

# Quantum Boltzmann Machines

Applications in Quantitative Finance

Cameron Perot

Master's Thesis

submitted to

the Faculty of Mathematics, Computer Science, and Natural Sciences  
of RWTH Aachen University

written at

Jülich Supercomputing Centre  
Forschungszentrum Jülich

First Examiner: Prof. Dr. Kristel Michielsen<sup>\*,†</sup>

Second Examiner: Prof. Dr. Holger Rauhut<sup>\*</sup>

Adviser: Dr. Dennis Willsch<sup>†</sup>

<sup>\*</sup>RWTH Aachen University, D-52056 Aachen, Germany

<sup>†</sup>Jülich Supercomputing Centre, Institute for Advanced Simulation,  
Forschungszentrum Jülich, D-52425 Jülich, Germany

July 4, 2022



# Abstract

In this thesis we explore using the D-Wave Advantage 4.1 quantum annealer to sample from quantum Boltzmann distributions and train quantum Boltzmann machines (QBM). We focus on the real-world problem of using QBMs as generative models to produce synthetic foreign exchange market data and analyze how the results stack up against classical models based on restricted Boltzmann machines (RBMs). Additionally, we study a small 12-qubit problem which we use to compare samples obtained from the Advantage 4.1 with theory, and in the process gain vital insights into how well the Advantage 4.1 can sample quantum Boltzmann random variables and be used to train QBMs. Through this, we are able to show that the Advantage 4.1 can sample classical Boltzmann random variables to some extent, but is limited in its ability to sample from quantum Boltzmann distributions. Our findings indicate that QBMs trained using the Advantage 4.1 are much noisier than those trained using simulations and struggle to perform at the same level as classical RBMs. However, there is the potential for QBMs to outperform classical RBMs if future generation annealers can generate samples closer to the desired theoretical distributions.



# Contents

|          |                                                               |           |
|----------|---------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                           | <b>1</b>  |
| <b>2</b> | <b>Data Analysis &amp; Preprocessing</b>                      | <b>3</b>  |
| 2.1      | Data Analysis . . . . .                                       | 3         |
| 2.2      | Data Preprocessing . . . . .                                  | 4         |
| 2.2.1    | Data Transformation . . . . .                                 | 6         |
| 2.2.2    | Additional Information . . . . .                              | 6         |
| <b>3</b> | <b>The Classical Restricted Boltzmann Machine</b>             | <b>11</b> |
| 3.1      | Theory . . . . .                                              | 11        |
| 3.1.1    | Optimizing an RBM . . . . .                                   | 12        |
| 3.2      | The Classical Market Generator . . . . .                      | 13        |
| 3.2.1    | Models . . . . .                                              | 14        |
| 3.2.2    | Results . . . . .                                             | 14        |
| 3.2.3    | Summary . . . . .                                             | 17        |
| <b>4</b> | <b>The Quantum Boltzmann Machine</b>                          | <b>21</b> |
| 4.1      | Theory . . . . .                                              | 21        |
| 4.1.1    | Optimizing a QBM . . . . .                                    | 22        |
| 4.1.2    | Quantum Annealing . . . . .                                   | 22        |
| 4.2      | 12-Qubit Problem . . . . .                                    | 26        |
| 4.2.1    | Sampling From a Quantum Boltzmann Distribution . . . . .      | 27        |
| 4.2.2    | Training Data . . . . .                                       | 30        |
| 4.2.3    | Simulation-based Model . . . . .                              | 30        |
| 4.2.4    | D-Wave Advantage 4.1-based Model . . . . .                    | 32        |
| 4.3      | The Quantum Market Generator . . . . .                        | 35        |
| 4.3.1    | Setting the Annealer’s Hyperparameters . . . . .              | 35        |
| 4.3.2    | Results . . . . .                                             | 36        |
| 4.3.3    | Comparison to Gate-Based Models . . . . .                     | 38        |
| 4.3.4    | Summary . . . . .                                             | 40        |
| 4.4      | Challenges of Using a D-Wave Annealer to Train QBMs . . . . . | 42        |
| 4.4.1    | Choosing an Embedding . . . . .                               | 42        |
| 4.4.2    | Sampling the Proper Distribution . . . . .                    | 42        |
| 4.4.3    | QPU Limitations and Imperfections . . . . .                   | 43        |
| <b>5</b> | <b>Conclusion</b>                                             | <b>45</b> |
| 5.1      | Summary . . . . .                                             | 45        |
| 5.2      | Future Directions . . . . .                                   | 46        |
|          | <b>Appendix A Definitions and Methodologies</b>               | <b>47</b> |
| A.1      | Correlation Coefficients . . . . .                            | 47        |
| A.2      | Annualized Volatility . . . . .                               | 47        |
| A.3      | Learning Rate Decay Schedule . . . . .                        | 48        |

|                                                |                                                      |           |
|------------------------------------------------|------------------------------------------------------|-----------|
| A.4                                            | Autocorrelation Analysis . . . . .                   | 48        |
| A.5                                            | Kullback-Leibler Divergence . . . . .                | 48        |
| A.5.1                                          | Kullback-Leibler Divergence in Practice . . . . .    | 49        |
| A.6                                            | Tail Concentration Functions . . . . .               | 49        |
| A.7                                            | Exact Computation of $\rho$ . . . . .                | 50        |
| A.8                                            | Constants . . . . .                                  | 50        |
| <b>Appendix B Restricted Boltzmann Machine</b> |                                                      | <b>51</b> |
| B.1                                            | Conditional Probabilities . . . . .                  | 51        |
| B.2                                            | Log-Likelihood Derivative . . . . .                  | 52        |
| <b>Appendix C Quantum Boltzmann Machine</b>    |                                                      | <b>53</b> |
| C.1                                            | Log-Likelihood Derivative . . . . .                  | 53        |
| C.2                                            | Log-Likelihood Lower Bound . . . . .                 | 54        |
| C.3                                            | Log-Likelihood Lower Bound Derivative . . . . .      | 55        |
| C.4                                            | Effective $\beta$ as a Learnable Parameter . . . . . | 56        |

# Chapter 1

## Introduction

In recent years we have seen the inception of cloud-based quantum computing, with a number of different providers offering various services. In terms of maturity, the quantum computing industry as a whole is still in the early stages and there are a lot of obstacles left to overcome before mainstream adoption. Quantum computing is not only trying to advance the theory and technology, but also yearning for practical applications in which quantum computing offers advantages over classical computing.

There are two main branches of quantum computing: universal quantum computing, i.e., gate-based quantum computing, and adiabatic quantum computing, i.e., quantum annealing. In our work here we focus on the latter, as current generation devices are slightly more mature and have much higher numbers of qubits than the former. We discuss the theory behind quantum annealing later in Section 4.1.2. One such cloud-based quantum computing service is D-Wave's Leap platform [1], which allows users to access quantum annealers and other solvers across the world.

D-Wave is a pioneer in this field, having been researching and developing quantum annealers since 1999. They revolutionized the field with the release of the world's first commercially available quantum annealer in 2011 [2]. Since then, they have released a new version every 2-3 years, each having more qubits and couplers than the previous. Their latest version, the D-Wave Advantage, has over 5000 qubits with 15 connections per qubit [3].

In this thesis we take a journey into the field of quantum machine learning and explore the possibilities of using quantum Boltzmann machines (QBM) as generative models for real-world financial data. As we will see, there is a deep connection between the quantum Boltzmann machine and quantum annealing, allowing one to train QBM using a quantum annealer.

Risk management is one of the most important components of the financial system, and in 2008 it failed, leading to the financial crisis which wreaked havoc on economies around the world. The success of risk management hinges on how accurately the underlying risk models capture the true behavior of the market. Therefore, it is essential that we continuously strive to find new and innovative ways of modeling that can help us understand the real risks involved and implement policies to effectively mitigate such risks.

In the globalized economy of today, foreign exchange (forex) fluctuations expose a number of firms to a lot of risk if not properly mitigated. Forex markets had a daily volume of \$6.6T in 2019 [4], the majority of which was concentrated in a few major pairs. In the 2019 paper *The Market Generator* [5], Kondratyev and Schwarz detail how a classical restricted Boltzmann machine (RBM) can be used to generate synthetic forex data, and the advantages it offers over traditional parametric models. We use their work as a basis to build our classical models upon, which we then use as a reference to compare our quantum models with.

In Chapter 2, we start by visualizing the data set in various ways to get an idea how it is distributed. We further analyze quantitative metrics to get a better understanding of some of the intricacies of the data set. Finally, we go through and detail how we preprocess the data set into a model-friendly format.

With the data set in hand, we move to explaining the theory behind the classical RBM in Chapter 3 and describing some of the difficulties associated with training and using classical RBMs. We then train several classical models on the data set discussed in Chapter 2 using different preprocessing methods and compare them with each other using visualizations and a number of quantitative metrics.

In Chapter 4, we start from the theory of quantum Boltzmann machines, detailing how they work and their connection to quantum annealing. We study a small 12-qubit problem which we can simulate, allowing us to compare annealer performance with that of theory, and gaining key insights into how to train and use QBMs. With those insights, we move to the final stage of training a model using the data set from Chapter 2, then assessing the performance versus the classical models from Chapter 3. Additionally, we cover some of the challenges of using D-Wave quantum annealers to train QBMs in Section 4.4.

Lastly, we summarize our findings in Chapter 5, as well as discuss future directions in which this research can be expanded.

In addition to the research and results presented here, we also introduce the open source Python package `qbm` [6] to make it easier for the community to train and study quantum Boltzmann machines. All work presented here is reproducible (except for that involving quantum measurements), and the code is available on the Forschungszentrum Jülich Git server <sup>1</sup>.

---

<sup>1</sup><https://jugit.fz-juelich.de/qip/qbm-quant-finance>

## Chapter 2

# Data Analysis & Preprocessing

### 2.1 Data Analysis

Our raw data set consists of the daily open, high, low, and close (OHLC) values for the time period 1999-01-01 through 2019-12-31 of the following major currency pairs

- EURUSD - Euro € / U.S. Dollar \$
- GBPUSD - British Pound Sterling £ / U.S. Dollar \$
- USDCAD - U.S. Dollar \$ / Canadian Dollar \$
- USDJPY - U.S. Dollar \$ / Japanese Yen ¥

obtained from Dukascopy historical data feed [7]. We filter the data set to remove days with zero volume, as well as NYSE and LSE holidays, resulting in 5165 training samples. Here we use the notation  $x_{\text{open}}$ ,  $x_{\text{high}}$ ,  $x_{\text{low}}$ , and  $x_{\text{close}}$  to denote the open, high, low, and close values of a currency pair on a particular day.

Given that the raw data values are on an absolute basis, we need to convert them to relative terms in order to be able to compare data from different time periods on a more equal footing. The natural way to do so is to use the intraday returns

$$r = \frac{x_{\text{close}} - x_{\text{open}}}{x_{\text{open}}}. \quad (2.1)$$

However, this is not necessarily the best way to approach this. Instead, we opt to use the log returns

$$\tilde{r} = \log(1 + r) = \log\left(\frac{x_{\text{close}}}{x_{\text{open}}}\right) \quad (2.2)$$

due to several advantages, such as log-normality and small  $r$  approximation [8].

We begin our analysis by taking a look at the histograms depicted in Fig. 2.1. From visual examination we see that the log returns are roughly normally distributed with the statistics given in Table 2.1.

We also visualize the log returns in a violin and box plot in Fig. 2.2 to identify outliers and see how they are distributed. Two major outliers clearly stand out from the rest: one to the downside for the GBPUSD pair, and another to the upside for the USDJPY pair. The former occurred on 2016-06-24, the day the Brexit referendum result was announced [9]. The latter occurred on 2008-10-28, right in the midst of the financial crisis when people were talking about the end of the Yen carry trade [10]. In the final training data set, we remove outliers greater than  $10\sigma$  from the mean, resulting in only removing the day corresponding to the Brexit referendum result, which lies  $11.1\sigma$  below the mean.

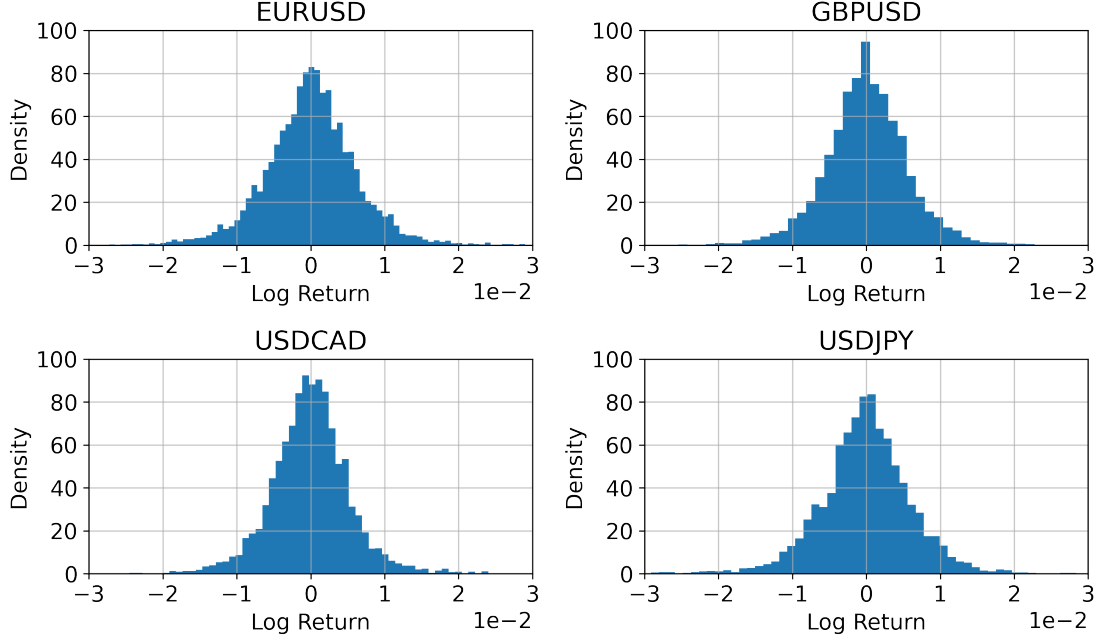


Figure 2.1: Histograms of the log returns data set.

| Log Returns Data Set Statistics |                       |                      |
|---------------------------------|-----------------------|----------------------|
| Currency Pair                   | Mean                  | Standard Deviation   |
| EURUSD                          | $5.15 \cdot 10^{-5}$  | $6.17 \cdot 10^{-3}$ |
| GBPUSD                          | $-8.49 \cdot 10^{-6}$ | $5.73 \cdot 10^{-3}$ |
| USDCAD                          | $-5.04 \cdot 10^{-5}$ | $5.40 \cdot 10^{-3}$ |
| USDJPY                          | $-6.31 \cdot 10^{-5}$ | $6.32 \cdot 10^{-3}$ |

Table 2.1: Statistics of the log returns data set.

Next we examine the correlations between the currency pairs to get an idea of the interdependencies between them. We visualize this with scatter plots shown in Fig. 2.3 where we observe a clear positive correlation between EURUSD/GBPUSD, and clear negative correlations between EURUSD/USDCAD and GBPUSD/USDCAD, where the  $/$  is used to denote the pairs being compared against each other. This is further verified by the Pearson  $r$ , Spearman  $\rho$ , and Kendall  $\tau$  correlation coefficients laid out in Table 2.2. Furthermore, we find the correlation coefficients to be positive for pairs of the form  $XUSD/YUSD$ , and negative for pairs of the form  $XUSD/USDY$ , for  $X, Y \in \{\text{EUR, GBP, CAD, JPY}\}$ , as expected. Details on how the correlation coefficients are computed and how to interpret them can be found in Appendix A.1.

## 2.2 Data Preprocessing

The models in the following chapters require the training data to be in the form of bit vectors, so we must first convert our data set to such a form. Let  $\mathbf{X} \in \mathbb{R}^{4 \times N}$  represent the training data set of log returns with  $N$  samples, where training samples are vectors in the column space, thus element  $x_{ij}$  represents the  $i$ th currency pair log return for the  $j$ th training sample.

To discretize the data, we rescale and round the entries of  $\mathbf{X}$  to integer values in

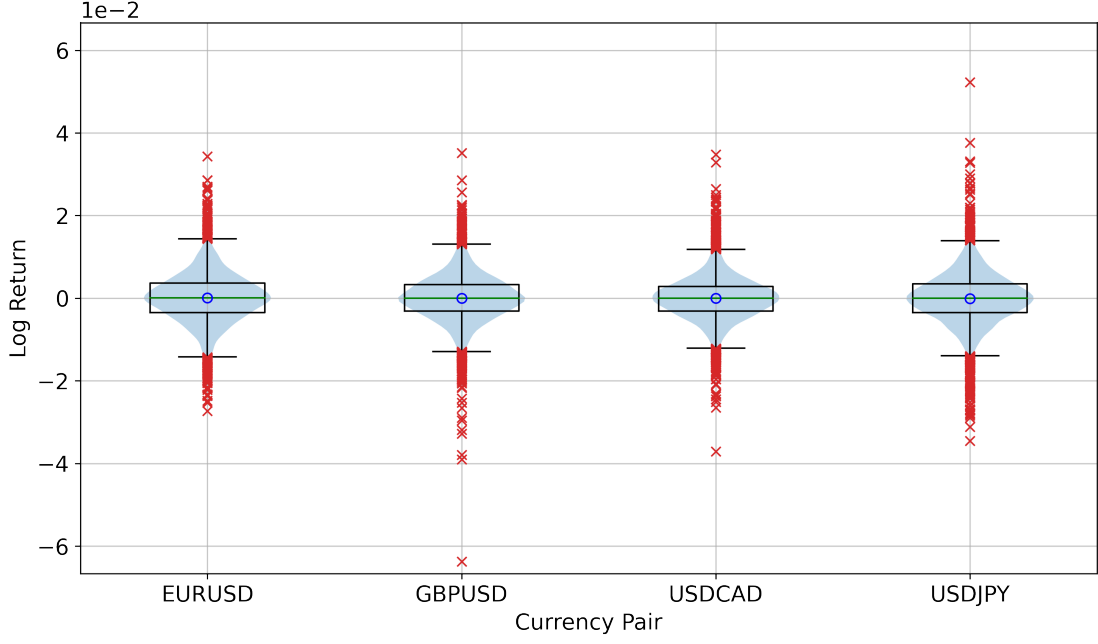


Figure 2.2: Violin and box plot of the log returns data set illustrating the distribution of the outliers.

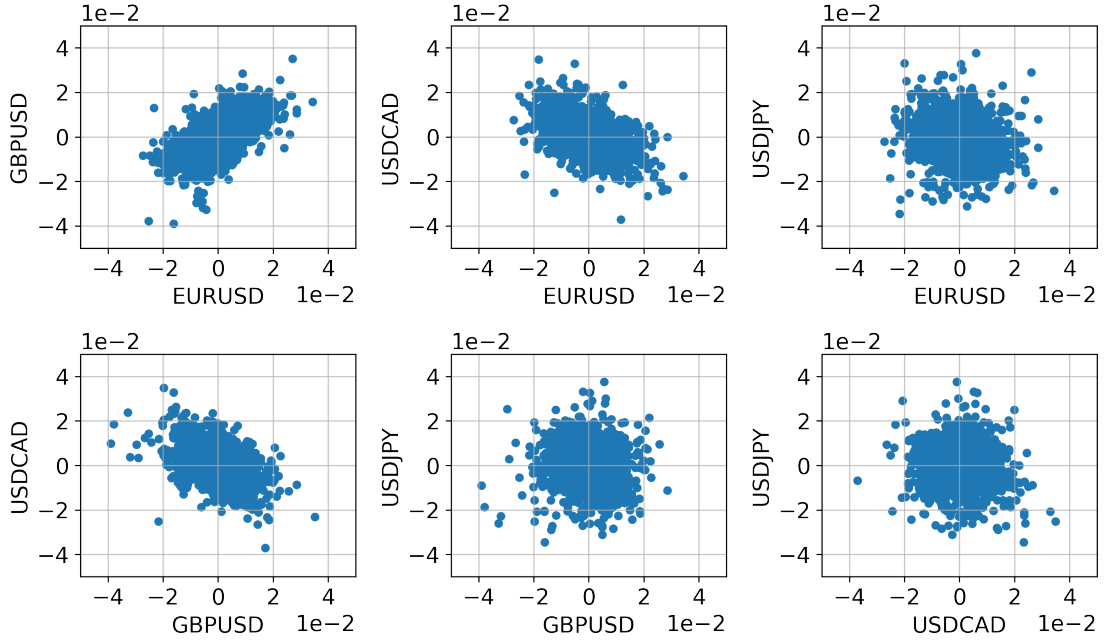


Figure 2.3: Scatter plots of the log returns data set.

$\{0, 1, \dots, 2^{n_{\text{bits}}} - 1\}$ , represented by the matrix  $\mathbf{X}' \in \mathbb{N}^{4 \times N}$  with entries

$$x'_{ij} = \left\lfloor \frac{x_{ij} - \min_k \{x_{ik}\}}{\max_k \{x_{ik}\} - \min_k \{x_{ik}\}} \cdot (2^{n_{\text{bits}}} - 1) \right\rfloor, \quad (2.3)$$

where  $\lfloor \cdot \rfloor$  denotes rounding to the nearest integer.

| Correlation Coefficients |         |          |         |
|--------------------------|---------|----------|---------|
| Currency Pairs           | Pearson | Spearman | Kendall |
| EURUSD/GBPUSD            | 0.62    | 0.62     | 0.44    |
| EURUSD/USDCAD            | -0.44   | -0.41    | -0.29   |
| EURUSD/USDJPY            | -0.26   | -0.30    | -0.21   |
| GBPUSD/USDCAD            | -0.42   | -0.37    | -0.26   |
| GBPUSD/USDJPY            | -0.14   | -0.21    | -0.15   |
| USDCAD/USDJPY            | 0.00    | 0.06     | 0.04    |

Table 2.2: Correlation coefficients of the log returns data set.

A new matrix  $\mathbf{V} \in \{0, 1\}^{4 \cdot n_{\text{bits}} \times N}$  is then created with the columns being the  $n_{\text{bits}}$ -length bit vectors corresponding to the binary representation of the entries of the columns of  $\mathbf{X}'$  concatenated together. For example, if  $\mathbf{x}' = (x'_1, x'_2, x'_3, x'_4)$  is a column of  $\mathbf{X}'$  and the function  $\text{bitvector}(x')$  takes in an integer  $x'$  and returns an  $n_{\text{bits}}$ -bit binary representation bit vector, then the corresponding column in  $\mathbf{V}$  is

$$\mathbf{v} = \begin{bmatrix} \text{bitvector}(x'_1) \\ \text{bitvector}(x'_2) \\ \text{bitvector}(x'_3) \\ \text{bitvector}(x'_4) \end{bmatrix} \in \{0, 1\}^{4 \cdot n_{\text{bits}}}. \quad (2.4)$$

For this research we take  $n_{\text{bits}} = 16$ , giving us a training set  $\mathbf{V} \in \{0, 1\}^{64 \times N}$ , thus our training samples are bit vectors of length 64. The discretization errors associated with this conversion and data set are on the order of  $10^{-7}$ , well within the desired tolerance for this purpose.

