DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Modeling the Term Structure of Interest Rates with Restricted Boltzmann Machines

MARKUS BERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Modeling the Term Structure of Interest Rates with Restricted Boltzmann Machines

MARKUS BERG

Degree Projects in Mathematical Statistics (30 ECTS credits)
Degree Programme in Applied and Computational Mathematics (120 credits)
KTH Royal Institute of Technology year 2018
Supervisor at KTH: Henrik Hult
Examiner at KTH: Henrik Hult

TRITA-SCI-GRU 2018:241
MAT-E 2018:45

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

Modeling the Term Structure of Interest Rates with Restricted Boltzmann Machines

Abstract

This thesis investigates if Gaussian restricted Boltzmann machines can be used to model the Swedish term structure of interest rates. The tested models are evaluated based on the ability to make one-day-ahead forecasts and the ability to generate plausible long term scenarios. The results are compared with simple benchmark models, such as assuming a random walk. The effects of principal component analysis as data preprocessing are also investigated.

The results show that the ability to make one-day-ahead forecasts, measured as a mean squared error, is comparable to a random walk benchmark both in-sample and out-of-sample. The ability to generate long term scenarios show promising results based on visual properties and one-year-ahead forecast error on semi-out-of-sample data.

The main focus of the thesis is not to optimize performance of the models, but instead to serve as an introduction to modeling the term structure of interest rates with Gaussian restricted Boltzmann machines.

Modellering av avkastningskurvan med restricted Boltzmann machines

Sammanfattning

Denna uppsats undersöker huruvida Gaussian restricted Boltzmann machines kan användas för att modellera avkastningskurvan baserad på svensk data. De testade modellerna utvärderas baserat på förmågan att förutsäga morgondagens avkastningskurva och förmågan att generera rimliga långsiktiga scenarier för avkastningskurvan. Resultaten jämförs med enkla jämförelsemodeller, så som att anta en slumpvandring. Effekten av att använda principalkomponentanalys för att preparera datan undersöks också.

Resultaten visar att förmågan att förutsäga morgondagens avkastningskurva, mätt som medelkvadratfel, är jämförbar med att anta en slumpvandring, både in-sample och out-of-sample. Förmågan att generera rimliga långsiktiga scenarier visar på lovande resultat baserat på synbara egenskaper och förmågan till att göra ettåriga förutsägelser för semi-out-of-sample data.

Uppsatsens huvudfokus är inte att optimera prestandan för modellerna, utan istället att vara en introduktion till hur avkastningskurvan kan modelleras med Gaussian restricted Boltzmann machines.

Contents

1 Introduction **1**
- 1.1 Previous work . . . 1
- 1.2 Objective . . . 3
- 1.3 Scope and limitations . . . 3
- 1.4 Disposition . . . 3

2 Financial background **4**
- 2.1 Bonds . . . 4
- 2.2 Yields . . . 4
- 2.3 Term structure of interest rates . . . 5

3 Mathematical background **5**
- 3.1 Artificial neural networks . . . 5
- 3.2 Graphs . . . 6
  - 3.2.1 Probabilistic graphical models . . . 6
  - 3.2.2 Markov random fields . . . 7
- 3.3 Restricted Boltzmann machines . . . 8
  - 3.3.1 Maximum likelihood estimation . . . 10
  - 3.3.2 Gradient ascent . . . 10
  - 3.3.3 Gibbs sampling . . . 11
  - 3.3.4 Contrastive Divergence . . . 12
  - 3.3.5 Annealed importance sampling . . . 13
- 3.4 Principal component analysis . . . 15

4 Method **16**
- 4.1 Data . . . 16
  - 4.1.1 Data preprocessing . . . 17
- 4.2 Models and training . . . 18
  - 4.2.1 Ensemble . . . 19
  - 4.2.2 Benchmarks . . . 20
- 4.3 Evaluations . . . 20
  - 4.3.1 One-day-ahead forecasts . . . 20
  - 4.3.2 Long term scenario generation . . . 20

5 Results **21**
- 5.1 One-day-ahead forecasts . . . 21
- 5.2 Long term scenario generation . . . 23

6 Discussion **30**
- 6.1 One-day-ahead forecasts . . . 30
- 6.2 Long term scenario generation . . . 31
- 6.3 Preprocessing . . . 32

7 Conclusion **33**
- 7.1 Future work . . . 33

A Ensemble results **35**

B Conditional probability distributions **36**

C Gradient of log-likelihood **38**

D Derivations for AIS **41**
D.1 Intermediate probabilities 41
D.2 Partition function of initial distribution 41

E Free energy **42**

1 Introduction

The term structure of interest rates plays an important role for many businesses, governments and other institutions around the world, and being able to understand and model the term structure is sometimes crucial. Institutions such as banks, investment funds and insurance companies with a large book of interest rate dependent assets rely heavily on models of the term structure to price, hedge and manage the risk of these assets.

To estimate the future value of an asset or a portfolio that depend on interest rates, you usually need to estimate what the term structure might look like in the future. Most classical term structure models are not adequate to do this for longer time horizons, so scenario analysis is often used instead, where a few interesting scenarios are evaluated. However, designing scenarios is an art and to estimate the risk associated with a scenario it must have a probability.

Most of the existing literature on term structure modeling is based on stochastic calculus. Affine short rate models are common, such as the Vasicek model, the Cox–Ingersoll–Ross model, the Ho–Lee model and the Hull–White model, where the short rate is modeled as a stochastic process with certain properties. Even though the model parameters are calibrated to market data, the models are designed by humans and can only capture a few desired properties.

This paper will use a different approach to term structure modeling, based on machine learning. Machine learning uses statistical techniques to learn useful structures of a data set without being explicitly programmed. Thanks to the constant increase in computing power can complex nonlinear relations of large data sets be modeled in a reasonable time. In recent years have machine learning become a very popular topic and shown amazing results in a wide variety of fields, but there is a lack of research in the field of term structure modeling. The purpose of this paper is to broadly investigate if Gaussian restricted Boltzmann machines can be used to model the term structure of interest rates and make one-day-ahead forecasts as well as generate plausible long term scenarios. A Gaussian restricted Boltzmann machine is chosen since it is a generative model with a rich theoretical foundation.

1.1 Previous work

The number of topics related to the term structure of interest rates is large and the amount of literature is vast. Two popular approaches to term structure modeling, discussed in [14], are no arbitrage models and equilibrium models. The models can further be classified as one-factor models or multi-factor models depending on the number of stochastic factors. A popular class of models are the affine one factor short-rate models, where the short rate is the stochastic factor, usually modeled by a drift term and a stochastic volatility term.

The focus of no arbitrage models is to fit a term structure at one point in time such that no arbitrage opportunities exists. The term structure of today is an input to the model. The drift term is usually a function of both the short rate and time to match today's term structure, and the volatility term is a constant

or a function of time to match today's volatility term structure. Since the drift and the volatility are functions of time fitted primarily to today's term structure, tend the fit of the term structure to disappear with time. An example of a no arbitrage model is the Hull-White model, introduced in [15].

The focus of equilibrium models is to model the dynamics of the short rate and construct the term structure based on assumptions about the risk premia. The term structure of today is an output of the model. The drift term is usually a function of the short rate, usually to obtain mean reversion, and the volatility term is a constant or a function of the short rate. These models usually cannot fit today's term structure perfectly, but the predictions tend to be better than for the no-arbitrage models. Examples of equilibrium models are the Vasicek model and the Cox–Ingersoll–Ross model, introduced in [26] and [4] respectively.

There exist different methods to assess the quality of forecasts, as discussed in [6]. One method is to look at the mean squared forecast error (MSFE) or the root mean squared forecast error (RMSFE). The errors can also be compared to some naive forecasting method such as assuming a random walk, and thus forecast no change, or base the forecast on the last observed change.

These classical affine models tend to generate poor forcasts of the term structure. In a study by Duffee is it shown that assuming a random walk tend to generate better forecasts both in-sample and out-of-sample for horizons of three months, six months and one year, see [7]. Similar results are also shown in a paper by Velásquez-Giraldo and Restrepo-Tobón for horizons of one day and five days out-of-sample, see [27].

More advanced models have shown better results with regards to forecasting, such as a no-arbitrage type of model based on the Nelson-Siegel function, developed by Christensen, Diebold and Rudebusch in [3]. This model outperforms the random walk benchmark out-of-sample for horizons of 6 months and 12 months. In a paper by Luo, Han and Zhang are three other extensions of the Nelson-Siegel function evaluated, which also outperform the random walk benchmark for a horizon of 21 days out-of-sample, see [18]. However, the one-day-ahead forecasts out-of-sample are slightly outperformed by the random walk benchmark.

In a more recent paper by Engle and Siriwardane is a forecasting method based on an econometric approach developed, see [8]. The model is a reduced rank vector autoregression with time varying volatilities and correlations.

Gaussian restricted Boltzmann machines have shown promising results regarding forecasting of financial time series when stacked and combined with feed-forward neural networks, see for example [13]. In that kind of approach are the restricted Boltzmann machines used to find good features of the data in an unsupervised way, and then is the feed-forward network used to fine tune the whole network in a supervised way. The success of this type of network indicate that the restricted Boltzmann machine can model useful features of financial time series.

No papers are found where the focus is to model the probability distribution of a time series with a single Gaussian restricted Boltzmann machine, such as is done in this thesis. Similar approaches have though been used with mixed results

on for example images, see [20]. One weakness with the restricted Boltzmann machine that has been shown, is a difficulty to model dependencies between the input nodes. An extended version called mean-covariance restricted Boltzmann machine tries to overcome this problem by modeling a non-diagonal covariance structure, see [11]. The way this problem will be handled in this thesis is by using a PCA transformation of the input data.

1.2 Objective

The objectives of this thesis are to:

- Investigate if Gaussian restricted Boltzmann machines can be used to make one-day-ahead forecasts of the term structure. Compare the results with simple benchmarks.
- Investigate if the one-day-ahead forecasts can be iterated to generate plausible long term scenarios. Compare the results with simple benchmarks.
- Investigate the effects of PCA as data preprocessing.

1.3 Scope and limitations

Since no previous studies that are using restricted Boltzmann machines as stated in Section 1.2 are found, the main focus of this thesis is to broadly explore the possibilities and lay the foundation for future work. Some performance investigation is done, such as comparing a wide range of hidden units and comparing different data preprocessings, but model optimization is not a main focus of the thesis. Deep architectures will not be investigated since it would make the thesis too extensive.

1.4 Disposition

This thesis is structured as follows. Section 2 contains some financial background, with a brief introduction to bonds, yields and the term structure of interest rates. Section 3 begins with an overview of artificial neural networks, which is one way to view restricted Boltzmann machines. Then follows a more in depth review of the theory behind graphs, graphical models and Markov random fields, which is the theoretical foundation of restricted Boltzmann machines. Then follows a description of the restricted Boltzmann machine, followed by training concepts such as maximum likelihood estimation, gradient ascent, Gibbs sampling, contrastive divergence, annealed importance sampling and principal component analysis.

Section 4 starts with a description of the raw data and how it was preprocessed and structured to fit the modeling problem. Then follows a description of the tested architectures, how the training and evaluation were done and which hyperparameters that were used, how the ensemble was set up and which benchmark models that were used. Then follows a description of how the ability to make one-day-ahead forecasts and generate long term scenarios were evaluated.

In Section 5 are the results of the study presented, starting with the results regarding the one-day-ahead forecasts followed by the results regarding long term scenario generation. Section 6 discuss the results and how they compare to the benchmark models and previous research, as well as strengths and weaknesses with the study. Section 7 finally concludes the thesis and discuss ideas for future research.

2 Financial background

This section presents a short introduction to bonds, yields and the term structure of interest rates.

2.1 Bonds

A *bond* is a tradable loan agreement, where the issuer sells a contract promising the holder a predetermined payment schedule. Bonds can be issued by governments, private and public corporations and financial institutions. They can have different credit quality, which refers to the probability that the agreed upon payments are made on time. Bonds can be traded at exchanges or over the counter (OTC) and the holders can be pension funds and other financial institutions, central banks, corporations and private households.

The *face value* of a bond, also called *principal*, *nominal* or *par value*, is the amount the issuer borrows and the amount on which interest may be calculated. A *zero-coupon bond* promises a single payment, corresponding to the face value, at a future date called the *maturity date*. A *coupon bond* promises more than one payment until maturity, usually at regular time intervals such as quarterly, semi-annualy or annualy. The size of the coupon payment is determined by the face value, the coupon rate and the amortization principle. The cupon rate is often quoted as annual regardless of the payment frequency.

