Networks for Financial Asset Forecasting

Abstract

Neural networks are powerful tools for modelling complex non-linear mappings, but they often suffer from overfitting and provide no measures of uncertainty in their predictions. Bayesian techniques are proposed as a remedy to these problems, as these both regularize and provide an inherent measure of uncertainty from their posterior predictive distributions.

By quantifying predictive uncertainty, we attempt to improve a systematic trading strategy by scaling positions with uncertainty. Exact Bayesian inference is often impossible, and approximate techniques must be used. For this task, this thesis compares dropout, variational inference and Markov chain Monte Carlo. We find that dropout and variational inference provide powerful regularization techniques, but their predictive uncertainties cannot improve a systematic trading strategy. Markov chain Monte Carlo provides powerful regularization as well as promising estimates of predictive uncertainty that are able to improve a systematic trading strategy. However, Markov chain Monte Carlo suffers from an extreme computational cost in the high-dimensional setting of neural networks.

Keywords: Bayesian neural networks, variational inference, Markov chain Monte Carlo, dropout, systematic trading, futures contracts.

Bayesianska Neurala Nätverk för Prediktion av Finansiella Tillgångar

Sammanfattning

Neurala nätverk är kraftfulla verktyg för att modellera komplexa icke-linjära avbildningar, men de lider ofta av överanpassning och tillhandahåller inga mått på osäkerhet i deras prediktioner. Bayesianska tekniker har föreslagits för att råda bot på dessa problem, eftersom att de både har en regulariserande effekt, samt har ett inneboende mått på osäkerhet genom den prediktiva posteriora fördelningen.

Genom att kvantifiera prediktiv osäkerhet försöker vi förbättra en systematisk tradingstrategi genom att skala modellens positioner med den skattade osäkerheten. Exakt Bayesiansk inferens är oftast omöjligt, och approximativa metoder måste användas. För detta ändamål jämför detta examensarbete dropout, variational inference och Markov chain Monte Carlo. Resultaten indikerar att både dropout och variational inference är kraftfulla regulariseringstekniker, men att deras prediktiva osäkerheter inte kan användas för att förbättra en systematisk tradingstrategi. Markov chain Monte Carlo ger en kraftfull regulariserande effekt, samt lovande skattningar av osäkerhet som kan användas för att förbättra en systematisk tradingstrategi. Dock lider Markov chain Monte Carlo av en enorm beräkningsmässig komplexitet i ett så högdimensionellt problem som neurala nätverk.

Nyckelord: Bayesianska neurala nätverk, variational inference, Markov chain Monte Carlo, dropout, systematisk trading, terminskontrakt.

Acknowledgements

We want to thank Lynx Asset Management for providing us the opportunity to write this thesis. Lynx has provided data, a place to work and above all; extensive experience and knowledge. A special thanks to our supervisors Max Nyström Winsa, Hannes Meier and Jesper Welander.

Finally we want to thank our supervisor at KTH, Timo Koski, for his help and guidance with the thesis.

Alexander Back & William Keith,
Stockholm, May 2019.

Contents

1 Introduction **10**
1.1 Research Questions . . . 11
1.2 Scope & Limitations . . . 11
1.3 Related Work . . . 12

2 Financial Background **13**
2.1 Futures Contracts . . . 13
2.2 Efficient Market Hypothesis . . . 14
2.2.1 Weak Form . . . 14
2.2.2 Semi-strong Form . . . 14
2.2.3 Strong Form . . . 15
2.3 Sharpe Ratio . . . 15

3 Neural Networks **16**
3.1 Feed-forward Neural Networks . . . 16
3.2 Activation Functions . . . 17
3.3 Loss Functions . . . 18
3.3.1 Mean Squared Error . . . 18
3.4 Regularization . . . 19
3.4.1 L^1-regularization . . . 19
3.4.2 L^2-regularization . . . 20
3.4.3 Dropout . . . 20

4 Bayesian Inference **21**
4.1 Introduction to Bayesian Inference . . . 21
4.2 Uncertainty . . . 22
4.3 Markov Chain Monte Carlo . . . 23
4.3.1 The Metropolis Algorithm . . . 24

4.3.2 Hamiltonian Monte Carlo 24
4.4 Variational Inference 27

5 Bayesian Neural Networks 29
5.1 Hamiltonian Monte Carlo 30
5.2 Variational Inference 30
5.2.1 Bayes by Backprop 31
5.2.2 Covariance estimation with high dimensionality 32
5.3 Dropout 33

6 Methodology 34
6.1 Data 34
6.1.1 Training, test and validation 35
6.1.2 Filters 36
6.1.3 Volatility Estimators 37
6.1.4 Features 37
6.1.5 Dimensionality Reduction 39
6.2 Bayesian Neural Networks 39
6.2.1 Network Architecture 39
6.2.2 Dropout 41
6.2.3 Markov Chain Monte Carlo 41
6.2.4 Variational Inference 44
6.3 Trading Strategies 45
6.3.1 Long Only 45
6.3.2 Frequentist 46
6.3.3 Bayesian 46
6.4 Hyperparameter Optimization 46
6.5 Software Implementation 49

7 Results 50
7.1 Uncertainties 50
7.1.1 Dropout 51
7.1.2 Variational Inference 52
7.1.3 Markov Chain Monte Carlo 53
7.2 Trading Performance 54
7.2.1 Long only 54
7.2.2 Dropout 55
7.2.3 Variational Inference 57

7.2.4 Markov Chain Monte Carlo 58
7.2.5 Markowitz Optimizaion 60

8 Discussion **61**
8.1 Dropout 61
8.2 Variational Inference 62
8.3 Markov Chain Monte Carlo 62

9 Conclusions **65**
9.1 Research Questions 65

Bibliography **66**

List of Figures

3.1 Feed-forward neural network with 2 hidden layers 17

4.1 Visualization of variational inference 27

5.1 Left: Standard NN with point estimates for weights. Right: Bayesian NN with probability distributions over weights [3]. . 29

6.1 Left: Unnormalized series of daily returns exhibiting clear volatility clustering. Right: Same return series, normalized with a rolling volatility estimator. 35
6.2 QQ-plot of the normalized returns of one market against an MLE fitted Gaussian. 42

7.1 Prediction error as a function of uncertainties, validation set . 51
7.2 Prediction error as a function of uncertainties, test set 52
7.3 Prediction error as a function of uncertainties, test set 52
7.4 Prediction error as a function of uncertainties, validation set . 53
7.5 Prediction error as a function of uncertainties, test set 53
7.6 Prediction error as a function of uncertainties, validation set . 54
7.7 Equity curve, long only strategy 55
7.8 Equity curve, frequentist strategy 56
7.9 Equity curve, Bayesian strategy 56
7.10 Equity curve, frequentist strategy 57
7.11 Equity curve, Bayesian strategy 58
7.12 Equity curve, frequentist strategy 59
7.13 Equity curve, Bayesian strategy 59
7.14 Equity curve, Markowitz optimized strategy 60

8.1 Log posterior probability 63

8.2 Autocorrelations of the MCMC chain for two different weights 63
8.3 MCMC chains for two different weights 64

List of Tables

6.1 Training, validation and test sets 36
6.2 Hyperparameter sampling procedure for the dropout model . . 47
6.3 Hyperparameter sampling procedure for the dropout model . . 48
6.4 Hyperparameters for HMC 49

7.1 Sharpe ratios for strategies based on the dropout model 55
7.2 Sharpe ratios for strategies based on MCMC 57
7.3 Sharpe ratios for strategies based on MCMC 58
7.4 Sharpe ratios for strategies based on MCMC 60

Chapter 1

Introduction

Neural networks have gained significant attention in the last decade due to successful applications to areas such as computer vision, natural language processing and reinforcement learning. Advances in hardware, learning algorithms and software libraries have enabled the use of deep neural networks, which are able to approximate complex, non-linear mappings. However, due to their inherent complexity, neural networks are prone to overfitting and require tuning of a large number of hyperparameters. Furthermore, neural networks lack the proper theoretical foundation, and practitioners often rely on common "best practices" for model and hyperparameter selection.

Bayesian approaches to neural networks have been suggested as a remedy to these problems. Bayesian neural networks (BNNs) are more robust to overfitting, and do not require quite as many hyperparameters. Moreover, Bayesian neural networks provide an inherent estimate of prediction uncertainty, expressed through the posterior predictive distribution. That is, in contrast to the frequentist approach we are provided with a probability distribution of predicted values, instead of single estimates. There are drawbacks of Bayesian approaches as well, which is the reason they have not gained momentum in current research. The main complication is the computational intractability of Bayesian posterior distributions. There have been significant advances on approximate inference, but these are still much more computationally expensive compared to frequentist inference.

There is an increasing interest in applications of neural networks to problems in finance. A fundamental task in finance is the prediction of future asset returns, using previous returns. Finding signal in financial data is notoriously difficult, as financial data is extremely noisy. Neural networks often struggle in this context, and provide poor generalization to unseen data. Furthermore, neural networks provide no confidence in their predictions except for softmax probabilities, which are often erroneously interpreted as confidence levels. In these regards, Bayesian approaches to neural networks show potential for applications in finance. A desirable property of Bayesian models is to increase uncertainty when presented with data that is very different from the training data. In particular, we desire the model to become uncertain when it is prone to error. Then, in a systematic trading strategy, we can decrease our position in an asset when the model is uncertain about future returns. Therefore, the connection between posterior predictive uncertainty and prediction error is at the focal point of this thesis.

1.1 Research Questions

The aim of this thesis is to research the application of Bayesian neural networks to financial time series. In particular, the connection between predictive uncertainty and error will be examined. This thesis will seek to answer the following questions:

- What method for approximate Bayesian inference in neural networks is most suitable for financial time series?
- How is the uncertainty of the posterior predictive distribution of a Bayesian neural network related to its prediction errors?
- By utilizing predictive uncertainty, can we improve a systematic trading strategy compared to the case with frequentist predictions?

1.2 Scope & Limitations

The scope of this thesis is to study the application of Bayesian neural networks for predicting financial asset returns, in a regression setting. The main focus for the study is the uncertainty estimates provided by the Bayesian models, and their relation to prediction error. The thesis is limited to studying Monte Carlo dropout, variational inference, and Markov chain Monte Carlo for approximate Bayesian inference, and compare these to a non-Bayesian benchmark.

The main limitation of this thesis is the computational complexity of the models in question. Thus, it is impossible to conduct exhaustive hyperparameter searches for all models. In particular, the methods for approximate Bayesian inference using Markov chain Monte Carlo will present the largest computational costs. Not only are these methods computationally expensive, but they are also notoriously difficult to diagnose.

1.3 Related Work

The research field on Bayesian neural networks has grown significantly since the early work of Hinton and van Camp in 1993 [15]. Neal [25] presented in his Ph.D. thesis an application of Hamiltonian Monte Carlo to neural networks, which has since served as the gold standard for sampling based methods of approximate inference. Other interesting approaches for MCMC methods have been proposed in recent years, that use stochastic gradient Langevin dynamics to perform online MCMC inference, in contrast to the HMC algorithm, which only works for a full batch of data [31], [5], [8].

Graves [13] introduced a variational approach for approximate posterior inference in neural networks, that assumes a diagonal Gaussian distribution on the posterior of the weights. Blundell et al. [3] extended this approach to allow for arbitrary variational posteriors, and used the local reparameterization trick from Kingma et al. [20] to allow for unbiased stochastic gradients. Further extensions to this approach have been proposed by Wen et al. [32], that use a technique called ”flipout” to decorrelate gradients over a mini-batch.

The above approaches suffer from computationally expensive training procedures, require explicitly defining a priors over weights and require significant alterations of standard NN models. In contrast to this, Gal [12] presented a method of approximate inference in NNs based on dropout, called Monte Carlo dropout. The idea is to randomly drop activations during both training and test, which can be shown to approximate Bayesian inference in deep Gaussian processes.

As for the application of BNNs to financial time series, there is not any published research, to the best of our knowledge. There is, however, research done on BNNs for time series modelling, where the predictive uncertainty is used to detect anomalies in the number of trips at Uber [34].

Chapter 2

Financial Background

This chapter gives an introduction to the financial aspects of this thesis. We aim to describe the futures contract, which is the financial asset this thesis is mainly concerned with. We also discuss the efficient market hypothesis and its implications on predicting futures returns. We continue by discussing the Sharpe ratio as a metric of investment performance and conclude the chapter with introducing some means of performing portfolio optimization.

2.1 Futures Contracts

A futures contract is most easily thought of as an extension of a forward contract. A forward contract is an agreement made at time 0 between two parties to trade an underlying asset X for the forward price K at time $T > 0$. Less stringently, this means that the buying party agrees to buy an asset, crude oil for example, at some specified time in the future, for some specified price. Thus, the parties agree to trade in the future, but the trade is not realized until the specified future date. The forward contract allows the parties to secure themselves from the risk of the underlying asset fluctuating in value. This is useful for a wide variety of market participants, but a simple example is a large coffee chain wishing to hedge against the future movements of the price of coffee beans and thus buying the beans ahead of time for a price specified in advance.

The futures contract is a natural extension of the forward contract, addressing some problems with the forward contract such as counter-party risk and the binding obligation to trade the underlying asset. The futures contract is set up like the forward contract, with the key difference that the contract is settled continuously instead of at the end time T. We formalize this by denoting the *future price* at time t by $F(t, T)$. The buyer then for every time step t, underlying asset X and delivery date T receives the amount:

$$F(t, T, X) - F(t - 1, T, X). \tag{2.1}$$

As such the underlying asset is not necessarily actually traded, instead the price movements of the underlying asset is settled so that for every time step t the buyer of the contract receives the difference in underlying value to the previous time step. This implies that the value of a futures contract is always equal to 0, making it free to enter and exit.

2.2 Efficient Market Hypothesis

Since the aim of this thesis can be boiled down to attempting to predict future market movements using only past market movements as input, a discussion of the efficient market hypothesis (EMH) seems appropriate. The EMH has been a core concept in the theory of financial markets since formalized by Fama [10] and can be summarized as a set of assumptions on the predictability of future market prices. The hypothesis gets its name from the definition of efficient markets; prices fully reflects all available information.