### 2.2.1 Data Transformation

Due to how the data is linearly converted to a discrete form before rounding, it opens up the possibility of the discretized data being clustered in the mid-range values if large outliers are present. To mitigate this, we use a transformation to reduce the gap between outliers by scaling outliers beyond a certain threshold  $\tau$  using the procedure detailed in Algorithm 1. We call this the *outlier power transformation*.

In practice, we take  $\tau = 1$  and  $\alpha = 0.5$ , thus the standardized data points above one standard deviation are mapped to their square roots, as illustrated in Fig. 2.4. We tested a few other combinations of  $\tau$  and  $\alpha$ , but found these values to produce the best model results out of those we tried; of course this could likely be further optimized. The effect this transformation has on the model results versus the base dataset can be seen in Section 3.2.2. This transformation is invertible when  $\bar{x}$ ,  $\sigma_x$ , and  $\delta$  are saved.

Histograms of the transformed data set are shown in Fig. 2.5, and a violin and box plot is shown in Fig. 2.6. In these, we observe the appearance of "shoulders" around the threshold  $\tau = 1$  standard deviation, and that the transformed outliers appear much less extreme, allowing us to better utilize the full range of discrete values. Table 2.3 shows that the transformation reduces the standard deviations to roughly 78% of their originals values given in Table 2.1.

### 2.2.2 Additional Information

As mentioned in [5], one can use additional binary indicator variables to enrich the training data set. One such bit of information is the rolling volatility relative to the historical median (see Appendix A.2 for definition of annualized volatility). If the 3-month rolling volatility

---

**Algorithm 1** Outlier Power Transformation

---

```
1: procedure TRANSFORM( $\mathbf{x}, \alpha, \tau$ ) ▷  $\alpha$  is the power,  $\tau$  is the threshold
2:    $N \leftarrow \text{length}(\mathbf{x})$ 
3:    $\bar{x} \leftarrow \frac{1}{N} \sum_{i=1}^N x_i$ 
4:    $\sigma_x \leftarrow \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}$ 
5:    $\delta \leftarrow \tau - \tau^\alpha$  ▷ ensures the transformation is bijective
6:   for  $i$  in 1 to  $N$  do
7:      $x_i \leftarrow (x_i - \bar{x})/\sigma_x$  ▷ standardize
8:     if  $x_i > \tau$  then
9:        $x_i \leftarrow (|x_i|^\alpha + \delta) \cdot \text{sign}(x_i)$  ▷ scale standardized values beyond  $\tau$ 
10:    end if
11:     $x_i \leftarrow x_i \cdot \sigma_x + \bar{x}$  ▷ undo standardization
12:  end for
13: end procedure
```

---

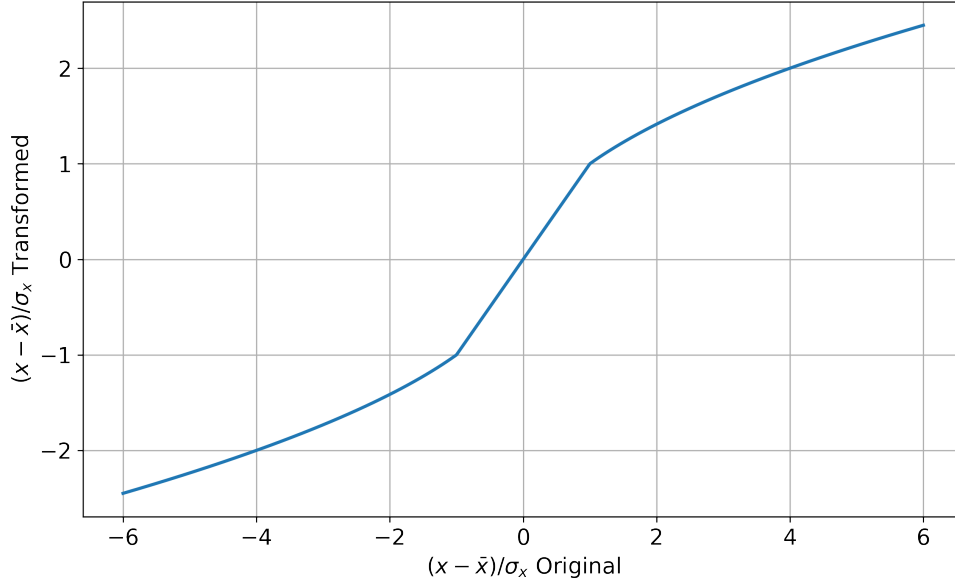


Figure 2.4: Transformation defined in Algorithm 1 using  $\tau = 1$  and  $\alpha = 0.5$ , for the purpose of reducing large gaps in the discretized data set by scaling outliers above  $\tau$  standard deviations.

---

**Transformed Log Returns Data Set Statistics**

---

| Currency Pair | Mean                  | Standard Deviation   |
|---------------|-----------------------|----------------------|
| EURUSD        | $5.54 \cdot 10^{-5}$  | $4.88 \cdot 10^{-3}$ |
| GBPUSD        | $1.66 \cdot 10^{-5}$  | $4.48 \cdot 10^{-3}$ |
| USDCAD        | $-6.42 \cdot 10^{-5}$ | $4.21 \cdot 10^{-3}$ |
| USDJPY        | $-4.68 \cdot 10^{-5}$ | $4.93 \cdot 10^{-3}$ |

---

Table 2.3: Statistics of the outlier power-transformed log returns data set.

is below (above) the historical median it is assigned a value of 0 (1) to indicate the low (high) volatility regime. The 3-month rolling volatilities versus their historical medians are

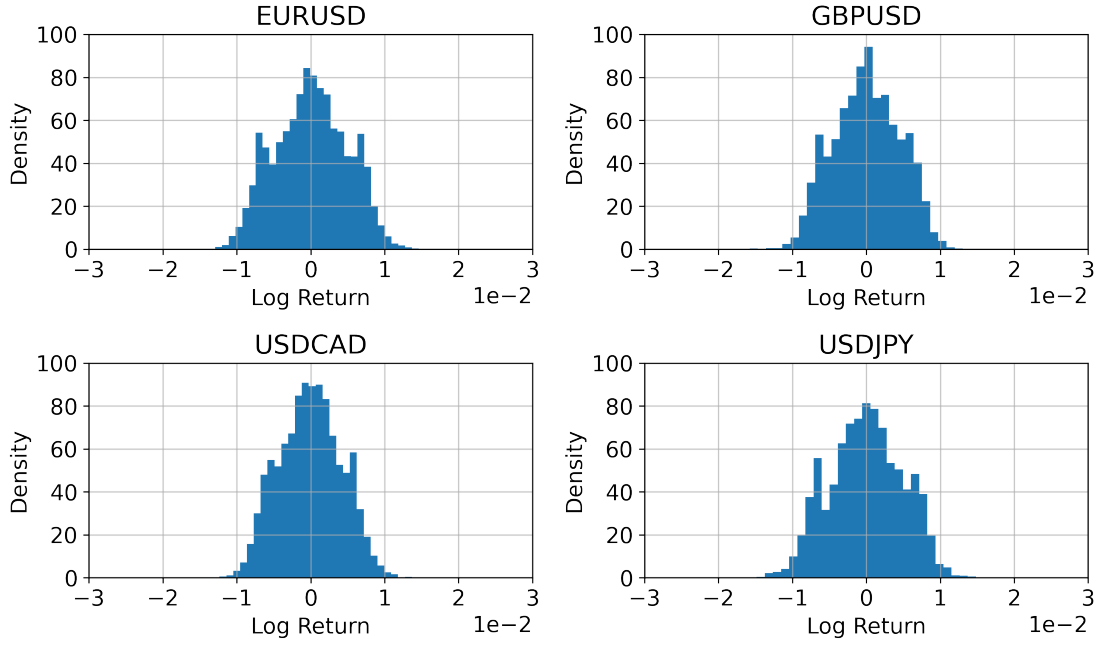


Figure 2.5: Histograms of the outlier power-transformed log returns data set.

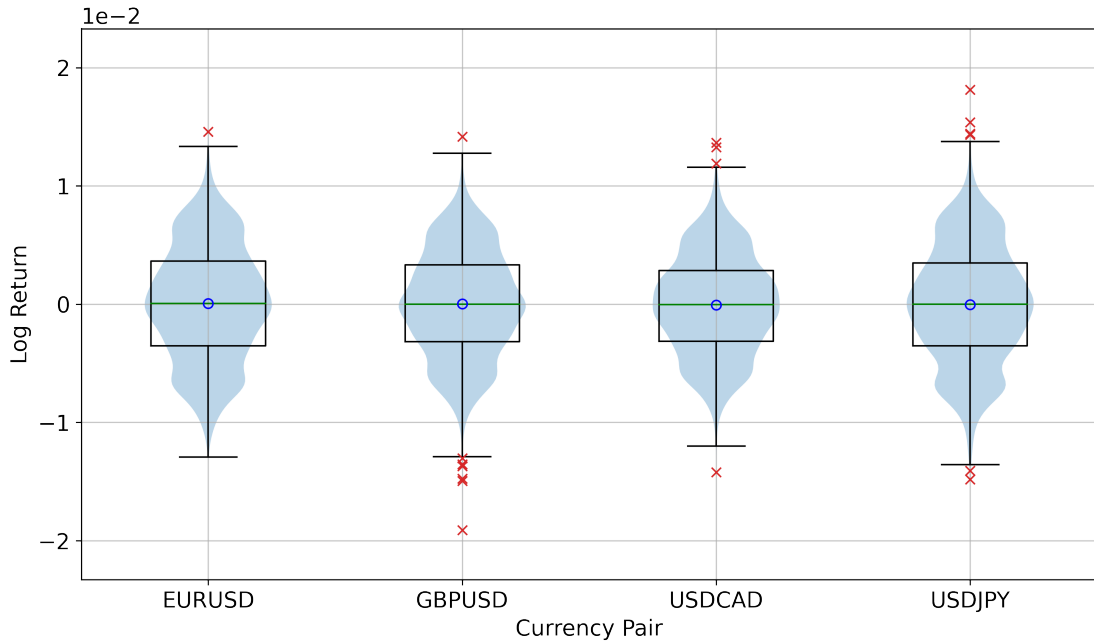


Figure 2.6: Violin and box plot of the outlier power-transformed log returns data set illustrating the distribution of the rescaled outliers.

plotted in Fig. 2.7.

These additional binary indicator variables are then concatenated onto the training data set and fed to the model to make it more flexible by allowing for the model outputs to be conditioned on a specific volatility regime. Adding one indicator for each of the four currency pairs increases the number of rows in our training data set by four, thus the

volatility-concatenated data set is in the space  $\{0, 1\}^{68 \times N}$ .

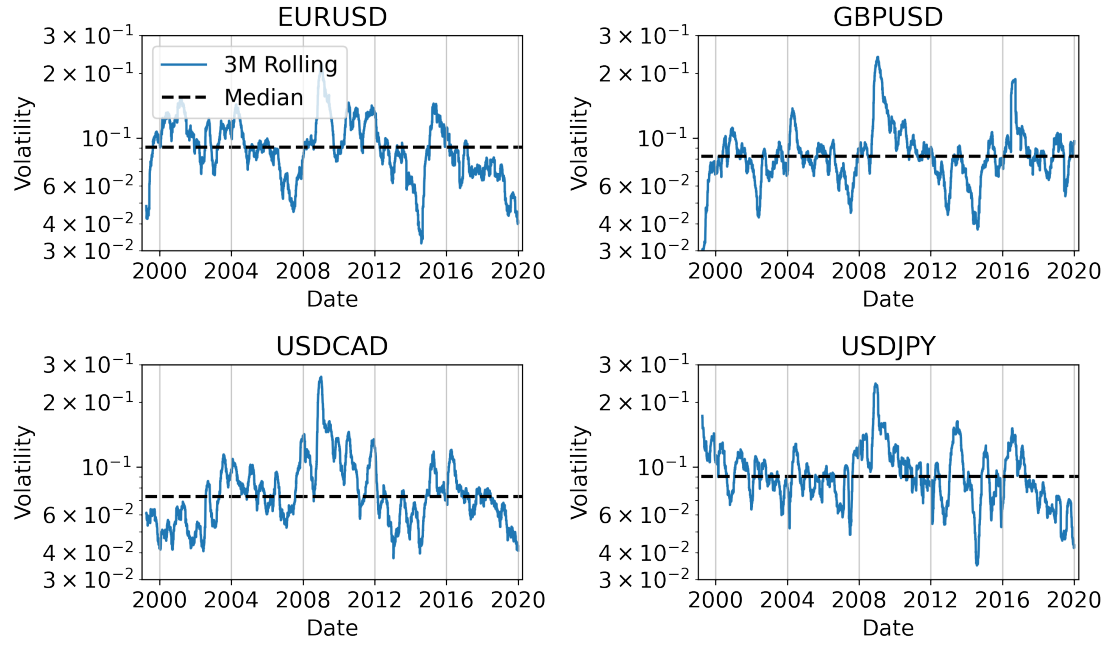


Figure 2.7: 3-month rolling volatilities of the log returns data set compared with their historical medians.



## Chapter 3

# The Classical Restricted Boltzmann Machine

### 3.1 Theory

The restricted Boltzmann machine (RBM) is an energy-based model defined by the energy function [11]

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}) &= - \sum_{i=1}^{n_v} a_i v_i - \sum_{j=1}^{n_h} b_j h_j - \sum_{i=1}^{n_v} \sum_{j=1}^{n_h} v_i w_{ij} h_j \\ &= -\mathbf{a}^\top \mathbf{v} - \mathbf{b}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h}, \end{aligned} \quad (3.1)$$

where

- $\mathbf{v} \in \{0, 1\}^{n_v}$  represents the visible units, with associated bias vector  $\mathbf{a} \in \mathbb{R}^{n_v}$ .
- $\mathbf{h} \in \{0, 1\}^{n_h}$  represents the hidden units, with associated bias vector  $\mathbf{b} \in \mathbb{R}^{n_h}$ .
- $\mathbf{W} \in \mathbb{R}^{n_v \times n_h}$  represents the weights corresponding to the interaction strengths between visible and hidden units.

It is termed *restricted* due to the fact that there are no intralayer connections, i.e., visible units are only connected to hidden units, and vice versa. An example diagram is depicted in Fig. 3.1.

The probability to find the system in the configuration  $(\mathbf{v}, \mathbf{h})$  is given by the Boltzmann distribution (with  $\beta = 1/kT = 1$ )

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})}, \quad (3.2)$$

with intractable [12] partition function

$$Z = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}, \quad (3.3)$$

where  $\sum_{\mathbf{v}, \mathbf{h}}$  denotes the sum over all possible configurations of  $\mathbf{v}$  and  $\mathbf{h}$ .

The imposed restrictions on intralayer connections enable us to write the conditional probabilities of the layers as the product of the individual units' probabilities<sup>1</sup> (see Ap-

---

<sup>1</sup>Here  $\sigma(x)$  is the element-wise logistic sigmoid function and  $\odot$  denotes element-wise multiplication.

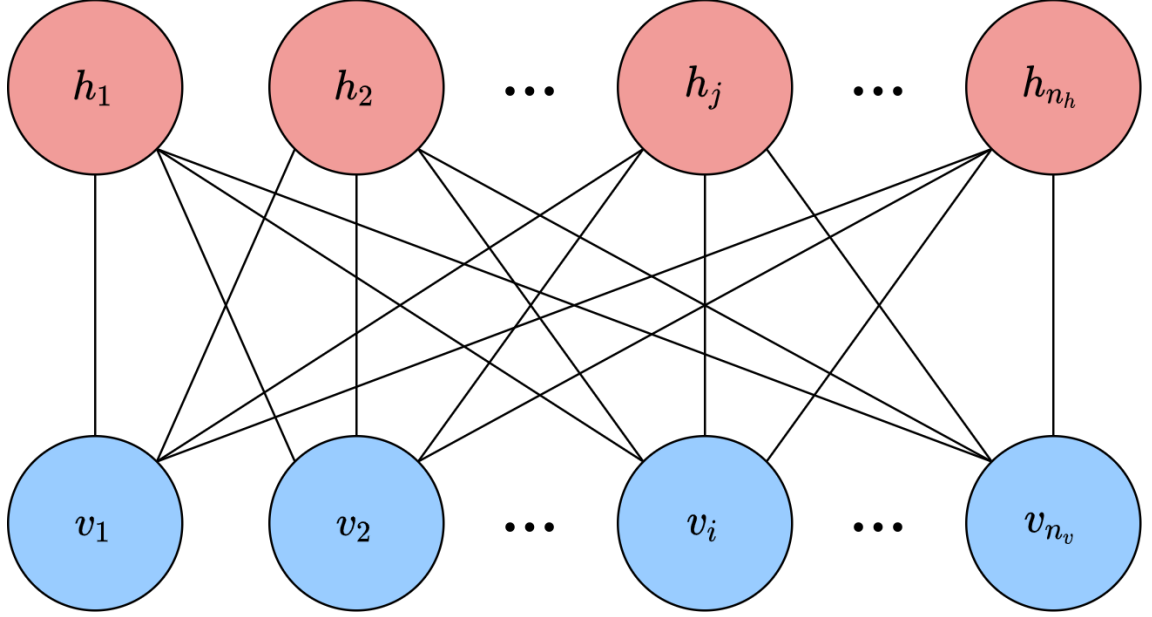


Figure 3.1: Diagram of a restricted Boltzmann machine with  $n_v$  visible units and  $n_h$  hidden units.

pendix B.1 for derivation)

$$\begin{aligned}
 p(\mathbf{h}|\mathbf{v}) &= \prod_{j=1}^{n_h} \sigma((2\mathbf{h} - 1) \odot (\mathbf{b} + \mathbf{W}^\top \mathbf{v}))_j, \\
 p(\mathbf{v}|\mathbf{h}) &= \prod_{i=1}^{n_v} \sigma((2\mathbf{v} - 1) \odot (\mathbf{a} + \mathbf{W}\mathbf{h}))_i.
 \end{aligned} \tag{3.4}$$

### 3.1.1 Optimizing an RBM

Due to the intractability of the partition function, the model cannot be solved exactly in general, thus we resort to other methods to optimize it such as likelihood maximization via gradient descent. For data set distribution  $p_{\text{data}}$  and parameters  $\theta = (\mathbf{W}, \mathbf{a}, \mathbf{b})$ , the log-likelihood is given by

$$\begin{aligned}
 \ell(\theta) &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log p(\mathbf{v}) \\
 &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \left( \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \right),
 \end{aligned} \tag{3.5}$$

with gradients (see Appendix B.2 for derivation)

$$\begin{aligned}
 \partial_{w_{ij}} \ell(\theta) &= \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}, \\
 \partial_{a_i} \ell(\theta) &= \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}, \\
 \partial_{b_j} \ell(\theta) &= \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}}.
 \end{aligned} \tag{3.6}$$

The part of the gradient under the data set distribution is referred to as the *positive* phase, and the part under the model distribution is referred to as the *negative* phase. It is trivial to compute the expectation values in the positive phase, but not so much in the negative phase because  $p(\mathbf{v})$  cannot be sampled directly.

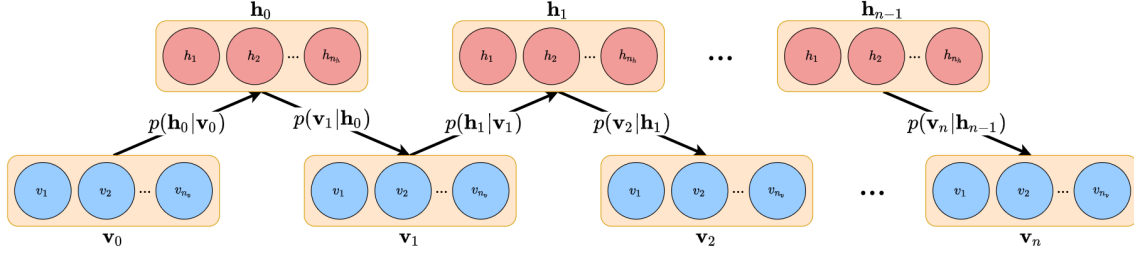


Figure 3.2: Illustration of the  $n$ -step Gibbs sampling procedure.

In practice the negative phase expectation values are sampled using a Markov chain Monte Carlo (MCMC) method. This is done via Gibbs sampling [13], which uses the conditional probabilities  $p(\mathbf{h}|\mathbf{v})$  and  $p(\mathbf{v}|\mathbf{h})$ . One starts with a visible vector and then samples the hidden units conditioned on the visible units, followed by sampling the visible units conditioned on the hidden units, and so forth until the desired thermalization threshold is reached. The number of steps required to reach thermalization is model dependent and can be estimated by analyzing the autocorrelations of a sample chain generated by the model. The algorithm for Gibbs sampling is given in Algorithm 2 and illustrated in Fig. 3.2. The algorithm is presented in a vectorized format for brevity.

---

**Algorithm 2** Gibbs Sampling

---

```

1: procedure GIBBS( $\mathbf{v}, n, \mathbf{W}, \mathbf{a}, \mathbf{b}$ )
2:    $n_v \leftarrow \text{length}(\mathbf{a})$ 
3:    $n_h \leftarrow \text{length}(\mathbf{b})$ 
4:   for  $k$  in 1 to  $n$  do
5:      $\mathbf{r} \sim \text{Uniform}(0, 1, n_h)$ 
6:      $\mathbf{h} \leftarrow \mathbf{r} < \sigma(\mathbf{b} + \mathbf{W}^\top \mathbf{v})$   $\triangleright \sigma, <$  applied element-wise
7:      $\mathbf{r} \sim \text{Uniform}(0, 1, n_v)$ 
8:      $\mathbf{v} \leftarrow \mathbf{r} < \sigma(\mathbf{a} + \mathbf{W}\mathbf{h})$   $\triangleright \sigma, <$  applied element-wise
9:   end for
10:  return  $\mathbf{v}$ 
11: end procedure

```

---

The  $\text{Uniform}(a, b, n)$  function in Algorithm 2 produces a length  $n$  vector of uniform i.i.d. random variables on the interval  $[a, b]$ , and the  $<$  operator acts element-wise with  $(\text{true}, \text{false}) \mapsto (1, 0)$ .