2.2 Yields

The *yield*, or *yield to maturity* (YTM), of a bond is the single discount rate at which the present value of all future payments equals the current price of the bond. To determine the yield, a compounding frequency needs to be specified. The convention can differ between markets, but often annual compounding is used. For a coupon bond with price B_t at time t and payments $Y_1, Y_2, ..., Y_n$ at times $T_1, T_2, ..., T_n$ respectively, the yield at time t using annual compounding, $\hat{y}_t^B$, is given by the equation

$$B_t = \sum_{T_i > t} Y_i (1 + \hat{y}_t^B)^{(T_i - t)}. \tag{2.1}$$

The yield at time t of a zero coupon bond with maturity time T, also called *zero-coupon yield*, *zero-coupon rate* or *spot rate* for time T, is given by

$$B_t^T = Y_T(1 + \hat{y}_t^T)^{(T-t)}, \tag{2.2}$$

where B_t^T is the price at time t and Y_T is the face value. Rearranged, the zero-coupon yield is explicitly given as

$$\hat{y}_t^T = (\frac{B_t^T}{Y_y^T})^{-\frac{1}{T-t}} - 1. \tag{2.3}$$

2.3 Term structure of interest rates

The *term structure of interest rates*, also known as the *yield curve*, shows yields or interest rates as a function of maturity for bonds with similar credit quality. When it is based on zero-coupon yields, it is called the *zero-coupon yield curve*. Most of the traded bonds are coupon bonds, but it is possible to extract or estimate zero coupon yields from the prices of the coupon bonds. A method based on arbitrage-free pricing is bootstrapping and methods based on regression are for example cubic splines and Nelson–Siegel parametrization.

For a more comprehensive review of bonds, yields and the term structure of interest rates, see for example [21].

3 Mathematical background

This chapter begins with a general overview of artificial neural networks. Then are graphs and graphical models presented, with an emphasis on Markov random fields, which are the theoretical foundation of restricted Boltzmann machines. Then follows a description of the restricted Boltzmann machine, followed by training concepts such as maximum likelihood estimation, gradient ascent, Gibbs sampling, contrastive divergence, annealed importance sampling and principal component analysis.

3.1 Artificial neural networks

Artificial neural networks (ANNs) are computational models inspired by biological neural networks of living beings. They are based on a set of processing units called *artificial neurons*, which are analogous to biological neurons. The neurons can be connected in different ways, where the connections correspond to biological synapses.

Artificial neural networks are often structured in *layers*. In general they have an *input layer* that receives external data, one or several *hidden layers* that extract patterns and an *output layer* that produces the final output of the network. Networks with many hidden layers are called *deep networks*. The *architecture* of an artificial neural network defines how the neurons and the connections are arranged.

Each neuron, except neurons in the input layer, receives input data which is a weighted sum of outputs from connected neurons. Then an *activation function* is applied, which gives the output of the neuron. The activation function is usually nonlinear and can be stochastic.

Artificial neural networks can model different kinds of nonlinear relations by varying the architecture and using different nonlinear activation functions. Artificial neural networks can for example be used for universal curve fitting, pattern recognition and classification, data clustering, prediction and as associative memory, see [5]. Restricted Boltzmann machines are stochastic generative neural networks, i.e. they have stochastic activation functions and models the joint probability distribution over the input data. They can also be regarded as probabilistic graphical models, which provide useful learning algorithms and theoretical results, neatly summarized in [9].

3.2 Graphs

A *graph* is a set of *vertices*, also called *nodes* or *points*, which are connected by *edges*, also called *lines* or *arcs*. It can be represented as an ordered pair $G = (V, E)$, where V is the set of vertices and E is the set of edges. An edge consists of a pair of vertices in V. The pair of vertices can be ordered or unordered, corresponding to *directed* and *undirected graphs* respectively. The model used in this thesis will be based on undirected graphs.

Below is a list of concepts related to undirected graphs, as defined in [9], that will be used in the sequel.

- If two vertices v and w are connected with an edge, i.e. $\{v, w\} \in E$, they are *neighbours* or *adjecent* to each other. Thus, the *neighbourhood* of a vertex v is the set of vertices connected to v, which can be written as $\mathcal{N}_v = \{w \in V : \{v, w\} \in E\}$.
- A *clique* is a subset of vertices such that there exists an edge between all pair of vertices in the subset, i.e. the set of vertices in a clique is fully connected. A clique is called *maximal* if it is not possible to include any other vertices from the graph without the clique ceasing to be a clique. The set of all maximal cliques will be denoted $\mathcal{C}$.
- A sequence of vertices $v_1, v_2, ..., v_n \in V$ with $\{v_i, v_{i+1}\} \in E$ for $i = 1, ..., n-1$ is a *path* from v_1 to v_n.
- A set $\mathcal{V} \subset V$ *separates* two vertices $v \notin \mathcal{V}$ and $w \notin \mathcal{V}$ if every path from v to w contains a vertex from $\mathcal{V}$.

3.2.1 Probabilistic graphical models

In a *probabilistic graphical model*, or simply a *graphical model*, do each vertex of a graph represent a random variable and the edges represents probabilistic relationships between these random variables. The graph can then describe conditional dependence and independence properties between the random variables

and how the joint distribution over all random variables can be factorized into factors that each only depend on a subset of the variables.

Two main types of graphical models are *Bayesian networks* and *Markov random fields*. Bayesian networks are acyclical and have directed edges between the vertices, whereas Markov random fields (MRF) can be cyclical and have undirected edges. This makes them able to represent different kinds of dependencies. Restricted Boltzmann machines are a type of Markov random field.

3.2.2 Markov random fields

Given an undirected graph $G = (V, E)$, a sequence of random variables $\boldsymbol{X} = (X_v)_{v \in V}$ forms a Markov random field with respect to G if the joint probability distribution $p(\boldsymbol{X} = \boldsymbol{x})$ satisfies the *local Markov property*. If the probability distribution is strictly positive for all $\boldsymbol{x}$, then the local Markov property is equivalent to the *pairwise Markov property* and the *global Markov property*. The Markov properties are defined as:

- Pairwise Markov property: Two non-adjacent variables are conditionally independent given all the other variables, i.e. if $\{v, w\} \notin E$, then $p(x_v, x_w|(x_t)_{t \in V \setminus \{v,w\}}) = p(x_v|(x_t)_{t \in V \setminus \{v,w\}})p(x_w|(x_t)_{t \in V \setminus \{v,w\}})$ for all $\boldsymbol{x}$.
- Local Markov property: For all $v \in V$, the random variable X_v is conditionally independent of all other variables given its neighborhood, i.e. $\forall v \in V$ and $\forall \boldsymbol{x}$, $p(x_v|(x_w)_{w \in V \setminus \{v\}}) = p(x_v|(x_w)_{w \in \mathcal{N}_v})$.
- Global Markov property: Two subsets of variables, A and B, are conditionally independent given a third separating subset S, i.e. $\forall \boldsymbol{x}$, $p((x_a)_{a \in A}|(x_t)_{t \in S \cup B}) = p((x_a)_{a \in A}|(x_t)_{t \in S})$.

To determine the Markov properties of a probability distribution can sometimes be hard. A theorem that provides a relation between the factorization properties of the joint probability distribution and the Markov properties is the Hammersley-Clifford theorem:

Theorem 1. *A strictly positive distribution p satisfies the Markov properties with respect to an undirected graph G if and only if p factorizes over G.*

A distribution *factorizes over an undirected graph G* with maximal cliques $\mathcal{C}$ if it can be written as

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C), \tag{3.1}$$

where ψ_C is a potential function for clique C. It can be any non-negative function. Z is the *partition function* which ensures that the probability distribution integrates to one. It is given by

$$Z = \int_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C) d\boldsymbol{x}, \tag{3.2}$$

where $\int_{\boldsymbol{x}} d\boldsymbol{x}$ refers to the integral over all possible values of $\boldsymbol{x}$, i.e. $\int_{\boldsymbol{x}} d\boldsymbol{x} = \int_{x_1} \int_{x_2} ... \int_{x_n} dx_1 dx_2 ... dx_n$ where n is the number of components of $\boldsymbol{x}$. Since ψ_C is strictly positive $\forall C \in \mathcal{C}$, the probability can be rewritten as

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C) = \frac{1}{Z} e^{\sum_{C \in \mathcal{C}} \ln \psi_C(\boldsymbol{x}_C)} = \frac{1}{Z} e^{-E(\boldsymbol{x}_C)}, \tag{3.3}$$

where $E = \sum_{C \in \mathcal{C}} \ln \psi_C(\boldsymbol{x}_C)$ is called the *energy function*. This is known as the *Boltzmann distribution* or *Gibbs distribution*.

For a more comprehensive review of graphs and graphical models, see for example [2] and [22].

3.3 Restricted Boltzmann machines

A *restricted Boltzmann machine* (RBM) is a Markov random field where the graph is *bipartite*, i.e. the vertices can be divided into two classes such that every edge has a vertex in each class. This bipartite property is what distinguishes restricted Boltzmann machines from "unrestricted" *Boltzmann machines*, which can be fully connected. The restricted connectivity allows for efficient learning algorithms based on for example block Gibbs sampling.

Restricted Boltzmann machines can also be viewed as stochastic artificial neural networks, where each vertex corresponds to a stochastic unit/neuron and each edge corresponds to a synaptic connection. In the sequel the term *unit* and *connection* will be used when referring to the random variables and their interactions.

The two classes of units are called *visible units* and *hidden units*. The visible units $\boldsymbol{V} = (V_1, V_2, ..., V_n)$ represents the observed data and the hidden units $\boldsymbol{H} = (H_1, H_2, ..., H_m)$ captures dependencies between the observed data. $\boldsymbol{V}$ and $\boldsymbol{H}$ are multivariate random variables with joint probability distribution given by the Gibbs distribution:

$$p(\boldsymbol{V} = \boldsymbol{v}, \boldsymbol{H} = \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})}. \tag{3.4}$$

(Note that $V_1, V_2, ..., V_n$ refers to random variables as opposed to V in section 3.2, which referred to a set of vertices). If the visible and hidden units take binary values, i.e. $\boldsymbol{v} \in \{0,1\}^n$ and $\boldsymbol{h} \in \{0,1\}^m$, the energy function is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i w_{ij} h_j, \tag{3.5}$$

where $a_i \in \mathbb{R}$ and $b_j \in \mathbb{R}$ are biases associated with the visible and hidden units respectively and $w_{ij} \in \mathbb{R}$ is the weight associated with the interaction

between visible unit V_i and hidden unit H_j. This is the standard restricted Boltzmann machine, sometimes called *Bernoulli-Bernoulli RBM*. A variant that can model continuous data is the *Gaussian-Bernoulli RBM*, sometimes simply called *Gaussian RBM*, where $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{h} \in \{0,1\}^m$ and where the visible units have normal distributions given the states of the hidden units and where the hidden units have Bernoulli distributions given the values of the visible units. The energy function is then given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j, \tag{3.6}$$

where σ_i is the standard deviation of visible unit i given the states of the hidden units.

Using these energy functions, the Gibbs distribution will factorize into factors that only depend on the maximal cliques, which ensures that the restricted Boltzmann machine is a Markov random field according to Theorem 1. Thus, since the restricted Boltzmann machine only have connections between the two layers and no connections between units in the same layer, all variables in each layer will be independent of each other conditioned on the values of the other layer, i.e.

$$\begin{aligned} p(\boldsymbol{v}|\boldsymbol{h}) &= \prod_{i=1}^{n} p(v_i|\boldsymbol{h}) \\ p(\boldsymbol{h}|\boldsymbol{v}) &= \prod_{j=1}^{m} p(h_j|\boldsymbol{v}). \end{aligned} \tag{3.7}$$

The conditional distributions are given by

$$\begin{aligned} p(v_i|\boldsymbol{h}) &= \mathcal{N}(v_i|a_i + \sum_j w_{ij} h_j, \sigma_i^2) \\ p(H_j = 1|\boldsymbol{v}) &= \text{sig}(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}), \end{aligned} \tag{3.8}$$

where $\mathcal{N}(\,\cdot\,|\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2 and $\text{sig}(\,\cdot\,)$ denotes the standard logistic function, also called the sigmoid function. The derivations are found in Appendix B.

Unsupervised learning of a restricted Boltzmann machine corresponds to adjusting the parameters of the energy function, denoted as $\theta = (\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})$, such that the marginal distribution $p(\boldsymbol{v}|\theta)$ is a good representation of the unknown distribution that has generated the observed data $\mathcal{D} = (\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_N)$, where N is the number of observations. The parameters σ_i for $i = 1, 2, ..., N$ are usually predefined. One way to estimate the parameters is by maximum likelihood estimation.

3.3.1 Maximum likelihood estimation

Maximum likelihood estimation (MLE) is a general method to estimate the parameters of a statistical model by maximizing the likelihood (or log-likelihood), given the observed data. The log-likelihood of a single observation $\boldsymbol{v}$ of a Gaussian Restricted Boltzmann machine is given by

$$\begin{aligned} \ln \mathcal{L}(\theta|\boldsymbol{v}) = \ln p(\boldsymbol{v}|\theta) = \ln \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} = \\ = \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln Z = \\ = \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln \int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} d\boldsymbol{v}, \end{aligned} \tag{3.9}$$

where $\sum_{\boldsymbol{h}}$ refers to the sum over all possible values of $\boldsymbol{h}$, i.e. $\sum_{h_1} \sum_{h_2} ... \sum_{h_m}$. The log-likelihood of the whole data set $\mathcal{D}$ is given by

$$\ln \mathcal{L}(\theta|\mathcal{D}) = \ln \prod_{k=1}^{N} \mathcal{L}(\theta|\boldsymbol{v}_k) = \sum_{k=1}^{N} \ln \mathcal{L}(\theta|\boldsymbol{v}_k). \tag{3.10}$$

In general it is not possible to analytically maximize the log-likelihood of a Markov random field, so numerical methods such as gradient ascent are used.