2.2.1 Weak Form

The weak form of the EMH states that current prices fully reflect the information to be gained from historical prices. This implies that it is impossible to consistently outperform the market using investment strategies that rely on past prices, such as attempted in this thesis. The claim has been widely discussed and has gone from being generally accepted in earlier works [10] to being generally dismissed and even disproven in certain settings [7],[19].

2.2.2 Semi-strong Form

The EMH in its' semi-strong form loosens the restriction of relying on past prices and includes all publicly available information. This form of the hypothesis therefore implies that one can not consistently outperform the market using any investment strategy based on public information.

2.2.3 Strong Form

In this most strict of the forms of EMH, the available information is generalized to include all information including insider information not known to the general public. The semi strong form as well as the strong form imply the weak form and make even further assumptions. Since even the weak form has been the target of some debate, the two stronger forms of the EMH are generally accepted as unfeasible, even by earlier work [10].

2.3 Sharpe Ratio

The Sharpe ratio is a metric designed to measure the performance of a set of investment, adjusted for risk. The ratio was originally designed by William Sharpe in 1965 [27] and later revised to its current definition [28].

$$\text{Sharpe} = \frac{E[R]}{\sqrt{\text{Var}[R]}}, \tag{2.2}$$

where R denotes the return of the assets and the risk free return is assumed to be zero. The variance of the returns of the set of investments is used as a metric for the financial risk of the set.

Chapter 3

Neural Networks

This chapter introduces artificial neural networks in a frequentist setting, describing the relevant concepts for introducing Bayesian neural networks. We present relevant regularization techniques, network architectures and activation functions.

Neural networks have in recent times gained significant attention, due to their advancements in fields such as image recognition, natural language processing and reinforcement learning. With developments in optimization, learning algorithms and hardware, neural networks continue to gain momentum. However, there are a number of problems that have not been properly addressed. Despite their performance on some data sets, neural networks tend to easily overfit to training data and provide poor generalization. Finally, they require tuning of a large amount of model hyperparameters.

A neural network is a parametric model that aims to approximate the mapping $f : \mathcal{X} \to \mathcal{Y}$, given a data set $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\} \subset (\mathcal{X}, \mathcal{Y})$. The set of parameters of the neural network is called weights $\mathbf{w}$, and the problem of finding the set of weights that best approximate the mapping f is solved via Maximum Likelihood Estimation (MLE).

$$\mathbf{w}^{\mathbf{MLE}} = \arg\max_{\mathbf{w}} \left(\log p(\mathcal{D}|\mathbf{w})\right). \tag{3.1}$$

3.1 Feed-forward Neural Networks

A feed-forward network is a generalization of the linear perceptron, where layers of artificial neurons are stacked together, with non-linearities applied

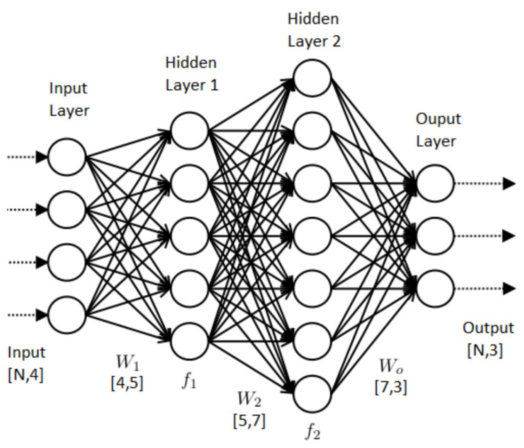


Figure 3.1: Feed-forward neural network with 2 hidden layers

between each layer. The reason behind the various successful applications of neural networks is due to the universal approximation theorem [17]. It proves that feed-forward neural networks with non-linear activation functions and at least one hidden layer are universal function approximators.

Given an input vector $\mathbf{x} \in \mathbb{R}^p$ and an output vector $\mathbf{y} \in \mathcal{Y}$, a feed-forward neural network with m hidden layers and h_i nodes in each layer and weights $\mathbf{w}$, bias $\mathbf{b}$ and non-linear activation function ϕ, produces a prediction by:

$$\begin{aligned}
\mathbf{h}^{(1)} &= \phi(\mathbf{w}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}),\\
\mathbf{h}^{(2)} &= \phi(\mathbf{w}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}),\\
&\vdots\\
\mathbf{y}^* &= g(\mathbf{w}^{(m)}\mathbf{h}^{(m-1)} + \mathbf{b}^{(m)}),
\end{aligned}$$

where $\mathbf{w}^{(i)} \in \mathbb{R}^{h_i \times h_{i-1}}$ and $\mathbf{b}^{(i)} \in \mathbb{R}^{h_i}$. The function g can be linear, which often is the case for regression problems, or softmax in the case of classification. However, many other choices for g exist.

3.2 Activation Functions

To enable neural networks to approximate complex non-linear transformations we must introduce some form of non-linearities to the model. There exist a myriad of possible choices for the activation function, but for this thesis we will consider two different functions, the first being the hyperbolic tangent:

$$\phi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-1}}, \tag{3.2}$$

which maps x to the interval $(-1, 1)$.

A common drawback of this activation function is the problem of vanishing gradients, where gradients converge to 0, and the network struggles to learn from the data. Since the derivative of the hyperbolic tangent quickly drops toward zero for inputs away from zero, we are at risk of vanishing gradients.

To overcome these problems, Rectified Linear Units (ReLU) were introduced

$$\phi(x) = \mathrm{ReLU}(x) = \max(0, x). \tag{3.3}$$

This has become a widely used approach in recent machine learning research, mostly due to its strong performance on image classification tasks, which has been at the center of focus in deep learning.

3.3 Loss Functions

To learn from the data using a neural network we must define what to learn. We are concerned with finding the set of weights, $\mathbf{w}$, that is able to fit to the training data, while also providing generalization to new data. Thus, we need to define a loss function that determines how well our current model actually fits the data.

3.3.1 Mean Squared Error

Since we are modelling a continuous response variable (the next day's return), we are faced with a regression problem. Thus, a natural loss metric is the mean squared error (MSE)

$$\mathrm{MSE} = \frac{1}{N}\sum_{i}^{N}(\mathbf{y}_i - \mathbf{y}_i^*)^2, \tag{3.4}$$

where $\mathbf{y}_i$ is the actual outcome and $\mathbf{y}_i^*$ is the model's prediction. Using MSE as loss function is equivalent to maximum likelihood estimation using a Gaussian likelihood for the data given the model's parameters.

3.4 Regularization

Due to the high complexity in neural networks, they have a tendency to overfit to the given training data, which yields poor generalization to unseen data points. To overcome this problem, several regularization techniques have been proposed. From a Bayesian viewpoint, this is often equal to introducing an informative prior on the weights, and finding the optimal set of weights via Maximum a posteriori estimation (MAP) [3]. More formally, given a data set $\mathcal{D}$ and a prior distribution of the weights, $p(\mathbf{w})$, the optimal set of weights is estimated by

$$\mathbf{w}^{\mathbf{MAP}} = \arg\max_{\mathbf{w}} \left(\log p(\mathcal{D}|\mathbf{w}) + \log p(\mathbf{w})\right). \tag{3.5}$$

3.4.1 L^1-regularization

L^1- regularization, also often called lasso regularization, penalizes large weights in terms of absolute value (i.e. L^1-norm), with a regularization parameter $\lambda \in \mathbb{R}^+$.

$$\mathcal{L}(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}) + \lambda \sum_{i}^{m} \sum_{j}^{h_i} |w_{i,j}|, \tag{3.6}$$

where $\mathcal{L}(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w})$ denotes the network's loss function of a new prediction, given data and trained weights. Large values of λ yields heavy regularization, while small values result in no regularization. L^1-regularization tends to force parameters to equal zero, creating model sparsity. From a probabilistic viewpoint, L^1-regularization equals introducing a Laplace-prior on the distribution of the weights in the network.

3.4.2 L^2-regularization

Also known as Ridge regression, L^2-regularization penalizes large weights by adding an L^2-penalty to the model's cost function according to:

$$\mathcal{L}(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}) + \lambda \sum_{i}^{m} \sum_{j}^{h_i} w_{i,j}^2. \tag{3.7}$$

From a probabilistic perspective, L^2-regularization equals introducing a Gaussian prior on the weights of the network.

3.4.3 Dropout

A different approach to regularization is to introduce noise to the model. The idea is that by introducing noise while training, the model will struggle to overfit and instead yield better generalization to new data. This idea has been introduced in a variety of approaches, e.g. adding Gaussian noise to the weights or adding noise to the training data. However, the most prominent approach is dropout, introduced by Srivastava et. al [29].

While training, dropout regularizes the model by randomly dropping activations in the network, with a given probability p as a hyperparameter. In the testing phase, no activations are dropped, but all activations in the network are scaled by p. This creates an ensemble of sparse networks, which can significantly increase the models ability to generalize [29].

Chapter 4

Bayesian Inference

Bayesian inference, in contrast to its frequentist counterpart, allows us to express our prior beliefs about the distribution of the model's parameters, and to model the entire distribution of possible parameter settings, freeing us from the point estimates of frequentist inference. This chapter introduces the techniques of Bayesian modelling, that will be used in Bayesian neural networks.

4.1 Introduction to Bayesian Inference

Bayesian inference is concerned with inferring an underlying model, given a set of data $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$, that is likely to have generated the data. The process of Bayesian inference starts with a *prior* belief, $p(\theta)$, about the distribution of the parameters $\theta \in \Theta$ of the model that has generated the data. When we observe new data $\mathcal{D}$, we update our beliefs regarding the model's parameters accordingly. This is done by specifying a *likelihood* distribution $p(\mathcal{D}|\theta)$, that links the parameters to the observed data. We then combine our prior beliefs with the likelihood distribution of observed data into a *posterior* distribution using Bayes' theorem:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{4.1}$$

If we assume independence between each observation in the data set, (4.1) can be re-written as:

$$p(\theta|\mathcal{D}) = \frac{p(\theta)\prod_i p(\mathbf{y}_i|\mathbf{x}_i,\theta)}{p(\mathcal{D})}. \tag{4.2}$$

Using the posterior distribution of model parameters given observed data, we can predict a new data point $\mathbf{y}^*$, given input $\mathbf{x}^*$, using the posterior predictive distribution [11]:

$$\begin{aligned} p(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D}) &= \mathbb{E}_{p(\theta|\mathcal{D})}\left[p(\mathbf{y}^*|\mathbf{x}^*,\theta)\right] \\ &= \int_\Theta p(\mathbf{y}^*|\mathbf{x}^*,\theta)P(\theta|\mathcal{D})d\theta. \end{aligned} \tag{4.3}$$

A key component of the posterior distribution is the denominator in (4.1). This is in fact not a probability distribution, but a normalizing scalar. Known as the *model evidence*, it is the marginalization of the likelihood over the parameters θ:

$$p(\mathcal{D}) = \int_\Theta p(\mathcal{D}|\theta)p(\theta)d\theta, \tag{4.4}$$

which explains the alternative name *marginal likelihood*. This integration quickly becomes intractable for all but the simplest Bayesian models, where the likelihood and the prior are *conjugate* distributions. For more complex models, we are left with approximate methods for estimating the posterior distribution of our model.

4.2 Uncertainty

In Bayesian inference, the posterior distribution embodies our uncertainty as to what model generated the data. Staring from a prior distribution over the model's parameters, we update our uncertainty when we observe data, and arrive at a posterior distribution. Given a new observation, $\mathbf{x}^*$, the posterior predictive distribution $p(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D})$ represents our belief regarding the label $\mathbf{y}^*$. In particular, the variance of this distribution can be used to quantify our uncertainty. We can use the law of total variance to decompose this into two terms [34]:

$$\mathrm{Var}(p(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D})) = \mathrm{Var}(\mathbb{E}[p(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D},\theta)]) + \mathbb{E}\left[\mathrm{Var}(p(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D},\theta))\right] \tag{4.5}$$

The first term represents our uncertainty regarding the model parameters θ, known as *model uncertainty*, or *epistemic uncertainty*. The second term reflects the *inherent noise*, known as *aleatoric uncertainty*. This can either be constant over the entire data set (homoscedastic noise), or vary with each observation (heteroscedastic noise). In contrast to epistemic uncertainty, which can be modelled away with infinite training data, aleatoroic uncertainty is an inherent feature of the data.

It is assumed that the training and test data have been generated from the same underlying distribution. However, this is not always the case. Financial time series provide an example of data when this assumption can be problematic, since correlations between markets are a function of time. A desirable quality of Bayesian models would be to increase uncertainty for observations that differ widely from their training data. Zhu et al. [34], modelled this connection explicitly by using an autoencoding on the training data, to measure the distance of a new observation to the training data, in the latent embedding space.

4.3 Markov Chain Monte Carlo

Due to the difficulty of analytically evaluating the posterior distribution $p(\theta|\mathcal{D})$ of our model, we have arrived at approximate methods for Bayesian inference. Markov Chain Monte Carlo (MCMC) is a powerful technique for approximating probability distributions, sprung from computational problems in statistical physics [24]. The main idea of MCMC is to sample from an ergodic Markov chain, that has the true posterior distribution as its stationary distribution. Thus, *eventually* the sampling will converge to the true distribution. These algorithms can be viewed as a non-parametric approach to Bayesian posterior inference, since we do not impose a restriction on the estimated posterior, but instead sample directly from the true posterior.

Broadly speaking, we are starting somewhere in the distribution, and jump to a new location if the posterior probability is higher there, otherwise, we try a new jump. To sample from the posterior, we want to move to the areas of the posterior where the density is high. Since we do not want to get stuck at the maximum of the density peaks of the distribution, we also accept some jumps to locations with lower posterior density.

To perform MCMC-sampling, there exist a number of different algorithms. The key difference among them is how they propose a new location to jump to, and how they select whether to accept a jump or not.

4.3.1 The Metropolis Algorithm

The Metroplis algorithm [24] is the original approach for MCMC sampling, and together with its extensions, is still one of the most widely used algorithms for MCMC. The Metropolis algorithm is a random walk that uses a pre-defined proposal distribution to generate a new proposal step. This step is accepted with a probability determined by the density of the target distribution at the proposal step and starting state. More formally, the Metropolis algorithm proposes a new step according to [25]:

1. Given an initial state $\theta^{(t)}$, a target distribution Q and a proposal distribution q.
2. Generate proposal state θ^* from the distribution $q(\theta^*|\theta^{(t)})$.
3. Accept θ^* with probability $\min(1, \frac{Q(\theta^*)}{Q(\theta^{(t)})})$.
4. If θ^* is accepted, set $\theta^{(t+1)} = \theta^*$. Otherwise $\theta^{(t+1)} = \theta^{(t)}$.