The standard procedure for training an RBM is called  $n$ -step contrastive divergence (CD- $n$ ), with  $n$  often taken to be one in practice [13]. The algorithm is detailed in Algorithm 3, where one can see that  $n$  corresponds to how many Gibbs sampling steps are between the positive and negative phase gradients. Applying the algorithm to a mini-batch is essentially the same except that one divides the learning rate by the size of the mini-batch to get a mini-batch averaged gradient.

## 3.2 The Classical Market Generator

In *The Market Generator* [5] by Kondratyev and Schwarz, they show how an RBM can be used as a generative model to produce synthetic market data. Specifically, they study how it performs on the log returns of forex data for the same currency pairs we use here for the time period 1999-2019. In this section we use some of the same metrics, as well as a couple additional ones, so that we can verify our models achieve similar performance to theirs, as

---

**Algorithm 3**  $n$ -Step Contrastive Divergence (CD- $n$ )

---

|                                                                                                               |                                                    |
|---------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| 1: <b>procedure</b> CD( $\mathbf{v}_+, n, \mathbf{W}, \mathbf{a}, \mathbf{b}, \eta$ )                         | $\triangleright \mathbf{v}_+$ is a training sample |
| 2: $\mathbf{h}_+ \leftarrow \sigma(\mathbf{b} + \mathbf{W}^\top \mathbf{v}_+)$                                | $\triangleright \sigma$ applied element-wise       |
| 3: $\mathbf{v}_- \leftarrow \text{Gibbs}(\mathbf{v}_+, n, \mathbf{W}, \mathbf{a}, \mathbf{b})$                |                                                    |
| 4: $\mathbf{h}_- \leftarrow \sigma(\mathbf{b} + \mathbf{W}^\top \mathbf{v}_-)$                                | $\triangleright \sigma$ applied element-wise       |
| 5: $\mathbf{W} \leftarrow \mathbf{W} + \eta(\mathbf{v}_+ \mathbf{h}_+^\top - \mathbf{v}_- \mathbf{h}_-^\top)$ |                                                    |
| 6: $\mathbf{a} \leftarrow \mathbf{a} + \eta(\mathbf{v}_+ - \mathbf{v}_-)$                                     |                                                    |
| 7: $\mathbf{b} \leftarrow \mathbf{b} + \eta(\mathbf{h}_+ - \mathbf{h}_-)$                                     |                                                    |
| 8: <b>return</b> $\mathbf{W}, \mathbf{a}, \mathbf{b}$                                                         |                                                    |
| 9: <b>end procedure</b>                                                                                       |                                                    |

---

well as give us a good reference point to compare our quantum models within Chapter 4.

### 3.2.1 Models

We train and analyze four RBM models using variations of the filtered data set from Chapter 2, each with slightly different preprocessing procedures denoted by:

- (B): base data set.
- (X): base data set transformed using Algorithm 1.
- (V): base data set with additional volatility indicators.
- (XV): base data set transformed using Algorithm 1 with additional volatility indicators.

The models here have 64 (68 for ones with volatility indicators) visible units and 30 hidden units (the same as in [5]) to act as regularized autoencoders. We use a mini-batch size of 10, and an initial learning rate of  $10^{-3}$  that decays by a factor of half every 1000 epochs after epoch 5000 as defined in Appendix A.3, for a total of  $10^4$  epochs. We base the models on a modified version of scikit-learn’s [14] BernoulliRBM class, which we forked <sup>2</sup> to implement the ability to use a learning rate schedule with the BernoulliRBM class.

One of the drawbacks of the RBM is that it is not easy to track the training progress for our use case, as the pseudolikelihood metric implemented by the scikit-learn package is not necessarily a good proxy for our models’ performances. The Kullback-Leibler (KL) divergence of  $p_{\text{model}}$  from  $p_{\text{data}}$ , denoted  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$ , is a suitable quantity to track model performance as it measures the information loss associated with using the model distribution  $p_{\text{model}}$  to approximate the data set distribution  $p_{\text{data}}$  (more information in Appendix A.5). However, due to the high number of epochs and the thermalization requirements of samples generated by the RBM, this is not very feasible because generating samples to compute the KL divergence every epoch significantly increases model training times. Therefore, we only present the final results of the models.

### 3.2.2 Results

#### Autocorrelations

As mentioned before, the classical RBM sampling method is based on an MCMC algorithm, and thus samples produced via this method are autocorrelated. Therefore, we first examine the autocorrelations to see how dependent samples are on the previous, so that we can get an idea of how many Gibbs steps are needed between samples to consider them statistically

---

<sup>2</sup><https://github.com/cameronperot/scikit-learn/>

independent. We use Gibbs sample chains of length  $10^8$  for this analysis. More information about the autocorrelation function and time can be found in Appendix A.4.

Fig. 3.3 shows the autocorrelation functions for the various models and currency pairs. It is immediately clear that the autocorrelations fall off much sooner for the models trained on the transformed data sets for all currency pairs. This observation is confirmed by examining the integrated autocorrelation times in Table 3.1.

It is not immediately clear why the transformed data sets lead to such shorter integrated autocorrelation times, but this is a welcome trend as it means that less sampling steps are required to reach thermalization.

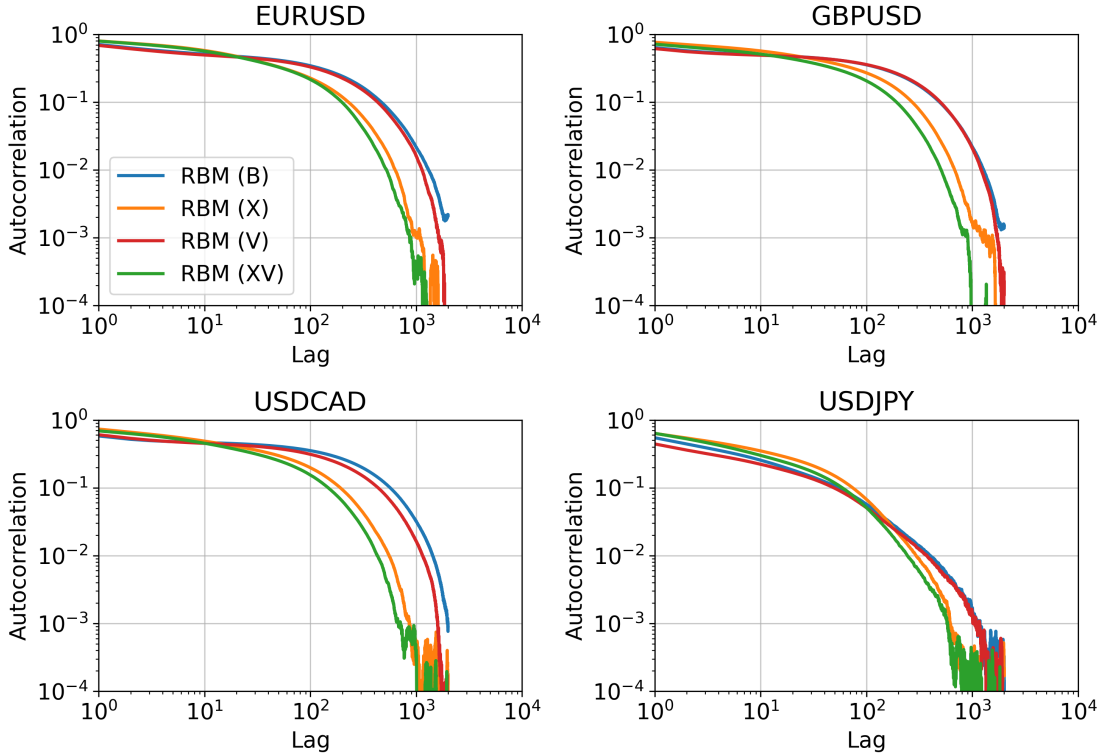


Figure 3.3: Autocorrelation functions of the RBM models.

| Integrated Autocorrelation Times |         |         |         |          |
|----------------------------------|---------|---------|---------|----------|
| Currency Pair                    | RBM (B) | RBM (X) | RBM (V) | RBM (XV) |
| EURUSD                           | 295.7   | 147.5   | 267.4   | 129.2    |
| GBPUSD                           | 307.0   | 173.2   | 308.9   | 121.6    |
| USDCAD                           | 340.6   | 120.0   | 258.8   | 91.3     |
| USDJPY                           | 33.9    | 46.7    | 28.8    | 36.7     |

Table 3.1: Integrated autocorrelation times of the RBM models.

The results in the rest of this section are derived from an ensemble of 100 sample sets consisting of  $10^4$  samples each, and  $10^4$  Gibbs sampling steps between samples to ensure thermalization.

## Marginal Distributions

To get an idea of how well the models perform, we examine the KL divergences of the marginal distributions of each currency pair in Table 3.2. Here we observe that all models reproduce the marginal distributions quite well, but the models trained on the transformed data sets perform slightly better, particularly on the USDCAD marginal. The performance of the models on the marginal distributions is also visualized with Q-Q plots in Fig. 3.4. More information on how the KL divergences are computed can be found in Appendix A.5.1.

| $D_{\text{KL}}(p_{\text{data}}    p_{\text{model}})$ |                   |                   |                   |                   |
|------------------------------------------------------|-------------------|-------------------|-------------------|-------------------|
| Currency Pair                                        | RBM (B)           | RBM (X)           | RBM (V)           | RBM (XV)          |
| EURUSD                                               | $0.010 \pm 0.001$ | $0.007 \pm 0.001$ | $0.011 \pm 0.002$ | $0.009 \pm 0.001$ |
| GBPUSD                                               | $0.007 \pm 0.001$ | $0.006 \pm 0.001$ | $0.011 \pm 0.001$ | $0.007 \pm 0.001$ |
| USDCAD                                               | $0.017 \pm 0.002$ | $0.007 \pm 0.001$ | $0.015 \pm 0.002$ | $0.008 \pm 0.001$ |
| USDJPY                                               | $0.008 \pm 0.001$ | $0.007 \pm 0.001$ | $0.010 \pm 0.001$ | $0.009 \pm 0.001$ |
| Mean                                                 | $0.010 \pm 0.001$ | $0.007 \pm 0.001$ | $0.011 \pm 0.002$ | $0.008 \pm 0.001$ |

Table 3.2: KL divergences of the RBM models. The values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

## Correlations

The distribution is in a sense more than just the sum of its parts. Beyond learning the marginal distributions, the models should also capture the correlations between the currency pairs. To verify this, we turn to the correlation coefficients in Table 3.3 to see how well the models capture the correlations. We find that the models reproduce the structure of the correlation coefficients reasonably well, with the models trained on the transformed data sets encoding more of the behavior.

## Volatilities

Examining the historical volatilities in Table 3.4 confirms the models can produce synthetic data with similar volatilities to the training data set, albeit marginally higher in all cases.

## Tails

It is extremely important for the models to learn the tail events because these play a crucial role in financial risk management. The models trained on the transformed data sets reproduce the lower tails a little better for most currency pairs, but overestimate some of the upper tails. It is difficult to say overall if one model performs better than another here, as it really depends on what one wants to do with the generated data.

We also study the tail concentration functions (see Appendix A.6 for definitions and interpretations) between currency pairs in Fig. 3.5. Here we see that all models perform quite well for the most part except for a few of the extreme regions in the EURUSD/GBPUSD, EURUSD/USDJPY, and GBPUSD/USDJPY plots.

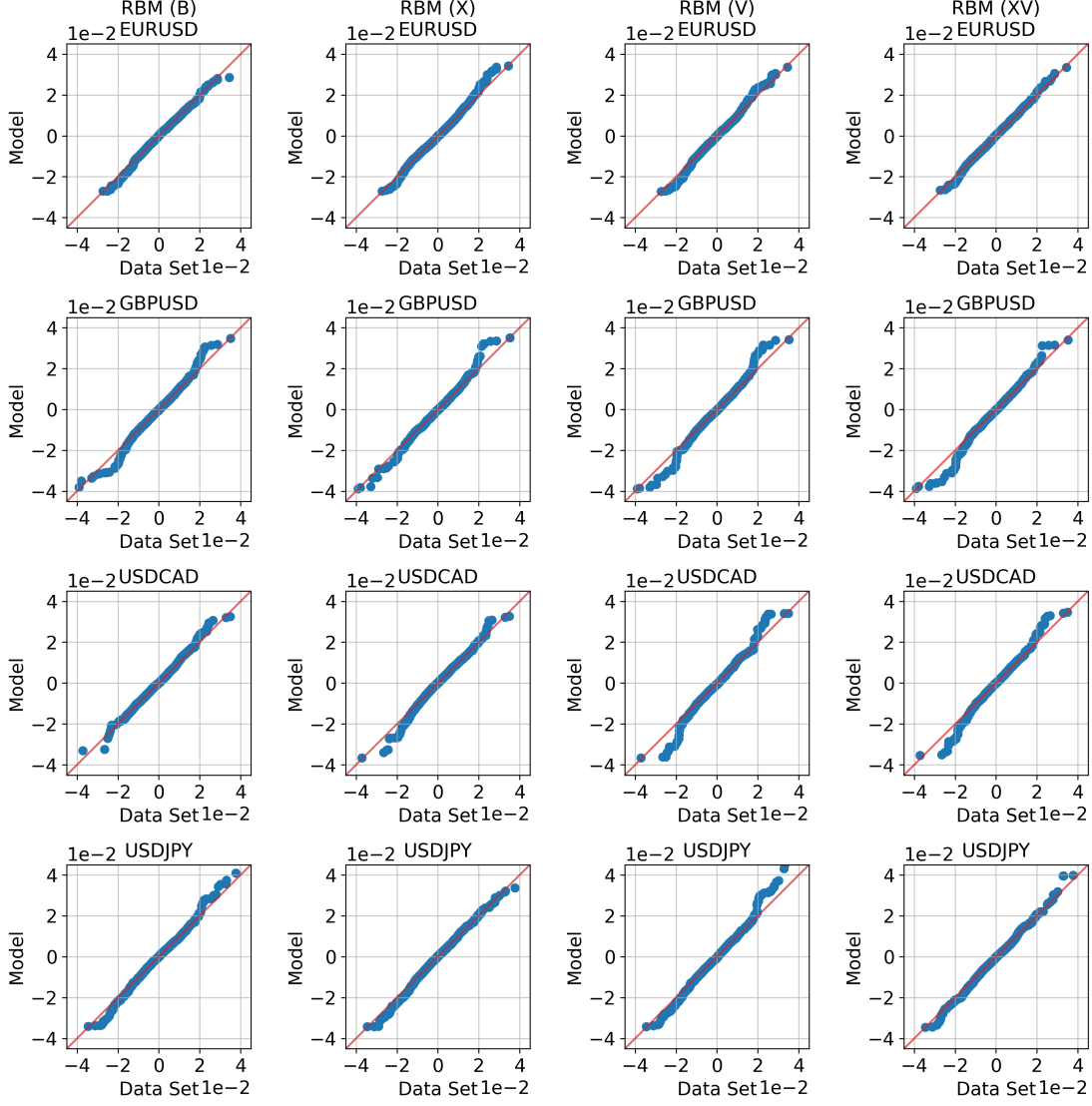


Figure 3.4: Log return Q-Q plots of the RBM models for each currency pair. Note that these plots only use the same number of samples as the size of the training data set (5165), and thus are not entirely representative of the models’ performances.

### Conditional Sampling

For the data sets with additional volatility indicators, we have the ability to condition on these indicators to sample from a specific volatility regime. This is useful, for example, if we are trying to generate real-world data that fits the current volatility landscape.

This leads us to look at the conditional volatilities, i.e., seeing how well the models reproduce the volatilities from the two volatility regimes. Laid out in Table 3.6, we observe that the samples produced by the RBMs have slightly lower (higher) volatilities in the high (low) regime, but are overall in good agreement with the data set.

#### 3.2.3 Summary

The classical RBM results presented in this section are in line with those obtained by Kondratyev and Schwarz in [5], and the differences can likely be accounted for by the different data sets used in training (e.g., different sources, different filtering, etc.), model

| Correlation Coefficients |          |          |         |                  |                  |                  |
|--------------------------|----------|----------|---------|------------------|------------------|------------------|
| Currency Pairs           | Data Set |          |         | RBM (B)          |                  |                  |
|                          | Pearson  | Spearman | Kendall | Pearson          | Spearman         | Kendall          |
| EURUSD/GBPUSD            | 0.62     | 0.62     | 0.44    | $0.48 \pm 0.01$  | $0.53 \pm 0.01$  | $0.38 \pm 0.01$  |
| EURUSD/USDCAD            | -0.44    | -0.41    | -0.29   | $-0.33 \pm 0.01$ | $-0.34 \pm 0.01$ | $-0.24 \pm 0.01$ |
| EURUSD/USDJPY            | -0.26    | -0.30    | -0.21   | $-0.21 \pm 0.01$ | $-0.25 \pm 0.01$ | $-0.17 \pm 0.01$ |
| GBPUSD/USDCAD            | -0.42    | -0.37    | -0.26   | $-0.31 \pm 0.01$ | $-0.33 \pm 0.01$ | $-0.22 \pm 0.01$ |
| GBPUSD/USDJPY            | -0.14    | -0.21    | -0.15   | $-0.15 \pm 0.01$ | $-0.18 \pm 0.01$ | $-0.13 \pm 0.01$ |
| USDCAD/USDJPY            | 0.00     | 0.06     | 0.04    | $0.06 \pm 0.01$  | $0.07 \pm 0.01$  | $0.05 \pm 0.01$  |

| Currency Pairs | RBM (X)          |                  |                  | RBM (V)          |                  |                  |
|----------------|------------------|------------------|------------------|------------------|------------------|------------------|
|                | Pearson          | Spearman         | Kendall          | Pearson          | Spearman         | Kendall          |
| EURUSD/GBPUSD  | $0.56 \pm 0.01$  | $0.59 \pm 0.01$  | $0.42 \pm 0.01$  | $0.48 \pm 0.01$  | $0.54 \pm 0.01$  | $0.38 \pm 0.01$  |
| EURUSD/USDCAD  | $-0.39 \pm 0.01$ | $-0.39 \pm 0.01$ | $-0.27 \pm 0.01$ | $-0.34 \pm 0.01$ | $-0.36 \pm 0.01$ | $-0.25 \pm 0.01$ |
| EURUSD/USDJPY  | $-0.24 \pm 0.01$ | $-0.29 \pm 0.01$ | $-0.19 \pm 0.01$ | $-0.20 \pm 0.01$ | $-0.23 \pm 0.01$ | $-0.16 \pm 0.01$ |
| GBPUSD/USDCAD  | $-0.36 \pm 0.01$ | $-0.35 \pm 0.01$ | $-0.24 \pm 0.01$ | $-0.30 \pm 0.01$ | $-0.33 \pm 0.01$ | $-0.22 \pm 0.01$ |
| GBPUSD/USDJPY  | $-0.16 \pm 0.01$ | $-0.20 \pm 0.01$ | $-0.13 \pm 0.01$ | $-0.14 \pm 0.01$ | $-0.17 \pm 0.01$ | $-0.12 \pm 0.01$ |
| USDCAD/USDJPY  | $0.05 \pm 0.01$  | $0.07 \pm 0.01$  | $0.05 \pm 0.01$  | $0.05 \pm 0.01$  | $0.07 \pm 0.01$  | $0.05 \pm 0.01$  |

| Currency Pairs | RBM (XV)         |                  |                  |
|----------------|------------------|------------------|------------------|
|                | Pearson          | Spearman         | Kendall          |
| EURUSD/GBPUSD  | $0.54 \pm 0.01$  | $0.59 \pm 0.01$  | $0.42 \pm 0.01$  |
| EURUSD/USDCAD  | $-0.39 \pm 0.01$ | $-0.38 \pm 0.01$ | $-0.26 \pm 0.01$ |
| EURUSD/USDJPY  | $-0.22 \pm 0.01$ | $-0.27 \pm 0.01$ | $-0.19 \pm 0.01$ |
| GBPUSD/USDCAD  | $-0.36 \pm 0.01$ | $-0.36 \pm 0.01$ | $-0.24 \pm 0.01$ |
| GBPUSD/USDJPY  | $-0.16 \pm 0.01$ | $-0.20 \pm 0.01$ | $-0.13 \pm 0.01$ |
| USDCAD/USDJPY  | $0.05 \pm 0.01$  | $0.07 \pm 0.01$  | $0.05 \pm 0.01$  |

Table 3.3: Correlation coefficients of the data set vs. samples generated by the RBM models. The RBM values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

| Historical Volatilities |          |                      |                      |                      |                      |
|-------------------------|----------|----------------------|----------------------|----------------------|----------------------|
| Currency Pair           | Data Set | RBM (B)              | RBM (X)              | RBM (V)              | RBM (XV)             |
| EURUSD                  | 9.78%    | $9.98\% \pm 0.11\%$  | $10.10\% \pm 0.10\%$ | $10.18\% \pm 0.11\%$ | $10.27\% \pm 0.10\%$ |
| GBPUSD                  | 8.98%    | $9.34\% \pm 0.11\%$  | $9.38\% \pm 0.12\%$  | $9.55\% \pm 0.12\%$  | $9.53\% \pm 0.11\%$  |
| USDCAD                  | 8.56%    | $8.98\% \pm 0.13\%$  | $9.01\% \pm 0.11\%$  | $9.29\% \pm 0.13\%$  | $9.12\% \pm 0.12\%$  |
| USDJPY                  | 10.02%   | $10.26\% \pm 0.13\%$ | $10.42\% \pm 0.14\%$ | $10.82\% \pm 0.16\%$ | $10.46\% \pm 0.12\%$ |

Table 3.4: Historical volatilities of the data set vs. samples generated by the RBM models. The RBM values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

hyperparameters, and the stochastic nature of the models. This further confirms that the RBM is performant and can be used to generate synthetic data from distributions with intricate structures, such as the correlations and volatilities seen here.