3.3.2 Gradient ascent

Gradient ascent is an optimization algorithm that tries to find the local maximum of a function by iteratively moving in the direction of the gradient. Applied to the log-likelihood function, the update rule is give by

$$\theta^{(t+1)} = \theta^{(t)} + \eta \frac{\partial}{\partial \theta^{(t)}} (\ln \mathcal{L}(\theta^{(t)}|\mathcal{D})), \tag{3.11}$$

where $\eta > 0$ is the *learning rate*. The update rule is iterated until some convergence criterion is satisfied or until some maximum number of iterations is reached. The initial parameter $\theta^{(0)}$ is set to some predefined value.

If the whole data set $\mathcal{D}$ is used in each iteration, the method is called *batch gradient ascent*. If one data point is used in each iteration, the method is called *stochastic gradient ascent* or *on-line gradient ascent*. Intermediate methods where a few data points are used in each iteration is called *mini-batch gradient ascent*. It is usually more effective to use one or a few data points in each iteration than to use the whole data set, see in [9] and [2].

The gradient of the log-likelihood of a single observation of a Gaussian restricted Boltzmann machine is given by

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial \theta} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta} + \int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta} d\boldsymbol{v} = \\ &= -\mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v})}[\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta}] + \mathbb{E}_{p(\boldsymbol{v},\boldsymbol{h})}[\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta}].\end{aligned} \tag{3.12}$$

The derivation is found in Appendix C, along with the derivations of the derivatives with respect to the components of θ:

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial w_{ij}} &= p(H_j = 1|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} - \int_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_j = 1|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} d\boldsymbol{v} \\ \frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial a_i} &= \frac{v_i - a_i}{\sigma_i^2} - \int_{\boldsymbol{v}} p(\boldsymbol{v}) \frac{v_i - a_i}{\sigma_i^2} d\boldsymbol{v} \\ \frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial b_j} &= p(H_j = 1|\boldsymbol{v}) - \int_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_j = 1|\boldsymbol{v}) d\boldsymbol{v}.\end{aligned} \tag{3.13}$$

The integral over $\boldsymbol{v}$ is usually intractable, so the expectations will be approximated using samples based on Gibbs sampling, which is a Markov chain Monte Carlo (MCMC) method.

3.3.3 Gibbs sampling

Gibbs sampling is a *Markov chain Monte Carlo* (MCMC) method that can be used to generate an approximate sequence of samples from a joint distribution $p(\boldsymbol{x}) = p(x_1, x_2, ..., x_n)$. Let $\boldsymbol{x}^{(i)}$ denote the ith sample, then the general Gibbs sampling procedure, starting at $i = 0$ with any $\boldsymbol{x}^{(0)}$, is as follows, see [2]:

1. Begin with a vector $\boldsymbol{x}^{(i)}$.
2. The next sample $\boldsymbol{x}^{(i+1)}$ is obtain by sampling one component $x_j^{(i+1)}$ of $\boldsymbol{x}^{(i+1)}$ at the time from the conditional distribution $p(x_j^{(i+1)}|x_1^{(i+1)}, ..., x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, ..., x_n^{(i)})$. The sampling order can be stochastic or follow some predefined schedule.
3. Repeat step 1 and 2 k times, where k is the number of samples to generate.

The generated sequence consists of n-dimensional samples whose joint distribution converges to a *stationary distribution* when $k \to \infty$, regardless of the initial vector $\boldsymbol{x}^{(0)}$, if the Markov chain is aperiodic and irreducible. For a Markov random field it holds that the Markov chain defined by the Gibbs sampling procedure (the so called Gibbs chain) is aperiodic and irreducible and that the stationary distribution is the joint distribution $p(\boldsymbol{x}) = p(x_1, x_2, ..., x_n)$. The *mixing time*, i.e. the time until the Markov chain is "close" to the stationary distribution, will depend on the model parameters.

For any subset of variables, the generated samples will asymptotically follow the corresponding marginal distribution of p, and if some subset of variables A are held fixed during the Gibbs sampling, the distribution of the remaining variables B will asymptotically follow the conditional distribution $p(B|A)$.

Instead of sampling one component of $\boldsymbol{x}$ at a time in the Gibbs sampling procedure, it is possible to sample many components at the time, called a *block*, from the corresponding conditional joint distribution. This is called *block Gibbs sampling*. Since all variables in each layer of a restricted Boltzmann machine are conditionally independent, one can sample all visible units given the hidden units simultaneously from $p(\boldsymbol{v}|\boldsymbol{h})$ and vise versa. Each Gibbs sample can thus be obtained in just two steps, see [9].

3.3.4 Contrastive Divergence

Obtaining samples from the stationary distribution of a Gibbs chain can require many iterations. However, it has been shown that only one or a few iterations can give good results when training a model if the Markov chain starts at a training example $\boldsymbol{v}^{(0)}$, see [12]. This method is called k-step *contrastive divergence* (CD-k), where k is the number of iterations.

Another approximation that is often done together with contrastive divergence is to approximate the expectation in the second term of the gradient of the log-likelihood 3.12 with a single sample, see [1]. A Python style pseudocode implementation of this approximation together with the contrastive divergence algorithm for a Gaussian restricted Boltzmann machine is given by

Algorithm 1: k-step contrastive divergence

Input: Training data $\mathcal{D}$

Output: Approximations of $\Delta \boldsymbol{W}$, $\Delta \boldsymbol{a}$ and $\Delta \boldsymbol{b}$

for $\boldsymbol{v}$ *in* $\mathcal{D}$ **do**
  $\boldsymbol{v}^{(0)} \leftarrow \boldsymbol{v}$
  for i = *0, ..., k* **do**
    sample $\boldsymbol{h}^{(i)} \sim p(\boldsymbol{h}|\boldsymbol{v}^{(i)})$
    sample $\boldsymbol{v}^{(i+1)} \sim p(\boldsymbol{v}|\boldsymbol{h}^{(i)})$
  end
  for $i = 1, 2, ..., n$ **do**
    for $j = 1, 2, ..., m$ **do**
      $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_j = 1|\boldsymbol{v}^{(0)})\frac{v_i^{(0)}}{\sigma_i^2} - p(H_j = 1|\boldsymbol{v}^{(k)})\frac{v_i^{(k)}}{\sigma_i^2}$
    end
  end
  for $i = 1, 2, ..., n$ **do**
    $\Delta a_i \leftarrow \Delta a_i + \frac{v_i^{(0)} - a_i^{(0)}}{\sigma_i^2} - \frac{v_i^{(k)} - a_i^{(k)}}{\sigma_i^2}$
  end
  for $j = 1, 2, ..., m$ **do**
    $\Delta b_j \leftarrow \Delta b_j + p(H_j = 1|\boldsymbol{v}^{(0)}) - p(H_j = 1|\boldsymbol{v}^{(k)})$
  end
end
return $\Delta \boldsymbol{W}$, $\Delta \boldsymbol{a}$, $\Delta \boldsymbol{b}$

The code can often run faster by using vector operations instead of the loops.

3.3.5 Annealed importance sampling

One way to monitor the progress of the unsupervised learning and to evaluate the final model, is by calculating the log-likelihood on the training data or on some unseen validation data. However, this requires calculating the partition function, which is usually analytically intractable. One way to estimate the partition function is by using annealed importance sampling, which is an extension of simple importance sampling.

Simple importance sampling
Suppose we have two probability distributions with density functions $p_A(\boldsymbol{x}) = \frac{p_A^*(\boldsymbol{x})}{Z_A}$ and $p_B(\boldsymbol{x}) = \frac{p_B^*(\boldsymbol{x})}{Z_B}$, where p^* denotes the unnormalized probability density. If $p_A(\boldsymbol{x}) \neq 0$ whenever $p_B(\boldsymbol{x}) \neq 0$, the ratio of the partition functions can be written as

$$\frac{Z_B}{Z_A} = \frac{\int p_B^*(\boldsymbol{x})d\boldsymbol{x}}{Z_A} = \int \frac{p_B^*(\boldsymbol{x})}{p_A^*(\boldsymbol{x})} p_A(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_{p_A(\boldsymbol{x})}[\frac{p_B^*(\boldsymbol{x})}{p_A^*(\boldsymbol{x})}]. \tag{3.14}$$

If independent samples can be drawn from p_A, an unbiased estimate of the ratio can be obtained using the following Monte Carlo approximation:

$$\frac{Z_B}{Z_A} \approx \frac{1}{M}\sum_{i=1}^{M} \frac{p_B^*(\boldsymbol{x}^{(i)})}{p_A^*(\boldsymbol{x}^{(i)})} := \frac{1}{M}\sum_{i=1}^{M} w^{(i)}, \tag{3.15}$$

where $\boldsymbol{x}^{(i)}$ is sampled from p_A and M is the number of samples. This method is called *simple importance sampling* (SIS), and if the distribution p_A is chosen such that Z_A can be calculated, it is possible to obtain an unbiased estimate of Z_B. However, if p_A and p_B are not similar enough, the estimate can have very high variance, see [24].

Annealed importance sampling
Annealed importance sampling (AIS) is a method used to avoid the high variance that can occur with simple importance sampling. Suppose we can define a sequence of intermediate distributions $p_A = p_0, p_1, ..., p_K = p_B$ that satisfy the following conditions:

- $p_k(\boldsymbol{x}) \neq 0$ whenever $p_{k+1}(\boldsymbol{x}) \neq 0$, for $k = 0, 1, ..., K-1$.
- The unnormalized probability p_k^* can be evaluated for $k = 0, 1, ..., K$.
- A sample $\boldsymbol{x}'$ can be drawn given $\boldsymbol{x}$ using a Markov chain transition operator $T_k(\boldsymbol{x}' \leftarrow \boldsymbol{x})$ that leaves p_k invariant, i.e. $\int T_k(\boldsymbol{x}' \leftarrow \boldsymbol{x})p_k(\boldsymbol{x})d\boldsymbol{x} = p_k(\boldsymbol{x}')$, for $k = 1, 2, ..., K-1$.
- Samples, preferably independent, can be drawn from $p_A(\boldsymbol{x})$.

The sequence is usually constructed as

$$p_k^*(\boldsymbol{x}) = p_A^*(\boldsymbol{x})^{1-\beta_k} p_B^*(\boldsymbol{x})^{\beta_k}, \tag{3.16}$$

where $k = 0, 1, ..., K$ and $0 = \beta_0 < \beta_1 < ... < \beta_K = 1$. The importance weight $w_{AIS}^{(i)}$ is given by

$$w_{AIS}^{(i)} = \frac{p_1^*(\boldsymbol{x}_1)}{p_0^*(\boldsymbol{x}_1)} \frac{p_2^*(\boldsymbol{x}_2)}{p_1^*(\boldsymbol{x}_2)} ... \frac{p_K^*(\boldsymbol{x}_K)}{p_{K-1}^*(\boldsymbol{x}_K)}, \tag{3.17}$$

where $\boldsymbol{x}_1$ is sampled from $p_0 = p_A$ and $\boldsymbol{x}_k$ is sampled given $\boldsymbol{x}_{k-1}$ using $T_{k-1}(\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1})$ for $k = 2, 3, ..., K$. The estimate of the ratio of the partition functions can then be obtained using the same Monte Carlo approximation as in 3.15:

$$\frac{Z_B}{Z_A} \approx \frac{1}{M} \sum_{i=1}^{M} w_{AIS}^{(i)}. \tag{3.18}$$

The variance of the estimate is proportional to $\frac{1}{MK}$ if K is sufficiently large, and if the samples from p_A are independent, the variance can be estimated using

$$\text{Var}(\frac{1}{M} \sum_{i=1}^{M} w_{AIS}^{(i)}) \approx \frac{\hat{s}^2}{M}, \tag{3.19}$$

where $\hat{s}^2$ is the sample variance of the weights $w_{AIS}^{(i)}$, see [24].

AIS for Gaussian RBMs

An initial distribution $p_A(\boldsymbol{v})$ that has been proposed when estimating the partition function of a Gaussian RBM is the marginal of the intractable distribution $p_B(\boldsymbol{v}, \boldsymbol{h})$ of the Gaussian RBM but with the weight matrix $\boldsymbol{W}$ and the bias vector of the hidden units $\boldsymbol{b}$ set to zero. [17] Thus, the parameters of the intractable distribution p_B is given by $\theta_B = (\boldsymbol{W}, \boldsymbol{a}^B, \boldsymbol{b})$ and the parameters of the initial distribution p_A is given by $\theta_A = (\boldsymbol{0}, \boldsymbol{a}^A, \boldsymbol{0})$. The two distributions also share the same variances of the visible units σ_i, where $i = 1, 2, ..., n$.

Intermediate distributions based on the following energy functions have also been proposed, see [17]:

$$\begin{aligned} E_{\beta_k}(\boldsymbol{v}, \boldsymbol{h}) &= \beta_k E(\boldsymbol{v}, \boldsymbol{h}; \theta_B) + (1 - \beta_k) E(\boldsymbol{v}, \boldsymbol{h}; \theta_A) = \\ &= \beta_k (\sum_i \frac{(v_i - a_i^B)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j) \\ &\quad + (1 - \beta_k) \sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2} \end{aligned} \tag{3.20}$$

for $k = 0, 1, ..., K$.