The original algorithm only allows for symmetrical proposal distributions, i.e. where $q(\theta^*|\theta^{(t)}) = q(\theta^{(t)}|\theta^*)$. This was extended by Hastings in 1970 to also allow for asymmetrical proposal distributions [14].

4.3.2 Hamiltonian Monte Carlo

The main drawback of the Metropolis algorithm is the random walk behaviour, which creates a correlated chain of states. This requires a larger number of samples for the chain to explore the full distribution. To remedy this, one can instead use more informed proposals for the next states in the chain. In this regard, the Hamiltonian Monte Carlo algorithm (HMC), or the hybrid Monte Carlo, is an extension of the Metropolis algorithm [9]. The HMC algorithm uses the gradients of the target distribution with respect to the parameters to propose new states. This enables proposal steps further from the initial steps, which de-correlates the chain and leads to faster convergence in high dimensions.

The HMC algorithm is based on the concept of Hamiltonian dynamics from classical physics. Hamiltonian dynamics can be visualized by a frictionless particle sliding over a surface of varying height [26]. This system is characterised by the position, q, and the momentum, p, of the particle. These, in turn, determine the potential energy, $U(q)$, and kinetic energy, $K(p)$, of the system.

With the intention of using the Hamiltonian dynamics to sample from a complex posterior distribution, the momentum variable p will be introduced artificially. We will let the potential energy $U(q)$ equal the negative logarithm of the target posterior distribution (up to a normalizing constant) [25]:

$$U(q) = -\log(\pi(q)\pi(\mathcal{D}|q)). \tag{4.6}$$

To avoid confusion with the position p, in this section we let $\pi(\cdot)$ denote density. The form of $K(P)$ is commonly assumed to be the standard form for kinetic energy:

$$K(p) = \sum_{i=1}^{d} \frac{p_i^2}{2m_i}, \tag{4.7}$$

where the m_i represent the mass of each respective component. However, the algorithm is not restricted to any particular choice of K [26]. This choice of K corresponds to a zero-mean Gaussian distribution for p, which can be seen through the canonical distribution from K, i.e. $\pi(p) \propto e^{-K(p)}$.

The Hamiltonian, $H(q,p) = U(q) + K(p)$, represents the total energy of the system. From this we can define a joint canonical distribution:

$$\pi(p,q) = \frac{1}{Z} e^{-H(p,q)}, \tag{4.8}$$

where Z is a normalizing constant. With p and q as d-dimensional vectors, the dynamics of the system are described by Hamilton's equations:

$$\begin{aligned} \frac{dq_i}{dt} &= \frac{\partial H}{\partial p_i}, \\ \frac{dp_i}{dt} &= -\frac{\partial H}{\partial q_i}, \end{aligned} \tag{4.9}$$

for $i = 1, ..., d$ [26]. There are three properties of these dynamics that are crucial to its use in Markov chain sampling. First of all, the dynamics are reversible, i.e. if we follow the dynamics backwards in time we will end up in our starting point. Second, the Hamiltonian dynamics preserve volume – if we let a number of points X, which have volume V, evolve through Hamiltonian dynamics to points X', then X' will have volume V. Finally, the Hamiltonian, H, of the system is constant as p and q evolve according to these dynamics.

In practice we approximate the Hamiltonian dynamics by discretizing time with some step size, ϵ. This is commonly done using the *leapfrog* integrator:

$$\begin{aligned} p_i\left(t+\frac{\epsilon}{2}\right) &= p_i(t) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t)), \\ q_i(t+\epsilon) &= q_i(t) + \epsilon\frac{p_i(t+\frac{\epsilon}{2})}{m_i}, \\ p_i(t+\epsilon) &= p_i\left(t+\frac{\epsilon}{2}\right) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t+\epsilon)), \end{aligned} \tag{4.10}$$

for $i = 1, ..., d$. To evolve the dynamics of the system by a time interval τ, we simply make $L = \tau/\epsilon$ updates via the leapfrog algorithm. Since this is just an approximation, it will introduce a systematic bias to the updates.

After L iterations of the leapfrog algorithm, we have generated a new proposal state, (q^*, p^*), for the HMC algorithm. This state is now accepted with probability:

$$\min(1, \exp(H(q,p) - H(q^*, p^*)). \tag{4.11}$$

This is the Metropolis step of the HMC algorithm. If the leapfrog algorithm were to approximate the dynamics perfectly, H would remain unchanged, and all proposals would be accepted. However, since the leapfrog algorithm introduces error to the dynamics, the Metropolis step ensures that the systematic bias is removed [2].

In our exploration of the target distribution, we are restricted to a certain total energy level, dictated by the Hamiltonian dynamics. To explore the full distribution, we can introduce randomness by sampling from the conditional distribution of p on q, which due to independence is the marginal distribution of p. This enables us to evolve the dynamics at different energy levels.

To summarize, one full iteration of the HMC algorithm is done via:

1. Sample $p_i \sim N(0, m_i)$.

2. Starting from (q, p) perform L iterations of the leapfrog integrator with step size ϵ, resulting in a proposal (q^*, p^*).

3. Accept (q^*, p^*) with probability $\min(1, \exp(H(q, p) - H(q^*, p^*))$. Otherwise remain at (q, p).

4.4 Variational Inference

An alternative approach to approximate inference in Bayesian modeling is known as variational inference. This can be viewed as a parametric alternative to the class of MCMC-sampling algorithms. The power of these methods is that they replace the integration part of inference with optimization, i.e. differentiation, which in general is computationally less expensive. The main downside of these algorithms, aside from the fact that the variational posterior might be very different from the true posterior, is that they assume independence between the model's parameters.

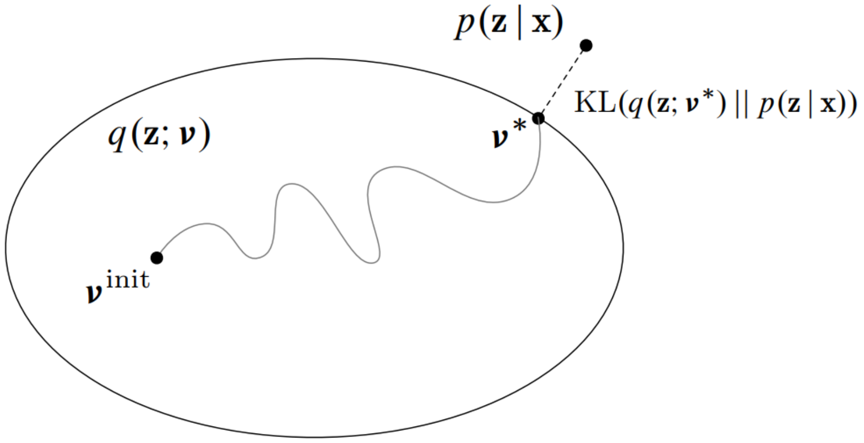

$p(\mathbf{z} \mid \mathbf{x})$
$\mathrm{KL}(q(\mathbf{z}; \boldsymbol{\nu}^*) \,||\, p(\mathbf{z} \mid \mathbf{x}))$
$q(\mathbf{z}; \boldsymbol{\nu})$
$\boldsymbol{\nu}^*$
$\boldsymbol{\nu}^{\text{init}}$

Figure 4.1: Visualization of variational inference

The main idea of variational inference is to use a known distribution, the *variational posterior*, and fit this distribution to the true posterior of our model, using some measure of similarity between probability distributions.

This measure is usually set to be the Kullback-Leibler divergence [22]:

$$\mathrm{KL}(q_\omega(\theta) \,||\, p(\theta|\mathcal{D})) = \int_\Theta q_\omega(\theta) \log \frac{q_\omega(\theta)}{p(\theta|\mathcal{D})} d\theta. \tag{4.12}$$

Note that the KL-divergence is *assymetric*, i.e. $\mathrm{KL}\,(p \,||\, q) \neq \mathrm{KL}\,(q \,||\, p)$. Minimizing the Kullback-Leibler divergence with respect to ω is equivalent to maximizing the Evidence Lower Bound (ELBO) [11]:

$$\mathcal{L}_{VI}(\omega) = \int_\Theta q_\omega(\theta) \log p(\mathcal{D}|\theta) d\theta - \mathrm{KL}(q_\omega(\theta) \,||\, p(\theta)), \tag{4.13}$$

which can be proven as a lower bound for the model evidence $p(\mathcal{D})$, by a use of Jensen's equality. The ELBO consists of two parts - the fist term, the *expected log-likelihood*, represents the data dependent part and second term, the *prior KL* represents the prior dependent part. Maximizing the expected log-likelihood equals finding a variational distribution that explains the data well. Minimizing the prior KL acts as regularization and forces the variational distribution to be as close to the prior distribution as possible. Thus, the ELBO represents a trade-off between model complexity and model expressiveness, in other words - a bias-variance trade-off.

Chapter 5

Bayesian Neural Networks

Neural networks are powerful universal function approximators, able to capture complex non-linear mappings by applying non-linearities to weighed sums of layerwise activations. In a Bayesian setting we introduce prior distributions over the weights of the network and combine this with a likelihood distribution of the model's predictions into a posterior distribution of the model's weights given training data. Due to the non-linear activation functions in the network, the conjugacy of prior and posterior is lost. Thus, we are left with approximate techniques for posterior inference. This chapter applies the approximate inference techniques introduced in chapter 4 to neural networks.

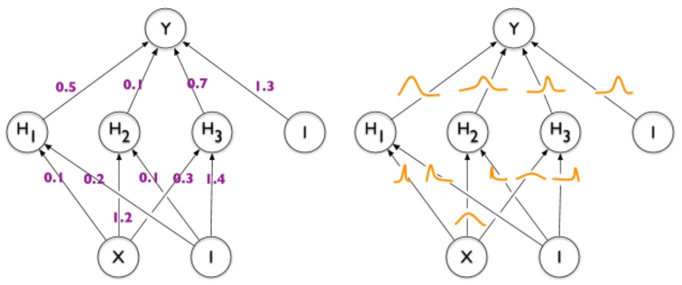


Figure 5.1: Left: Standard NN with point estimates for weights. Right: Bayesian NN with probability distributions over weights [3].

5.1 Hamiltonian Monte Carlo

Neal [25] presented in his Ph.D. thesis the first application of Markov chain Monte Carlo sampling applied to posterior inference in Bayesian neural networks. He used the Hamiltonian Monte Carlo algorithm of Duane et al. [9] to deal with the high dimensionality of the posterior distribution. This has since then served as the gold standard of MCMC based BNNs. There is a famous extension by Hoffman et al. [16], which eliminates the need to set the number of leapfrog steps L. However, we will consider the standard HMC algorithm.

While HMC applied to BNNs does produce strong results compared to other approaches for approximate Bayesian inference, there are reasons why this has not become a widespread approach. Markov chain Monte Carlo, in any form, is computationally expensive compared to the maximum likelihood estimates of backpropagation. These methods scale poorly with both model complexity and amount of data. Furthermore, the methods for MCMC sampling require difficult tuning of a number of hyperparamters, which goes against some of the the reasons for using a Bayesian approach in the first place.

5.2 Variational Inference

Applying variational inference, as explained in chapter 4, to neural networks can be summarized as setting a prior distribution on the weights of the networks' layers, choosing a variational posterior distribution and then fitting this variational posterior as close to the true posterior distribution as possible using some optimization algorithm and the training data. In practice, this means optimizing the parameters of the variational posterior to maximize the ELBO defined in (4.13). Because neural networks are large systems, it is essential to choose an optimization procedure so that updating the parameters of the model is computationally inexpensive. The gradient descent algorithm using the backpropagation procedure is widely used for training neural networks for this reason specifically.

In order to apply the backpropagation algorithm efficiently in a Bayesian setting, it is necessary to efficiently compute the gradients resulting from differentiating the ELBO with regards to the networks parameters. Comparing the ELBO to a standard frequentist loss function such as the mean squared error, an immediate difference in computational complexity is the presence of an integral. The integral occurs in the first term of the ELBO, over the likelihood and variational posterior in the first term of the expression. This intergral does not necessarily have any analytically solutions and will have to be estimated numerically, i.e. using Monte Carlo simulation. This introduces a step of unwanted complexity in calculating the necessary gradients, which has to be addressed for even a small Bayesian neural network to be trained to a feasible computational cost. Recent literature in the field is in large part concerned with different ways of solving this problem; Bayes by Backprop [3], the local reparametrization trick [21] and Flipout [32] being the most famous methods. This thesis applies the Bayes by Backprop methodology.