Overall it is difficult to say if one of the models performs better than the others, as it depends on the desired use case, but the models trained on the transformed data sets do yield lower KL divergence values and capture more of the correlations between currency pairs. This offers evidence that the results might be able to be further improved through the use of more advanced data preprocessing methods. We do not investigate these possibilities any further though, given that this is not the main scope of this thesis. The results in this section act mainly as a point of reference to compare the quantum models within the next chapter.

| Lower Tails (1st Percentile) |          |                    |                    |                    |                    |
|------------------------------|----------|--------------------|--------------------|--------------------|--------------------|
| Currency Pair                | Data Set | RBM (B)            | RBM (X)            | RBM (V)            | RBM (XV)           |
| EURUSD                       | -1.64%   | -1.80% $\pm$ 0.04% | -1.68% $\pm$ 0.05% | -1.90% $\pm$ 0.05% | -1.76% $\pm$ 0.06% |
| GBPUSD                       | -1.47%   | -1.59% $\pm$ 0.04% | -1.57% $\pm$ 0.05% | -1.63% $\pm$ 0.05% | -1.66% $\pm$ 0.06% |
| USDCAD                       | -1.40%   | -1.54% $\pm$ 0.05% | -1.57% $\pm$ 0.06% | -1.59% $\pm$ 0.05% | -1.58% $\pm$ 0.06% |
| USDJPY                       | -1.70%   | -2.03% $\pm$ 0.07% | -1.92% $\pm$ 0.06% | -2.17% $\pm$ 0.09% | -1.96% $\pm$ 0.06% |

| Upper Tails (99th Percentile) |          |                   |                   |                   |                   |
|-------------------------------|----------|-------------------|-------------------|-------------------|-------------------|
| Currency Pair                 | Data Set | RBM (B)           | RBM (X)           | RBM (V)           | RBM (XV)          |
| EURUSD                        | 1.62%    | 1.59% $\pm$ 0.04% | 1.84% $\pm$ 0.06% | 1.70% $\pm$ 0.04% | 1.81% $\pm$ 0.05% |
| GBPUSD                        | 1.42%    | 1.45% $\pm$ 0.04% | 1.54% $\pm$ 0.04% | 1.53% $\pm$ 0.03% | 1.57% $\pm$ 0.04% |
| USDCAD                        | 1.51%    | 1.61% $\pm$ 0.04% | 1.53% $\pm$ 0.05% | 1.60% $\pm$ 0.05% | 1.56% $\pm$ 0.05% |
| USDJPY                        | 1.59%    | 1.56% $\pm$ 0.04% | 1.60% $\pm$ 0.05% | 1.61% $\pm$ 0.04% | 1.61% $\pm$ 0.05% |

Table 3.5: Lower and upper tails, i.e., 1st and 99th percentiles, of the data set vs. samples generated by the RBM models. The RBM values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

| Conditional Volatilities |            |                   |                   |             |                    |                    |
|--------------------------|------------|-------------------|-------------------|-------------|--------------------|--------------------|
| Currency Pair            | Low Regime |                   |                   | High Regime |                    |                    |
|                          | Data Set   | RBM (V)           | RBM (XV)          | Data Set    | RBM (V)            | RBM (XV)           |
| EURUSD                   | 6.72%      | 7.67% $\pm$ 0.23% | 7.60% $\pm$ 0.24% | 13.04%      | 13.19% $\pm$ 0.31% | 12.92% $\pm$ 0.27% |
| GBPUSD                   | 6.67%      | 7.45% $\pm$ 0.21% | 7.50% $\pm$ 0.22% | 12.69%      | 12.13% $\pm$ 0.31% | 11.61% $\pm$ 0.28% |
| USDCAD                   | 6.05%      | 6.72% $\pm$ 0.22% | 6.40% $\pm$ 0.21% | 12.86%      | 12.53% $\pm$ 0.37% | 12.14% $\pm$ 0.30% |
| USDJPY                   | 7.36%      | 9.01% $\pm$ 0.32% | 8.76% $\pm$ 0.27% | 13.15%      | 12.63% $\pm$ 0.38% | 12.41% $\pm$ 0.31% |

Table 3.6: Conditional historical volatilities of the data set vs. samples generated by the RBM models. The RBM values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

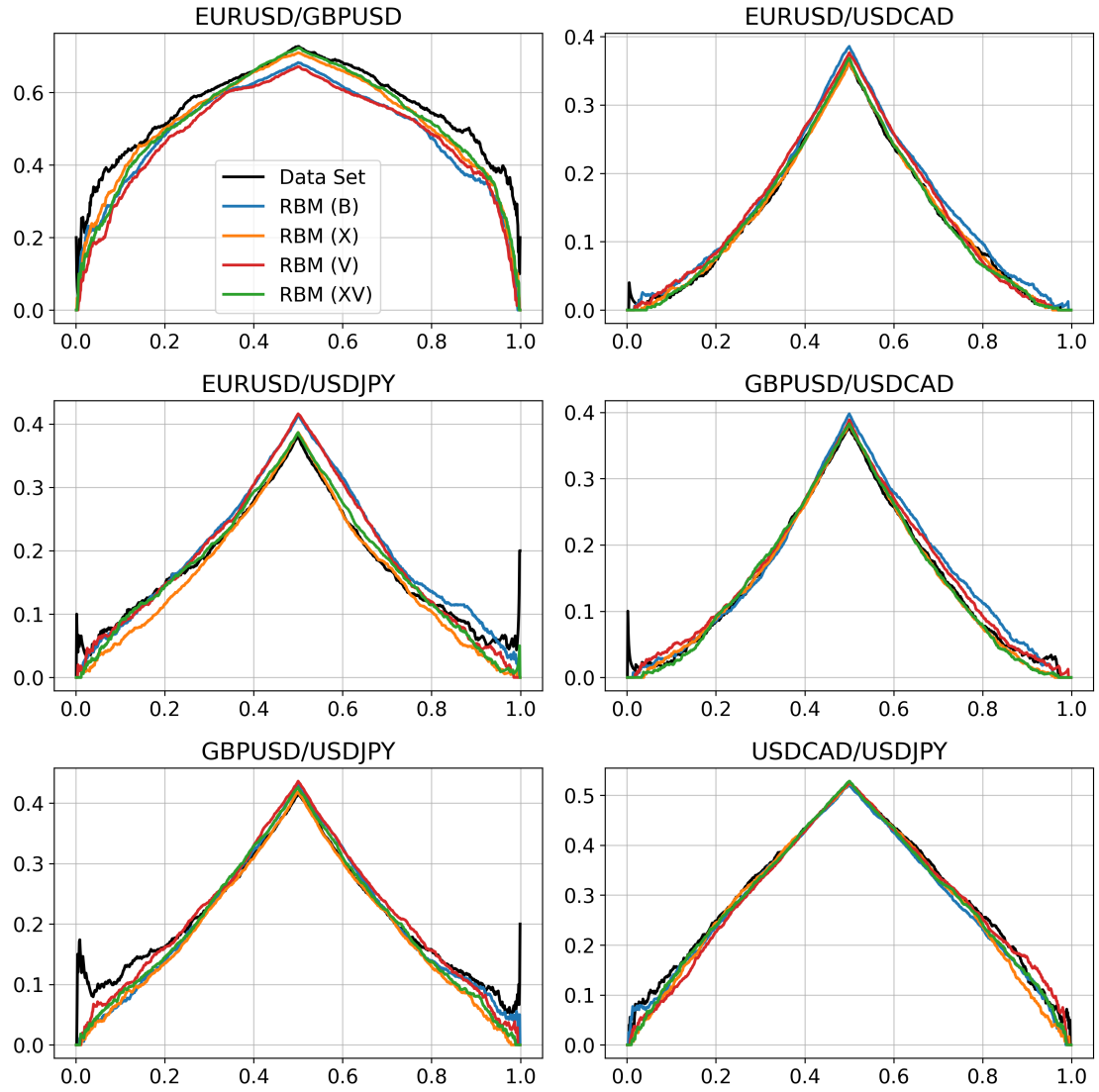


Figure 3.5: Tail concentration functions of the data set vs. samples generated by the RBM models.

## Chapter 4

# The Quantum Boltzmann Machine

### 4.1 Theory

The Quantum Boltzmann Machine detailed here is based on the work in *Quantum Boltzmann Machine* by Amin et al. [15]. In this section we use spin eigenvalues  $+1$  and  $-1$  rather than binary values  $0$  and  $1$ , respectively, in order to maintain consistency with the language of quantum mechanics. We start with the  $n$ -qubit Hamiltonian

$$H = - \sum_{i=1}^n \Gamma_i \sigma_i^x - \sum_{i=1}^n b_i \sigma_i^z - \sum_{i=1}^n \sum_{j=i+1}^n w_{ij} \sigma_i^z \sigma_j^z, \quad (4.1)$$

where

$$\begin{aligned} \sigma_i^x &= I^{\otimes i-1} \otimes \sigma_x \otimes I^{\otimes n-i}, \\ \sigma_i^z &= I^{\otimes i-1} \otimes \sigma_z \otimes I^{\otimes n-i}, \end{aligned} \quad (4.2)$$

with  $\sigma_x$  and  $\sigma_z$  being the Pauli  $x$  and  $z$  matrices, and  $I$  being the  $2 \times 2$  identity matrix. We denote the first  $n_v$  qubits as the visible units and the last  $n_h$  qubits as the hidden units, thus we have a total of  $n_v + n_h = n$  qubits.

The system's distribution is modeled by the density matrix

$$\rho = \frac{1}{Z} e^{-H}, \quad (4.3)$$

where  $e^{-H} = \sum_{n=0}^{\infty} \frac{1}{n!} (-H)^n$  is the matrix exponential, and  $Z = \text{tr}(e^{-H})$  is the partition function. The probability to observe the system in state  $|\mathbf{v}, \mathbf{h}\rangle$  is given by

$$p(\mathbf{v}, \mathbf{h}) = \text{tr}(|\mathbf{v}, \mathbf{h}\rangle \langle \mathbf{v}, \mathbf{h}| \rho), \quad (4.4)$$

and if we define the projection operator

$$\Lambda_{\mathbf{v}} = |\mathbf{v}\rangle \langle \mathbf{v}| \otimes I^{\otimes n_h}, \quad (4.5)$$

then the marginal probability to measure the visible units in state  $|\mathbf{v}\rangle$  is given by

$$p(\mathbf{v}) = \text{tr}(\Lambda_{\mathbf{v}} \rho). \quad (4.6)$$

Using the probabilities above we can obtain the log-likelihood, which for data set distribution  $p_{\text{data}}$  and parameters  $\theta = (\mathbf{W}, \mathbf{a}, \mathbf{b})$  is

$$\ell(\theta) = \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \text{tr}(\Lambda_{\mathbf{v}} \rho), \quad (4.7)$$

where  $\sum_{\mathbf{v}}$  denotes the sum over all possible configurations of  $\mathbf{v}$ .

#### 4.1.1 Optimizing a QBM

When optimizing a QBM, it is preferable to maximize the lower bound of the log-likelihood rather than maximizing the log-likelihood itself. The reason for this is that the partial derivative of the log-likelihood with respect to the parameters has a term which is computationally expensive to compute, as discussed in Appendix C.1. The lower bound of the log-likelihood is given by (see Appendix C.2 for derivation)

$$\tilde{\ell}(\theta) = \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \text{tr}(\rho_{\mathbf{v}}), \quad (4.8)$$

where we have what is referred to as the *clamped* Hamiltonian, which for a given visible vector  $\mathbf{v}$  is

$$H_{\mathbf{v}} = \langle \mathbf{v} | H | \mathbf{v} \rangle, \quad (4.9)$$

with corresponding clamped density matrix

$$\rho_{\mathbf{v}} = \frac{1}{Z_{\mathbf{v}}} e^{-H_{\mathbf{v}}}, \quad (4.10)$$

and  $Z_{\mathbf{v}} = \text{tr}(e^{-H_{\mathbf{v}}})$ . This is called clamped because the visible qubits are held to the classical state of the visible vector  $\mathbf{v}$ .

The associated derivatives with respect to the parameters of the lower bound are given by (see Appendix C.3 for derivation)

$$\begin{aligned} \partial_{w_{ij}} \tilde{\ell}(\theta) &= \langle \sigma_i^z \sigma_j^z \rangle_{\text{data}} - \langle \sigma_i^z \sigma_j^z \rangle_{\text{model}}, \\ \partial_{b_i} \tilde{\ell}(\theta) &= \langle \sigma_i^z \rangle_{\text{data}} - \langle \sigma_i^z \rangle_{\text{model}}, \end{aligned} \quad (4.11)$$

where  $\langle \cdot \rangle_{\text{data}}$  is the expectation value with respect to the data set, and  $\langle \cdot \rangle_{\text{model}}$  is the expectation value with respect to the original density matrix.

If connections are restricted within the hidden layer, then the hidden unit probabilities are independent in the positive phase and can be computed easily, as shown in Appendix C.3. This leads to positive phase expectation values of

$$\begin{aligned} \langle \sigma_i^z \rangle_{\text{data}} &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) v_i, \quad i \in \mathcal{I}_v, \\ \langle \sigma_i^z \rangle_{\text{data}} &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \frac{b'_i(\mathbf{v})}{D_i(\mathbf{v})} \tanh(D_i(\mathbf{v})), \quad i \in \mathcal{I}_h, \\ \langle \sigma_i^z \sigma_j^z \rangle_{\text{data}} &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) v_i v_j, \quad i, j \in \mathcal{I}_v, \\ \langle \sigma_i^z \sigma_j^z \rangle_{\text{data}} &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) v_i \frac{b'_j(\mathbf{v})}{D_j(\mathbf{v})} \tanh(D_j(\mathbf{v})), \quad i \in \mathcal{I}_v, \quad j \in \mathcal{I}_h, \end{aligned} \quad (4.12)$$

where  $b'_i(\mathbf{v}) = b_i + (\mathbf{W}^T \mathbf{v})_i$ ,  $D_i(\mathbf{v}) = \sqrt{\Gamma_i^2 + b'_i(\mathbf{v})^2}$ ,  $\mathcal{I}_v = \{1, \dots, n_v\}$  represents the visible qubit indices, and  $\mathcal{I}_h = \{n_v + 1, \dots, n\}$  represents the hidden qubit indices.

#### 4.1.2 Quantum Annealing

Quantum annealing, also known as adiabatic quantum computing, is a branch of quantum computing that is based on the adiabatic theorem, which in the (translated) words of Born and Fock [16]: "A physical system remains in its instantaneous eigenstate if a given perturbation is acting on it slowly enough and if there is a gap between the eigenvalue

and the rest of the Hamiltonian's spectrum." This can be achieved by implementing a Hamiltonian of the form [17]

$$H(s) = A(s)H_{\text{initial}} + B(s)H_{\text{final}}, \quad (4.13)$$

where  $s \in [0, 1]$ . For a linear anneal schedule  $s(t) = t/t_a$ , where  $t_a$  is the annealing time.  $H_{\text{initial}}$  is the initial Hamiltonian which describes the system at  $s = 0$  and is responsible for introducing quantum fluctuations.  $H_{\text{final}}$  is the final Hamiltonian which describes the system at  $s = 1$  and is responsible for encoding the problem defined by the user.

The functions  $A(s)$  and  $B(s)$  must be such that they satisfy the relations

$$\begin{aligned} A(0) &\gg B(0), \\ A(1) &\ll B(1). \end{aligned} \quad (4.14)$$

In essence, a quantum annealer starts in the ground state of the initial Hamiltonian, then slowly evolves the system over time so that it remains in the instantaneous ground state. By the time the annealing process is completed, the Hamiltonian is just that of the problem, and if the system evolved adiabatically, then it should have remained in the instantaneous ground state. Therefore, when the qubits are measured at the end, they should correspond to a low energy solution of the final Hamiltonian.

## D-Wave Quantum Annealer

D-Wave quantum annealers implement a time-dependent Hamiltonian of the form [18]

$$H(s) = A(s) \left( - \sum_{i=1}^n \sigma_i^x \right) + B(s) \left( \sum_{i=1}^n h_i \sigma_i^z + \sum_{i=1}^n \sum_{j=i+1}^n J_{ij} \sigma_i^z \sigma_j^z \right). \quad (4.15)$$

From this we see the initial Hamiltonian has the ground state where all qubits are aligned in the  $x$ -direction, i.e.,  $|+\rangle^{\otimes n}$ , which corresponds to an equal superposition of all possible states in the computational basis. The final Hamiltonian corresponds to the Ising model described by the  $h_i$  and  $J_{ij}$  values.

The quantum processing unit (QPU) is made up of superconducting qubits under the influence of external magnetic fluxes [17] that change the Hamiltonian from the initial to the final over the duration of the annealing process. These qubits are arranged in a graph structure similar to that seen in Fig. 4.1. The default anneal schedule for the D-Wave Advantage 4.1 is shown in Fig. 4.2

## Mapping the QBM to the D-Wave Quantum Annealer

As stated in [15], in order get a quantum annealer to sample from a quantum Boltzmann distribution, one would need to freeze the evolution at some point  $s^*$  during the annealing process and then perform the measurements. The authors go on to say that this can be done in practice using a nonuniform  $s(t)$  that anneals slowly in the beginning, then quenches the system (completes the annealing as fast as possible) at the freeze-out point  $s^*$ , if  $s^*$  is in the quasistatic regime. In an earlier paper [21], Amin showed that the quasistatic regime begins around 1  $\mu$ s for the D-Wave 2000Q, so it should not be an issue to reach the quasistatic regime for annealing times longer than 5  $\mu$ s.

Because a quantum annealer is a real-world physical device, samples generated with it have an associated temperature called the effective temperature. To be more specific, the corresponding density operator is of the form

$$\rho(s, T) = \frac{1}{Z} e^{-\beta H(s)}, \quad (4.16)$$

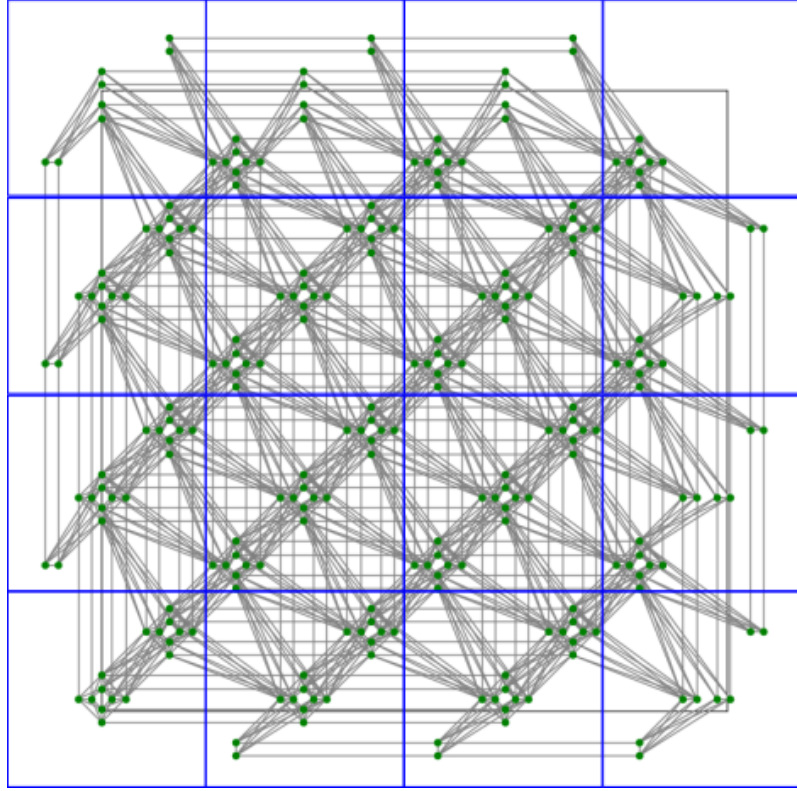


Figure 4.1: A lattice with  $4 \times 4$  Pegasus unit cells ( $P_4$ ). The D-Wave Advantage QPU is based on a lattice with  $16 \times 16$  Pegasus unit cells ( $P_{16}$ ) [19].

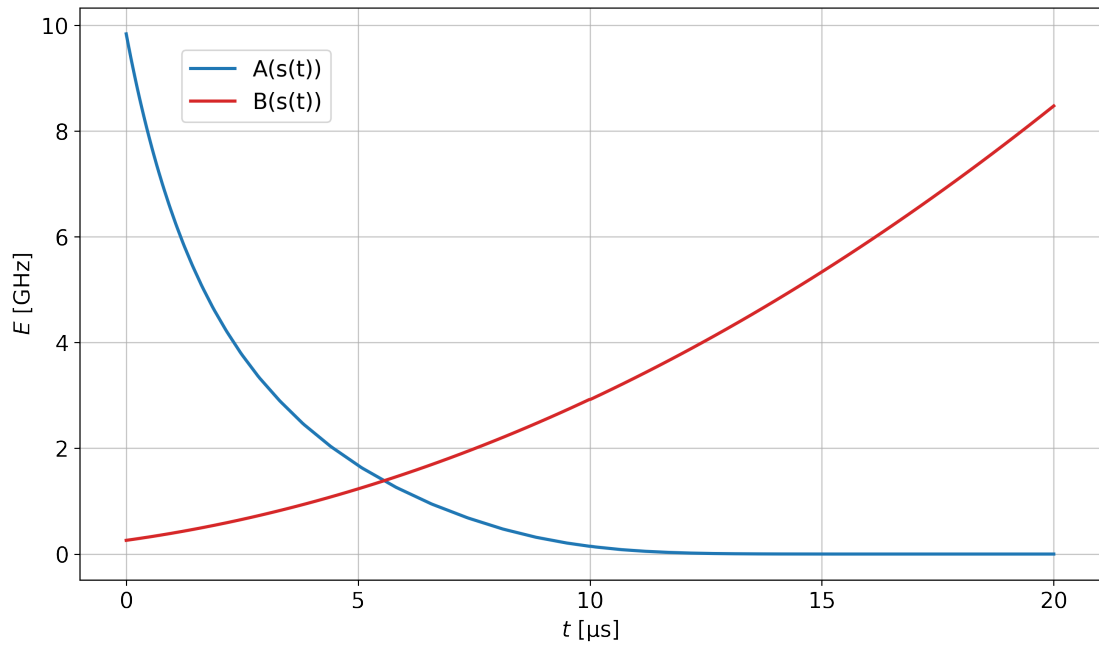


Figure 4.2: Default anneal schedule of the D-Wave Advantage 4.1 with linear  $s(t) = t/t_a$  and  $t_a = 20 \mu\text{s}$  [20].

where  $\beta = 1/kT$  is the effective inverse temperature. In principle,  $\beta$  is an unknown quantity and must be determined in order to effectively use the annealer to generate samples from a quantum Boltzmann distribution.

Comparing the density operator of the QBM in Eq. (4.3) to the one in Eq. (4.16) at the freeze-out point  $s^*$ , we find

$$\begin{aligned}\Gamma_i &= \beta A(s^*), \\ b_i &= -\beta B(s^*)h_i, \\ w_{ij} &= -\beta B(s^*)J_{ij}.\end{aligned}\tag{4.17}$$

This enables us to map the QBM to the annealer if  $\beta$  can be determined to some reasonable degree of accuracy.

### Learning the Effective Inverse Temperature

There is the possibility to treat  $\beta$  as a learnable parameter rather than having to choose a value empirically, as detailed by Xu and Oates in [22]. The method is based on a log-likelihood maximization approach leading to parameter updates of the form (see Appendix C.4 for how one arrives at this result)

$$\Delta\hat{\beta} = \frac{\eta_{\hat{\beta}}}{\hat{\beta}^2}(\langle E \rangle_{\text{data}} - \langle E \rangle_{\text{model}}),\tag{4.18}$$

where  $\hat{\beta} = 1/k\hat{T}$  is the estimator of the effective inverse temperature, and  $\eta_{\hat{\beta}}$  is the associated learning rate. In practice, we forego the factor of  $\hat{\beta}^2$  in the denominator for simplicity and to keep a similar form to the other gradient updates. We must note though, that this approach is only valid for classical Boltzmann distributions, but this fits our current use case as we will see in Section 4.2.1.