The unnormalized intermediate probabilities are given by

$$p^*_{\beta_k}(\boldsymbol{v}) = e^{-\beta \sum_i \frac{(v_i - a_i^B)^2}{2\sigma_i^2} - (1-\beta) \sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2}} \prod_j (1 + e^{\beta_k (b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij})}). \quad (3.21)$$

The derivation is found in Appendix D.1. The transition operator is obtained from

$$\begin{aligned} p_{\beta_k}(H_j = 1|\boldsymbol{v}) &= \text{sig}(\beta_k(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij})) \\ p_{\beta_k}(v_i|\boldsymbol{h}) &= \mathcal{N}(v_i|\beta_k(a_i^B + \sum_j w_{ij} h_j) + (1 - \beta_k) a_i^A, \sigma_i^2), \end{aligned} \quad (3.22)$$

which are derived in the same way as the conditional distributions given by 3.8. Sampling from 3.22 gives a sample $\boldsymbol{v}'$ given $\boldsymbol{v}$, and it is not necessary to compute $T_{\beta_k}(\boldsymbol{v}' \leftarrow \boldsymbol{v})$, see [23].

Samples from p_A can be obtained by block Gibbs sampling. The samples will not be independent, so the variance estimate 3.19 can not be used.

The partition function of the tractable distribution, Z_A, is given by

$$Z_A = 2^m \prod_i \sqrt{2\pi\sigma_i^2}. \quad (3.23)$$

The derivation is found in Appendix D.2. Finally, the partition function of the intractable distribution, Z_B, can be estimated using annealed importance sampling.

3.4 Principal component analysis

Principal component analysis (PCA) is an orthogonal transformation of a data set of possibly correlated variables into a data set of uncorrelated variables, called *principal components*. The transformation is defined such that the first principal component accounts for the largest possible variance in the data, the second component accounts for the largest possible variance given that it is orthogonal to the first component and so forth. The number of principal components is equal to the smallest of the number of original variables and the number of data points minus one. If only a subset of the principal components are used to represent the data, PCA can be used for dimensionality reduction, data compression, feature extraction and data visualization, see [22].

Let the matrix $\boldsymbol{X} \in \mathbb{R}^{N \times n}$ represent a data set, where each row is an observation $\boldsymbol{x}_i \in \mathbb{R}^n$ and each column is a variable, and let the PCA transformation of $\boldsymbol{X}$ be denoted as $\boldsymbol{Z} \in \mathbb{R}^{N \times n}$, where each row is an observation $\boldsymbol{z}_i \in \mathbb{R}^n$ and each column is a principal component. The PCA transformation can be obtained by

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{U}, \quad (3.24)$$

where $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ is the orthogonal transformation matrix with the new basis vectors, called *principal component axes*, as columns.

One way to find the new basis vectors is by finding the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and the corresponding eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n$ of the sample covariance matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ of the data $\boldsymbol{X}$. The eigenvectors will be the new basis vectors and the eigenvalues will equal the variance of the data in the corresponding direction.

To avoid that variables with large variance dominates the first principal components, each variable can be scaled to have unit variance, see [16]. This corresponds to calculating the eigenvalues and eigenvectors on the correlation matrix instead of the covariance matrix. Let $\tilde{\boldsymbol{U}}$ denote the matrix with the eigenvectors of the correlation matrix as column and let $\tilde{\boldsymbol{L}}$ denote the diagonal matrix with the corresponding eigenvalues on the diagonal.

PCA can be used to obtain a data set with zero mean and unit covariance, which is called *whitening* or *sphering*. The whitened data can be obtained as

$$\tilde{\boldsymbol{z}}_i = \tilde{\boldsymbol{L}}^{-1} \tilde{\boldsymbol{U}}^T (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \tag{3.25}$$

for $i = 1, 2, ..., N$, where $\bar{\boldsymbol{x}}$ is the sample mean given by $\bar{\boldsymbol{x}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i$, see [2].

4 Method

This sections starts with information about the raw data, followed by a description of how the raw data was preprocessed and structured to fit the modeling problem. Then follows a description of the tested architectures, how the training and evaluation were done and which hyperparameters that were used. Then is the ensemble described, followed by the benchmark models. The section ends with a description of how the ability to make one-day-ahead forecasts and generate long term scenarios were evaluated.

4.1 Data

The raw data that were used consisted of daily quotes of the yield to maturity of Swedish government bonds and Treasury bills. The data covered the period 1993-01-01 to 2017-09-11, which corresponds to 6201 observations, and the following time to maturities were included: 1M, 3M, 6M, 2Y, 5Y, 7Y and 10Y. The data was provided by the central bank of Sweden, Sveriges Riksbank. Missing values were linearly interpolated. The raw data is plotted in Figure 1.

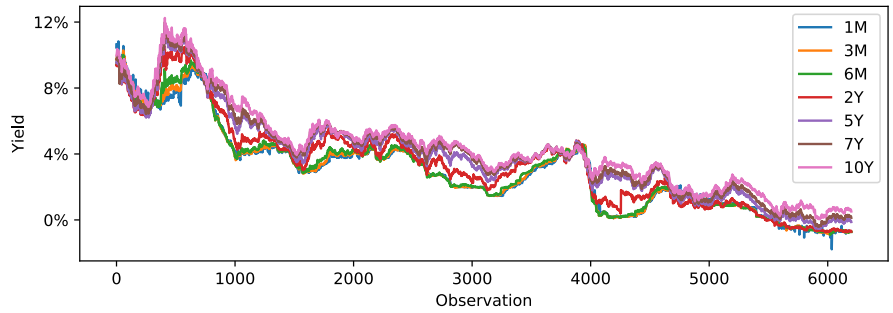

Figure 1: The raw yield data used in this thesis.

4.1.1 Data preprocessing

The raw data was randomly shuffled over the time dimension and divided into a training set, a validation set and a test set. The training set was used for training, the validation set was used for model selection, the test set was used to evaluate the selected models and the shuffling was done to avoid differences between the sets. The training set, the validation set and test set contained 4000, 1100 and 1100 observations respectively.

Three different preprocessings were compared in the thesis: the standardized raw data, a standardized PCA representation of the standardized raw data with all principal components included, and a standardized PCA approximation of the standardized raw data with a subset of the principal components included.

Let the yield to maturity at time t be denoted as $\boldsymbol{y}_t \in \mathbb{R}^7$ and the difference between the yield to maturity at time $t+1$ and the yield to maturity at time t be denoted as $\boldsymbol{x}_t = \boldsymbol{y}_{t+1} - \boldsymbol{y}_t \in \mathbb{R}^7$, where $t = 1, 2, ..., 6200$. Let $\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{y}}_t$ denote the standardized data, with components given by

$$\begin{aligned} \hat{x}_i &= \frac{x_i - \bar{x}_i}{s_i^x} \\ \hat{y}_i &= \frac{y_i - \bar{y}_i}{s_i^y}, \end{aligned} \tag{4.1}$$

where i is the component index, $\bar{x}_i$ and $\bar{y}_i$ are sample means of the training set and s_i^x and s_i^y are sample standard deviations of the training set.

The components of $\hat{\boldsymbol{y}}_t$ may be correlated, as well as the components of $\hat{\boldsymbol{x}}_t$. By using PCA, $\hat{\boldsymbol{y}}_t$ and $\hat{\boldsymbol{x}}_t$ can be transformed without loss of information into uncorrelated principal components, denoted as $\boldsymbol{z}_t^y$ and $\boldsymbol{z}_t^x$ respectively. This transformation might be useful since a limitation of the restricted Boltzmann machine is that the visible units are conditionally independent given the hidden states.

By discarding some of the principal components, dimensionality reduction is obtained. The table below shows how much of the variance that was cumulatively

explained by different number of principal components.

Component:	1st	2nd	3rd	4th	5th	6th	7th
Level $\hat{\boldsymbol{y}}$:	0.9749	0.9974	0.9994	0.9997	0.9999	0.9999	1
Return $\hat{\boldsymbol{x}}$:	0.6135	0.8216	0.9056	0.9480	0.9832	0.9933	1

Table 1: Cumulative percentage of the total variance explained by the principal components.

In the literature, it is common to use the first three principal components to explain the changes of the yield curve. The first component tend to explain parallel shifts, the second component tend to explain rotations and the third component tend to explain changes in the curvature, see [21]. Thus, let $\tilde{\boldsymbol{z}}_t^x$ denote the first three principal components of $\hat{\boldsymbol{x}}_t$. For simplicity, the first three components were also used to explain the level of the yield curve, so let $\tilde{\boldsymbol{z}}_t^y$ denote the first three principal components of $\hat{\boldsymbol{y}}_t$.

The three input sets, before standardization, that were compared in this study are given by

$$\begin{aligned} \boldsymbol{v}_t^1 &= (\boldsymbol{y}_t, \boldsymbol{x}_t) \in \mathbb{R}^{14} \\ \boldsymbol{v}_t^2 &= (\boldsymbol{z}_t^y, \boldsymbol{z}_t^x) \in \mathbb{R}^{14} \\ \boldsymbol{v}_t^3 &= (\tilde{\boldsymbol{z}}_t^y, \tilde{\boldsymbol{z}}_t^x) \in \mathbb{R}^6. \end{aligned} \tag{4.2}$$

Let $\hat{\boldsymbol{v}}_t^1$, $\hat{\boldsymbol{v}}_t^2$ and $\hat{\boldsymbol{v}}_t^3$ denote the standardized input sets, with components given by

$$\hat{v}_i = \frac{v_i - \bar{v}_i}{s_i}, \tag{4.3}$$

where i is the component index, $\bar{v}_i$ is the sample mean of the training set and s_i is the sample standard deviation of the training set. Thus, each component of the training data had zero mean and unit variance, while each component of the validation and test data may have had a slightly different mean and standard deviation.

4.2 Models and training

The models used in this thesis were restricted Boltzmann machines with Gaussian visible units and binary hidden units. The number of visible units equaled the dimension of the input vector, i.e. 14 or 6 depending on the preprocessing, and the number of hidden units was a hyperparameter that had to be tuned. The unstandardized input vectors are given by 4.2.

Since no previous studies that were using a similar model on a similar data set were found, a large span of hidden units was investigated. The following architectures, i.e. number of hidden units, were compared:

$$\mathcal{M} = (4, 8, 16, 32, 64, 128, 256). \tag{4.4}$$

Ideally, the remaining hyperparameters would also have been tuned, but since the training and evaluation of the models were too time consuming, the remaining hyperparameters were set to values that were mentioned in the literature and that performed reasonably well during initial testing. The initial learning rate was set to 0.01 and decreased by a constant factor each epoch down to 0.001. The batch size was set to 100 and the number of Gibbs step in the contrastive divergence algorithm was set to 1. The initial weights were sampled from a Gaussian distribution with zero mean and a variance of 0.01 and the biases were initialized to zero, as mentioned in [10].

There is no simple and universal way to evaluate unsupervised generative models. A popular measure is the likelihood, but it cannot be used to compare the different preprocessings of the data and it is usually analytically intractable for restricted Boltzmann machines. It can be estimated using annealed importance sampling, but this method sometimes fails to estimate the true likelihood, see [25]. A measure that fits the objectives of this thesis is the ability to reconstruct the returns given the levels, measured as a mean squared error. This type of reconstruction will also be used to make one-day-ahead forecasts, either with or without sampling. In the sequel will the phrases *return reconstruction* and *one-day-ahead forecast* thus be used interchangeably, and the phrase *reconstruction error* will refer to the mean squared error of the reconstructed normalized returns. Both the log-likelihood and the return reconstruction were evaluated during training.

The training was evaluated every 100th epoch by first calculating the log-likelihood on the training and validation set using annealed importance sampling with 100 samples, 1000 intermediate distributions with 100 evenly spaced betas from 0 to 0.5, 400 evenly spaced betas from 0.5 to 0.9 and 500 evenly spaced betas from 0.9 to 1, which are the same proportions as in [24]. A burn in period of length 100 was used before sampling from the initial distribution.

The ability to reconstruct the returns $\boldsymbol{x}_t$ given the levels $\boldsymbol{y}_t$ was also evaluated every 100th epoch. The samples of $\boldsymbol{x}_t$ were obtained by clamping the levels and alternately updating the hidden and visible units without sampling until convergence. The initial returns were set to random normal values with mean 0 and standard deviation 1. The mean squared error between the reconstructed normalized returns and the true normalized returns was then calculated.

The number of training epochs was set to 20000 and no early stopping was used. Each evaluated model were saved and the optimal training lengths were chosen afterwards based on either the log-likelihood or the reconstruction error on the validation set.

4.2.1 Ensemble

To investigate and potentially limit the effects of randomness that originate from the weight initialization and the training, an ensemble of 10 models were trained. Ideally, this would have been done for all tested architectures, but it was not

possible since the training and evaluation were too time consuming. Instead, it was done for the architecture that performed best with respect to return reconstruction on the validation set. The models in the ensemble were trained for 10000 epochs and evaluated every 100th epoch. Each evaluated model were saved and the optimal training lengths were chosen afterwards based on the reconstruction errors on the validation set.