5.2.1 Bayes by Backprop

To train the network given our prior distribution, variational posterior and training data, we define the loss function as the negative ELBO:

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) &= \mathrm{KL}(q_\theta(\mathbf{w}) \,||\, p(\mathbf{w})) - \mathbb{E}_{q_\theta(\mathbf{w})}\left[\log(\mathcal{D}|\mathbf{w})\right] \\ &= \int_\Theta q_\theta(\mathbf{w}) \log \frac{q_\theta(\mathbf{w})}{p(\mathbf{w})} d\theta - \int_\Theta q_\theta(\mathbf{w}) \log p(\mathcal{D}|\mathbf{w}) d\theta.\end{aligned} \tag{5.1}$$

With the goal of minimizing $\mathcal{F}$ using gradient descent, we face the problem of finding the gradients of the two expectations in 5.1. This cannot be done explicitly, but must be done by Monte Carlo sampling. Blundell [3] uses a generalization of the local reparameterization trick [21] to allow for unbiased estimates of the gradients. We can use the fact that under certain conditions, the derivative of an expectation is the expectation of the derivative. For a random variable ϵ with distribution $q(\epsilon)$, a function $f(\mathbf{w},\theta)$, and a $\mathbf{w} = t(\theta,\epsilon)$ with marginal distribution such that $q_\theta(\mathbf{w})d\mathbf{w} = q(\epsilon)d\epsilon$, the following holds:

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q_\theta(\mathbf{w})}\left[f(\mathbf{w},\theta)\right] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial\theta}\right]. \tag{5.2}$$

By using $f(\mathbf{w}_i,\theta) = \log q_\theta(\mathbf{w}_i) - \log p(\mathbf{w}_i)p(\mathcal{D}|\mathbf{w}_i)$, where $\mathbf{w}_i$ denotes the ith Monte Carlo sample from the variational posterior $q_\theta(\mathbf{w})$, we can approximate the gradients of the loss function 5.1 via Monte Carlo sampling. The number of samples, M, required for this is typically low, often with sufficient results for $M = 1$. For a Gaussian variational posterior the minimization process aims at finding the mean, μ, and variance, ρ^2, of the variational posterior, assuming a diagonal covariance matrix. These variables are initialized to set the variational posterior to a standard normal distribution, i.e μ as 0 and ρ as 1. The optimization procedure of Blundell [3], can then be generalized to allow for an arbitrary M as:

1. Sample $\epsilon_i \sim N(0, I)$, for $i = 1, ..., M$.
2. Set $\mathbf{w}_i = \mu + \log(1 + \exp(\rho)) \circ \epsilon_i$.
3. Set $\theta = (\mu, \rho)$.
4. Let $f(\mathbf{w}_i, \theta) = \log q_\theta(\mathbf{w}_i) - \log p(\mathbf{w}_i)p(\mathcal{D}|\mathbf{w}_i)$.
5. Find the gradient with respect to μ as:
$$\frac{\partial f(\mathbf{w}_i, \theta)}{\partial \mu} = \frac{\partial f(\mathbf{w}_i, \theta)}{\partial \mathbf{w}_i} + \frac{\partial f(\mathbf{w}_i, \theta)}{\partial \mu}.$$
6. Find the gradient with respect to ρ as:
$$\frac{\partial f(\mathbf{w}_i, \theta)}{\partial \rho} = \frac{\partial f(\mathbf{w}_i, \theta)}{\partial \mathbf{w}_i} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}_i, \theta)}{\partial \rho}.$$
7. Update the variational parameters using the estimated expectation of the gradients:
$$\mu \leftarrow \mu - \alpha \frac{1}{M} \sum_{i=1}^{M} \frac{\partial f(\mathbf{w}_i, \theta)}{\partial \mu},$$
$$\rho \leftarrow \rho - \alpha \frac{1}{M} \sum_{i=1}^{M} \frac{\partial f(\mathbf{w}_i, \theta)}{\partial \rho}.$$

5.2.2 Covariance estimation with high dimensionality

Because the weights of a neural network can be very numerous, the problem of fitting the variational posterior will often be of a very high dimensionality. Since the entries of a covariance matrix scales with quadratic complexity it is generally unfeasible to incorporate covariance between weights in the prior or variational posterior. This makes it necessary to set the variational posterior distribution under the assumption that the weights of the network are independent of each other. The veracity of this assumption is dependent on the specific model and data, and therefore can not be known without knowing the true posterior. Because variational inference is used to approximate an unobservable true posterior distribution, the assumption of independence can't be evaluated and has the potential to generate a faulty model. This could be considered the largest downside of using variational inference for training Bayesian neural networks as it is an error source with potential for large impact, that can not be observed or evaluated.

5.3 Dropout

By using variational inference or Markov chain Monte Carlo for approximate inference, one must drastically alter existing neural network models to allow for Bayesian modelling. An alternative to this was proposed by Gal [12], which is based on extending stochastic regularization techniques to allow for approximate inference. By using dropout during training *and* during test, we can approximate a posterior predictive distribution by repeatedly sampling from the weight distribution. This methods only results in a posterior predictive distribution, not a posterior distribution over the weights in the network. Gal [12] proves that this procedure approximates Bayesian inference in deep Gaussian processes. The choice of activation function governs the covariance function of the Gaussian process that is approximated.

Chapter 6

Methodology

This chapter introduces the methodology that was used to apply Bayesian neural networks to the problem of predicting financial asset returns, and to utilize predictive uncertainty in systematic trading strategies. For this purpose, we first introduce the data that was used, together with the various techniques for pre-processing that were necessary. Using this processed data, we then create a set of features for each market. After this, we introduce the different types of models that were used, with respect to loss functions and network architecture. These models are then extended by setting a prior distribution over the weights, and a likelihood distribution over the data. The resulting posterior distribution is inferred using different approximation methods. Finally, these models and their inherent uncertainty are used to create and evaluate systematic trading strategies, on a hold out test set.

6.1 Data

This section introduces the data that was used to train and test all models. This includes pre-processing of data such as scaling and normalization of rolling volatility, together with the features created from the data to aid the learning process. Finally, we present the methods for dimensionality reduction used in the data processing.

We are using a data set consisting of daily opening (O_t^*) and closing (C_t^*) prices, as well as daily high (H_t^*) and low (L_t^*) prices for 78 different futures markets, with data ranging from 2001-06-01 to 2018-12-31. The futures markets are of different asset classes, including stock indices, commodities,

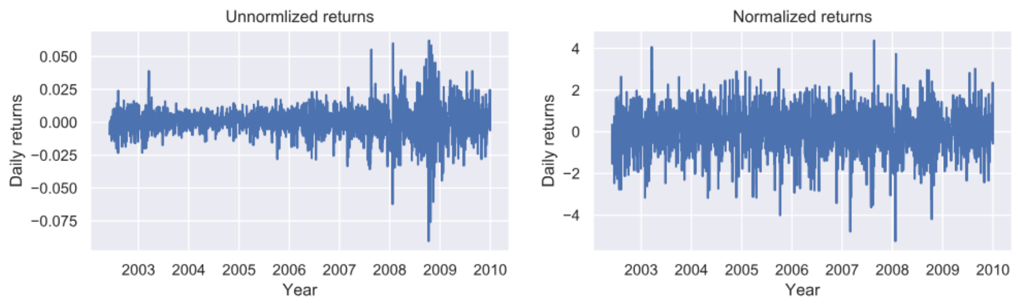


Figure 6.1: Left: Unnormalized series of daily returns exhibiting clear volatility clustering. Right: Same return series, normalized with a rolling volatility estimator.

interest rates, and currency pairs.

Since financial time series data tends to exhibit volatility clustering, i.e. distinct periods of high variance and distinct periods of low variance, a common technique is to normalize the daily returns by a rolling estimate of volatility. Thereafter we can create a normalized series of prices from the cumulative sum over normalized returns. Let O_t, C_t, H_t and L_t denote the normalized open, close, high and low prices respectively.

6.1.1 Training, test and validation

The data is split into a training set, a validation set and a final test set. We train the models on the training set and test their performance on the validation set during the hyperparameter optimization. Finally, the models with the optimal hyperparameter configurations are trained on both the training set and validation set, and thereafter evaluated on the test set.

To train, validate and test models for time series prediction, we must keep the temporal structure of the data to avoid any forward-looking bias. The training set must therefore be chronologically before the test set. The data is split as follows:

	Time period
Training	2001-06-01 — 2009-01-01
Validation	2009-01-01 — 2014-01-01
Test	2014-01-01 — 2018-12-31

Table 6.1: Training, validation and test sets

6.1.2 Filters

We filter the data by applying a weighting of observations across time. The filters that will be used is the simple moving average (SMA), which weights observations equally across a rolling time window, and the exponential moving average (EMA) which decays the weights on each observations exponentially in time.

Simple Moving Average

The simple moving average applies an equal weight to all predictions within the rolling window, and is thus the average of the m last days of the time series $\{X_t\}$ [4]:

$$\mathrm{SMA}(\{X_t\}, m) = \frac{1}{m}\sum_{i=0}^{m-1} X_{t-i}. \tag{6.1}$$

This can be defined recursively as:

$$\mathrm{SMA}(\{X_t\}, m) = \mathrm{SMA}(\{X_{t-1}\}, m) + \frac{1}{m}(X_t - X_{t-m}). \tag{6.2}$$

Exponential Moving Average

The simple moving average weights all observations equally, even though we often consider recent observations to be more relevant. In contrast, the exponential moving average decays the weights of older observations. This can be defined recursively as [4]:

$$\mathrm{EMA}(\{X_t\}, \alpha) = \begin{cases} X_0, \text{ if } t = 0, \\ \alpha X_t + (1-\alpha)\mathrm{EMA}(\{X_{t-1}\}, \alpha), \text{ if } t > 0, \end{cases} \tag{6.3}$$

for any $\alpha \in [0, 1]$. This parameter is often set to $\alpha = \frac{1}{1+m}$, which lets m represent the filters center of mass.

6.1.3 Volatility Estimators

To normalize asset returns, in an attempt to remove volatility clusters and create a stationary time series, we need a rolling estimate of time dependent volatility. To avoid a forward-looking bias in evaluation, we normalize returns using the *previous* day's volatility. This has the added benefit of not suppressing large movements. There exist a number of methods for this, ranging from simpler ones which are only based on closing prices, to more advanced ones which include open, close, high and low prices in its calculations.

6.1.4 Features

The power of neural networks resides in their ability to create a hierarchy of features from the data. With large amounts of training data at hand, we could utilize this power by using deep networks. However, for this problem we are limited to a relatively small set of training examples. Therefore we can aid the model's learning by feeding it a set of handcrafted features. This is not in the spirit of current trends in deep learning research, which would be to use as raw data as possible, but results from [30] suggest that using handcrafted features aids learning for this particular domain.

Momentum

Momentum is defined as the m day lagged difference in closing prices:

$$Momentum(\{C_t\}, m) = C_t - C_{t-m}. \tag{6.4}$$

This can be interpreted as an indication of trend in the time series, where positive values indicate an upwards trend in prices. As for the parameter m, we use values 50, 100 and 250.

Commodity Channel Index

The Commodity Channel Index (CCI) is defined as:

$$\begin{aligned} X_t &= \frac{C_t + H_t + L_t}{3}, \\ CCI(m) &= \frac{X_t - SMA(\{X_t\}, m)}{SMA(\{|X_t - SMA(\{X_t\}, m)|\}, m)}. \end{aligned} \tag{6.5}$$

For the parameter m, we use 50, 100 and 250 days.

Exponential Moving Average

The Exponential Moving Average feature (EMAF) is defined here as the difference of two exponential moving averages over normalized closing prices, with different lag. Thus, it is defined as:

$$EMAF(\{C_t\}, m, n) = EMA(\{C_t\}, m) - EMA(\{C_t\}, n), \tag{6.6}$$

where $m < n$. The values for (m, n) are set as $(30, 100), (50, 200)$ and $(50, 250)$.

Relative Strength Index

The Relative Strength Index (RSI) is defined using the ratio of upward movements to downward movements, using simple moving averages:

$$\begin{aligned} \Delta C_t &= C_t - C_{t-1}, \\ U_t &= \Delta C_t \mathbb{1}_{\Delta C_t \geq 0}, \\ D_t &= \Delta C_t \mathbb{1}_{\Delta C_t < 0}, \end{aligned} \tag{6.7}$$

where $\mathbb{1}_{\Delta C_t \geq 0}$ equals 1 if the return is non-negative, and 0 otherwise. Using this, the RSI is defined as:

$$RSI(m) = 1 - \frac{1}{1 + \frac{SMA(\{U_t\}, m)}{SMA(\{D_t\}, m)}}. \tag{6.8}$$

As for the value of m, we again use 50, 100 and 250 days.

6.1.5 Dimensionality Reduction

Since we are modelling a multivariate regression problem, making predictions for all markets in parallel, we have created a high-dimensional input for our model by creating a variety of trend features for each market. With 78 markets and 15 features per market, we have a 1170-dimensional input. To make predictions for all markets separately would be undesirable, since we want to exploit the fact that the markets are correlated and may carry information with predictive power across different markets. With only 1958 observations in our training set, the model will have a difficult time learning in such a high-dimensional setting. However, we might exploit the fact that the features across different markets are correlated. Thus, we can turn to techniques for dimensionality reduction.

Principal Components Analysis

Principal component analysis (PCA) is an unsupervised dimensionality reduction technique. The idea of PCA is to reduce the number of m features into a number k of orthogonal principal components, while retaining as much of the variance in the data as possible. This is done by projecting the data set onto the subspace spanned by the eigenvectors of the covariance matrix of the data corresponding to the k largest eigenvalues.

To avoid any forward-looking bias when we are evaluating the different models on our hold out test set, we cannot apply PCA reduction to the entire data set. Instead, we first find the k principal components on the *training* set, and project the entire data set to those principal components.

6.2 Bayesian Neural Networks

This section introduces the methodology of applying the different approaches of Bayesian inference presented in chapter 5 to the problem of predicting future financial returns. These models have some traits in common, but each model requires some specific choices regarding training and prediction.

6.2.1 Network Architecture

This thesis is concerned with feed-forward neural networks. The problem we are facing has a temporal structure, why recurrent networks would be a natural choice. However, by creating a set of features that take different time periods into account, we can use feed-forward networks.

Hidden Layers

The number of hidden layers, together with the number of nodes in each respective layer, govern the complexity of the functions that the network is able to approximate. With more layers and more nodes the network can find increasingly complex mappings. However, this comes at the cost of increased variance.

In this thesis, we are dealing with incredibly noisy data and a relatively small data set. Thus we are unable to use deep models, since these would easily overfit to our training data. Results from Widegren [33] suggest that there is no notable increase in prediction accuracy with deeper networks for predicting returns of futures contracts. There is also an increased computational cost of deep or wide networks, which for some of the methods used in this thesis might become intractable. Markov chain Monte Carlo scales poorly with network size and data, and we therefore limit the number of hidden layers in each model to one.

Activation Function

ReLU only saturates in one direction, and therefore is not as sensitive to vanishing gradients. However, a network with ReLU activations might learn a bias, despite having excluded bias terms in the network. For this thesis, hyperbolic tangent was used for the dropout and variational inference models, while ReLU was used for the MCMC model. This was due to a large problem with vanishing gradients in the MCMC model, which ReLU helped solve.

Bias

All network architectures considered in this thesis will exclude the bias term for all layers. In financial data, which is both noisy and often has an inherent positive drift, the bias term will dominate predictions and predict only positive values for the entire data set. Experimental results confirm this hypothesis, why the bias term is excluded for all models.

6.2.2 Dropout

The dropout model does not require any alteration from a standard frequentist neural network, thus all methodology presented in the previous section applies.