### D-Wave Ocean SDK

D-Wave offers an easy-to-use Python package called Ocean SDK [23] to interact with their Leap [1] cloud-based quantum annealing platform, which allows users to access various quantum annealers and other solvers around the globe.

One of the most important steps in solving a problem using a D-Wave annealer is finding an embedding, i.e., a mapping of the logical qubits to the physical qubits, and the SDK offers a heuristic method to do so. If the problem cannot be directly embedded (1:1 logical:physical qubits), then a cluster of physical qubits called a chain is created to represent one logical qubit. Chains introduce added complexity into the problem, because one then needs to tune the chain strength, i.e., the coupling constant between the qubits in the chains. If the measured values of the qubits in a chain differ, this is called a chain break, and the system will report back the majority vote of the measured values in the chain. Therefore, it is best to avoid chains if possible, but they are often a necessary evil for larger problems due to connectivity limitations.

Samples can be easily generated by the annealer using the `sample_ising(h, J)` function which takes in the user-defined  $h_i$  and  $J_{ij}$  values and returns a sample set of specified size (maximum  $10^4$ ). The returned sample set contains the sampled state vectors (an array of shape  $(n_{\text{samples}}, n)$  with values  $\pm 1$  corresponding to the qubit measurements), their energies, and other information about the run.

It must be noted that for the purposes of using a D-Wave annealer for quantum Boltzmann sampling, one must disable autoscaling to properly estimate the effective temperature, as per Eq. (4.17). The `sample_ising(h, J)` function has the keyword argument

`autoscale=True`, which rescales the  $h_i$  and  $J_{ij}$  values by the factor [24]

$$r_{\text{autoscale}} = \max \left\{ \max \left\{ \frac{\max\{h_i\}}{\max\{h_{\text{range}}\}}, 0 \right\}, \max \left\{ \frac{\min\{h_i\}}{\min\{h_{\text{range}}\}}, 0 \right\}, \right. \\ \left. \max \left\{ \frac{\max\{J_{ij}\}}{\max\{J_{\text{range}}\}}, 0 \right\}, \max \left\{ \frac{\min\{J_{ij}\}}{\min\{J_{\text{range}}\}}, 0 \right\} \right\}. \quad (4.19)$$

This is because the main use case of D-Wave annealers is to maximize the probability of measuring the ground state, thus the problem is rescaled so that the  $h_i$  and  $J_{ij}$  values fully utilize the allowed range of values, essentially decreasing the effective temperature; therefore, we set `autoscale=False` to avoid this. For the Advantage 4.1 system the allowed value ranges are  $h_{\text{range}} = [-4, 4]$  and  $J_{\text{range}} = [-1, 1]$  [25].

### Generating More Robust Statistics

QPUs are not perfect, and sometimes specific qubits or parts of the chip might have readout biases. To mitigate such issues, one can perform a gauge transformation on the problem. If we have an  $n$ -qubit problem, then we can generate a random vector  $\mathbf{r} \in \{+1, -1\}^n$  which allows us to change the submission to the solver without actually changing the underlying problem. This is done by taking

$$\begin{aligned} h_i &\rightarrow r_i h_i, \\ J_{ij} &\rightarrow r_i r_j J_{ij}, \\ (s_1, \dots, s_n) &\rightarrow (r_1 s_1, \dots, r_n s_n), \end{aligned} \quad (4.20)$$

and then transforming the results back using the third relation above, where  $s_i$  is the measured value of qubit  $i$ .

### Previous Work in This Field

In recent years, a number of researchers have studied using D-Wave quantum annealers to train Boltzmann machines [22, 26–33]. The most common approach is to train a classical RBM with quantum assistance, i.e., using the annealer to generate the samples in the negative phase rather than using Gibbs sampling. Classical RBMs trained with quantum assistance are a special case of the QBM, i.e., when  $s^* = 1$  the problem reduces to a classical RBM because  $\lim_{s \rightarrow 1} \Gamma_i = 0$  in Eq. (4.17).

One thing that stands out the most about some of the previous research is that very few discuss embeddings and anneal schedules, which as we will see in the next section are important for getting the best possible performance out of the annealer. Therefore, we aim to create a basic framework with which one can use to approach the problem of using a D-Wave annealer to sample from a (quantum) Boltzmann distribution.

## 4.2 12-Qubit Problem

In order to get a better understanding of how the QBM works, we study a small 12-qubit problem that can be solved exactly. For this purpose we take a QBM with restrictions in both the visible and hidden layers and train it using the log-likelihood lower bound maximization approach; we call this a bound-based quantum restricted Boltzmann machine, or BQRBM for short. We configure the model with 8 visible and 4 hidden units to act as a regularized autoencoder.

### 4.2.1 Sampling From a Quantum Boltzmann Distribution

Before training the model, we first need to assess the Advantage 4.1's ability to sample from quantum Boltzmann distributions. To this end, we randomly generate the values of  $h_i$  and  $J_{ij}$  from a normal distribution with  $\mu = 0$  and  $\sigma = 0.1$ , then use the KL divergence to compare samples generated by the Advantage 4.1 with theoretical distributions.

#### Anneal Schedule Format

The  $A(s)$  and  $B(s)$  values for a D-Wave annealer are fixed and depend on the specific system [20], but the Ocean SDK allows us to define a nonuniform  $s(t)$  using a list of  $(t, s)$  tuples, which then determine the  $A(s(t))$  and  $B(s(t))$  curves. In this section we use what we call pause-and-quench anneal schedules that

1. start at  $(t = 0, s = 0)$ ,
2. pause the system at  $(t_{\text{pause}}, s_{\text{pause}})$  for a duration of  $\Delta_{\text{pause}}$ ,
3. quench the system at  $(t_{\text{quench}}, s_{\text{quench}})$  over a duration of  $\Delta_{\text{quench}}$ .

Thus, the anneal schedules provided to the solver are of the form

$$[(0, 0), (t_{\text{pause}}, s_{\text{pause}}), (t_{\text{quench}}, s_{\text{quench}}), (t_{\text{quench}} + \Delta_{\text{quench}}, 1)], \quad (4.21)$$

where

$$\begin{aligned} s_{\text{quench}} &\equiv s_{\text{pause}}, \\ t_{\text{pause}} &= s_{\text{pause}} \cdot t_{\text{relative}}, \\ t_{\text{quench}} &= t_{\text{pause}} + \Delta_{\text{pause}}. \end{aligned} \quad (4.22)$$

An annotated example of a custom pause-and-quench anneal schedule with  $s_{\text{quench}} = 0.55$ ,  $t_{\text{relative}} = 20 \mu\text{s}$ , and  $\Delta_{\text{pause}} = 10 \mu\text{s}$  is given in Fig. 4.3.

The minimum quench duration  $\Delta_{\text{quench}}$  is a function of  $s_{\text{quench}}$  and is limited by the system's fastest anneal rate  $\alpha_{\text{quench}}$

$$\Delta_{\text{quench}}(s_{\text{quench}}) = \frac{1 - s_{\text{quench}}}{\alpha_{\text{quench}}}. \quad (4.23)$$

The Advantage 4.1 system allows a maximum of  $\alpha_{\text{quench}} = 2 \mu\text{s}^{-1}$  [24].

#### Verifying the Distribution

We use the KL divergence  $D_{\text{KL}}(p_{\text{theory}} \parallel p_{\text{samples}})$  to compare the probabilities of the energies computed from the samples returned by the Advantage 4.1 with the theoretical energy distributions for  $s = 0.01, 0.02, \dots, 1$  and  $T = 10^{-3}, 2, 4, \dots, 200 \text{ mK}$ , which we visualize as heatmaps in Fig. 4.4. More information on how the KL divergences are computed can be found in Appendix A.5.1, how the density matrix (from which the theoretical distributions are obtained) is computed in Appendix A.7, and the required constants in Appendix A.8.

In the right heatmap, where  $s_{\text{quench}} = 0.55$ , we observe a narrow band in which the Advantage 4.1-generated samples closely resemble a quantum Boltzmann distribution, and in fact the samples approximate multiple distributions depending on the effective temperature. Marshall et al. present similar results using a D-Wave 2000Q in [34], in which they discuss if the distribution returned by the annealer fits that of a quantum Boltzmann distribution late in the anneal process when  $A(s^*)/B(s^*) \ll 1$ , then the distribution at  $s^*$  should be close to a classical Boltzmann distribution, i.e.,

$$e^{-\beta H(s^*)} \approx e^{-\beta B(s^*) H_{\text{final}}}. \quad (4.24)$$

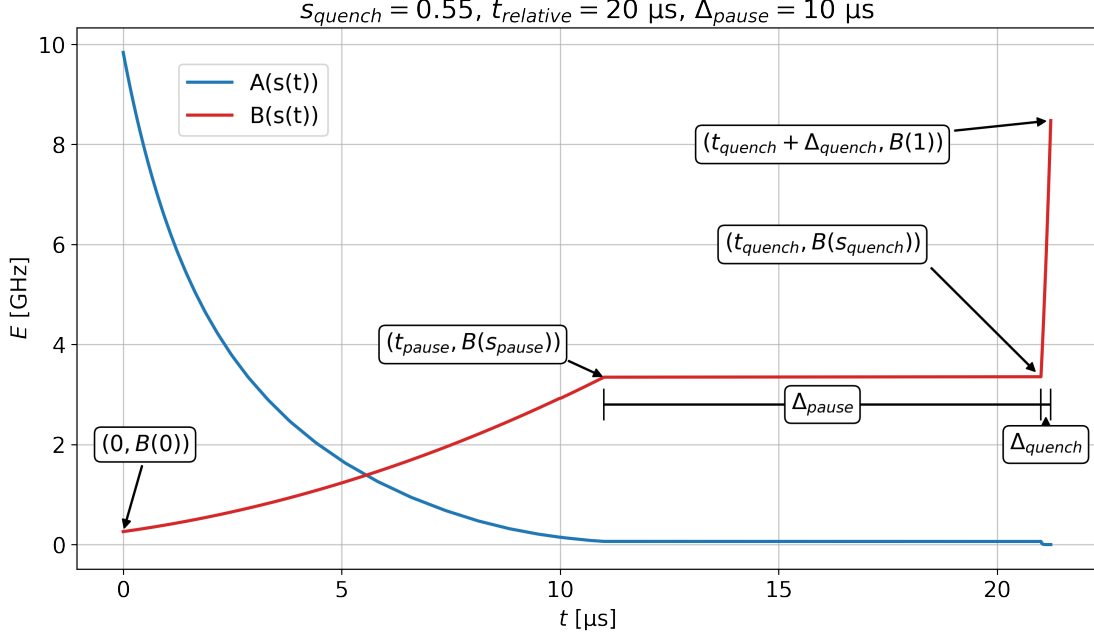


Figure 4.3: Example of a custom pause-and-quench anneal schedule for the D-Wave Advantage 4.1 [20]. Annotations indicate the points  $(t, B(s(t)))$ , as well as the periods over which the annealing is paused and quenched.

This in turn means that not only is there one optimal  $s^*$  and effective temperature which models the distribution, but rather a set of them corresponding to a family of distributions for which  $\beta B(s^*)$  is constant. Therefore, this explains the streak pattern in the heatmaps.

Furthermore, we observe that the left heatmap, where  $s_{\text{quench}} = 0.25$ , is quite similar to the right one where  $s_{\text{quench}} = 0.55$ , but with higher KL divergence values and temperatures. This indicates that quenching at  $s_{\text{quench}} = 0.25$  produces samples that are distributed more as a classical Boltzmann distribution, and that we cannot generate samples from quantum Boltzmann distributions with  $s^* \lesssim 0.45$ , at least not with the anneal schedules we use here.

It must also be noted that the effective temperature corresponding to the classical Boltzmann distribution ( $s^* = 1$ ) is significantly higher than that of the D-Wave temperature of  $T_{\text{DW}} = 15.4 \pm 0.1$  mK [1]<sup>1</sup>. It is not entirely clear exactly why the effective temperature of the distribution is so much higher than the device temperature, but in [34] they give several possible reasons, including the discrepancy between the temperature of the device and the qubits, fluctuations in the temperature while annealing, and control errors masquerading as higher temperatures. In principle, higher effective temperatures are unwanted because they shrink the range of allowed values for the weights and biases as per Eq. (4.17), but there is not much one can do about this.

From analysis of the heatmaps and the fact that we cannot produce distributions with  $s^* \lesssim 0.45$ , we conclude that nontrivial dynamics occur while the system is quenching, i.e., the system cannot quench fast enough. It is difficult to compare directly since the 2000Q is a different system than the Advantage 4.1 we study here, but in [34] they also allude to the possibility of nontrivial dynamics occurring. The 2000Q allows for quenching with  $\alpha_{\text{quench}} = 1$ , which is only a factor of two smaller than that of the Advantage 4.1. Therefore, if as supposed in [34] that the quench is not fast enough, then likely such a small difference in how fast the system can be quenched would not drastically change the

<sup>1</sup>Temperature obtained from the system properties in the Leap interface.

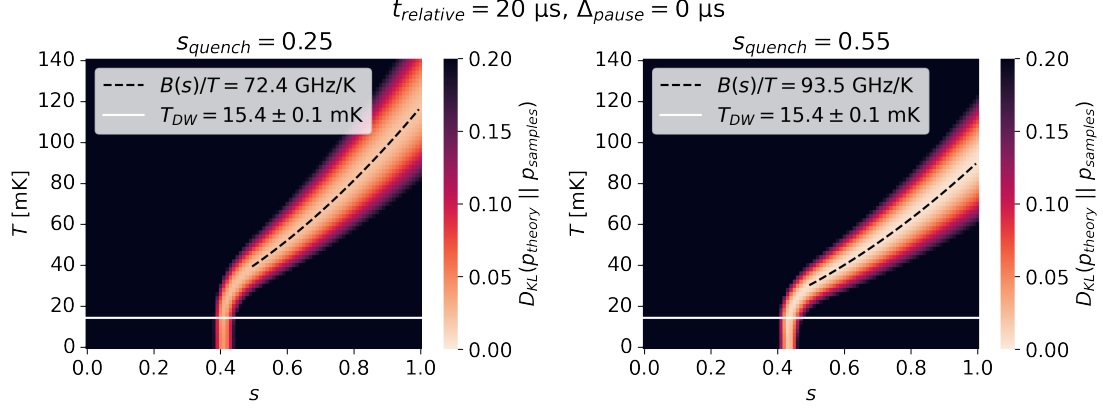


Figure 4.4: Heatmaps of  $D_{\text{KL}}(p_{\text{theory}} \parallel p_{\text{samples}})$  comparing the distribution produced by samples from the D-Wave Advantage 4.1 to a set of theoretical QBM distributions for two different quench points using embedding 10. The dashed lines represent the optimal values of  $B(s)/T = \text{constant}$ , computed by taking the value of  $T$  which produces the lowest KL divergence for each  $s \geq 0.5$ . Data represents an ensemble average over 10 random gauge sample sets consisting of  $10^4$  samples each.

results.

If we take a second to think about it, the qubits are oscillating at a frequency in terms of gigahertz. This means that a quench duration of a few hundred nanoseconds still allows for a number of oscillations in the qubits, which is likely enough time for nontrivial dynamics to take place. It would be interesting to verify via simulation how fast a quench must be in order to freeze out the distribution at the desired point  $s^*$ .

We conclude that we are unable to reliably generate arbitrary quantum Boltzmann distributed samples using the Advantage 4.1 system. Therefore, for the remainder of this thesis we focus on training models with  $s^* = 1$  using classical Boltzmann distributed samples generated by the Advantage 4.1, also enabling us to use the aforementioned method of learning the effective temperature.

### Choosing an Embedding

We compare 10 different heuristically generated embeddings based on how well they approximate the desired distribution. In this embedding comparison, we use only direct embeddings (no chains), so the embeddings only differ by the location of the qubits on the chip, and pause-and-quench anneal schedules with  $t_{\text{relative}} = 20 \mu\text{s}$  and  $\Delta_{\text{pause}} = 0 \mu\text{s}$ .

It is difficult to compare the heatmaps of all embeddings and quench points due to the higher dimensionality of the data, so we take the minimum KL divergence over  $s$  and  $T$ , and plot it as a function of  $s_{\text{quench}}$  in Fig. 4.5. We immediately see how varied the results are depending on the embedding and quench point, highlighting the importance of choosing a good embedding and anneal schedule.

Our findings indicate that embedding 10 is likely a good choice because it produces the best results at  $s_{\text{quench}} = 0.55$ . The rest of the results in this subsection use embedding 10.

### Choosing an Anneal Schedule

With the chosen embedding we want to see if there is a way in which we can alter the anneal schedule to further reduce the KL divergence. We start with the same anneal

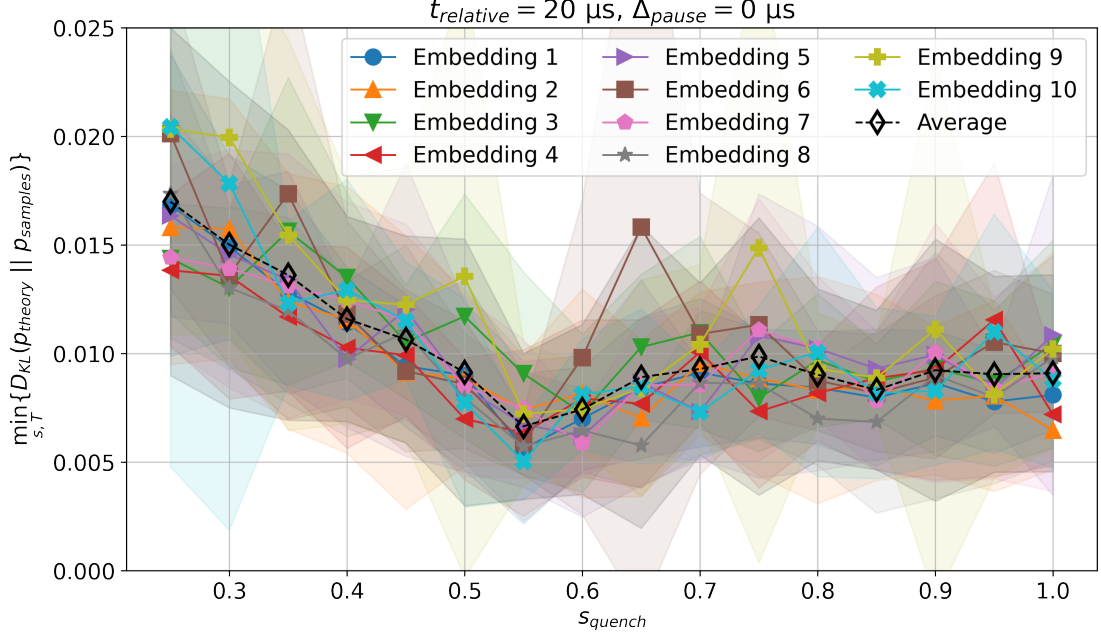


Figure 4.5: Comparison of  $\min_{s,T} \{D_{\text{KL}}(p_{\text{theory}} \parallel p_{\text{samples}})\}$  for different embeddings and  $s_{\text{quench}}$  values. Data represents an ensemble average over 10 random gauge sample sets consisting of  $10^4$  samples each. Shaded regions represent one standard deviation.

schedule formula as before, except we introduce pausing before initiating the quench for durations  $\Delta_{\text{pause}} = 0, 10, 100 \mu\text{s}$ , as well as the addition of  $t_{\text{relative}} = 100 \mu\text{s}$ .

Fig. 4.6 illustrates that pausing and longer annealing times have little effect, and that quenching in the range of  $s_{\text{quench}} \in [0.55, 0.6]$  produces the best results. With this information, we opt to use an anneal schedule with  $s_{\text{quench}} = 0.55$ ,  $t_{\text{relative}} = 20 \mu\text{s}$ , and  $\Delta_{\text{pause}} = 0 \mu\text{s}$ , as it offers a good balance between performance and QPU usage time.

## 4.2.2 Training Data

Having verified that the Advantage 4.1 can indeed produce Boltzmann distributed samples to some degree of accuracy, we proceed with training models using both a simulation and the Advantage 4.1. We randomly generate a training data set consisting of 1500 samples, 1000 from a  $\mathcal{N}(-2, 1)$  distribution and 500 from a  $\mathcal{N}(3, 1)$  distribution, visualized in Fig. 4.7.

## 4.2.3 Simulation-based Model

The first step is training a model using a simulation in which the samples are generated using the probabilities obtained from computing  $\rho$  exactly. Here we use a mini-batch size of 10,  $s^* = 1$ , and an initial learning rate of  $\eta = 0.1$  with a schedule that exponentially decays the learning rate every 10 epochs by a factor of 2 beginning at epoch 50 as defined in Appendix A.3. The learning rate  $\eta_{\hat{\beta}}$  for the parameter  $\hat{\beta}$  follows a similar schedule, except it has a decay period of 20 as opposed to 10, to allow for more range of motion in the  $\hat{\beta}$  parameter later in the training process if the estimate needs to adapt more quickly to a new effective  $\beta$ .

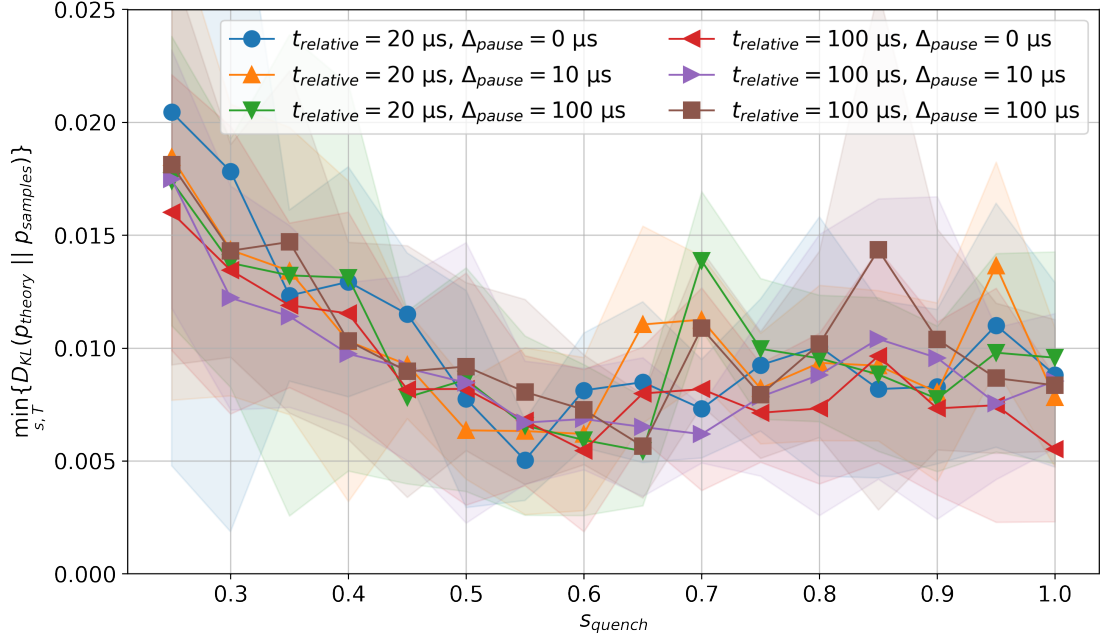


Figure 4.6: Comparison of  $\min_{s,T} \{D_{\text{KL}}(p_{\text{theory}} \parallel p_{\text{samples}})\}$  for various pause-and-quench anneal schedules using embedding 10. Data represents an ensemble average over 10 random gauge sample sets consisting of  $10^4$  samples each. Shaded regions represent one standard deviation. Some of the sample sets with longer annealing times and pause durations contain less than  $10^4$  samples as to satisfy the maximum allowed run time of the D-Wave Advantage 4.1.