4.2.2 Benchmarks

The results of the one-day-ahead forecasts and the scenario generation, described in Section 4.3.1 and 4.3.2, were compared to some simple benchmark models. Two benchmark models were used for the one-day-ahead forecasts: one that assumed a random walk and thus forecasted a zero change of the standardized returns and one vector autoregressive model of order 1, fitted to the standardized returns of the training set. Assuming a random walk on the standardized data corresponds to assuming a random walk with the trend of the training set on the non-standardized data. For the scenario generation, three probabilistic benchmark models were used. One Gaussian random walk with zero mean and the variance structure of the training set but with no covariance, one Gaussian random walk with zero mean and the full covariance structure of the training set and one PCA model based on the standardized training set with Gaussian principal components having the same mean and variance as the principal components of the standardized training set.

4.3 Evaluations

4.3.1 One-day-ahead forecasts

The one-day-ahead forecasts were obtained by reconstructing the returns given the levels without sampling, as described in Section 4.2. To evaluate the forecasts, the mean squared error between the forecasted normalized returns and the true normalized returns was calculated.

The one-day-ahead forecast error was calculated for each preprocessing and each architecture on both the validation and test set. The training length for each model were chosen based on the reconstruction error on the validation set.

The one-day-ahead forecast error was also calculated for each model in the ensemble on both the validation and test set. Then were the mean and variance estimated.

These results were then compared to the results from the benchmark models.

4.3.2 Long term scenario generation

The long term scenarios were generated by iterating the one-day-ahead forecasts, but with sampling. Each sample of returns $\boldsymbol{x}_t$ were obtained by clamping the current levels and alternately updating the hidden and visible units, each with random sampling, 10 times. The initial returns for each time step were set to

random normal values with mean 0 and standard deviation 1. The levels of the next time step were obtained by adding the sampled returns to the current levels, i.e. $\boldsymbol{y}_{t+1} = \boldsymbol{y}_t + \boldsymbol{x}_t$. From the sampled returns were the mean values of the training set subtracted to avoid catching the negative trend of the yields in the raw data, see Figure 1. It is not obvious how to handle the trend, but it is more of an economic topic that is outside the scope of this thesis. The length of the generated scenarios were set to 10000 days, starting at the last observation in the unshuffled data set, i.e. the yield curve at 2017-09-08.

For each preprocessing were scenarios generated from the models with the lowest reconstruction error and the highest log-likelihood on the validation set during training. From each model were 100 scenarios generated and the mean of these scenarios calculated.

For each model in the ensemble were 10 scenarios generated, i.e. a total of 100 scenarios. From these scenarios were the mean, 5th percentile and 95th percentile calculated.

The generated scenarios were then visually compared to the scenarios generated from the benchmark models.

The ability to make one year ahead forecasts was also evaluated in a more formal way, by estimating the mean squared forecast error. For each model in the ensemble were 10 scenarios of length 250 days, approximately 1 year, generated, i.e. a total of 100 scenarios. The mean of these scenarios were used to forecast the term structure. The trends of the generated scenarios were not removed, since it would make the forecasts worse. This procedure was used to forecast the last 750 yields of the unshuffled data, and then the mean squared error between the forecasted yields and the true yields was calculated. Since the raw data was shuffled before it was divided into training, validation and test sets, this is not a true out-of-sample forecast. Some of the values to forecast will be in the training and validation set.

The mean squared forecast errors were compared to the benchmark models.

5 Results

This section begins with a presentation of the results regarding the one-day-ahead forecasts, followed by the results regarding the long term scenario generation.

5.1 One-day-ahead forecasts

Model comparison
The purpose of this part was to evaluate a wide range of architectures and the effects of PCA as data preprocessing. The results in Table 2 were obtained from models trained on raw data and show the mean squared error (MSE) of normalized one-day-ahead returns.

Hidden	MSE valid	MSE test	Epochs
4	1.2962	1.3114	19300
8	1.2973	1.3833	6700
16	1.3004	1.3065	3500
32	1.3023	1.3099	6400
64	1.3040	1.3101	9400
128	1.3040	1.3083	9500
256	1.3035	1.3071	18900

Table 2: MSE of normalized one-day-ahead returns from models trained on raw data.

The results in Table 3 were obtained from models trained on PCA transformed data using all PCA components.

Hidden	MSE valid	MSE test	Epochs
4	1.3029	1.3031	1400
8	1.2976	1.3036	1400
16	1.2962	1.3002	1700
32	1.2967	1.3026	2300
64	1.2965	1.2985	3400
128	1.2968	1.2968	3000
256	1.2952	1.3044	6300

Table 3: MSE of normalized one-day-ahead returns from models trained on PCA transformed data with all components included.

The results in Table 4 were obtained from models trained on PCA transformed data using 3 PCA components.

Hidden	MSE valid	MSE test	Epochs
4	1.2994	1.3089	1400
8	1.2990	1.3055	2400
16	1.2989	1.3043	2400
32	1.2987	1.3054	4400
64	1.2983	1.3081	8900
128	1.2985	1.3060	14400
256	1.2984	1.3088	18600

Table 4: MSE of normalized one-day-ahead returns from models trained on PCA transformed data with 3 components included.

The MSE was higher on the test set than on the validation set for almost all models and data sets. The differences in MSE between the three data sets were small, but PCA transformed data performed in general slightly better than raw data, and using all PCA components performed slightly better than using only 3 components, at least on the validation set. There was no clear and general relation between the number of hidden nodes and the MSE.

Ensemble
The purpose of the ensemble was to get a rough estimate of the standard deviation of the one-day-ahead return MSE. The architecture that corresponded to the lowest one-day-ahead return MSE on the validation set, i.e. the architecture with 256 hidden nodes trained on PCA transformed data with all components included, was chosen to create the ensemble of 10 models. The mean and standard deviation of the one-day-ahead return MSE for the models in the ensemble are presented in Table 5.

	MSE valid	MSE test	Epochs
Mean	1.2956	1.3008	5570
SD	0.000619	0.002853	750

Table 5: Mean and standard deviation of the one-day-ahead return MSE for the models in the ensemble.

Benchmarks
The purpose of the benchmark models were to compare the one-day-ahead forecast ability of the Gaussian restricted Boltzmann machine to some simple models. The results of the benchmarks models, described in section 4.2.2, are shown in Table 6.

Model	MSE train	MSE valid	MSE test
Random walk	1.0	1.3050	1.3022
VAR(1)	1.0095	1.3194	1.3160

Table 6: MSE of normalized one-day-ahead returns for the benchmark models.

5.2 Long term scenario generation

In this section are the results from the generation of long term scenarios presented.

Model selection
The purpose of this part is to compare log-likelihood and return reconstruction as measures of ability to generate long term scenarios. The figures show the mean of 100 generated scenarios, based on the best performing models with respect to log-likelihood and reconstruction error from Section 5.1. The scenarios in Figure 2 are based on models trained on raw data.

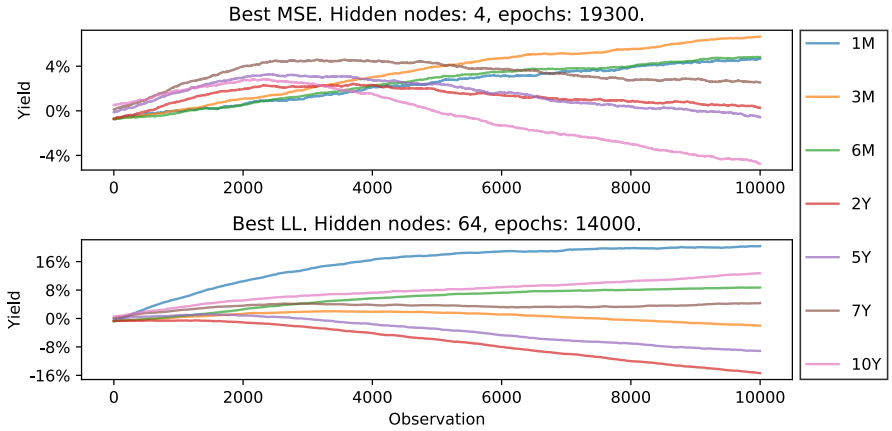

Figure 2: Mean of 100 generated scenarios based on the best performing models with respect to log-likelihood and reconstruction error. The models are trained on raw data.

Figure 3 show scenarios based on models trained on PCA transformed data with all components included.

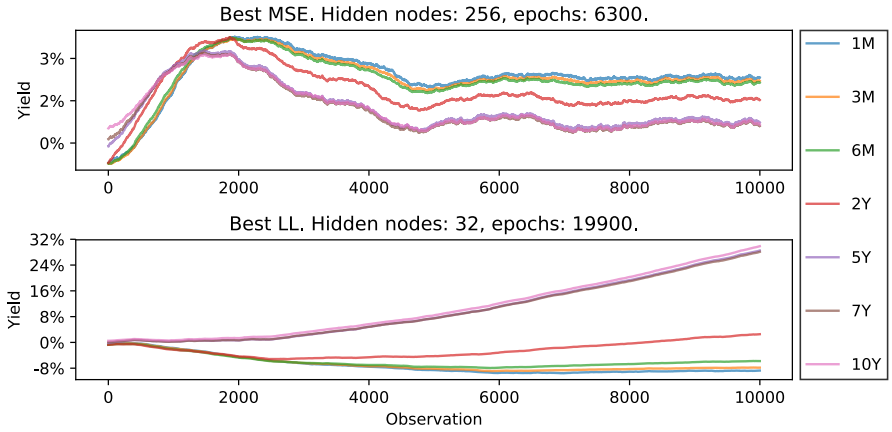

Figure 3: Mean of 100 generated scenarios based on the best performing models with respect to log-likelihood and reconstruction error. The models are trained on PCA transformed data with all components included.

Figure 4 show scenarios based on models trained on PCA transformed data with 3 components included.

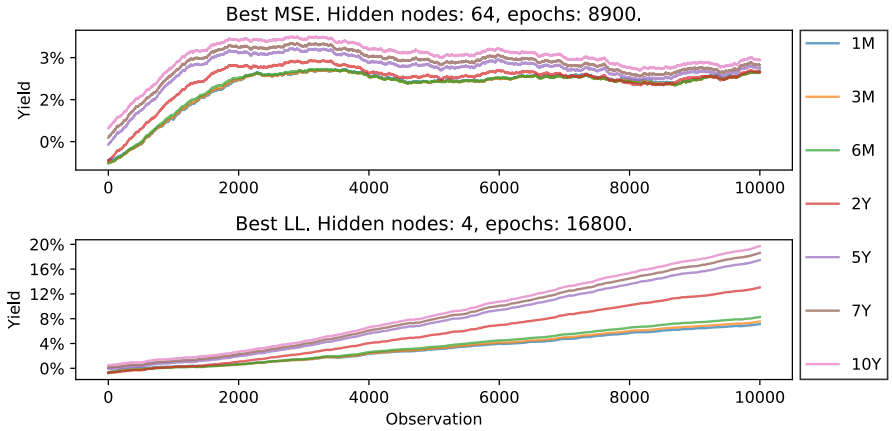

Figure 4: Mean of 100 generated scenarios based on the best performing models with respect to log-likelihood and reconstruction error. The models are trained on PCA transformed data with 3 components included.

The generated scenarios should ideally be drawn from a probability distribution similar to the unknown probability distribution of the raw data, but without the downward trend. Therefore, the generated scenarios should have a visual resemblance to the raw data, shown in Figure 1, but without the trend. But note that the plots above show mean values of 100 samples and not a single sample.

The generated yields from models selected based on log-likelihood tended to diverge and drift to extreme values, which was also the case for the yields generated from the model selected based on reconstruction error and trained on raw data. The yields generated from the models trained on PCA transformed data and selected based on reconstruction error seemed to hold together well and had mean values that resemble the raw data. However, for the yields generated from the model trained on PCA transformed data with all components included, the ordering of the yields seemed reversed.

Ensemble

The purpose of the ensemble was to eliminate potential effects of randomness in the model initialization and model training. The models in this ensemble were the same as in Section 5.1. In Figure 5 are the mean, 5th percentile and 95th percentile of the generated scenarios from the models in the ensemble plotted. From each model in the ensemble were 10 scenarios of length 10000 generated, i.e. a total of 100 scenarios.

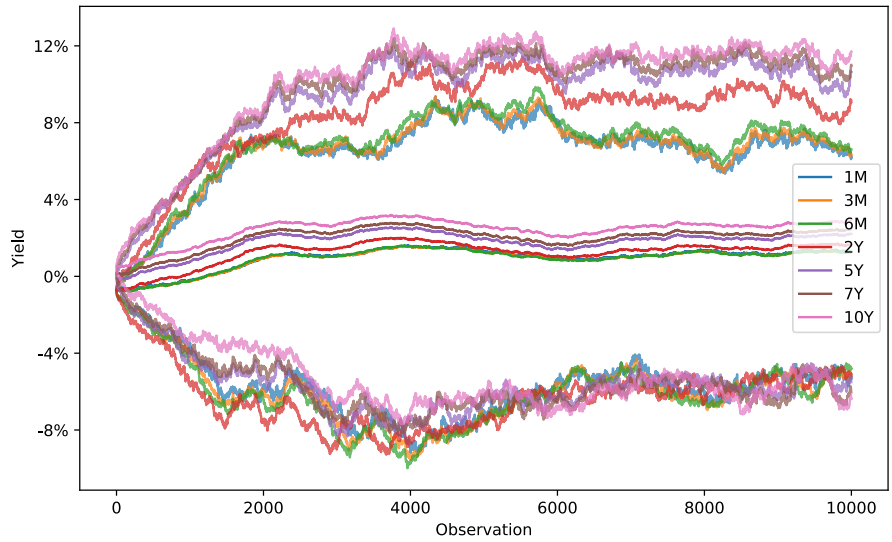

Figure 5: Mean, 5th percentile and 95th percentile of 100 generated scenarios from 10 different models with 256 hidden units, trained on PCA transformed data with all components included.