Loss Function

In this thesis we are dealing with the regression problem of predicting future asset returns for a number of futures contracts. For this setting, a natural loss function is the mean squared error (MSE). However, MSE can be regarded as an unnatural loss metric for a financial trading strategy. For example, given that tomorrow's actual return is 0.1, the MSE loss would penalize a prediction of 0.3 as hard as it would -0.1, even though we would have made money with a prediction of 0.3, and incurred a loss with a prediction of -0.1.

6.2.3 Markov Chain Monte Carlo

To perform approximate Bayesian infernce with MCMC, using the Hamiltonian algorithm to propose new states, we must decide on a prior distribution of the model's weights, a likelihood distribution of the data, and select the parameters that govern the Hamiltonian algorithm.

Priors

The sampling based methods for Bayesian inference provide no inherent regularization in their learning procedure. Therefore, our only way to regularize the model is through restrictive priors. By decreasing the variance of the priors over the network's weights, we limit its ability to overfit to the training data, and thus regularize the model. However, the drawback of this approach is that tight priors which have much density close to zero might result in posterior distribution with many areas of zero density. This puts our sampling procedure at risk of getting stuck in a local maxima of the posterior, without being able to escape through areas of low density, where all proposed steps would be rejected. This would require very long chains of sampling, that would eventually escape to different modes of the posterior.

A starting point for informative priors is a standard normal distribution. However, a Gaussian prior adds a quadratic penalty to the log posterior, which might trap the chain.

A common choice of prior for BNNs is a Gaussian scale mixture [3]. This is essentially a weighted sum of two Gaussians, with different variances. By using a very low variance of the first distribution ($\sigma_1^2 \ll 1$), it will *a priori* force some of the weights close to zero. The second mixture component has a higher variance, and allows weights to vary more freely. While the results from Blundell [3], using Gaussian scale mixtures are promising, these are for variational inference. For MCMC, the low variance of the first mixture component might create areas in the resulting posterior distribution with zero density, and thus trap the chain. Experimental results using scale mixture priors, suggest that this is the case. It quickly becomes impossible to reach convergence when decreasing the variance of the first mixture component.

Finally, the Laplace distribution is considered. This distribution will provide more regularization than the Gaussian prior, but will also penalize larger steps less than the Gaussian during the MCMC sampling. Therefore, the Laplace distribution will be used as prior for the HMC algorithm.

Likelihood

Financial asset returns are often said to suffer from heavier tails than a normal distribution [18]. However, we are working with normalized time series, why the assumption of normality is more reasonable.

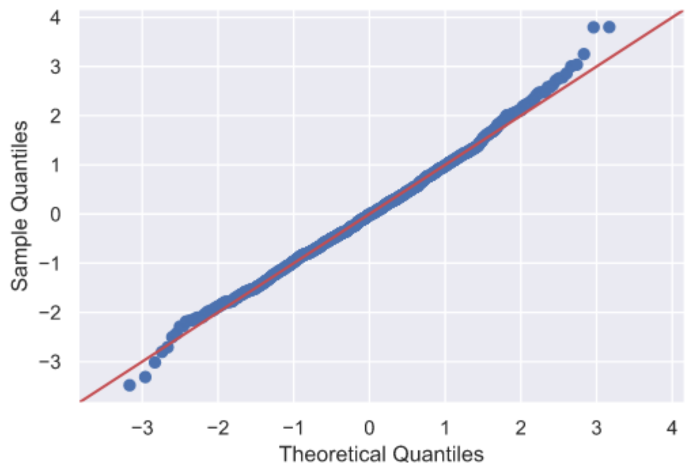


Figure 6.2: QQ-plot of the normalized returns of one market against an MLE fitted Gaussian.

figure 6.2 suggests that the normalized data is approximately Gaussian. Thus, we will use a Gaussian likelihood:

$$\mathbf{y}^*|\mathbf{w}, \mathbf{x}^* \sim N(f_{\mathbf{w}}(\mathbf{x}^*), \sigma^2), \tag{6.9}$$

where $f_{\mathbf{w}}(\mathbf{x}^*)$ is the prediction of the model given input $\mathbf{x}^*$, and σ^2 is the variance of the likelihood distribution. There are a number of ways to deal with this hyperparameter. The most basic assumption is that it is constant across time, i.e. homoscedastic noise. An extension of this would be to let it vary with time, and predict it together with the response. Since we are dealing with time series that are normalized by their rolling volatility, it could be assumed constant. The noise σ^2 could also be constant across markets, which would be reasonable using normalized time series. However, by letting σ^2 vary across markets, the model could increase noise for markets where it is more prone to errors and thereby not penalize them as hard. When using the Bayesian uncertainties in a trading strategy, this would result in the model scaling down its predictions for markets where it performs worse.

We treat the noise in the likelihood as random variable itself, introduce a hyperprior for it and infer it from the data together with the models posterior distribution using the HMC algorithm. The choice of hyperprior for σ^2 is a Gamma distribution with shape parameter 3 and scale parameter 2, as this introduces most mass around 1.

Sampling

After defining a prior distribution of the weights and a likelihood of the data, we are ready to infer the distribution of the weights given the data, using the HMC algorithm. This procedure is as follows:

1. Start with an MLE estimate of the weights, $\mathbf{w}^{MLE}$.
2. Perform $N + M$ iterations of the HMC algorithm, where each iteration in turn performs L steps of the leapfrog algorithm with a step size ϵ.
3. The first M samples of the chain are discarded and serve as a burn-in period for the chain to reach its stationary distribution.
4. Perform steps 1-3 k times, initialized with different random seeds.

The parameters that control this process are the number of samples of the burn-in, M, the number of samples that are kept, N, together with the parameters of the leapfrog algorithm, i.e. the number of leapfrog steps, L and the step size ϵ. Also, the sampling is dependent on the initial value of $\mathbf{w}$. Because of this, we perform k parallel chains of sampling. The parallel chains are joined into one model by creating an ensemble of the predictions from each chain.

The results from the HMC algorithm are sensitive to changes in all of these parameters. If N and M are too small, the chain will not reach convergence, and we will not have sampled from the target distribution. The parameters L and ϵ govern the approximation of the Hamiltonian dynamics, and will thus determine the acceptance rate of proposed samples. High values of L will generate long trajectories of the dynamics, and result in low acceptance rates (possibly 0%), while low values will result in a high acceptance rate but autocorrelated samples. Analogously, high values of ϵ will result in poor approximations of the Hamiltonian dynamics, resulting in low acceptance rates in the Metropolis step.

6.2.4 Variational Inference

As with the MCMC models, a prior distribution over the weights and a likelihood distribution over the data must be set. The variational inference models, unlike the MCMC models, also require setting a variational posterior.

Priors

As mentioned in the previous Hamiltonian Monte Carlo chapter, a common prior used for Bayesian neural networks is a scale mixture of two normal distribution with one of the two having a very small variance. This distribution provides the nice regularizing effects of a high probability mass close to zero without confining the weights overtly, because of the very fat tails. Another property of this prior is the added hyperparameters. Instead of choosing only a variance for the distributions, one has to choose two variances and the weight between the two normal distributions.

Because of the computational cost and extreme sensitivity to hyperparameters of the HMC algorithm, scale mixture priors were discarded in favor of the simpler, but somewhat similar Laplace distribution. These problems are not as prevalent in the variational posterior approach but to keep results comparable, the same prior is used in variational inference as in the HMC algorithm, namely a Laplace distribution with mean zero. The variance of this distribution is considered a hyperparameter to be optimized.

Likelihood

Like in the HMC approach, a Gaussian likelihood was assumed. The reasoning for this is the same as explained in the previous HMC section. In short, the reasoning is that the data is normalized to follow a standard normal distribution, so the likelihood should represent this.

Variational Posterior

The variational posterior, q_θ, is set to be a multivariate normal distribution $N(\theta)$, where $\theta = (\mu, \rho)$ are the parameters to be learned. The normal distribution is chosen for its simplicity, with relatively few parameters to optimize. The Bayes by backprop algorithm is designed for variational posterior distributions with diagonal covariance matrices. This is not only imposed by the algorithm, but also by the fact that the amount of parameters to learn would scale very poorly with the amount of weights in the network without this constraint. Hence, the covariance matrix of q_θ is assumed to be diagonal, leaving the individual means and variances of each weight to be learned.

6.3 Trading Strategies

The outputs of the Bayesian neural networks can be applied to trading in different ways. This section outlines the different approaches taken.

6.3.1 Long Only

As a benchmark to compare against, we use the long only strategy, defined as taking equal risk in long positions across all markets. This is achieved by taking an equal position across all markets after the data has been normalized to unit variance.

The reason for using this as a benchmark is that it is the most naive investment strategy, that incurs no trading cost, and still often performs well.

6.3.2 Frequentist

The frequentist trading strategy takes positions according to the models predictions of tomorrow's returns. Using an ensemble of N models, the prediction equals the mean of the posterior predictive distribution. This is equivalent to the MAP estimate in a frequentist setting. With predicted returns of the ith model at time t as $y_t^{(i)}$, the trading strategies positions are:

$$p_t = \frac{1}{N}\sum_{i=1}^{N} y_t^{(i)}. \tag{6.10}$$

6.3.3 Bayesian

The Bayesian trading strategy aims to exploit the predictive uncertainty of the posterior distribution. The assumption behind this strategy is that the predictive uncertainty is high when the model is more prone to errors. A high predictive uncertainty should therefore cause the strategy to decrease its positions. As a measure of predictive uncertainty we use the standard deviation of the posterior predictive distribution. Thus, the trading strategy, given N samples $y_t^{(i)}$ from the posterior predictive distribution, can be defined as:

$$\begin{aligned} s_t &= \sqrt{\frac{\sum_{i=1}^{N}\left(y_t^{(i)} - \bar{y_t}\right)^2}{N-1}} \\ p_t &= \frac{1}{s_t N}\sum_{i=1}^{N} y_t^{(i)}. \end{aligned} \tag{6.11}$$

6.4 Hyperparameter Optimization

All three algorithms for generating a predictive posterior: Dropout, Markov chain Monte Carlo and variational inference, require hyperparameters whose settings impact the effectiveness of the algorithms. Finding good values for these hyperparameters is in itself a large task requiring intimate knowledge and experience of each algorithm, as there often does not exist efficient algorithms for computing the optimal settings.

To find reasonable values of the hyperparameters of the dropout and variational inference methods, 1000 models of each type were trained on the training set and evaluated on the validation set. For each model a set of hyperparameters was randomly sampled from a set of values deemed reasonable. The hyperparameters associated with the model performing best on the validation set were then chosen. The chosen hyperparameters were used in a new model trained on the training and validation sets, to be evaluated on the test set. For the Markov chain Monte Carlo methodology, the randomized methodology of hyperparameter optimization was abandoned because of the associated computational costs.

The tables found below show how parameters were randomized or set for each model. In all the tables $U(a, b)$ denotes the uniform distribution between a and b, while $U_d(a, b)$ denotes a discrete uniform distribution.

Dropout

The hyperparameters of the dropout methodology are much the same as one would encounter in a frequentist neural network. The ranges to test for each hyperparameter are mostly set in accordance with the previous literature mentioned in chapter 1, though some problem specific intuition was applied. An example of this is the number of PCA components of the features to feed the models, which can be analyzed by what percentage of the data's variance is kept, in relation to the complexity of the optimization task.

Hyperparameter	Randomization procedure
PCA components used	$U_d(50, 100)$
Hidden dimension size	$U_d(30, 100)$
Dropout rate	$U(0.05, 0.75)$
Epochs	$U_d(10, 150)$
Activation function	$tanh$

Table 6.2: Hyperparameter sampling procedure for the dropout model

PCA components used refers to the number of principal components extracted from the original features that are kept and fed to the neural network. This range of PCA components retain around 95%-99% of the variance of the data. *Hidden dimension size* refers to the number of nodes in the single hidden layer of the model. The *dropout rate* refers to the probability of a weights activation being set to zero in the dropout schematic. *Epochs* denote the number of times the backpropagation algorithm is run over the full set of data.

Variational Inference

The contextually most important hyperparameter for the variational inference methodology is the *prior scale*, referring to the variance of the Laplace prior distribution placed over the weights of the neural network. The range of this parameter was set in reference to the variance of the scale mixture prior distribution used by Blundell et al [3].

Hyperparameter	Randomization procedure
PCA components used	$U_d(50, 100)$
Hidden dimension size	$U_d(30, 100)$
Prior scale	$U(0.01, 1)$
Epochs	$U_d(250, 1000)$
MC samples	$U_d(1, 10)$
Activation function	*tanh*

Table 6.3: Hyperparameter sampling procedure for the dropout model

PCA components used and *hidden dimension size* are defined as in the dropout section above. *Prior scale* refers to the variance of the Laplace prior over the network weights. *Epochs* refers to the number of times the Bayes by backprop algorithm is run over the full data set. *MC samples* refers to the number of samples drawn for each Monte Carlo estimate of the gradients in the Bayes by backprop algorithm.

Markov Chain Monte Carlo

Markov chain Monte Carlo can be extremely computationally expensive for high-dimensional problems. It can be difficult to converge to a stationary distribution of the chain, and also difficult to determine if the chain has converged. Furthermore, small changes to the hyperparameters can drastically change the acceptance rate of the algorithm. Therefore, it is deemed infeasible to perform a structured hyperparameter optimization. Instead, we manually select hyperparameters to ensure a high enough acceptance rate.

Hyperparameter	Value
PCA components	80
Hidden dimension size	80
Prior scale	1
Leapfrog steps	10
Leapfrog step size	0.001
Number of samples	2000
Thinning	1000
Parallel chains	10
Activation function	*ReLU*

Table 6.4: Hyperparameters for HMC

Leapfrog step size and *leapfrog steps* refer to the parameters that govern the leapfrog algorithm in the approximation of the Hamiltonian dynamics in subsection 4.3.2. The *number of samples* denotes the total number of steps of the MCMC chain, before discarding the steps of the burn-in. Finally, the *thinning* hyperparameter govern the number of steps taken between two states in the result. That is, to decrease autocorrelation we only keep every 1000th step of the chain, therefore making a total of $2 \cdot 10^6$ steps. *Parallel chains* are the number of different MCMC chains that are run in parallel, with different initializations.