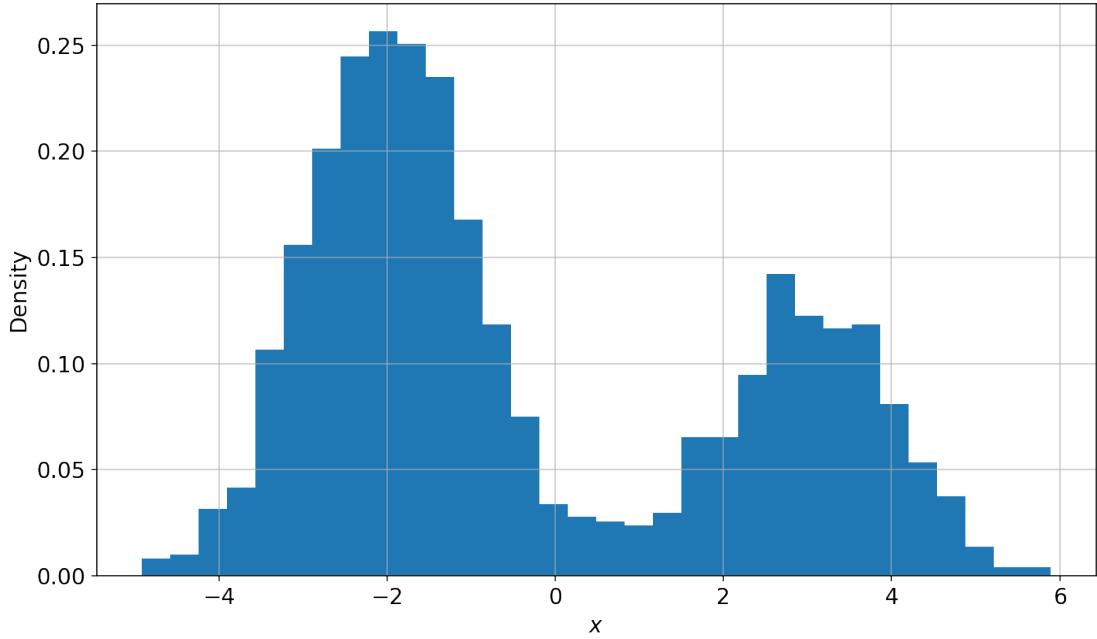


Figure 4.7: Histogram of the training data set used in the 12-qubit problem.

## Results

Fig. 4.8 shows the results of training the simulation-based model on the aforementioned data set. We use the KL divergence  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  as a way to track the progress of the training and get a read on how well the model learns the data set distribution, because minimizing the KL divergence is equivalent to maximizing the log-likelihood [35]. The KL divergence is computed at the end of every epoch using a sample set of size  $10^4$ . In the left plot of Fig. 4.8, we observe a clear trend of the KL divergence being minimized. The learning curve reaches an optimal value after about 80 epochs, then remains steady for the next 20 epochs until the end of training.

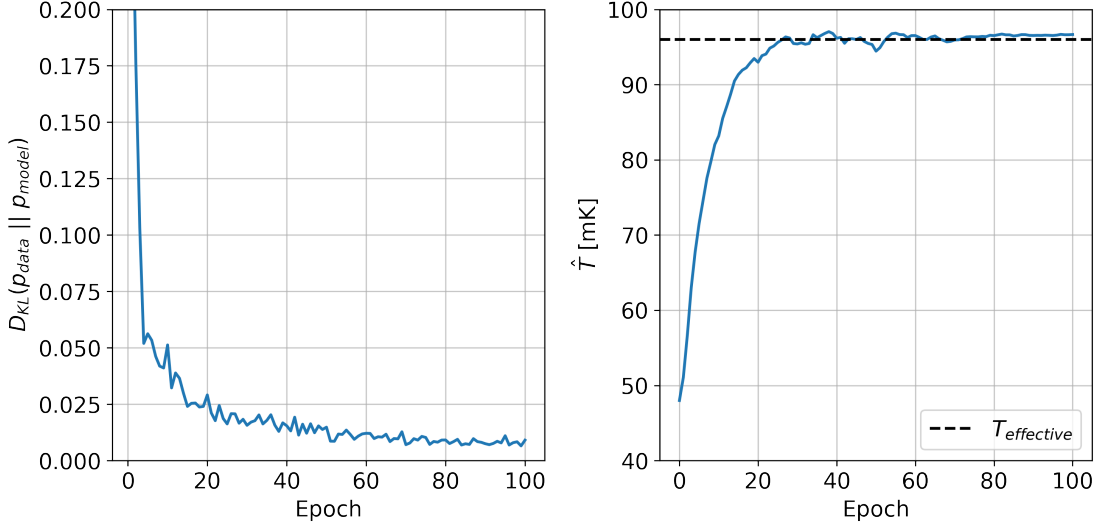


Figure 4.8: Training results of the 12-qubit model trained using the simulation. On the left is the KL divergence  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  plotted against the epochs; each data point was generated using  $10^4$  samples at the end of every epoch. On the right is the learned temperature estimator  $\hat{T}$  plotted against the epochs, as well as the effective temperature that the simulation was configured to generate samples at.

We designed the simulation such that we can set the effective  $\beta$  to any value we desire. To verify that the model can learn an accurate value for the estimator  $\hat{\beta}$ , we configure the simulation to generate samples at an effective value of  $\beta = 0.5 \text{ GHz}^{-1}$  ( $T \approx 96 \text{ mK}$ ) and initialize the model with a value of  $\hat{\beta} = 1 \text{ GHz}^{-1}$  ( $\hat{T} \approx 48 \text{ mK}$ ). The results in the right plot of Fig. 4.8 confirm that it is able to learn a value of  $\hat{\beta}$  close to the actual effective  $\beta$ .

Overall, the results show that the model can generate samples similar to the training distribution reasonably well when trained using the simulation, i.e., the best case scenario. Additionally, we are able to verify that the model can accurately learn an estimate of the effective temperature. We use the results of this model as a baseline to compare the models trained using the Advantage 4.1 in the next subsection with.

### 4.2.4 D-Wave Advantage 4.1-based Model

Having successfully trained the 12-qubit BQRBM using samples generated via exact simulation, we move to switching the sample generation part to the Advantage 4.1. We take the same hyperparameters as the simulation and an anneal schedule using  $s_{\text{quench}} = 0.55$ ,  $t_{\text{relative}} = 20 \text{ }\mu\text{s}$ , and  $\Delta_{\text{pause}} = 0 \text{ }\mu\text{s}$ . We see in Fig. 4.4 that  $s_{\text{quench}} = 0.55$  has an optimal temperature of around 90 mK for  $s^* = 1$ , thus we take  $\hat{\beta} = 0.5 \text{ GHz}^{-1}$  ( $\hat{T} \approx 96 \text{ mK}$ ) as

our initial guess for the effective  $\beta$ , and let the model learn from there.

## Results

The KL divergences in Fig. 4.9 and Table 4.1 show the model trained using the Advantage 4.1 produces samples that resemble the training distribution to some extent, but still underperforms when compared with the simulation and the classically trained RBM. This is possibly due to the information loss associated with using the D-Wave to approximate the distribution, which likely arises due to noise and errors (see Section 4.4), because after all, real-world systems governed by quantum mechanics are highly sensitive to their environment.

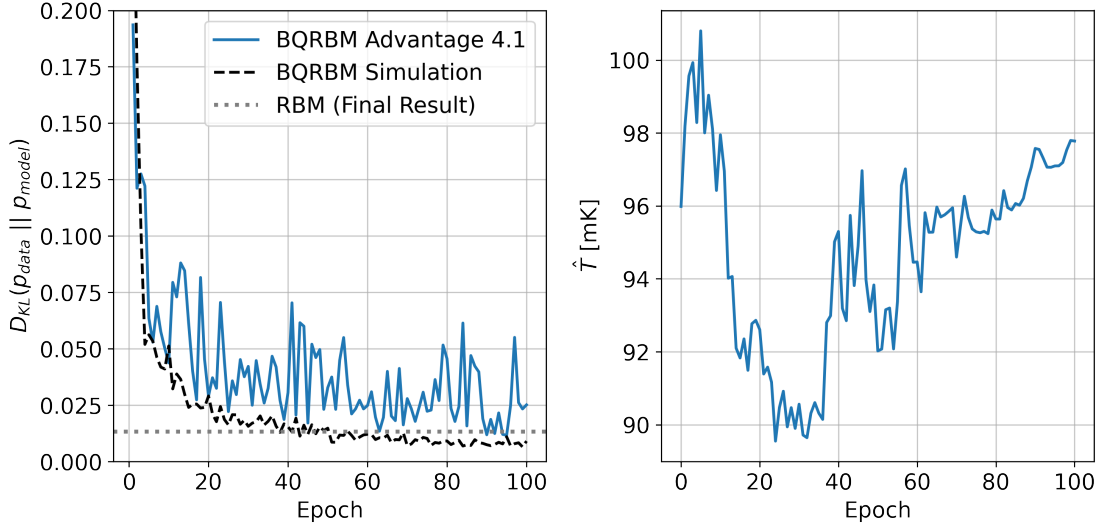


Figure 4.9: Training results of the 12-qubit model trained using samples generated with the D-Wave Advantage 4.1 compared with that of the simulation and the final results of a classical RBM. On the left is the KL divergence  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  plotted against the epochs; each data point was generated using  $10^4$  samples at the end of every epoch. On the right is the learned temperature estimator  $\hat{T}$  plotted against the epochs.

| $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$ |                   |                   |
|-------------------------------------------------------------|-------------------|-------------------|
| BQRBM Advantage 4.1                                         | BQRBM Simulation  | RBM               |
| $0.034 \pm 0.014$                                           | $0.008 \pm 0.001$ | $0.015 \pm 0.002$ |

Table 4.1: KL divergences of the 12-qubit BQRBM models vs. the classical RBM. The values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

We notice that the Advantage 4.1-based model struggles most with the trough between the two Gaussian peaks in the training distribution based on the Q-Q plots in Fig. 4.10.

Although we cannot track the true effective temperature throughout the training process, we are able to see how close the learned effective temperature estimate at the end matches that generated by samples using the final learned weights and biases. The heatmap shown in Fig. 4.11 confirms that the Advantage 4.1-based model's  $\hat{\beta}$  value of 97.8 mK is quite close to the 95.6 mK computed from the optimal  $B(s)/T$  value.

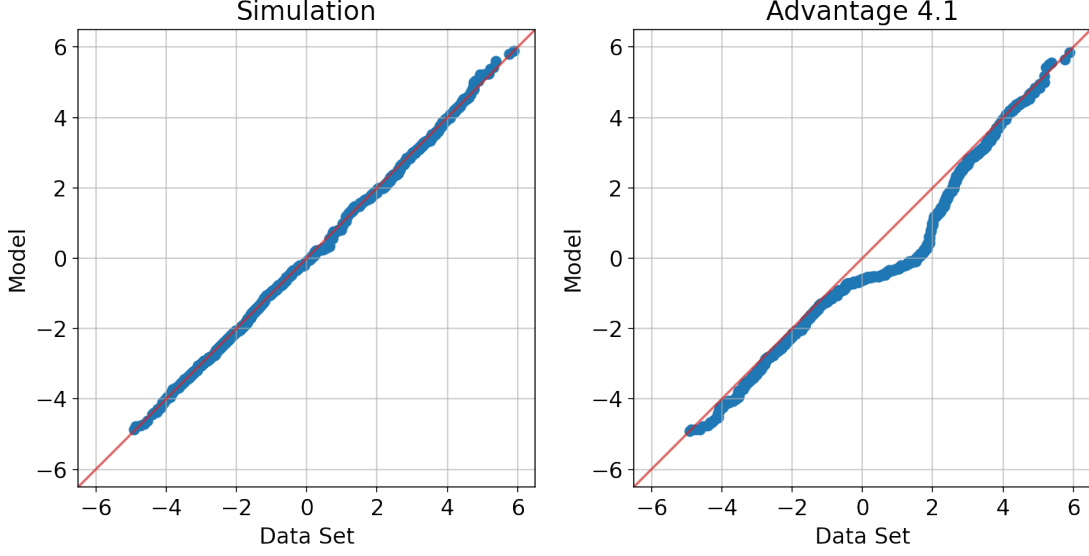


Figure 4.10: Log return Q-Q plots of the 12-qubit model trained using the simulation (left) and the D-Wave Advantage 4.1 (right).

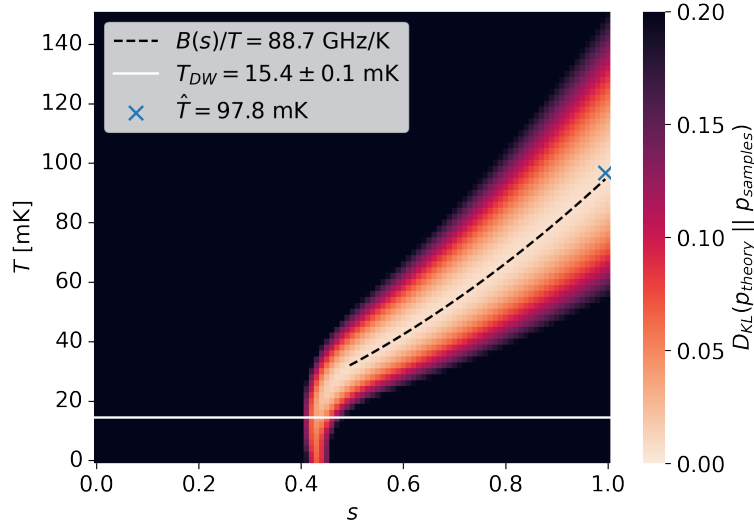


Figure 4.11: Heatmap of  $D_{KL}(p_{theory} || p_{samples})$  comparing the distributions produced by samples from the Advantage 4.1 to a set of theoretical QBM distributions, using the final  $h_i$  and  $J_{ij}$  values learned by the 12-qubit model trained using the Advantage 4.1. The blue cross indicates the learned estimate of the effective temperature. The dashed line represents the optimal value of  $B(s)/T = \text{constant}$ , computed by taking the value of  $T$  which produces the lowest KL divergence for each  $s \geq 0.5$ . Data represents an ensemble average over 10 random gauge sample sets consisting of  $10^4$  samples each.

The results in this section show that a BQRBM can indeed be trained using a D-Wave quantum annealer. The 12-qubit problem plays a crucial role in our understanding of how one can use a D-Wave quantum annealer to generate Boltzmann distributed samples. Although the results are not spectacular, and underperform the simulation and classical model, they still show promise. It will be interesting to rerun this analysis on the next

generation of D-Wave quantum annealers to see how much they improve.

### 4.3 The Quantum Market Generator

With deeper insights into the workings of the BQRBM from the 12-qubit problem, we move to the final stage of training a quantum market generator with 64 visible and 30 hidden units. All results in this section use the same baseline data set (B) as in Section 3.2.

#### 4.3.1 Setting the Annealer’s Hyperparameters

Unlike the 12-qubit problem, the larger problem restricts our ability to perform an in-depth analysis to compare the sample distributions produced by the Advantage 4.1 with that of theory. Therefore, we have to take a more practical approach when choosing some of the annealer hyperparameters such as the relative chain strength, quench point, and embedding. To this end, we train a number of models with various settings of these hyperparameters for 20 epochs to get a read on the trend direction, and then choose their values empirically. For this, we use a mini-batch size of 10,  $s^* = 1$ , constant learning rates of  $\eta = 0.02$  and  $\eta_{\hat{\beta}} = 0.01$ , and an initial value of  $\hat{\beta} = 0.25 \text{ GHz}^{-1}$  ( $\hat{T} \approx 192 \text{ mK}$ ).

The reason we only train for 20 epochs is that the epoch duration is quite high (10-25 minutes) due to latency and load on the annealer, ergo it is not very feasible to train every model for a higher number of epochs. With an average epoch duration of around 15 minutes, training a model for 20 epochs takes roughly 5 hours, so training for 100 epochs would take around a day.

#### Choosing a Relative Chain Strength

The chain strength  $\gamma$  is computed using the relative chain strength  $\gamma_{\text{relative}}$  as

$$\gamma = \gamma_{\text{relative}} \cdot \min \left\{ \max\{J_{\text{range}}\}, \max \left\{ \{|h_i|\} \cup \{|J_{ij}|\} \right\} \right\}. \quad (4.25)$$

We train models using various values of  $\gamma_{\text{relative}} \in [0.3, 2]$ , and a pause-and-quench anneal schedule with  $s_{\text{quench}} = 0.55$ ,  $t_{\text{relative}} = 20 \text{ }\mu\text{s}$ , and  $\Delta_{\text{pause}} = 0 \text{ }\mu\text{s}$ . The results are plotted in Fig. 4.12 (only a subset depicted). We find that too low values of  $\gamma_{\text{relative}}$  lead to more chain breaks early on in the training process, which then cause the model to learn a higher temperature, in turn shrinking the allowed range of weights and biases to the point where the model can no longer learn effectively. Everything indicates that higher relative chain strengths produce better results. After a number of epochs, we observe  $\max \left\{ \{|h_i|\} \cup \{|J_{ij}|\} \right\}$  grow to a value larger than 0.5, implying that values of  $\gamma_{\text{relative}} \geq 2$  would not change the results since  $\gamma$  is reaching its limit of  $\max\{J_{\text{range}}\} = 1$ . Therefore, we choose a value of  $\gamma_{\text{relative}} = 2$ .

#### Choosing an Anneal Schedule

We keep  $t_{\text{relative}} = 20 \text{ }\mu\text{s}$  and  $\Delta_{\text{pause}} = 0 \text{ }\mu\text{s}$  as in the 12-qubit problem, but check to see if a different value of  $s_{\text{quench}}$  improves performance. We try values of  $s_{\text{quench}} = 0.5, 0.55, 0.6$ , plotted in Fig. 4.13, and find that  $s_{\text{quench}} = 0.55$  leads to the best KL divergence curve, and thus choose that value going forward.

#### Choosing an Embedding

The final annealer hyperparameter we seek to tune is the embedding. We try 5 different heuristically generated embeddings each composed of around 400 physical qubits and maximum chain lengths of 7. The comparison plotted in Fig. 4.14 indicates to us that embedding 1 is likely a good choice to continue with.

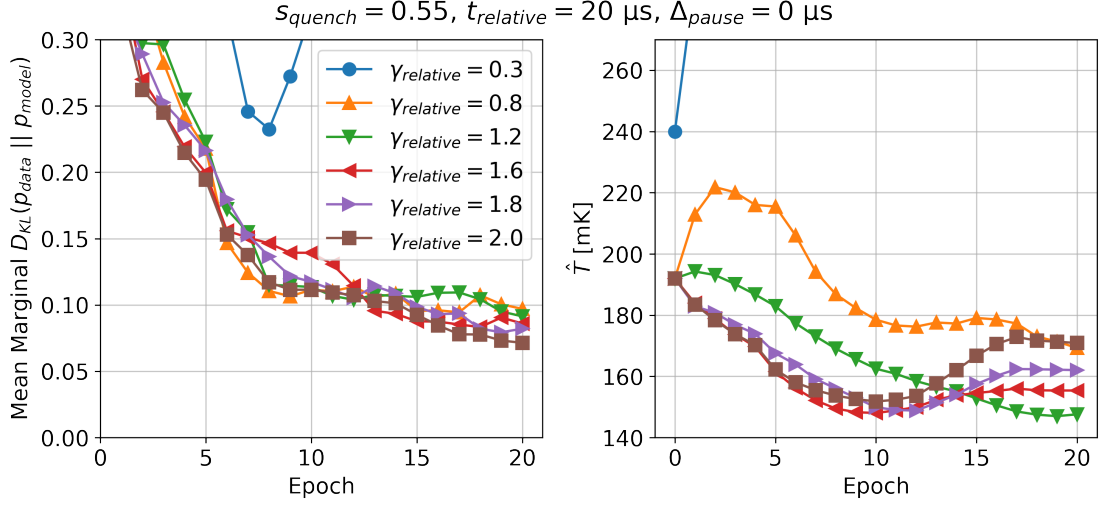


Figure 4.12: Training results of the relative chain strength  $\gamma_{\text{relative}}$  scan for embedding 1. On the left are the mean marginal  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  values, i.e., the average of the KL divergences of the individual currency pairs. On the right are the learned estimates of the effective temperature. Data plotted on a 5 epoch simple moving average basis to reduce visual noise.

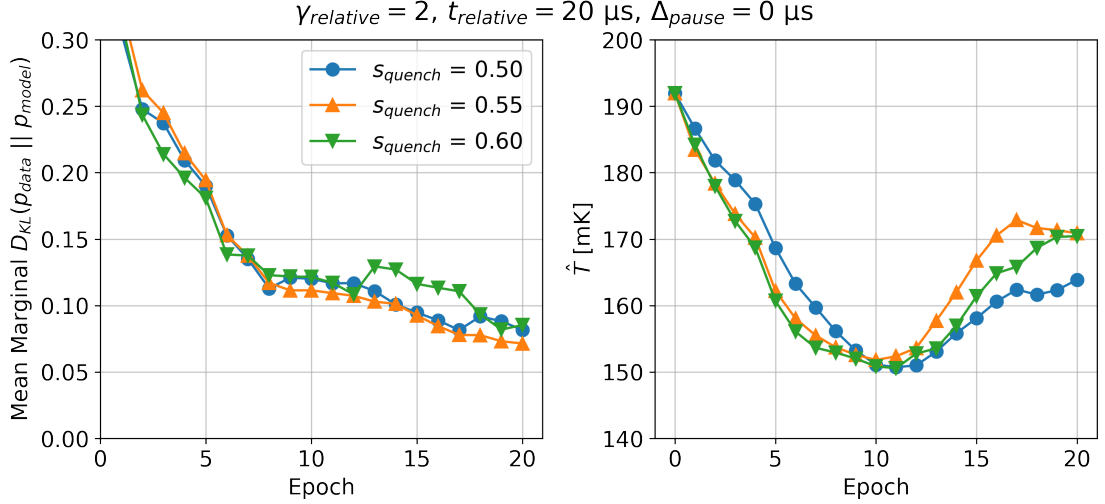


Figure 4.13: Training results of the  $s_{\text{quench}}$  scan for embedding 1. On the left are the mean marginal  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  values, i.e., the average of the KL divergences of the individual currency pairs. On the right are the learned estimates of the effective temperature. Data plotted on a 5 epoch simple moving average basis to reduce visual noise.