In Figure 6 are the means, 5th percentiles and 95th percentiles of the generated yield curves at 1 year, 10 years and 35 years ahead plotted. The curves are based on the same data as Figure 5

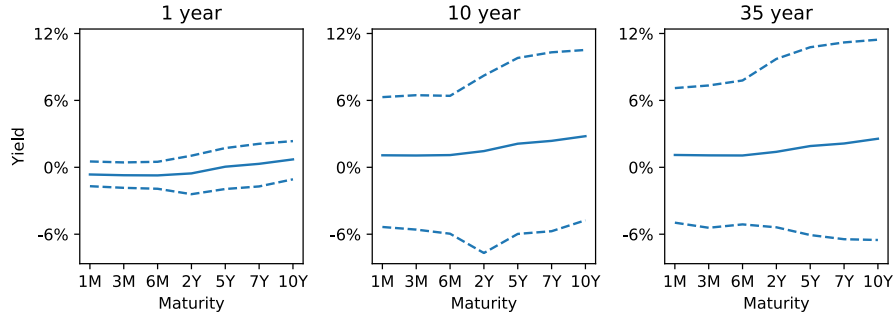

Figure 6: Mean, 5th percentile and 95th percentile of 100 generated yield curves 1 year, 10 years and 35 years ahead.

Benchmarks

The purpose of the benchmark models were to compare the generated scenarios of the Gaussian restricted Boltzmann machine to some simple models. In the figures below are the means, 5th percentiles and 95th percentiles of 100 generated scenarios from the benchmark models plotted. Figure 7 show the results from the uncorrelated Gaussian random walk.

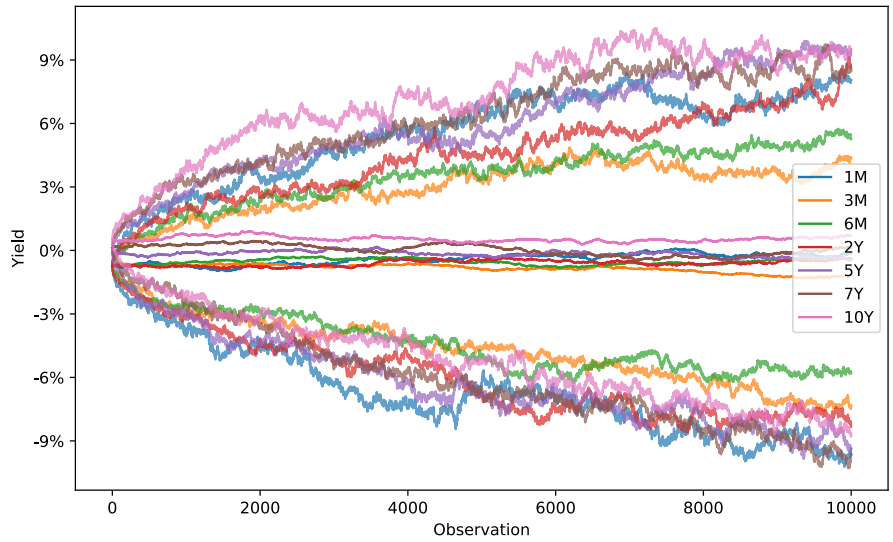

Figure 7: Mean, 5th percentile and 95th percentile of 100 generated scenarios based on the Gaussian random walk.

In Figure 6 are the means, 5th percentiles and 95th percentiles of the generated yield curves at 1 year, 10 years and 35 years ahead plotted. The curves are based on the same data as Figure 7

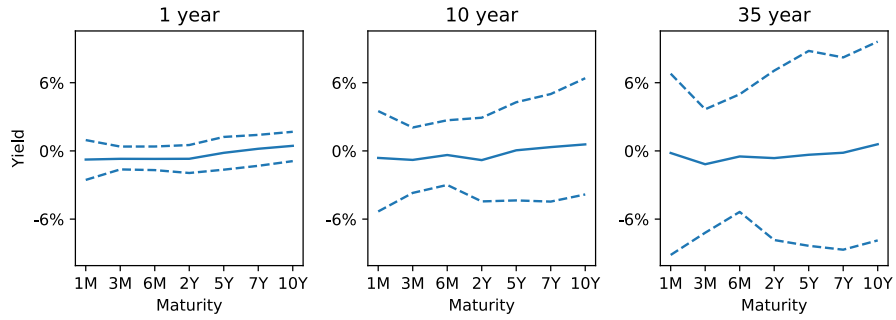

Figure 8: Mean, 5th percentile and 95th percentile of 100 generated yield curves 1 year, 10 years and 35 years ahead, based on the Gaussian random walk.

Figure 9 and 10 show the results from the correlated Gaussian random walk.

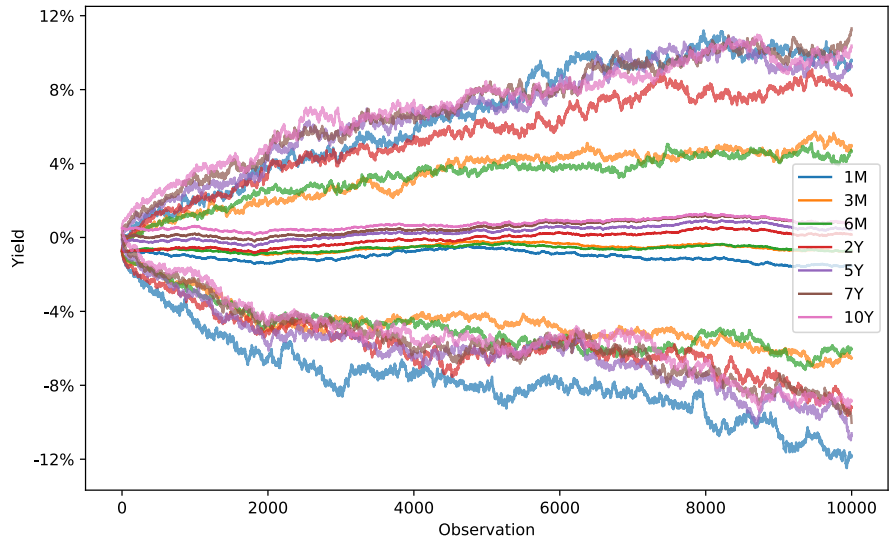

Figure 9: Mean, 5th percentile and 95th percentile of 100 generated scenarios based on the correlated Gaussian random walk.

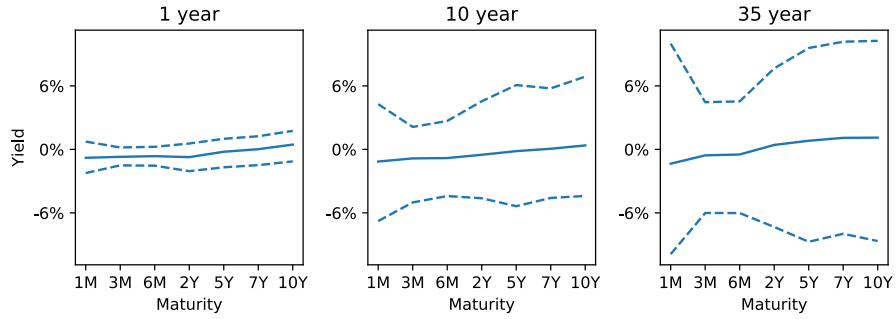

Figure 10: Mean, 5th percentile and 95th percentile of 100 generated yield curves 1 year, 10 years and 35 years ahead, based on the correlated Gaussian random walk.

Figure 11 and 12 show the results from the PCA based benchmark described in section 4.2.2.

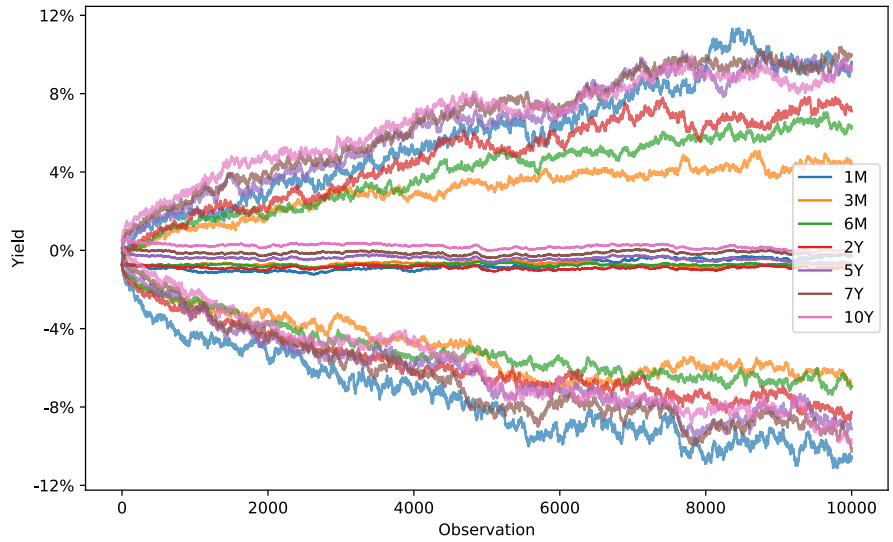

Figure 11: Mean, 5th percentile and 95th percentile of 100 generated scenarios from the PCA based benchmark.

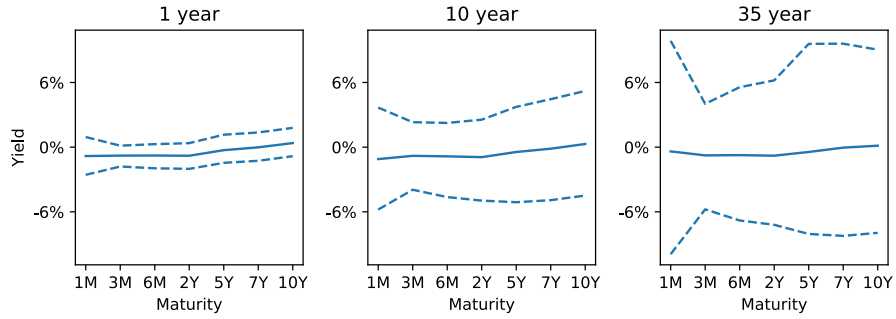

Figure 12: Mean, 5th percentile and 95th percentile of 100 generated yield curves 1 year, 10 years and 35 years ahead from the PCA based benchmark.

Numeric comparison

The purpose of this part was to evaluate the ability to generate scenarios in a more rigorous way. Table 7 shows the one year ahead mean squared forecast errors for the different maturities. It is not a true out-of-sample forecast since most of the forecasted values were used during training, as described in Section 4.3.2.

Maturity	GRBM	RW	Cov	PCA
1M	0.0439	0.1114	0.1207	0.1231
3M	0.0365	0.0918	0.0957	0.0976
6M	0.0446	0.0893	0.0920	0.0930
2Y	0.1075	0.1539	0.1618	0.1616
5Y	0.3029	0.3214	0.3325	0.3281
7Y	0.3377	0.3649	0.3760	0.3704
10Y	0.3896	0.3915	0.4021	0.3974

Table 7: MSE between the forecasted level of the term structure and the true level of the term structure for the Gaussian restricted Boltzmann machine and the benchmark models.

6 Discussion

In this section are the results obtained in Section 5 discussed. It begins with a discussion about the results regarding the return reconstructions/one-day-ahead forecasts, followed by a discussion regarding the the scenario generation. Then follows a discussion about PCA as data preprocessing.

6.1 One-day-ahead forecasts

The results from all tested architectures were fairly similar. The ensemble of the best performing architecture performed slightly better than the vector autoregressive benchmark, but close to the random walk benchmark. These results are slightly better than for the classical no arbitrage affine models tested in the paper by Velásquez-Giraldo and Restrepo-Tobón and the Nelson-Siegel extensions tested in the paper by Luo, Han and Zhang, which were slightly outperformed by the random walk benchmark, see [27] and [18] respectively.

The reason why the models cannot significantly outperform the random walk benchmark might be explained by the efficient market hypothesis, which states that financial markets are efficient and that the current price on a financial asset reflects all available information. It should thus be impossible to consistently outperform the market, see [19]. There exist different formulations of the efficient market hypothesis that differ regarding which information is reflected in the prices of financial assets, but predicting future prices by analyzing prices from the past are considered impossible. In the light of this, the results are not surprising.

However, there are a lot of researchers and practitioners that have disputed the efficient market hypothesis, for example [19]. One rationale behind the efficient market hypothesis is that inefficiencies which are found will quickly be exploited and thus disappear. The ability to use machine learning to model complex nonlinear relations is a fairly new thing, so there is reason to believe that it can be used to find formerly unknown inefficiencies. However, this study could not find any inefficiencies by using Gaussian restricted Boltzmann machines that

could be used to predict the yields of tomorrow significantly better than the random walk benchmark.