6.5 Software Implementation

The models used in this thesis have been implemented using Python 3.7, together with a number of libraries for numerical processing and machine learning. For the Bayesian models based on MC dropout, the library Keras [6] has been used, with Tensorflow [1] as backend. The HMC models, together with the VI based models have used Tensorflow, together with the sub-library Tensorflow Probability. This contains a library called Edward 2 that was used for creating the probabilistic models. Tensorflow Probability also contains an implementation of the Hamiltonian Monte Carlo algorithm, that was used in this thesis. This exploits Tensorflow's automatic differentiation engine for computing the gradients of the posterior.

Chapter 7

Results

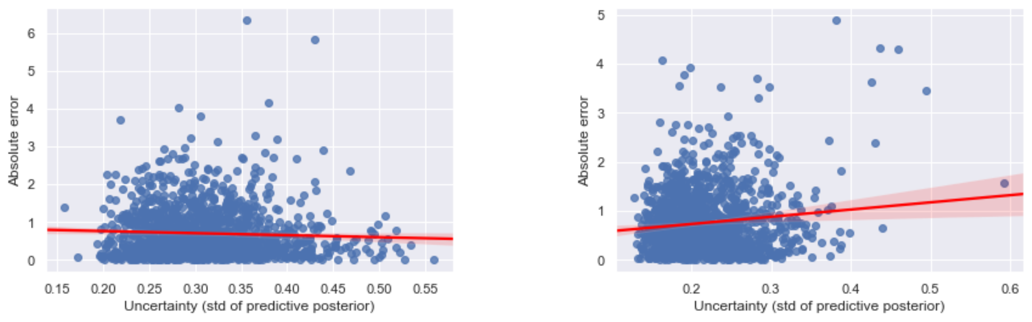
There are mainly two types of results displayed in this chapter; correlations between prediction errors and prediction uncertainties and Sharpe ratios for trading strategies. Both of these metrics are accompanied by some graphical illustrations intended to show a more nuanced overview of the results. The uncertainties section as well as the Sharpe ratio section of this chapter are divided into three subsections, each showing the same results for the three different algorithms being compared; Dropout, variational inference and Markov chain Monte Carlo. In general, results and illustrations are presented for the validation set and the test set seperately. It is important to note that a large number of models were evaluated and the best ones chosen on the validation set, while the test set was only used to measure one final model per algorithmic family. Therefore, the results from the test set are the most interesting as a reference for future performance.

7.1 Uncertainties

An estimate of model uncertainties can be considered successful when the correlation between prediction error and the measured uncertainty is high. When measuring this correlation for a trading problem including different markets, it is important to measure the correlation separately on each market, as different markets have different aleatoric uncertainties. Because of this, the correlation between error and uncertainty is measured separately for each market, and the mean over markets is reported. This is illustrated as a scatter plot of the errors and uncertainties for two markets, the one with the lowest correlation metric and the one with the highest correlation metric. Including all markets in one illustration would mostly show the varying uncertainties over markets rather than the correlation to the prediction error. The red line shown in the scatter plots show a least squares fitted regression line with a shaded confidence interval.

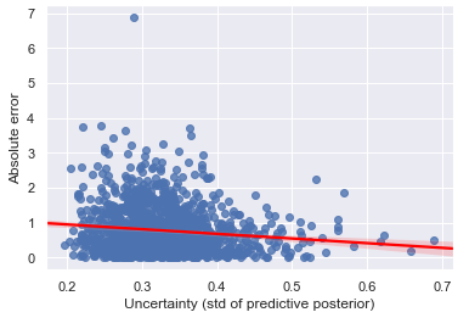
7.1.1 Dropout

The resulting correlations between prediction error and uncertainties of the dropout model are very close to zero in the validation set as well as the test set. The average correlation is 0.017 in the validation set and 0.014 in the test set. This result implies that the Bayesian trading strategies based on the dropout model will not outperform the frequentist strategies.

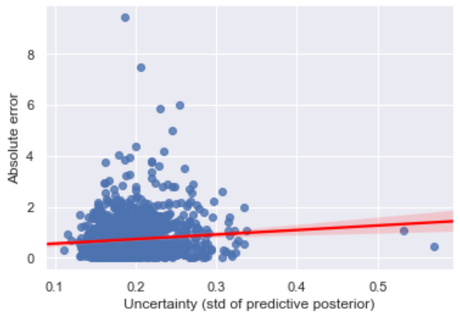


(a) Market with lowest correlation (b) Market with highest correlation

Figure 7.1: Prediction error as a function of uncertainties, validation set



(a) Market with lowest correlation

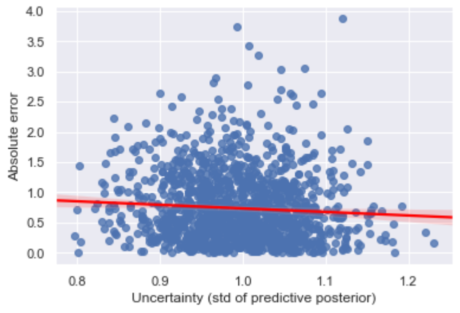


(b) Market with highest correlation

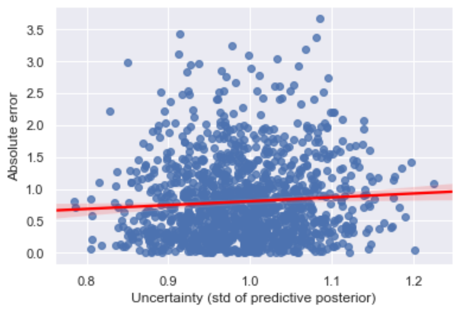
Figure 7.2: Prediction error as a function of uncertainties, test set

7.1.2 Variational Inference

The correlations between expressed uncertainty and prediction error of the variational inference model, like the dropout model, appears to be almost nonexistent. The average correlation over markets is 0.006 in the validation data, and 0.001 in the test data.

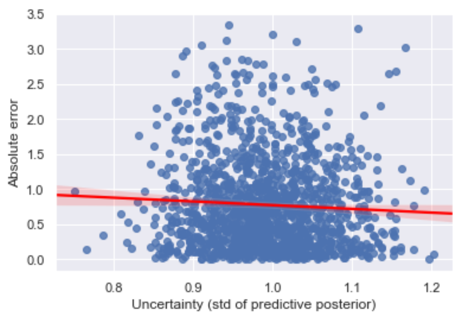


(a) Market with lowest correlation

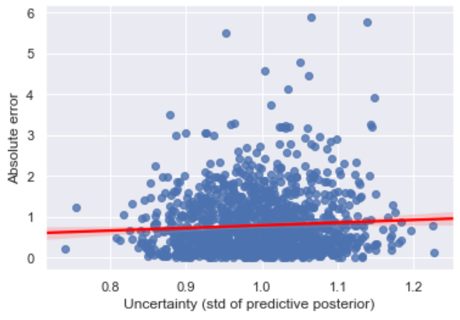


(b) Market with highest correlation

Figure 7.3: Prediction error as a function of uncertainties, test set



(a) Market with lowest correlation

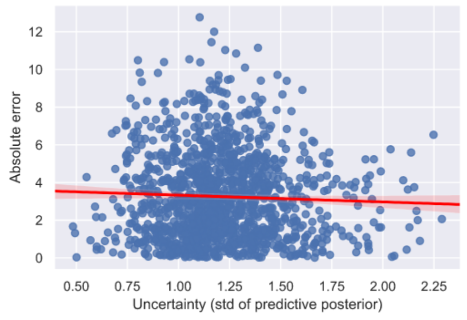


(b) Market with highest correlation

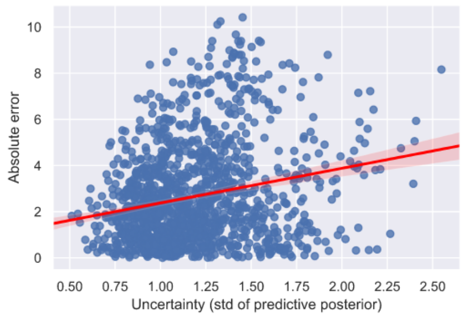
Figure 7.4: Prediction error as a function of uncertainties, validation set

7.1.3 Markov Chain Monte Carlo

The uncertainties from the model based on the HMC algorithm appears to have a (weak) connection to its prediction errors. The average correlation on the validation set was 0.0887, and 0.2579 on the test set.

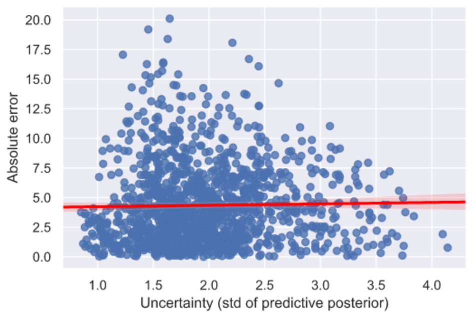


(a) Market with lowest correlation

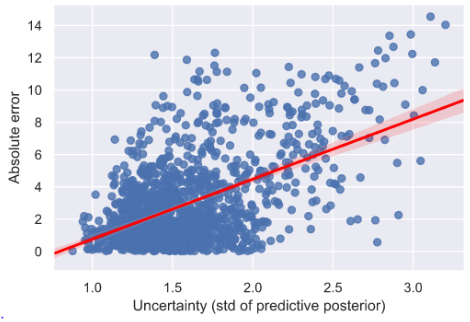


(b) Market with highest correlation

Figure 7.5: Prediction error as a function of uncertainties, test set



(a) Market with lowest correlation



(b) Market with highest correlation

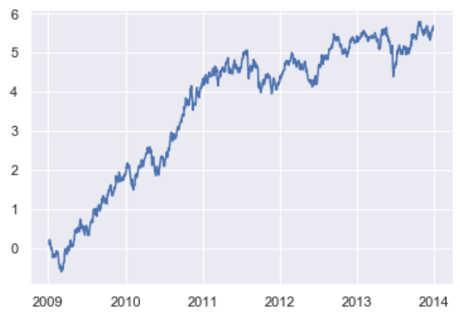
Figure 7.6: Prediction error as a function of uncertainties, validation set

7.2 Trading Performance

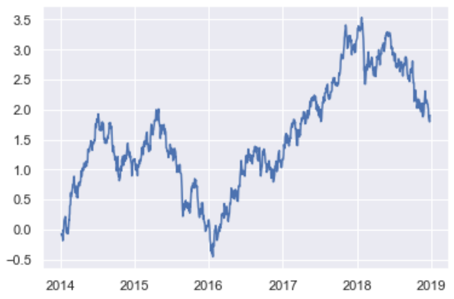
This section presents Sharpe ratios and equity curves for the different trading strategies generated from the model predictions and uncertainties. An explanation of the Sharpe ratio and its properties can be found in chapter 2. The equity curves show the performance of a given trading strategy over time. The returns of the strategy are normalized to unit variance over the entire evaluation period, in order to plot the risk adjusted returns and not simply the returns. Consequently, the x-axis of the graphs show the dates of the set the strategy is being evaluated on and the y-axis shows monetary gains in terms of portfolio variance. For each family of models, i.e. dropout, variational inference and Markov chain Monte Carlo, the results of the frequentist and Bayesian strategies are reported. The results of a buy and hold baseline strategy are presented for the validation and test periods separately. A buy and hold strategy refers to buying all assets and not selling them for the duration of the period, selling them at the end of the period.

7.2.1 Long only

The equity curves for the long only strategy, as well as for the two strategies based on the dropout model are displayed in this chapter. Each row of figures correspond to a specific strategy and displays the equity curve evaluated on the test set to the left and the validation set to the right. Because the long only strategy is unaffected by training algorithm and strategies, its equity curves are displayed only once here.



(a) Validation set



(b) Test set

Figure 7.7: Equity curve, long only strategy

There is a large difference in the success of the long only strategy between the two periods, which could affect the efficiency of the trading strategies that are the result of this thesis.

7.2.2 Dropout

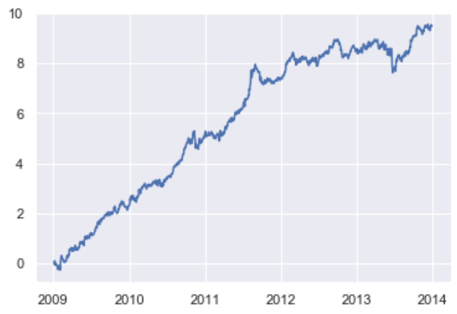
The following table displays the Sharpe ratios of the strategies based on the predictions and uncertainties of the dropout model.

	Validation	Test
Frequentist	1.837	0.673
Bayesian	1.959	0.597
Long only	1.066	0.370

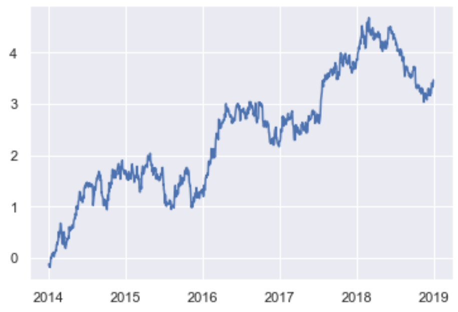
Table 7.1: Sharpe ratios for strategies based on the dropout model

The results are significantly lower on the test set than on the validation set. The Bayesian strategies perform worse than the frequentist strategies which is reasonable, given that the generated uncertainties do not have a strong correlation to the prediction errors.

Equity curves



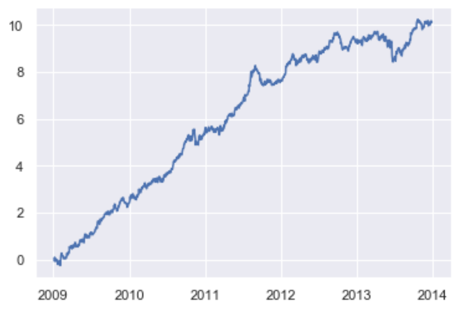
(a) Validation set



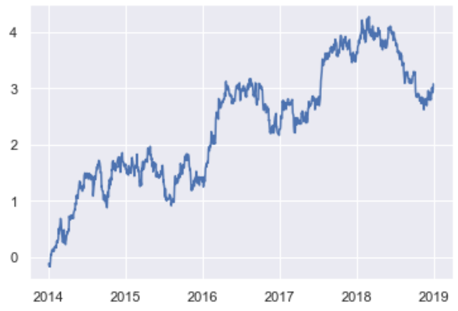
(b) Test set

Figure 7.8: Equity curve, frequentist strategy

We see a large difference in stability and overall trend between the test and validation set for the frequentist strategy. The large number of models evaluated in the hyperparameter search seems to have performed some overfitting to the validation set. Even so, the strategy outperforms the long only baseline.