### 4.3.2 Results

With annealer hyperparameters of  $\gamma_{\text{relative}} = 2$ ,  $s_{\text{quench}} = 0.55$ , and embedding 1, we move to training a full model over 100 epochs. The training curves are depicted in Fig. 4.15, where we observe the KL divergence decrease until around epoch 40 where it then oscillates

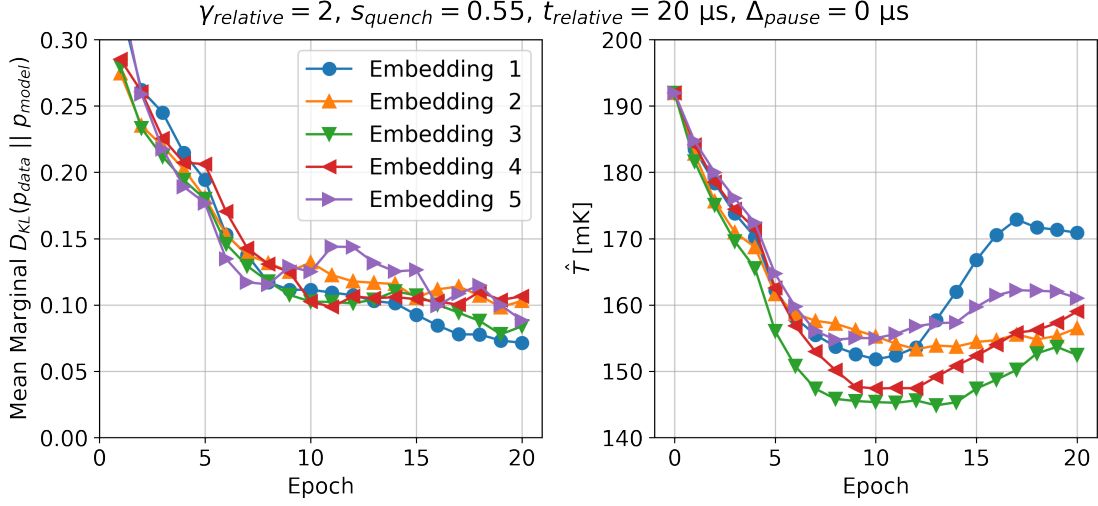


Figure 4.14: Training results comparing 5 different embeddings. On the left are the mean marginal  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  values, i.e., the average of the KL divergences of the individual currency pairs. On the right are the learned estimates of the effective temperature. Data plotted on a 5 epoch simple moving average basis to reduce visual noise.

for the remainder of the training process. Unfortunately, the training results show that the BQRBM model significantly underperforms the classical model.

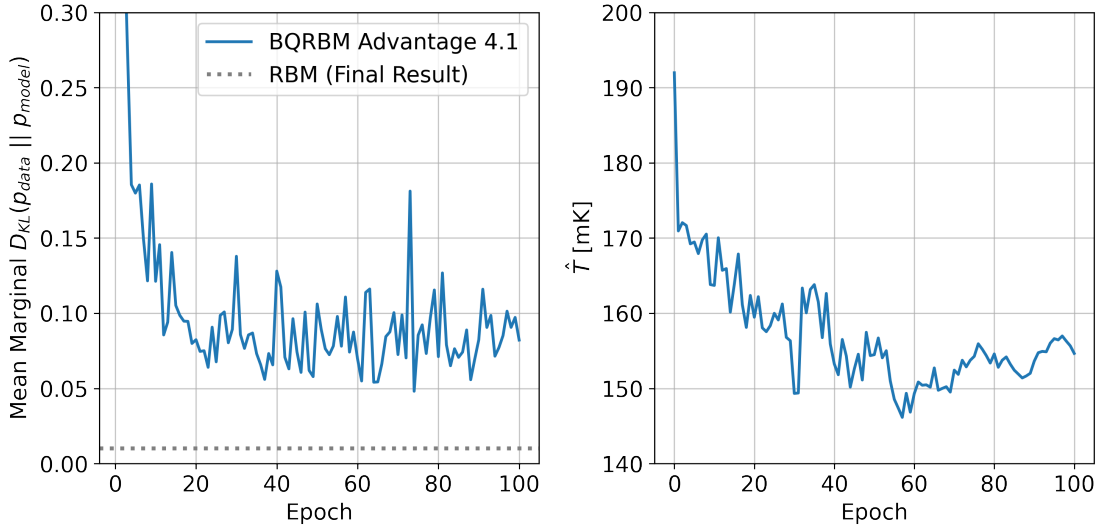


Figure 4.15: Training results of the BQRBM compared with the final results of the classical RBM. On the left is the mean marginal  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  value, i.e., the average of the KL divergences of the individual currency pairs. On the right is the learned estimate of the effective temperature.

Poor model performance is further confirmed by the KL divergences in Table 4.2. We see the KL divergences of the BQRBM are about eight times higher than those of the classical RBM.

| $D_{\text{KL}}(p_{\text{data}}    p_{\text{model}})$ |                   |                   |
|------------------------------------------------------|-------------------|-------------------|
| Currency Pair                                        | BQRBM             | RBM               |
| EURUSD                                               | $0.086 \pm 0.044$ | $0.010 \pm 0.001$ |
| GBPUSD                                               | $0.062 \pm 0.037$ | $0.007 \pm 0.001$ |
| USDCAD                                               | $0.064 \pm 0.028$ | $0.017 \pm 0.002$ |
| USDJPY                                               | $0.103 \pm 0.037$ | $0.008 \pm 0.001$ |
| Mean                                                 | $0.079 \pm 0.037$ | $0.010 \pm 0.001$ |

Table 4.2: KL divergences of the BQRBM model vs. the classical RBM. The values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

The Q-Q plots in Fig. 4.16 point out that the BQRBM model struggles the most with the USDJPY and USDCAD marginals.

Table 4.3 show that the BQRBM is able to reproduce the structure of the correlation coefficients, albeit to a lesser extent than the classical RBM.

| Correlation Coefficients |          |          |         |                  |                  |                  |
|--------------------------|----------|----------|---------|------------------|------------------|------------------|
| Currency Pairs           | Data Set |          |         | BQRBM            |                  |                  |
|                          | Pearson  | Spearman | Kendall | Pearson          | Spearman         | Kendall          |
| EURUSD/GBPUSD            | 0.62     | 0.62     | 0.44    | $0.37 \pm 0.04$  | $0.44 \pm 0.05$  | $0.30 \pm 0.04$  |
| EURUSD/USDCAD            | -0.44    | -0.41    | -0.29   | $-0.25 \pm 0.03$ | $-0.30 \pm 0.04$ | $-0.20 \pm 0.03$ |
| EURUSD/USDJPY            | -0.26    | -0.30    | -0.21   | $-0.12 \pm 0.03$ | $-0.16 \pm 0.04$ | $-0.11 \pm 0.02$ |
| GBPUSD/USDCAD            | -0.42    | -0.37    | -0.26   | $-0.24 \pm 0.02$ | $-0.28 \pm 0.03$ | $-0.19 \pm 0.02$ |
| GBPUSD/USDJPY            | -0.14    | -0.21    | -0.15   | $-0.12 \pm 0.02$ | $-0.15 \pm 0.03$ | $-0.10 \pm 0.02$ |
| USDCAD/USDJPY            | 0.00     | 0.06     | 0.04    | $0.05 \pm 0.02$  | $0.06 \pm 0.02$  | $0.04 \pm 0.01$  |

| RBM            |                  |                  |                  |
|----------------|------------------|------------------|------------------|
| Currency Pairs | Pearson          | Spearman         | Kendall          |
| EURUSD/GBPUSD  | $0.48 \pm 0.01$  | $0.53 \pm 0.01$  | $0.38 \pm 0.01$  |
| EURUSD/USDCAD  | $-0.33 \pm 0.01$ | $-0.34 \pm 0.01$ | $-0.24 \pm 0.01$ |
| EURUSD/USDJPY  | $-0.21 \pm 0.01$ | $-0.25 \pm 0.01$ | $-0.17 \pm 0.01$ |
| GBPUSD/USDCAD  | $-0.31 \pm 0.01$ | $-0.33 \pm 0.01$ | $-0.22 \pm 0.01$ |
| GBPUSD/USDJPY  | $-0.15 \pm 0.01$ | $-0.18 \pm 0.01$ | $-0.13 \pm 0.01$ |
| USDCAD/USDJPY  | $0.06 \pm 0.01$  | $0.07 \pm 0.01$  | $0.05 \pm 0.01$  |

Table 4.3: Correlation coefficients of the data set vs. samples generated by the BQRBM and classical RBM models. The BQRBM and RBM values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

Interestingly, the BQRBM is able to reproduce the volatilities for the most part except for the USDJPY, as seen in Table 4.4.

Table 4.5 shows the quantum model struggles more on the tails, particularly with the USDJPY, as well as EURUSD. This is further confirmed by the tail concentration functions in Fig. 4.17.

### 4.3.3 Comparison to Gate-Based Models

In the paper *Quantum Versus Classical Generative Modelling in Finance* by Coyle et al. [36], they use a 32-qubit gate-based quantum computer to train both classical RBMs and quantum circuit Born machines (QCBMs) on a similar forex log returns data set. Their work shows that the QCBM produces better results than the RBM, but they mostly focus on smaller models using 4, 6, 8, and 12 qubits due to the limited number of qubits

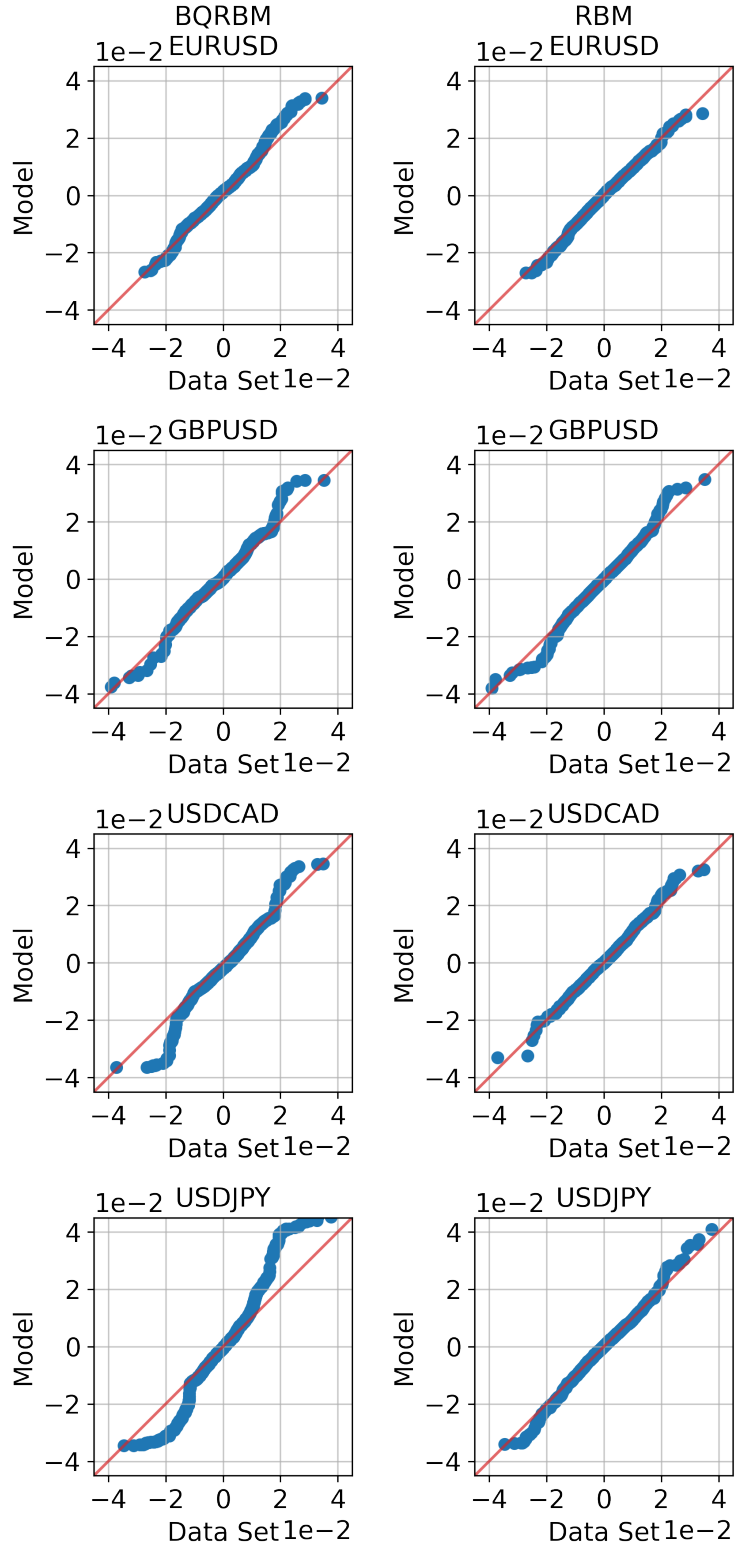


Figure 4.16: Log return Q-Q plots of the BQRBM and classical RBM models for each currency pair. Note that these plots only use the same number of samples as the size of the training data set (5165), and thus are not entirely representative of the models' performances.

| Historical Volatilities |          |                      |                      |
|-------------------------|----------|----------------------|----------------------|
| Currency Pair           | Data Set | BQRBM                | RBM                  |
| EURUSD                  | 9.78%    | $10.17\% \pm 0.79\%$ | $9.98\% \pm 0.11\%$  |
| GBPUSD                  | 8.98%    | $8.86\% \pm 0.47\%$  | $9.34\% \pm 0.11\%$  |
| USDCAD                  | 8.56%    | $8.44\% \pm 0.39\%$  | $8.98\% \pm 0.13\%$  |
| USDJPY                  | 10.02%   | $11.58\% \pm 1.44\%$ | $10.26\% \pm 0.13\%$ |

Table 4.4: Historical volatilities of the data set vs. samples generated by the BQRBM and classical RBM models. The BQRBM and RBM values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

| Lower Tails (1st Percentile) |          |                      |                      |
|------------------------------|----------|----------------------|----------------------|
| Currency Pair                | Data Set | BQRBM                | RBM                  |
| EURUSD                       | -1.64%   | $-2.02\% \pm 0.24\%$ | $-1.80\% \pm 0.04\%$ |
| GBPUSD                       | -1.47%   | $-1.45\% \pm 0.12\%$ | $-1.59\% \pm 0.04\%$ |
| USDCAD                       | -1.40%   | $-1.43\% \pm 0.17\%$ | $-1.54\% \pm 0.05\%$ |
| USDJPY                       | -1.70%   | $-2.32\% \pm 0.38\%$ | $-2.03\% \pm 0.07\%$ |

| Upper Tails (99th Percentile) |          |                     |                     |
|-------------------------------|----------|---------------------|---------------------|
| Currency Pair                 | Data Set | BQRBM               | RBM                 |
| EURUSD                        | 1.62%    | $1.70\% \pm 0.23\%$ | $1.59\% \pm 0.04\%$ |
| GBPUSD                        | 1.42%    | $1.49\% \pm 0.07\%$ | $1.45\% \pm 0.04\%$ |
| USDCAD                        | 1.51%    | $1.50\% \pm 0.12\%$ | $1.61\% \pm 0.04\%$ |
| USDJPY                        | 1.59%    | $2.13\% \pm 0.50\%$ | $1.56\% \pm 0.04\%$ |

Table 4.5: Lower and upper tails, i.e., 1st and 99th percentiles, of the data set vs. samples generated by the BQRBM and classical RBM models. The BQRBM and RBM values are shown in the format mean  $\pm$  one standard deviation from an ensemble of 100 sample sets consisting of  $10^4$  samples each.

available. Therefore, it is not exactly a fair comparison against the results of our 94-qubit model here since the bits of precision are significantly higher, but the BQRBM trained in this section appears to perform better than their QCBM when comparing the Q-Q plots.

#### 4.3.4 Summary

Overall, the BQRBM model trained using the Advantage 4.1 produces lackcluster results when compared with the classical RBM, and does not motivate training additional models on the transformed and volatility indicator enhanced data sets. This could possibly be due to hyperparameters, as our grid search of the space was limited by the training time requirements, but it is not immediately clear if there is a better way to do this. This could also be due to the fact that the model is so large that it uses a significant portion of the QPU and requires chains with lengths of up to 7, resulting in added complexity.

In conclusion, it does not appear that the Advantage 4.1-trained BQRBM can produce results good enough to replace the classical RBM. It will be interesting to see how much this improves with future generations of annealers, although it will likely take serious advances in the technology to outperform the classical RBM.

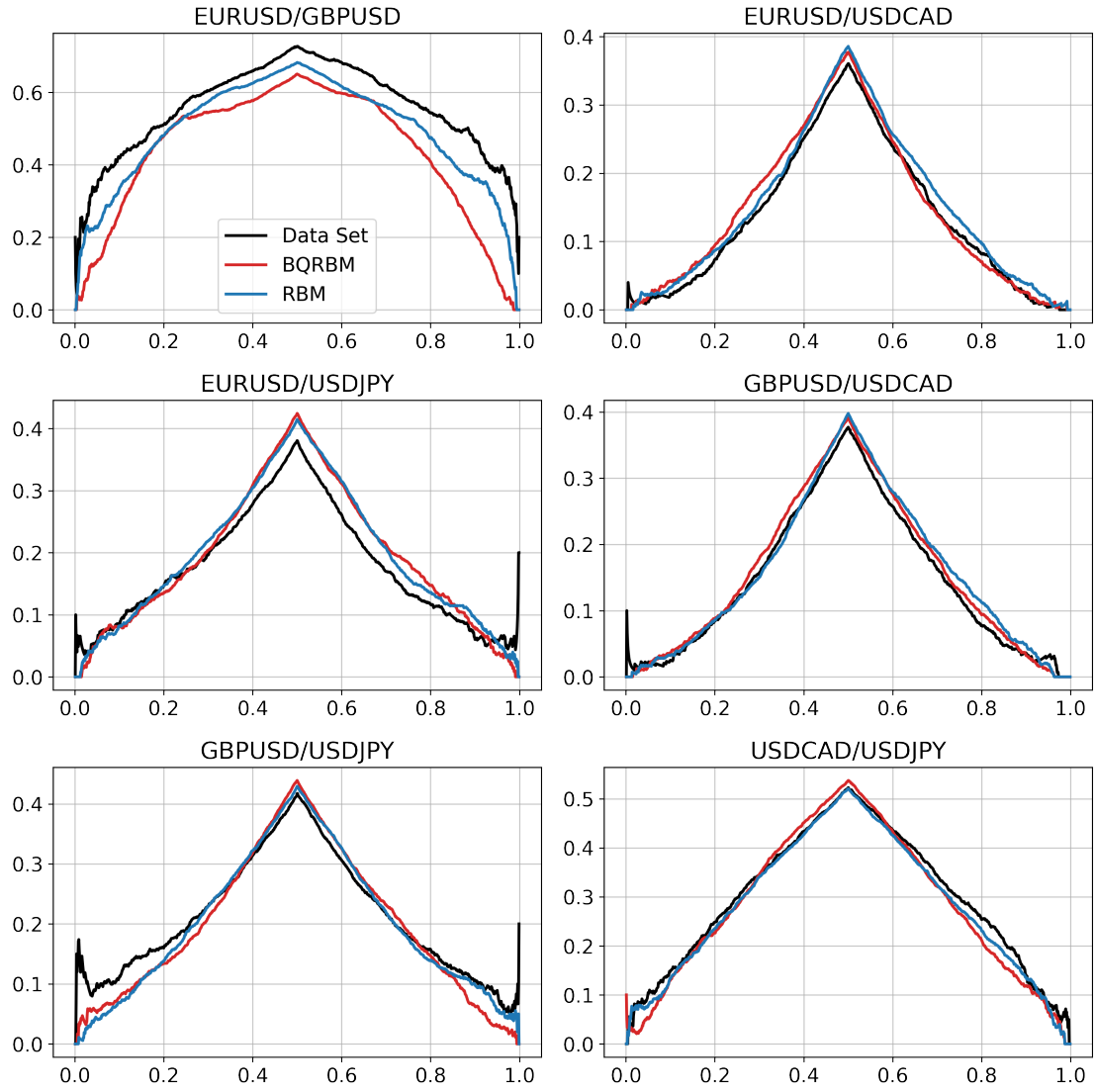


Figure 4.17: Tail concentration functions of the data set vs. samples generated by the BQRBM and classical RBM models.

## 4.4 Challenges of Using a D-Wave Annealer to Train QBMs

Using a D-Wave quantum annealer to train quantum Boltzmann machines is a difficult task, and there are many challenges which need to be overcome in order to do so. In this section we touch on some of these difficulties and discuss some possible methods to mitigate them. Around the time this thesis was started, Pochart et al. released a paper [37] in which they discuss challenges associated with using a D-Wave annealer to sample Boltzmann random variables.

### 4.4.1 Choosing an Embedding

Mapping the logical qubits to physical qubits is nontrivial. D-Wave provides a heuristic method to find embeddings, but in practice it cannot be guaranteed that the returned embedding is optimal. As we saw first hand, different embeddings can produce different results. Therefore, it is recommended to generate multiple embeddings and compare them against each other, and choose the one that performs the best. Additionally, it is worth noting that an optimal embedding on one QPU might not be optimal on another of the same generation.

### Chain Strength

Depending on how large the problem is, one will likely need to use an embedding that is not direct, i.e., one that requires chains of physical qubits to represent single logical qubits due to limited connectivity. This brings about an additional hyperparameter that needs to be tuned. Rather than setting the chain strength directly though, it is recommended to use the relative chain strength as mentioned in Section 4.3.1. It is best to do a comparison in the beginning to get an idea of what a good relative chain strength might be, as it is problem dependent.