The differences in performance between the architectures were fairly small and the standard deviations, based on the models in the ensemble, were fairly large compared to these differences, which makes it hard to draw any conclusions regarding how the number of hidden units affect performance. Since the differences were so small regarding one-day-ahead forecasts, it would be interesting to investigate if the differences would be as small for long term scenario generation. If so, there is no reason to use large architectures since the smaller architectures are less time consuming to train and evaluate. This is something for future work to look at.

The lower forecast error on the validation set compared to the test set was expected, since the models were chosen based on performance on the validation set.

A fairly large number of hidden units were compared, but a limitation of this study is that the remaining hyperparameters were not tuned rigorously, even though it is unlikely that it would have changed the results substantially. This is something for future research to look into, as well as if there might be performance differences between the maturities.

6.2 Long term scenario generation

Even if the one day ahead returns could not be predicted better than assuming a random walk, there may still be important structures in the data that can be modeled to generate plausible long term scenarios. For example if the yield curve is particularly steep, it can be expected to flatten out, but exactly how the flattening proceeds with respect to time and size of the daily changes can be hard to predict.

The ability to generate scenarios showed promising results. A few conclusions can be drawn by comparing the generated scenarios shown in Figure 2, 3 and 4. The models selected based on log-likelihood performed considerably worse than the models selected based on reconstruction error. The yields from the models selected based on log-likelihood diverge from each other and drift to extreme values in a way that is not consistent with the raw data plotted in Figure 1. By looking at the number of hidden nodes and the number of training epochs for each model, it is apparent that the log-likelihood and the reconstruction error favour different models, where the latter perform better with regards to long term scenario generation. The reasons for this is something to investigate in future work.

Another conclusion is that the models trained on PCA transformed data performed better than the models trained on raw data, given that reconstruction error was used for model selection. The yields generated from the model trained on raw data diverged from each other in a way that is not consistent with the raw data in Figure 1. This difficulty to model dependencies between the visible units has been discussed in previous research, and might be due to the diagonal

covariance matrix of the restricted Boltzamnn machine. The PCA transformation seems to be one way to deal with this difficulty.

One unexpected thing with the scenario generated from the model trained on PCA transformed data with all components included, shown in the top plot of Figure 3, is that the ordering of the yields seems inverted. The reason for this was not investigated further, but a potential explanation is that it is caused by randomness in the weight initialization or in the training process, since the results of the ensemble look better. This is something that has to be investigated in future work.

The scenario generated from the ensemble, shown in Figure 5, has many properties that agree with the raw data in Figure 1. The ordering of the yields seems to agree, at least for the mean and the 95th percentile. The relations between the yields of different maturities do also seem to agree to a large extent, where the 1M and 3M and 6M stick closely together, as well as the 5Y, 7Y and 10Y, with the 2Y somewhere in between. Another desired property is that the yields tend to become somewhat stationary after some time and that the mean level seems to agree fairly well with the raw data shown in Figure 1. The ensemble outperform all benchmark models with respect to these properties.

One weakness with the machine learning approach used in this paper, compared to more classical approaches, is that the model is kind of a black box. The dynamics of the model is not as apparent as for the classical models, which usually can be hand crafted to obtain certain properties. In the classical approach can you for example obtain mean reversion and non-negative yields, but this is harder to explicitly obtain in the machine learning approach.

The results of the one-year forecast errors given in Table 7 do also show promising results. The errors of the ensemble were lower than the errors of all benchmark models for all maturities, especially the shorter ones. However, it should be noted that it is partly an in-sample forecast, as described in Section 4.3.2. The results are better than for the classical affine models evaluated by for example Duffee in [7], which were outperformed by the random walk benchmark both in-sample and out-of-sample. The studies that were found which used more advanced models did only evaluate the forecast ability out-of-sample, which makes a comparison unfair. The forecast ability is probably also be dependent on the data set used, for example with respect to country and period of time, which makes comparisons hard. To get a fair comparison between models, it is best to compare the results on the same data set. This is something for future research to look at.

6.3 Preprocessing

The results indicate that modeling PCA preprocessed data is to prefer both regarding one-day-ahead forecasting and long term scenario generation. The most considerable advantage of modeling PCA transformed data is that the yields of the generated scenarios show correlations which are more in line with the raw data in Figure 1. This might seem obvious considering the diagonal covariance structure of the restricted Boltzmann machine, but it is not a very common transformation in the literature.

Regarding how many components to include, there is not enough evidence in this thesis to draw any conclusions. The number of components to include can probably be seen as a hyperparameter that should be tuned.

7 Conclusion

One objective of this thesis was to investigate if Gaussian restricted Boltzmann machines can be used to forecast the term structure of interest rates one day ahead. The results suggest that the forecast ability one day ahead is similar to a benchmark model that assumes a random walk and slightly better than a benchmark based on a vector autoregression model of order 1. The results hold both in-sample and out-of-sample and outperform many classical affine models tested in previous work, which tend to perform worse than a random walk benchmark both in-sample and out-of-sample.

Another objective of this thesis was to investigate if the one-day-ahead forecasts can be iterated to generate plausible long term scenarios. These results are promising. The generated scenarios have many desirable properties, such as correct ordering of the yields and mutual relations that resemble the original yield data. The generated scenarios do also seem somewhat stationary after some time, with mean values that agree fairly well with the original yield data. With regard to these properties do the Gaussian restricted Boltzmann machine outperform the tested benchmark models.

The ability to make one-year-ahead forecasts do also show good results, measured as mean squared forecast error on semi-out-of-sample data. The results outperform the tested benchmark models, especially for shorter maturities. The results do also outperform many classical affine models tested in previous work, which fail to beat the random walk benchmark both in-sample and out-of sample. How the results compare to more advanced models out-of-sample is something to investigate in future work.

The last objective was to investigate the effects of PCA as data preprocessing. The results indicate that modeling PCA preprocessed data is to prefer regarding both one-day-ahead forecasting and long term scenario generation. For the scenario generation do PCA preprocessing seem like a necessity to obtain good results.

7.1 Future work

This thesis lay the groundwork for modeling the term structure of interest rates with machine learning in general and Gaussian restricted Boltzmann machines in particular. The results are promising and this thesis has just scratched the surface of the possibilities. Some ideas for future work are to:

- Do a more extensive parameter optimization
- Investigate the dynamics of the generated scenarios

- Compare the Gaussian restricted Boltzmann machine with other term structure models on the same data set
- Try deep architectures, such as a deep Boltzmann machines, or other generative models

A Ensemble results

In Table 8 are the MSE of the return reconstruction for the models in the ensemble presented.

Ensemble	MSE valid	MSE test	Epochs
1	1.2963	1.3003	5500
2	1.2960	1.3032	5500
3	1.2958	1.2961	5500
4	1.2966	1.2972	3900
5	1.2951	1.3056	7200
6	1.2949	1.3024	5800
7	1.2945	1.3032	5600
8	1.2957	1.2982	5300
9	1.2951	1.3000	5700
10	1.2957	1.3018	5700
Mean	1.2956	1.3008	5570
SD	0.000619	0.002853	750

Table 8: MSE between inferred normalized returns and true normalized returns for the models in the ensemble.

B Conditional probability distributions

The joint probability distribution of a Gaussian-Bernoulli restricted Boltzmann machine is obtained by plugging in the energy function 3.6 into the expression for the Gibbs distribution 3.4. [20]

$$
\begin{aligned}
p(\boldsymbol{v}|\boldsymbol{h}) &= \frac{p(\boldsymbol{v},\boldsymbol{h})}{\int_{\boldsymbol{v}} p(\boldsymbol{v},\boldsymbol{h})d\boldsymbol{v}} = \\
&= \frac{e^{-\sum_i \frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_j b_j h_j+\sum_{i,j}\frac{v_i}{\sigma_i^2}w_{ij}h_j}}{\int_{\boldsymbol{v}} e^{-\sum_i \frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_j b_j h_j+\sum_{i,j}\frac{v_i}{\sigma_i^2}w_{ij}h_j} d\boldsymbol{v}} = \\
&= \frac{\prod_i e^{-\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_j \frac{v_i}{\sigma_i^2}w_{ij}h_j}}{\int_{\boldsymbol{v}} \prod_i e^{-\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_j \frac{v_i}{\sigma_i^2}w_{ij}h_j} d\boldsymbol{v}} = \\
&= \prod_i \frac{e^{-\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_j \frac{v_i}{\sigma_i^2}w_{ij}h_j}}{\int_{v_i} e^{-\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_j \frac{v_i}{\sigma_i^2}w_{ij}h_j} dv_i} = \\
&= \prod_i \frac{e^{\frac{-(v_i-a_i-\sum_j w_{ij}h_j)^2+(\sum_j w_{ij}h_j)^2+2\sum_j a_i w_{ij}h_j}{2\sigma_i^2}}}{\int_{v_i} e^{\frac{-(v_i-a_i-\sum_j w_{ij}h_j)^2+(\sum_j w_{ij}h_j)^2+2\sum_j a_i w_{ij}h_j}{2\sigma_i^2}} dv_i} = \\
&= \prod_i \frac{e^{\frac{-(v_i-a_i-\sum_j w_{ij}h_j)^2}{2\sigma_i^2}}}{\int_{v_i} e^{\frac{-(v_i-a_i-\sum_j w_{ij}h_j)^2}{2\sigma_i^2}} dv_i} = \\
&= \prod_i \mathcal{N}(v_i|a_i+\sum_j w_{ij}h_j,\sigma_i^2)
\end{aligned}
\tag{B.1}
$$

The integral $\int_{\boldsymbol{v}} d\boldsymbol{v}$ for $\boldsymbol{v}=(v_1,v_2,...,v_n)$ refers to $\int_{v_1}\int_{v_2}...\int_{v_n} dv_1 dv_2 ... dv_n$. The following calculation is used to complete the square:

$$
\begin{aligned}
&-\frac{(v_i-a_i)^2}{2\sigma_i^2}+\sum_j \frac{v_i}{\sigma_i^2}w_{ij}h_j = \\
&= \frac{-v_i^2-a_i^2+2v_ia_i+2\sum_j v_i w_{ij}h_j}{2\sigma_i^2} = \\
&= \frac{-v_i^2-a_i^2+2v_ia_i+2\sum_j v_i w_{ij}h_j+(\sum_j w_{ij}h_j)^2-(\sum_j w_{ij}h_j)^2+2\sum_j a_i w_{ij}h_j-2\sum_j a_i w_{ij}h_j}{2\sigma_i^2} = \\
&= \frac{-(v_i-a_i-\sum_j w_{ij}h_j)^2+(\sum_j w_{ij}h_j)^2+2\sum_j a_i w_{ij}h_j}{2\sigma_i^2}.
\end{aligned}
\tag{B.2}
$$

$$
\begin{aligned}
p(\boldsymbol{h}|\boldsymbol{v}) = \frac{p(\boldsymbol{v},\boldsymbol{h})}{\sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h})} = \\
&= \frac{e^{-\sum_i \frac{(v_i-a_i)^2}{2\sigma_i^2} + \sum_j b_j h_j + \sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j}}{\sum_{\boldsymbol{h}} e^{-\sum_i \frac{(v_i-a_i)^2}{2\sigma_i^2} + \sum_j b_j h_j + \sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j}} = \\
&= \frac{\prod_j e^{(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}) h_j}}{\sum_{\boldsymbol{h}} \prod_j e^{(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}) h_j}} = \\
&= \prod_j \frac{e^{(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}) h_j}}{\sum_{h_j} e^{(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}) h_j}} = \\
&= \prod_j \frac{e^{(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}) h_j}}{1 + e^{b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}}}
\end{aligned}
\tag{B.3}
$$

The sum $\sum_{\boldsymbol{h}}$ for $\boldsymbol{h} = (h_1, h_2, ..., h_m)$ refers to $\sum_{h_1} \sum_{h_2} ... \sum_{h_m}$. In the last step the sum over $h_j \in \{0, 1\}$ is evaluated.

$$
\begin{aligned}
p(H_j = 1|\boldsymbol{v}) = \frac{e^{b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}}}{1 + e^{b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}}} = \\
&= \frac{1}{1 + e^{-(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij})}} = \\
&= \mathrm{sig}(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij})
\end{aligned}
\tag{B.4}
$$

C Gradient of log-likelihood

For a Gaussian-Bernoulli restricted Boltzmann machine can the gradient of the log-likelihood of a single observation be expressed as [9]

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial \theta} &= \frac{\partial}{\partial \theta} \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \frac{\partial}{\partial \theta} \ln \int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} d\boldsymbol{v} = \\ &= -\frac{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta}}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} + \frac{\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta} d\boldsymbol{v}}{\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} d\boldsymbol{v}} = \\ &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta} + \int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \theta} d\boldsymbol{v}.\end{aligned} \tag{C.1}$$

In the last step the following relations are used:

$$p(\boldsymbol{h}|\boldsymbol{v}) = \frac{p(\boldsymbol{v},\boldsymbol{h})}{p(\boldsymbol{v})} = \frac{\frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \tag{C.2}$$

$$p(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})} = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} d\boldsymbol{v}}. \tag{C.3}$$

Component $\boldsymbol{W}$
The first term of the derivative of the log-likelihood C.1 with respect to the components of $\boldsymbol{W}$ can be rewritten as [20]

$$\begin{aligned}\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} h_j = \\ &= -\sum_{\boldsymbol{h}} p(h_j|\boldsymbol{v}) p(\boldsymbol{h}_{-j}|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} h_j = \\ &= -\sum_{h_j} \sum_{\boldsymbol{h}_{-j}} p(h_j|\boldsymbol{v}) p(\boldsymbol{h}_{-j}|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} h_j = \\ &= -\sum_{h_j} p(h_j|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} h_j \sum_{\boldsymbol{h}_{-j}} p(\boldsymbol{h}_{-j}|\boldsymbol{v}) = \\ &= -p(H_j = 1|\boldsymbol{v}) \frac{v_i}{\sigma_i^2}.\end{aligned} \tag{C.4}$$

The factorization is possible since the hidden units are independent of each other given the values of the visible units, as stated in 3.7. $\boldsymbol{h}_{-j}$ refers to the vector $\boldsymbol{h}$ without component j. In the last step the sum over h_j is evaluated and it is noted that $\sum_{\boldsymbol{h}_{-j}} p(\boldsymbol{h}_{-j}|\boldsymbol{v}) = 1$.