(a) validation set



(b) Test set

Figure 7.9: Equity curve, Bayesian strategy

The equity curve of the Bayesian strategy is very similar to the frequentist one, implying that the positions of the strategies are very similar, although somewhat less accurate in the Bayesian strategy.

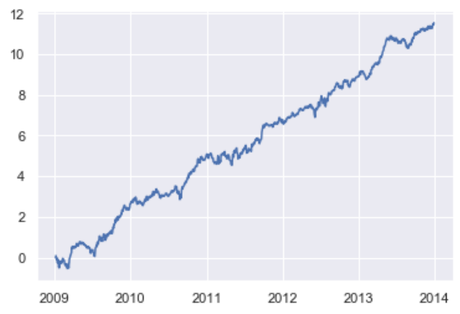
7.2.3 Variational Inference

The following table displays the Sharpe ratios of the strategies based on the predictions and uncertainties of the variational inference model. Like in the dropout model, we see a significant drop in performance on the test set compared to the validation set.

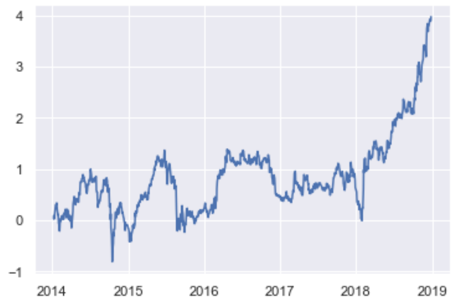
	Validation	Test
Frequentist	2.240	0.774
Bayesian	2.232	0.737
Long only	1.066	0.370

Table 7.2: Sharpe ratios for strategies based on MCMC

Equity curves



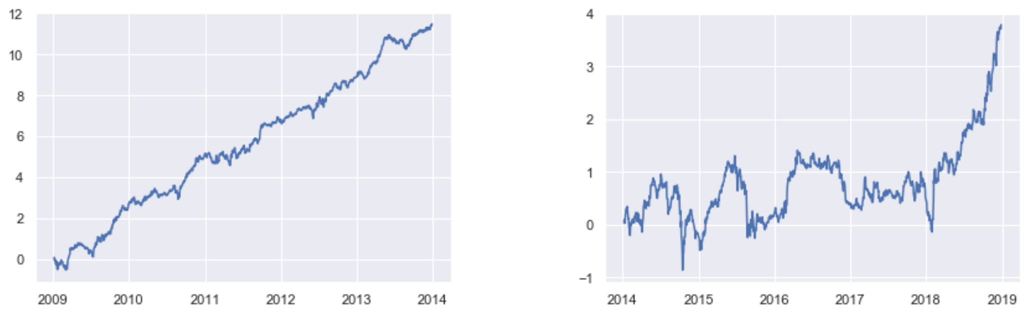
(a) Validation set



(b) Test set

Figure 7.10: Equity curve, frequentist strategy

The strategy is profitable and outperforms the long only strategy on the validation data as well as on the test data. As implied by the Sharpe ratios, we see a significant drop in profitability and overall stability of the strategy on the test data compared to the validation data.



(a) Validation set (b) Test set

Figure 7.11: Equity curve, Bayesian strategy

The Bayesian strategy shows result very similar to the frequentist one. As shown in the uncertainties section of this chapter, the Bayesian strategy suffers from uncertainties that are not correlated with prediction errors, so these results are expected.

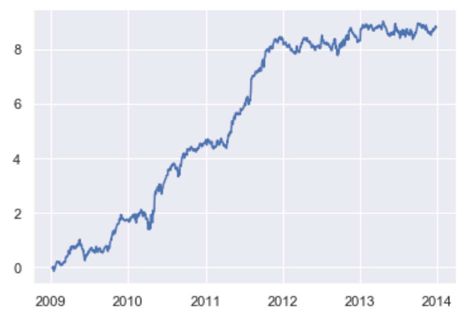
7.2.4 Markov Chain Monte Carlo

Markov chain Monte Carlo provided uncertainties with a weak correlation to its predictive errors. These uncertainties resulted in an improvement of the Bayesian strategy, compared to the frequentist one, on both validation and test. This is remarkable, since a hyperparameter search was not performed for MCMC. Due to the computational complexity of the MCMC model it was also unfeasible to retrain the model including the validation set, for prediction on the hold out test set. This might explain the poor performance on the test set, compared to dropout and variational inference. The following table summarizes the results from the models based on MCMC sampling:

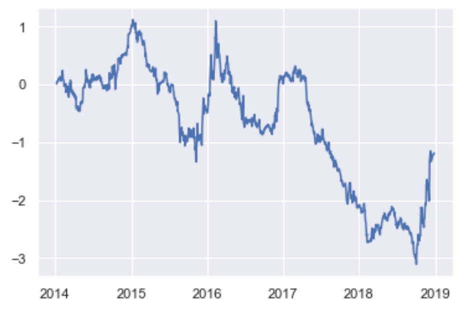
	Validation	Test
Frequentist	1.708	-0.230
Bayesian	1.922	-0.208
Long only	1.066	0.370

Table 7.3: Sharpe ratios for strategies based on MCMC

Equity Curves



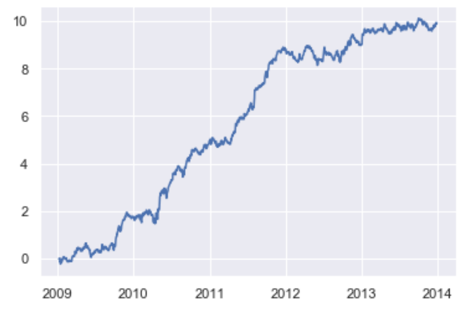
(a) Validation set



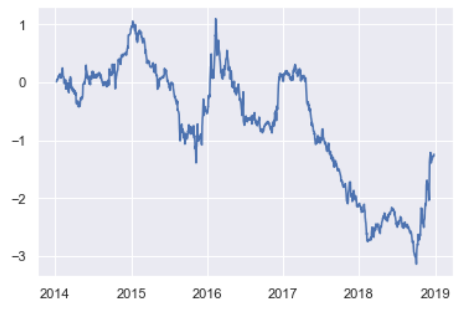
(b) Test set

Figure 7.12: Equity curve, frequentist strategy

The frequentist strategy performed well on the validation set, although most of the profits come from the first three years. The test set provided more difficulty for the model, but this might in part be attributed to the fact that it has not been trained on the data from the validation set.



(a) Validation set



(b) Test set

Figure 7.13: Equity curve, Bayesian strategy

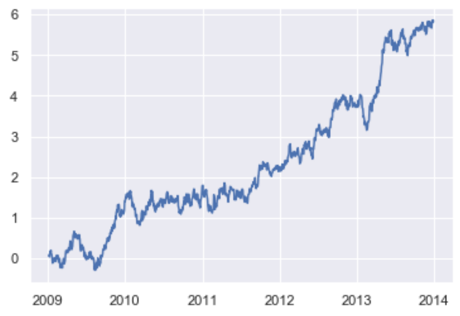
The Bayesian strategy provided an improvement over the frequentist strategy on both the validation and test sets. However, on the test set this improvement is very small, and its performance is still poor.

7.2.5 Markowitz Optimizaion

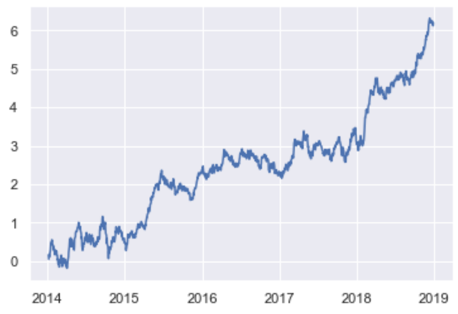
We performed a Markowitz optimization of the models, with the hopes of generating a higher Sharpe ratio. For theory and methodology on this, see the appendix. We present the results of the Markowitz optimized dropout model, since this presented the greatest improvement on the strategy.

	Validation	Test
Frequentist	1.923	1.049
Bayesian	2.242	0.938
Long only	1.066	0.370

Table 7.4: Sharpe ratios for strategies based on MCMC



(a) Validation set



(b) Test set

Figure 7.14: Equity curve, Markowitz optimized strategy

Chapter 8

Discussion

8.1 Dropout

When reviewing the results of this thesis in general, but of the dropout model especially, it is important to make a distinction between regularization effects and evaluating uncertainty. The Bayesian strategies based on the dropout model differ from the frequentist approach by scaling the predictions with the generated uncertainties, but also in the predictions themselves as the frequentist ones are not samples from a predictive posterior, but point estimates from a frequentist neural network trained with dropout. As such, the fact that the Bayesian strategies perform on par with, but not better than, the frequentist strategies shows that the Bayesian approach to dropout works at least as well as the frequentist approach as a regularization technique, even for this very noisy data set.

The uncertainties generated by the dropout model does however seem to be completely uncorrelated to the prediction error, and are therefore not useful in the trading setting of this thesis. This is inconsistent with results shown by Gal and Ghahramani [12], who show results that indicate that the modelled uncertainties are reasonable proxies for the actual underlying uncertainties. The data modelled by Gal and Ghahramani is however significantly less noisy than the data used in this thesis, which is a likely reason for the very different results.

8.2 Variational Inference

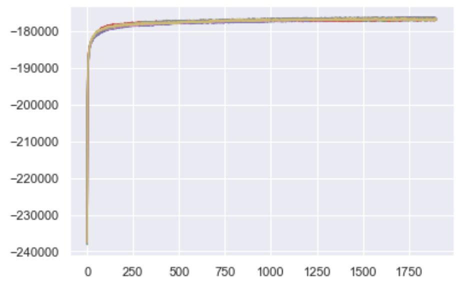
The results from the variational inference method indicate that the method succeeds in learning patterns from the training data that generalize to the validation set. As such, the method can be said to be successful in the regularization task discussed in the methodology chapter. This result is interesting in itself because it opens for an explicit definition of the prior distribution used to regularize the model, unlike conventional regularization techniques in frequentist neural networks where a prior distribution is implied. These results also show that the Bayes by backprop modification of the backpropagation algorithm is applicable and generates a trained Bayesian neural network even for the very noisy financial data used in this thesis.

The uncertainties expressed by the standard deviation of samples from the predictive posterior does not correlate with the prediction error at all. These uncertainties are therefore poor approximators of the actual model uncertainty, which makes their usefulness in a trading strategy limited. A probable cause for these poor results is that the variational posterior is misspecified and can not converge towards the true posterior. There is no reason to believe that the covariance matrix of the true posterior is diagonal, i.e. that the true covariance between the networks weights are zero. The enormous amount of parameters to learn for a variational posterior without this restraint is a limitation that can only be circumvented by equally enormous computational power, although for any larger Bayesian neural network this problem becomes unfeasible for almost any machine.

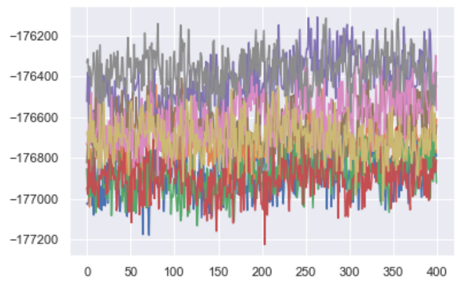
8.3 Markov Chain Monte Carlo

The models based on MCMC provided both powerful regularization, yielding results on par with dropout and variational inference on the validation set, while also providing meaningful uncertainties, that allowed the Bayesian trading strategy to outperform the frequentist strategy. The MCMC model was outperformed on the test set, but this can be explained by the fact that the model evaluated on test was not retrained including the validation data, due to computational limitations.

We can diagnose the sampling procedure from the MCMC model to try to assess whether the chain has converged to its stationary distribution or not. First of all, we can inspect the log posterior distribution of each of the sampling chains. This can provide insight into if the chains get stuck in different local minima, or if they are still trying to escape from areas of low density.



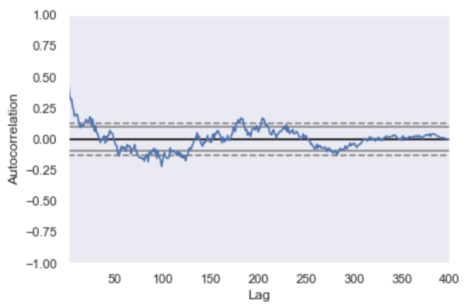
(a) All samples included



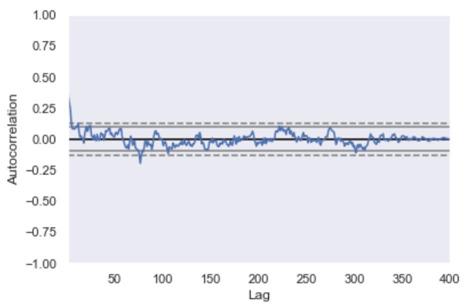
(b) Samples after burn-in

Figure 8.1: Log posterior probability

As seen from figure 8.1a, all 10 different chains converge to the same posterior density, despite different initializations. By discarding a burn-in of 1500 steps, we can view the resulting log posterior densities in figure 8.1b. From this, the chains appear to to be stationary, but further diagnostics are required.



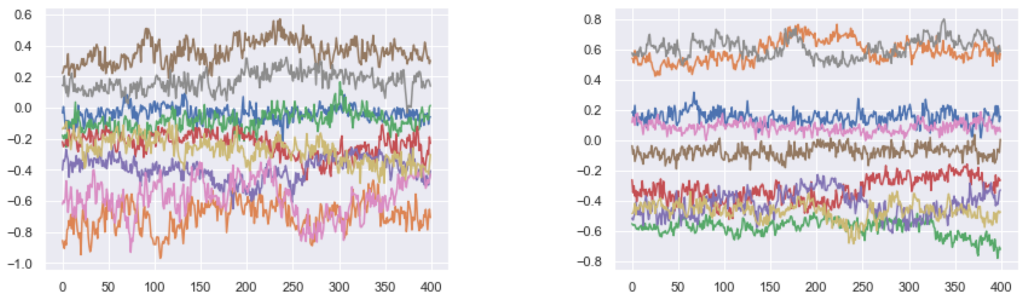
(a) Weight in the input layer



(b) Weight in the output layer

Figure 8.2: Autocorrelations of the MCMC chain for two different weights

figure 8.2 displays the autocorrelations for two different weights (one from the input layer, and one from the output layer) in one of the sampling chains. While these only display a small subset of all weights of the network and thus only provide anecdotal evidence of convergence, the autocorrelations appear to be close to zero for all weights in the network.



(a) Weight in the input layer (b) Weight in the output layer

Figure 8.3: MCMC chains for two different weights

Figure 8.3 displays the samples after burn-in from one weight in the input layer and one weight in the output layer. Each sub-figure displays the samples from the same weight, but from the different initializations. This indicates that the chains still have not reached convergence. If they had converged to their stationary distribution, and because they all have the same stationary distribution (they only have different initializations), they would have sampled from the same range of values. Instead, it appears as if they have all converged to different local minima. If the chains have not converged, we have not sampled from the true posterior distribution. This brings us to question the resulting posterior predictive uncertainties. If the samples are not from the true posterior, the uncertainties cannot represent the true predictive uncertainty.

Chapter 9

Conclusions

We conclude that Bayesian techniques for neural networks provide powerful regularization methods. Both dropout and variational inference generalize to unseen data. However, there is not a connection between their predictive uncertainty and prediction error, nor is it possible to improve a trading strategy by scaling the positions by a measure of predictive uncertainty.

Markov chain Monte Carlo, on the other hand, provides both regularization and produces predictive uncertainties that correlated with the prediction error. These uncertainties were able to improve a trading strategy. However, the method does not fully converge towards the desired true posterior distribution and more research into this is required.

9.1 Research Questions

This section attempts to answer the three research questions posed in chapter 1, using the results from chapter 7, and the following discussion from chapter 8.

What method for approximate Bayesian inference in neural networks is most suitable for financial time series?

Both dropout and variational inference provide efficient regularization, and require relatively few computations. Dropout has the advantage of not requiring any alterations of a standard neural network model, with the drawback that it becomes more difficult to control. We cannot control the prior distribution over the weights, and therefore not the posterior. Variational inference is a more explicit way of regularization, with the freedom to choose a prior distribution, likelihood over data and variational posterior over the weights. This has the disadvantage of requiring more parameters, even with the restrictive assumption of no covariance between weights.

Markov chain Monte Carlo presents a massive computational challenge, even for relatively small networks. However, it generalizes well to unseen data and provides a lot of flexibility for defining the problem, e.g. market specific aleatoric uncertainty. Furthermore, the MCMC model provides predictive uncertainties correlated with prediction errors, and is able to outperform a frequentist trading strategy.

How is the uncertainty of the posterior predictive distribution of a Bayesian neural network related to its prediction errors?

For dropout and variational inference, there appears to be no relation between posterior predictive uncertainty and prediction error, when comparing the correlation between squared errors and standard deviations of the posterior predictive uncertainty. We conclude that both dropout and variational inference requires too restrictive assumptions on the posterior of the model, which washes out the connection between predictive errors and uncertainty.

With Markov chain Monte Carlo, there is a slight connection. However, since these uncertainties are from a sampling chain that has not yet reached convergence, and thus have not sampled from the true posterior distribution, it is not clear what these uncertainties actually represent.

By utilizing predictive uncertainty, can we improve a systematic trading strategy compared to the case with frequentist predictions?

Without any correlation between predictive uncertainty and error, we cannot improve a systematic trading strategy by scaling the models positions by its predictive uncertainty, as expected. Thus, neither dropout nor variational inference can improve a trading strategy using their predictive uncertainty. For Markov chain Monte Carlo, where there is a connection between uncertainty and error, we saw an improvement from using uncertainties.

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[2] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] Charles Blundell et al. "Weight Uncertainty in Neural Networks". In: *arXiv preprint arXiv:1505.05424* (2015).

[4] Peter J Brockwell, Richard A Davis, and Matthew V Calder. *Introduction to time series and forecasting*. Vol. 2. Springer, 2002.

[5] Tianqi Chen, Emily Fox, and Carlos Guestrin. "Stochastic gradient hamiltonian monte carlo". In: (2014), pp. 1683–1691.

[6] François et al. Chollet. *Keras*. https://keras.io. 2015.

[7] Werner F.M. De Bondt, Richard Thaler, and Peter L. Bernstein. "Does the stock market overreact? (and discussion) (papers and proceedings of the forty-third annual meeting of the American Finance Association, held in Dallas, Texas, on December 28 through December 30, 1984)". In: *Journal of Finance* 40.3 (1985). ISSN: 0022-1082.

[8] Nan Ding et al. "Bayesian sampling using stochastic gradient thermostats". In: (2014), pp. 3203–3211.

[9] Simon Duane et al. "Hybrid monte carlo". In: *Physics letters B* 195.2 (1987), pp. 216–222.

[10] Eugene F. Fama. "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *The Journal of Finance* 25.2 (1970). ISSN: 00221082.

[11] Yarin Gal. "Uncertainty in Deep Learning". In: *University of Cambridge* (2016).

[12] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *International Conference on Machine Learning*. 2016, pp. 1050–1059.

[13] Alex Graves. "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*. 2011, pp. 2348–2356.

[14] W Keith Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: (1970).

[15] Geoffrey Hinton and Drew Van Camp. "Keeping neural networks simple by minimizing the description length of the weights". In: (1993).

[16] Matthew D Hoffman and Andrew Gelman. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1593–1623.

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks are Universal Approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[18] Filip; Hammarlid Ola; Rehn Carl Johan Hult Henrik; Lindskog. *Risk and Portfolio Analysis : Principles and Methods*. eng. New York: Springer, 2012. ISBN: 9781493900312.

[19] Narasimhan Jegadeesh and Sheridan Titman. "Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency". In: *Journal of Finance* 48.1 (1993), pp. 65–91. ISSN: 0022-1082.

[20] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[21] D.P. Kingma et al. "Variational Dropout and the Local Reparameterization Trick". eng. In: *29th Annual Conference on Neural Information Processing Systems 2015* 3 (2015), pp. 2575–2583. ISSN: 1049-5258.

[22] Solomon Kullback and Richard A Leibler. "On Information and Sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[23] Harry Markowitz. "Portfolio Selection". In: *Journal of Finance* 7.1 (1952), pp. 77–91. ISSN: 0022-1082.

[24] Nicholas Metropolis et al. "Equation of State Calculations by Fast Computing Machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[25] Radford M. Neal. *Bayesian Learning for Neural Networks*. Vol. 118. Springer Science & Business Media, 2012.

[26] Radford M Neal et al. "MCMC using Hamiltonian dynamics". In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.

[27] William F Sharpe. "Mutual fund performance". und. In: *The journal of business : B* 39.1 (1965), pp. 119–138. ISSN: 00219398.

[28] William F Sharpe. "The sharpe ratio". In: *Journal of portfolio management* 21.1 (1994), pp. 49–58.

[29] Nitish Srivastava et al. "Dropout: a Simple Way to Prevent Neural Networks from Overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[30] Gustaf Tegnér. *Recurrent neural networks for financial asset forecasting*. 2018.

[31] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics". In: (2011), pp. 681–688.

[32] Yeming Wen et al. "Flipout: Efficient pseudo-independent weight perturbations on mini-batches". In: *arXiv preprint arXiv:1803.04386* (2018).

[33] Philip Widegren. *Deep learning-based forecasting of financial assets*. 2017.

[34] Lingxue Zhu and Nikolay Laptev. "Deep and confident prediction for time series at uber". In: (2017), pp. 103–110.

Appendix

Portfolio Optimization

A set of investments are often referred to as an investment portfolio. Given a market $\mathbf{S}_t \in \mathbb{R}^n$ of n assets at time t, a portfolio can be expressed as the set of positions $\mathbf{h}_t$ in the assets. We write

$$\mathbf{R}_{t+1} = \frac{\mathbf{S}_{t+1}}{\mathbf{S}_t} \in \mathbb{R}^n \tag{9.1}$$

$$V_{t+1} = \mathbf{h} \cdot \mathbf{R}_{t+1} \in \mathbb{R}, \quad \mathbf{h} \in \mathbb{R}^n \tag{9.2}$$

Where, for a time increment 1, $\mathbf{R}_{t+1}$ denotes the returns of the market and V_{t+1} denotes the return of the portfolio defined by $\mathbf{h}$. The process of portfolio optimization can then be defined as optimizing some utility function $f(\mathbf{h}, R_{t+1})$ with respect to $\mathbf{h}$, subject to some set of restrictions. It is common to choose f so that it somehow reflects the expected returns $\mathbb{E}[V_{t+1}]$ and the risk of the portfolio. The Sharpe ratio defined in (2.2) is an example of such a function.

Modern Portfolio Theory

Harry Markowitz introduced modern portfolio theory (MPT) [23] in 1952, but the framework is still widely in use. An important assumption for MPT is that investors are risk averse, meaning that when presented with a choice between two portfolios with similar expected return, a rational investor will choose the less risky one. This assumption introduces the concept of diversification - to spread ones portfolio over several assets to reduce risk. MPT formalizes this school of thought by quantifying portfolio risk as the variance of the portfolio value. Using (9.2) as our expression for portfolio value, the variance can be calculated as

$$Var(V_{t+1}) = \mathbf{h}^T \mathbf{\Sigma}_{t+1} \mathbf{h}. \tag{9.3}$$

where $\mathbf{\Sigma}_{t+1}$ denotes the covariance matrix of asset returns.

$$\mathbf{\Sigma}_{t+1} = \begin{bmatrix} Var(R_{t+1,1}) & Cov(R_{t+1,1}, R_{t+1,2}) & ... & Cov(R_{t+1,1}, R_{t+1,n}) \\ Cov(R_{t+1,2}, R_{t+1,1}) & Var(R_{t+1,2}) & ... & Cov(R_{t+1,2}, R_{t+1,n}) \\ ... & ... & ... & ... \\ Cov(R_{t+1,n}, R_{t+1,1}) & ... & ... & Var(R_{t+1,n}) \end{bmatrix}$$

Using this metric for portfolio risk, it is possible to define a wide variety of optimization problems balancing expected returns against risk. Perhaps the most common of these is the "trade-off problem", given by the following.

$$\underset{\mathbf{h}}{\text{maximize}} \quad \mathbf{h}^T \cdot \mathbb{E}[\mathbf{R}_{t+1}] - \frac{C}{2} \mathbf{h}^T \mathbf{\Sigma}_{t+1} \mathbf{h} \tag{9.4}$$

Where $C \in \mathbb{R}$ is some constant set by the investors level of risk aversion and the expected value in the objective function is derived by

$$\mathbb{E}[V_{t+1}] = \mathbb{E}[\mathbf{h} \cdot \mathbf{R}_{t+1}] = \mathbf{h} \cdot \mathbb{E}[\mathbf{R}_{t+1}].$$

The trade-off problem is a convex optimization problem with a solution given by the expression for **h** below. Proof for this can be found in [18].

$$\mathbf{h} = \frac{1}{C} \mathbf{\Sigma}_{t+1}^{-1} (\mathbb{E}[\mathbf{R}_{t+1}]) \tag{9.5}$$

The constant C in this solution controls the leverage of the portfolio. It's obvious from the solution in (9.5) that this modeling is dependent on set values for the expected return $\mathbb{E}[\mathbf{R}_{t+1}]$ and the covariance matrix Σ_{t+1}. This is problematic since $\mathbf{R}_{t+1}$ is an unobserved random variable whose distribution can never be fully known. Using MPT is therefore dependent on some exogenous model for estimating the expected return and covariance of assets. This thesis employs the predictions of neural networks trained on past returns as proxies for $\mathbb{E}[\mathbf{R}_{t+1}]$ and estimates Σ_{t+1} directly from past returns.

Optimizing the Sharpe Ratio

It can be shown that the solution to (9.4) shown in (9.5) corresponds to maximizing the Sharpe ratio defined in (2.2). Furthermore, the maximum Sharpe value found by this optimization procedure is

$$\text{Sharpe}_{\text{max}} = \sqrt{\mathbb{E}[\mathbf{R}_{t+1}]\Sigma_{t+1}^{-1}\mathbb{E}[\mathbf{R}_{t+1}]}. \tag{9.6}$$

Again, proof of this can be found in [18].

Portfolio Optimization Methodology

The predictions of the neural networks are optimized, using the different training algorithms, using the mean squared prediction error as loss function. This provides a prediction of the coming market movement, but does not take the entire portfolios value into account. To remedy this, the portfolio optimization procedure outlined in chapter 2 is performed. The procedure for using modern portfolio theory to optimize the Sharpe ratio of a portfolio requires an estimation of the expected value of the coming movements for each markets and an estimation of the covariance between the markets. The predictions of the neural networks trained to minimize MSE can be used as an estimation of the expected values, but the covariance of the markets must be estimated in some other fashion. Covariance between markets must be considered a function of time, as they are far from constant. Accordingly, the covariance matrix should be estimated for each set of predictions, resulting in a daily estimation in the context of this thesis.

A simple but noisy estimation of the covariance of market returns is the outer product $\mathbf{r}_t \otimes \mathbf{r}_t^T$ where $\mathbf{r}_t$ denotes the vector of all market returns on day t. This estimation assumes that $\mathbb{E}[\mathbf{r}_t] = 0$, since otherwise we would subtract the term $\mathbb{E}[\mathbf{r}_t]^2$. This assumption is reasonable since the data is normalized. These estimates can be made more robust by applying an exponential moving average (EMA) to the initial estimations, smoothing the covariance over time according to the equation below.

$$\Sigma_t = (1-\alpha)\Sigma_{t-1} + \alpha(\mathbf{r}_t \otimes \mathbf{r}_t^T). \tag{9.7}$$

Finally, the EMA smoothed covariance matrices are regularized by adding weight to the diagonal entries of the covariance matrix, i.e. the individual variance of the markets.

$$\Sigma_t^* = (1 - \rho)\Sigma_t + \rho I. \tag{9.8}$$

The resulting estimates of the covariance matrix Σ_t^* and the predictions of the neural networks can then be inserted in (9.5) to retrieve new positions, optimized for a high Sharpe ratio.

The value for α is set to 1/200, to be consistent with the values set in the features containing exponential moving averages. The regularization term ρ has been set to 0.2.

TRITA -SCI-GRU 2019:095