The second term of the derivative of the log-likelihood C.1 with respect to the components of $\boldsymbol{W}$ can be rewritten as

$$\begin{aligned}\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} d\boldsymbol{v} &= -\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) p(\boldsymbol{v}) \frac{v_i}{\sigma_i^2} h_j d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} h_j d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{h_j} p(h_j|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} h_j \sum_{\boldsymbol{h}_{-j}} p(\boldsymbol{h}_{-j}|\boldsymbol{v}) d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_j = 1|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} d\boldsymbol{v}.\end{aligned} \tag{C.5}$$

The derivative of the log-likelihood C.1 with respect to w_{ij} then becomes:

$$\frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial w_{ij}} = p(H_j = 1|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} - \int_{\boldsymbol{v}} p(\boldsymbol{v}) p(H_j = 1|\boldsymbol{v}) \frac{v_i}{\sigma_i^2} d\boldsymbol{v}. \tag{C.6}$$

Component $\boldsymbol{a}$

The first term of the derivative of the log-likelihood C.1 with respect to the components of $\boldsymbol{a}$ can be rewritten as

$$\begin{aligned}\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial a_i} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{v_i - a_i}{\sigma_i^2} = \\ &= -\frac{v_i - a_i}{\sigma_i^2}.\end{aligned} \tag{C.7}$$

The second term of the derivative of the log-likelihood C.1 with respect to the components of $\boldsymbol{a}$ can be rewritten as

$$\begin{aligned}\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial a_i} d\boldsymbol{v} &= -\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) p(\boldsymbol{v}) \frac{v_i - a_i}{\sigma_i^2} d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v}) \frac{v_i - a_i}{\sigma_i^2} \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v}) \frac{v_i - a_i}{\sigma_i^2} d\boldsymbol{v}.\end{aligned} \tag{C.8}$$

The derivative of the log-likelihood C.1 with respect to a_i then becomes:

$$\frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial a_i} = \frac{v_i - a_i}{\sigma_i^2} - \int_{\boldsymbol{v}} p(\boldsymbol{v}) \frac{v_i - a_i}{\sigma_i^2} d\boldsymbol{v}. \tag{C.9}$$

Component $\boldsymbol{b}$
The first term of the derivative of the log-likelihood C.1 with respect to the components of $\boldsymbol{b}$ can be rewritten as:

$$\begin{aligned}\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial b_j} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})h_j = \\ &= -\sum_{h_j} p(h_j|\boldsymbol{v})h_j \sum_{\boldsymbol{h}_{-j}} p(\boldsymbol{h}_{-j}|\boldsymbol{v}) = \\ &= -p(H_j = 1|\boldsymbol{v}).\end{aligned} \tag{C.10}$$

The second term of the derivative of the log-likelihood C.1 with respect to the components of $\boldsymbol{b}$ can be rewritten as

$$\begin{aligned}\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h})\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial b_j} d\boldsymbol{v} &= -\int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})p(\boldsymbol{v})h_j d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})h_j d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{h_j} p(h_j|\boldsymbol{v})h_j \sum_{\boldsymbol{h}_{-j}} p(\boldsymbol{h}_{-j}|\boldsymbol{v}) d\boldsymbol{v} = \\ &= -\int_{\boldsymbol{v}} p(\boldsymbol{v})p(H_j = 1|\boldsymbol{v}) d\boldsymbol{v}.\end{aligned} \tag{C.11}$$

The derivative of the log-likelihood C.1 with respect to b_j then becomes:

$$\frac{\partial \ln \mathcal{L}(\theta|\boldsymbol{v})}{\partial b_j} = p(H_j = 1|\boldsymbol{v}) - \int_{\boldsymbol{v}} p(\boldsymbol{v})p(H_j = 1|\boldsymbol{v}) d\boldsymbol{v}. \tag{C.12}$$

D Derivations for AIS

D.1 Intermediate probabilities

The unnormalized intermediate probabilities $p^*_{\beta_k}(\boldsymbol{v})$ for $k = 0, 1, ..., K$ are derived by plugging in the energy function 3.20 into the expression for the Gibbs distribution 3.4, and then marginalizing over $\boldsymbol{h}$:

$$
\begin{aligned}
p^*_{\beta_k}(\boldsymbol{v}) &= \sum_{\boldsymbol{h}} p^*_{\beta_k}(\boldsymbol{v}, \boldsymbol{h}) = \sum_{\boldsymbol{h}} e^{-E_{\beta_k}(\boldsymbol{v},\boldsymbol{h})} = \\
&= \sum_{\boldsymbol{h}} e^{-\beta_k(\sum_i \frac{(v_i - a_i^B)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j) - (1-\beta_k) \sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2}} = \\
&= e^{-\beta_k \sum_i \frac{(v_i - a_i^B)^2}{2\sigma_i^2} - (1-\beta_k) \sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2}} \sum_{\boldsymbol{h}} e^{\beta_k(\sum_j b_j + \sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij}) h_j} = \\
&= e^{-\beta_k \sum_i \frac{(v_i - a_i^B)^2}{2\sigma_i^2} - (1-\beta_k) \sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2}} \sum_{\boldsymbol{h}} \prod_j e^{\beta_k(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}) h_j} = \\
&= e^{-\beta_k \sum_i \frac{(v_i - a_i^B)^2}{2\sigma_i^2} - (1-\beta_k) \sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2}} \prod_j \sum_{h_j} e^{\beta_k(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij}) h_j} = \\
&= e^{-\beta_k \sum_i \frac{(v_i - a_i^B)^2}{2\sigma_i^2} - (1-\beta_k) \sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2}} \prod_j (1 + e^{\beta_k(b_j + \sum_i \frac{v_i}{\sigma_i^2} w_{ij})})
\end{aligned}
\tag{D.1}
$$

D.2 Partition function of initial distribution

The partition function Z_A of the initial distribution p_A is obtained by marginalizing the unnormalized distribution p^*_A over all variables.

$$
\begin{aligned}
Z_A &= \int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p^*_A(\boldsymbol{v}, \boldsymbol{h}) d\boldsymbol{v} = \int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E_A(\boldsymbol{v},\boldsymbol{h})} d\boldsymbol{v} = \\
&= \int_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-\sum_i \frac{(v_i - a_i^A)^2}{2\sigma_i^2}} d\boldsymbol{v} = \\
&= \sum_{\boldsymbol{h}} \int_{\boldsymbol{v}} \prod_i e^{-\frac{(v_i - a_i^A)^2}{2\sigma_i^2}} d\boldsymbol{v} = \\
&= 2^m \prod_i \int_{v_i} e^{-\frac{(v_i - a_i^A)^2}{2\sigma_i^2}} dv_i = \\
&= 2^m \prod_i \sqrt{2\pi\sigma_i^2}
\end{aligned}
\tag{D.2}
$$

E Free energy

The free energy is given by:

$$\begin{aligned} F(\boldsymbol{v}) &= -\log \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} = \\ &= -\log \sum_{\boldsymbol{h}} e^{-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^2 + \boldsymbol{b}^T\boldsymbol{h} + \boldsymbol{v}\boldsymbol{W}\boldsymbol{h}} = \\ &= \frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^2 - \log \sum_{\boldsymbol{h}} e^{(\boldsymbol{b}^T + \boldsymbol{v}\boldsymbol{W})\boldsymbol{h}} = \\ &= \frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^2 - \log \sum_{h_i} e^{\sum_i (b_i + vW_i)h_i} = \\ &= \frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^2 - \log \sum_{h_i} \prod_i e^{(b_i + vW_i)h_i} = \\ &= \frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^2 - \log \prod_i \sum_{h_i} e^{(b_i + vW_i)h_i} = \\ &= \frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^2 - \sum_i \log 1 + e^{b_i + vW_i} \end{aligned} \tag{E.1}$$

I have changed from vector to index notation. Since the product factorizes in factors that only depend on i, the sum and product can change place. In the last equality I have used simply calculated the sum, where h_i can take values 0 or 1.

References

[1] Bengio, Yoshua et al. "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[2] Bishop, Christopher M. *Pattern recognition and machine learning*. eng. Information science and statistics. New York, NY: Springer, 2006. ISBN: 0-387-31073-8.

[3] Christensen, Jens HE, Diebold, Francis X, and Rudebusch, Glenn D. "The affine arbitrage-free class of Nelson–Siegel term structure models". In: *Journal of Econometrics* 164.1 (2011), pp. 4–20.

[4] Cox, John C, Ingersoll Jr, Jonathan E, and Ross, Stephen A. "A theory of the term structure of interest rates". In: *Theory of Valuation*. World Scientific, 2005, pp. 129–164.

[5] Da Silva, Ivan Nunes et al. *Artificial Neural Networks*. Springer, 2017.

[6] Danıelsson, Ásgeir et al. "Accuracy in forecasting macroeconomic variables in Iceland". In: (2008).

[7] Duffee, Gregory R. "Term premia and interest rate forecasts in affine models". In: *The Journal of Finance* 57.1 (2002), pp. 405–443.

[8] Engle, Robert and Siriwardane, Emil. "TERM STRUCTURE FORECASTING AND SCENARIO GENERATION". In: (2015).

[9] Fischer, Asja and Igel, Christian. "Training restricted Boltzmann machines: An introduction". In: *Pattern Recognition* 47.1 (2014), pp. 25–39.

[10] Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines". In: *Momentum* 9.1 (2010), p. 926.

[11] Hinton, Geoffrey E et al. "Modeling pixel means and covariances using factorized third-order Boltzmann machines". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 2551–2558.

[12] Hinton, Geoffrey E. "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[13] Hrasko, Rafael, Pacheco, André GC, and Krohling, Renato A. "Time series prediction using restricted boltzmann machines and backpropagation". In: *Procedia Computer Science* 55 (2015), pp. 990–999.

[14] Hull, John C and Basu, Sankarshan. *Options, futures, and other derivatives*. Pearson Education India, 2016.

[15] Hull, John and White, Alan. "Pricing interest-rate-derivative securities". In: *The Review of Financial Studies* 3.4 (1990), pp. 573–592.

[16] Jolliffe, Ian T. *Principal component analysis*. eng. 2nd ed.. Springer series in statistics. New York: Springer, 2002. ISBN: 1-280-00966-7.

[17] Kiwaki, Taichi and Aihara, Kazuyuki. "An effect of initial distribution covariance for annealing Gaussian restricted Boltzmann machines". In: *Artificial Intelligence Research* 4.1 (2015), p. 53.

[18] Luo, Xingguo, Han, Haifeng, and Zhang, Jin E. "Forecasting the term structure of Chinese Treasury yields". In: *Pacific-Basin Finance Journal* 20.5 (2012), pp. 639–659.

[19] Malkiel, Burton Gordon and McCue, Kerin. *A random walk down Wall Street*. Vol. 8. Norton New York, 1985.

[20] Melchior, Jan, Wang, Nan, and Wiskott, Laurenz. "Gaussian-binary restricted Boltzmann machines for modeling natural image statistics". In: *PloS one* 12.2 (2017), e0171015.

[21] Munk, Claus. *Fixed income modelling*. Oxford University Press, 2011.

[22] Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 0262018020, 9780262018029.

[23] Neal, Radford M. "Annealed importance sampling". In: *Statistics and computing* 11.2 (2001), pp. 125–139.

[24] Salakhutdinov, Ruslan. "Learning and evaluating Boltzmann machines". In: *Tech. Rep., Technical Report UTML TR 2008-002, Department of Computer Science, University of Toronto* (2008).

[25] Schulz, Hannes, Müller, Andreas, and Behnke, Sven. "Investigating convergence of restricted boltzmann machine learning". In: *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*. 2010.

[26] Vasicek, Oldrich. "An equilibrium characterization of the term structure". In: *Journal of financial economics* 5.2 (1977), pp. 177–188.

[27] Velásquez-Giraldo, Mateo and Restrepo-Tobón, Diego. "Affine term structure models: Forecasting the yield curve for Colombia". In: *Lecturas de Economıa* 85 (2016), pp. 53–90.

TRITA -SCI-GRU 2018:241