### 4.4.2 Sampling the Proper Distribution

The most important thing when using a quantum annealer to train a Boltzmann machine is making sure the annealer is sampling from the proper distribution. In the case of the quantum Boltzmann machine we need samples generated according to  $\rho = \frac{1}{Z}e^{-\beta H(s^*)}$ . For smaller problems it is easy to compare results obtained from the annealer with exact computed distributions (as in the 12-qubit problem), but it is not as simple for larger problems. For larger problems there is the possibility to use advanced methods, such as they did in [34] with the use of an entropic sampling technique [38] based on population annealing to estimate degeneracies, and in turn use those to compute classical Boltzmann distributions to compare with, but that might not always be practical. Alternatively, one can try a hyperparameter grid search as in Section 4.3.1

### Effective Temperature

One of the most important hyperparameters is that of the effective inverse temperature  $\beta$ . In practice, we divide our weights and biases by a factor of  $-\hat{\beta}B(s^*)$  (as per Eq. (4.17)) in order to cancel out the effective temperature so that we can sample the problem we wish to, thus it is crucial for proper parameter scaling. For the case of  $s^* = 1$  we have the ability to treat the effective temperature as a learnable parameter (as in Section 4.1.2), for which we use  $\hat{\beta}$  as an estimator of. This is not so straightforward for  $s^* < 1$  though, because of the initial Hamiltonian and the D-Wave’s inability to measure the qubits in the  $x$ -direction.

## Anneal Schedules

The ability to configure the anneal schedule as allowed by the D-Wave annealer means that there are a number of different ways one can tweak the annealing process such that the results returned minimize the KL divergence between the theoretical distribution one wishes to approximate and the samples returned by the annealer. In an ideal world, the way to get the desired distribution is to anneal slowly at first, then quench the system at the point  $s^*$  and measure the qubits [15]. Unfortunately, the research conducted in this thesis seems to indicate that the current generation of D-Wave annealers cannot quench fast enough to prevent any nontrivial dynamics occurring after  $s^*$ , and all sample sets we collected are more similar to classical Boltzmann distributions than quantum ones. With that said, the annealer can still be used to assist in the training of a classical Boltzmann machine.

### 4.4.3 QPU Limitations and Imperfections

The properties of the QPU itself must also be taken into account. There is no doubt that D-Wave is a top-notch manufacturer of quantum annealers, but even with all of their expertise the QPUs are still subject to imperfections and errors. It is possible for some areas of the chip to perform better than others, or for some of the qubits to have readout biases (although biases can be mitigated by using gauge transformations as detailed in Section 4.1.2).

### Maximum Sample Set Size

One of the main limitations of the D-Wave annealer for this purpose is that of the maximum sample set size. When sampling, the D-Wave one can only generate sample sets with a maximum size of  $10^4$  samples, which is adequate for the intended purpose of optimization, but can fall short when one wants to use it as a sampler for a QBM. It is natural to think that one could just combine the results from multiple sample sets, but this is not necessarily the case. Due to the spin-bath polarization effect (see error sources below), one cannot combine sample sets because of the possibility of previous samples affecting future ones [37].

### Time Requirements

Another issue is the time requirements. Most models detailed here require little QPU access time to train, around 5-10 minutes, but this can add up and get expensive if one is doing a hyperparameter grid search. Additionally, there is the total training time, which as we saw can be quite substantial due to the latency to the cloud platform combined with the load queue of the solver. To illustrate this point, training the simulation-based 12-qubit model takes roughly 2 seconds per epoch (the majority of which is spent computing the density matrix), whereas the Advantage 4.1-based model takes 2-5 minutes per epoch.

### Error Sources

There are a number of sources from which errors can arise on a D-Wave quantum annealer. D-Wave does an excellent job at detailing these errors in their documentation [39, 40], so we will only briefly touch on them here with high-level information obtained from the aforementioned references.

- **Integrated Control Errors (ICE)** are errors due to the accuracy at which the  $h_i$  and  $J_{ij}$  values can be implemented. In mathematical terms this is because the

problem the QPU solves is closer to

$$H_{\text{Ising}}^{\delta} = \sum_{i=1}^n (h_i + \delta_{h_i}) \sigma_i^z + \sum_{i=1}^n \sum_{j=i+1}^n (J_{ij} + \delta_{J_{ij}}) \sigma_i^z \sigma_j^z, \quad (4.26)$$

for some small  $\delta_{h_i}$  and  $\delta_{J_{ij}}$ .

- **Temperature** errors arise due to fluctuations in the physical temperature of the device, which can change depending on how frequently the QPU is programmed.
- **High-Energy Photon Flux** errors can occur in the presence of photons with energies higher than that expected at the effective temperature dependent equilibrium, which can lead to higher energy solutions. These photons originate from cryogenic filtering at higher temperature phases.
- **Readout Fidelity** errors can occur when the bit string returned by the annealer differs from that arrived at by the QPU by one or more bit flips. For reference, D-Wave annealers have a readout fidelity of  $>99\%$ .
- **Programming Errors** can occur when the problem implemented by the QPU suffers from programming issues resulting in the implemented problem's low-energy subspace not having an overlap with that of the desired problem.
- **Spin-Bath Polarization Effect** errors can arise when the current flowing through the qubits during the annealing process causes the spins to obtain a polarization which can bias the measurements.

# Chapter 5

## Conclusion

### 5.1 Summary

We started with an analysis of the forex log returns data set in Chapter 2, analyzing the data from a number of aspects to get an understanding of the intricacies. After that, we moved to training classical RBM models in Chapter 3, where we were able to produce good results similar to those in [5]. The outlier power transformation detailed in Section 2.2.1 shows much promise, as models trained on the transformed data sets perform noticeably better than those trained on the base data sets.

After establishing a classical baseline to compare our quantum models with, we studied a small 12-qubit problem in Section 4.2, through which we gained a deeper understanding of how to sample quantum Boltzmann random variables using a D-Wave quantum annealer, specifically the Advantage 4.1. There, we were able to match sample distributions returned by the annealer to theoretical distributions from the family of distributions corresponding to the density operator  $\rho(s, T) = \frac{1}{Z} e^{-\beta H(s)}$ . Our findings indicate that, with the anneal schedules and parameters used here, the Advantage 4.1 is not able to sample from just any quantum Boltzmann distribution, rather only those that are classical Boltzmann-like in nature. To be more specific, the samples we obtained from the annealer resemble a subset of the family of distributions that satisfies  $B(s)/T = \text{constant}$ , as indicated by the streak patterns observed in Fig. 4.4. This occurs when the distribution is similar to one late in the anneal process, i.e., when  $e^{-\beta H(s^*)} \approx e^{-\beta B(s^*) H_{\text{final}}}$ . This is likely due to the annealer not being able to quench the system fast enough, allowing for nontrivial dynamics to occur, as the shortest allowed quench durations are still quite long relative to the qubit oscillation frequency in terms of gigahertz. How closely the annealer can approximate a desired classical Boltzmann distribution was found to be dependent on both the embedding and the anneal schedule, thus it is highly recommended to tune these accordingly.

With the information that we can only reliably sample classical Boltzmann distributions, we moved to training a bound-based quantum restricted Boltzmann machine (BQRBM) with a freeze-out point of  $s^* = 1$ , essentially reducing the problem to a classical RBM trained using quantum assistance. The difficulty of choosing the effective temperature was in this case easily circumvented by treating  $\beta$  as a learnable parameter as described in Section 4.1.2 and verified in Section 4.2.3. We trained BQRBM models using both a simulation and the Advantage 4.1 annealer, allowing us to compare exactly how close the Advantage 4.1-trained model is to the theory. Additionally, we trained a classical RBM to use as a reference point. In short, the BQRBM model trained using the Advantage 4.1 underperforms both the classical RBM and the simulation, as seen in Section 4.2.4. The simulation-based model shows promise though, outperforming the classical RBM, offering hope for future annealer-trained models if annealers can further reduce the information loss associated with sampling (quantum) Boltzmann distributions.

Finally, we used the knowledge gained about how to train a small BQRBM and applied it to training a larger one in Section 4.3 using the log returns data set, mapping 94 logical qubits to 398 physical qubits with chain lengths of up to 7. This model proved to be more challenging to train because setting the annealer hyperparameters (chain strength, anneal schedule, and embedding) cannot be done as in the 12-qubit problem due to the fact that we cannot simulate such a large system. In practice, we had to choose these values by doing a limited hyperparameter scan, which was difficult due to increased training times that averaged around 15 minutes per epoch. Longer epoch times originated from a combination of solver load and latency from Europe to the North American West Coast. This meant that training a model for 100 epochs would have taken around a day, and if we wanted to fully train the models for all hyperparameters in our scan it would have taken weeks.

The results in Section 4.3.2 show that the BQRBM was able to learn to produce synthetic data similar to the log returns data set distribution to some extent, but drastically underperforms the classical RBM. This could likely have been improved with a more exhaustive hyperparameter scan, but that was not necessarily feasible given the time requirements, and it is unclear if the results would have been significantly better given that even the 12-qubit BQRBM trained on the annealer underperforms the classical RBM.

In this thesis we laid out a framework with which one can train quantum Boltzmann machines using both simulations and D-Wave quantum annealers. As part of this thesis, the Python package `qbm` [6] was developed to make it easier to train and study QBMs. This package is open source and available to the public to encourage further study of QBMs.

Overall, this thesis furthered not only our understanding of QBMs, but that of D-Wave annealer sampling in general. We hope that this work will be useful for future research and development.

## 5.2 Future Directions

Throughout this thesis we came across several directions which we would have liked to explore more in depth but did not have the time to.

It would be interesting to investigate if adding technical indicators to the log returns data set could increase model performance. Given that the log returns data set used here only takes into account the currency pairs' behavior over one day (excluding the volatility indicators), technical indicators calculated using data over a historical window could enrich the data set with vital information to help the model better learn the complexities of the distribution.

The discretization procedure for converting continuous data into bit vectors could probably be further improved. As we saw in Section 3.2, the models that used the outlier power-transformed data sets generate samples with lower KL divergences and better reproduced the correlations between the currency pairs.

Of most interest is simulating the time-dependent Schrödinger equation of the D-Wave annealer to determine how fast the system needs to quench in order to freeze out the dynamics. This would give a good indication of how much quantum annealers need to improve in order to be able to sample from arbitrary quantum Boltzmann distributions.

Studying additional anneal schedule formats would also be a very interesting direction. Reverse annealing was tested to a small extent here only to see if it produced drastically different results than forward annealing, but was left out of the final research because the results did not show any significant improvements and led to added complexity due to the need to choose what state the system was initialized in and if the system was reinitialized to the same state after each measurement or not.

# Appendix A

## Definitions and Methodologies

### A.1 Correlation Coefficients

The Pearson correlation coefficient is defined as

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \in [-1, 1], \quad (\text{A.1})$$

and measures the linear correlation between the random variables  $X$  and  $Y$ . Therefore, it must be noted that this does not capture nonlinear relations, and should not be relied upon to tell the full story. Additionally, this measure is quite sensitive to outliers.

The Spearman rank correlation coefficient is defined as

$$r_s = \rho_{R(X),R(Y)} = \frac{\text{cov}(R(X), R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}} \in [-1, 1], \quad (\text{A.2})$$

and is the Pearson correlation coefficient of the rank of the random variables  $X$  and  $Y$ . The main difference to the Pearson correlation coefficient is that the Spearman measures the monotonic relationship, regardless of linearity. The Spearman correlation coefficient is also less sensitive to outliers than the Pearson.

The Kendall rank correlation coefficient is defined as

$$\tau = \frac{2}{n(n-1)} \sum_{i=1}^n \sum_{j=i+1}^n \text{sign}(x_i - x_j) \text{sign}(y_i - y_j) \in [-1, 1], \quad (\text{A.3})$$

where  $(x_1, y_1), \dots, (x_n, y_n)$  are pairs of observations of the random variables  $X$  and  $Y$ .

It is important to keep in mind how one interprets the correlation coefficients. The sign of the correlation coefficient determines whether the variables are negatively or positively correlated, and the magnitude determines how strong the correlation effects are. As a loose guide, correlation coefficient values of 0.1, 0.3, and 0.5 can be termed small, medium, and large, respectively [41]. In general, one must be careful when interpreting the correlation coefficients; it is important to understand what the values mean, and what they do not. Section 3.4.2 "Interpreting the Correlation Coefficient" of [41] offers further insight and points out some pitfalls to watch out for.

In this thesis the correlation coefficients are computed using the respective functions from the SciPy Python package [42].

### A.2 Annualized Volatility

In finance, the annualized volatility of a time series vector  $\mathbf{x}$  is computed as

$$\text{vol}(\mathbf{x}) = \sqrt{252} \cdot \text{std}(\mathbf{x}), \quad (\text{A.4})$$

where the factor of  $\sqrt{252}$  comes from the square root of the number of trading days in a year, i.e., it's the annualization factor.

### A.3 Learning Rate Decay Schedule

The learning rate at epoch  $t$  is given by

$$\eta^{(t)} = \eta^{(0)} \cdot \min \left\{ 1, 2^{-\frac{t-t_{\text{decay}}}{T_{\text{decay}}}} \right\}, \quad (\text{A.5})$$

where  $\eta^{(0)}$  is the initial learning rate,  $t_{\text{decay}}$  is the epoch at which the decay begins, and  $T_{\text{decay}}$  is the decay period.

### A.4 Autocorrelation Analysis

When studying results from an MCMC-based model it is important to be aware that sequentially generated samples are not always statistically independent, that is, there is some thermalization threshold that corresponds to the minimum number of sampling steps between samples to consider them as statistically independent.

For a time series  $x_1, \dots, x_n$ , the lag- $k$  autocorrelation function is defined as [43]

$$\rho_k = \frac{\text{cov}(x_t, x_{t+k})}{\sigma_x^2}. \quad (\text{A.6})$$

The autocorrelation function is essentially the Pearson correlation coefficient, except instead of comparing two different variables it compares the same variable at different times. In this thesis we use the statsmodels Python package [44] to compute the autocorrelation function as there are some caveats when computing it in practice with large chains, e.g., there are some tricks such as using a Fourier transformation to make the computations more efficient.

The integrated autocorrelation time is a reasonable estimate of how many steps in between samples we should have before we can consider them to be (to a degree) statistically independent. In this thesis we use the emcee Python package [45] to estimate the integrated autocorrelation time, which follows the approach laid out by Goodman and Weare in [46].

### A.5 Kullback-Leibler Divergence

The Kullback-Leibler divergence [47] is a measure of how much the probability distribution  $q$  differs from the reference probability distribution  $p$ . It is defined as

$$D_{\text{KL}}(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}, \quad (\text{A.7})$$

where  $\mathcal{X}$  is the probability space. It can be interpreted as the amount of information loss associated with using  $q$  to approximate  $p$ . We also note that the KL divergence is a distance, but not a metric (rather a divergence), because of the asymmetry that  $D_{\text{KL}}(p \parallel q) \neq D_{\text{KL}}(q \parallel p)$ .

### A.5.1 Kullback-Leibler Divergence in Practice

Due to the limited maximum sample size of  $10^4$  when using a D-Wave annealer and the inability to concatenate sample sets due to spin-bath polarization effects [37], it makes computation of the KL divergence quite difficult because we cannot get a proper read on the probability distribution when the number of possible states is high. Even for the small 12-qubit problem there are still  $2^{12} = 4096$  possible states, thus  $10^4$  samples are not entirely representative of the true distribution. This problem is only exacerbated when working with larger system sizes.

Therefore, in this thesis we take a histogram-based approach to approximate the KL divergence. All KL divergences are computed using 32 bins since this is close to the number of bins computed using the Freedman-Diaconis rule on some of the sample sets for the 12-qubit problem.

When computing the  $q$  distribution from a sample set of limited size, it is often the case that some probabilities come out to zero, which in turn leads to issues computing the KL divergence due to zeros in the denominator of the argument of the log. Luckily, there is a way around this if we know the true probability of measuring such a state to be nonzero. Due to the quantum nature of this problem and the fact that no state has a truly zero probability (although some infinitesimally small), we can take such an approach.

The method we use to mitigate this problem is called smoothing [48], in which we add some small probability  $\epsilon$  to the  $q$  distribution probabilities that are observed to be zero, then take the sum of the added probabilities and evenly subtract it from the nonzero probabilities in order to ensure the distribution remains normalized. For example, if  $\{q_1 = 1/3, q_2 = 2/3, q_3 = 0, q_4 = 0\}$ , then the corresponding smoothed distribution is  $\{q_1 = 1/3 - \epsilon, q_2 = 2/3 - \epsilon, q_3 = \epsilon, q_4 = \epsilon\}$ .

Furthermore, we call it *relative* smoothing when the smoothed probabilities are taken to be relative to the reference distribution  $p$ . This is useful when it is difficult to choose a constant value of  $\epsilon$ , e.g., when the reference distribution probabilities vary widely and can coincide with  $\epsilon$ . For example, if  $\{q_1 = 1/3, q_2 = 2/3, q_3 = 0, q_4 = 0\}$ , then the corresponding relative smoothed distribution is  $\{q_1 = 1/3 - \epsilon(p_3 + p_4)/2, q_2 = 2/3 - \epsilon(p_3 + p_4)/2, q_3 = \epsilon p_3, q_4 = \epsilon p_4\}$ .

We take a value of  $\epsilon = 10^{-6}$  when computing  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  because it is small enough that it will not coincide with any of the  $p_{\text{data}}$  values since the data sets contain only a few thousand samples, thus the smallest value of  $p_{\text{data}}$  is roughly on the order of  $10^{-4}$ . When computing  $D_{\text{KL}}(p_{\text{theory}} \parallel p_{\text{samples}})$  though, we opt to use relative smoothing with a value of  $\epsilon = 10^{-6}$  since sometimes some probabilities of  $p_{\text{theory}}$  can be close to  $\epsilon$  and give a false sense of agreement with the smoothed  $p_{\text{samples}}$ .

## A.6 Tail Concentration Functions

The lower tail concentration function is defined as [49]

$$\begin{aligned} L(z) &= \frac{p(U_1 \leq z, U_2 \leq z)}{z} \\ &= \frac{C(z, z)}{z}, \end{aligned} \tag{A.8}$$

and the upper as

$$\begin{aligned} R(z) &= \frac{p(U_1 > z, U_2 > z)}{1 - z} \\ &= \frac{1 - 2z + C(z, z)}{1 - z}, \end{aligned} \tag{A.9}$$

where  $U_1$  and  $U_2$  are uniform random variables on the interval  $[0, 1]$ , and  $C(u_1, u_2)$  is the copula of  $(U_1, U_2)$ .

In practice, we compute  $U_1$  and  $U_2$  as the normalized rank of the observations of the random variables  $X$  and  $Y$ , respectively. The way to interpret the concentration functions is that they represent the probability that  $X$  and  $Y$  simultaneously take on extreme values. When plotted, the lower tail concentration function is used for  $0 \leq z \leq 0.5$  and the upper for  $0.5 < z \leq 1$ .

A nice explanation with animations can be found at [50].

## A.7 Exact Computation of $\rho$

For the density matrix

$$\rho = \frac{1}{Z} e^{-\beta H}, \quad (\text{A.10})$$

we can compute this as

$$\rho = \frac{1}{\text{tr}(A)} S A S^{-1}, \quad (\text{A.11})$$

where

$$A = \text{diag}\left(e^{-\beta(\lambda_1 - \min_i \{\lambda_i\})}, \dots, e^{-\beta(\lambda_{2^n} - \min_i \{\lambda_i\})}\right), \quad (\text{A.12})$$

where  $\{\lambda_i\}$  are the eigenvalues of  $H$ , and  $S$  is the matrix of eigenvectors that transforms  $H$  to and from its eigenspace. We subtract  $\min_i \{\lambda_i\}$  from the eigenvalues in practice to avoid computing the exponential of a large number which can lead to divergence in floating point calculations.

## A.8 Constants

The values of  $A(s)$  and  $B(s)$  are in terms of GHz, and consequently so is the Hamiltonian. The density matrix is of the form  $\rho = e^{-\beta H(s)}$ , where  $\beta = 1/kT$ , so in order to obtain the (effective) temperature we need the argument of the exponential to be dimensionless, i.e.,  $k$  must be in terms of  $\text{GHz} \cdot \text{K}^{-1}$ . This is achieved by taking a value of

$$\begin{aligned} k &= \frac{k_B}{h} \\ &\approx \frac{1.380649 \cdot 10^{-23} \text{ J} \cdot \text{K}^{-1}}{6.62607015 \cdot 10^{-34} \text{ J} \cdot \text{Hz}^{-1}} \\ &\approx 2.083661912 \cdot 10^{10} \text{ Hz} \cdot \text{K}^{-1} \\ &= 20.83661912 \text{ GHz} \cdot \text{K}^{-1}. \end{aligned} \quad (\text{A.13})$$

## Appendix B

# Restricted Boltzmann Machine

### B.1 Conditional Probabilities

This derivation follows along the lines of that found on p. 658-659 of [11]. We start by noting Eq. (3.2)

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})}. \quad (\text{B.1})$$

From this we can derive the conditional probability using Eqs. (3.1) and (3.3)

$$\begin{aligned} p(\mathbf{h}|\mathbf{v}) &= \frac{p(\mathbf{v}, \mathbf{h})}{p(\mathbf{v})} \\ &= \frac{1}{p(\mathbf{v})} \frac{1}{Z} \exp(\mathbf{a}^\top \mathbf{v} + \mathbf{b}^\top \mathbf{h} + \mathbf{v}^\top \mathbf{W} \mathbf{h}) \\ &= \frac{1}{Z'} \exp \left( \sum_{j=1}^{n_h} b_j h_j + \sum_{j=1}^{n_h} (\mathbf{v}^\top \mathbf{W})_j h_j \right) \\ &= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp(b_j h_j + (\mathbf{v}^\top \mathbf{W})_j h_j), \end{aligned} \quad (\text{B.2})$$

with

$$Z' = \sum_{\mathbf{h}} \exp(\mathbf{b}^\top \mathbf{h} + \mathbf{v}^\top \mathbf{W} \mathbf{h}), \quad (\text{B.3})$$

where  $\sum_{\mathbf{h}}$  denotes the sum over all possible configurations of  $\mathbf{h}$ . This leads us to

$$\begin{aligned} p(h_j = 1|\mathbf{v}) &= \frac{\tilde{p}(h_j = 1|\mathbf{v})}{\tilde{p}(h_j = 0|\mathbf{v}) + \tilde{p}(h_j = 1|\mathbf{v})} \\ &= \frac{\exp(b_j + (\mathbf{v}^\top \mathbf{W})_j)}{1 + \exp(b_j + (\mathbf{v}^\top \mathbf{W})_j)} \\ &= \sigma(b_j + (\mathbf{v}^\top \mathbf{W})_j). \end{aligned} \quad (\text{B.4})$$

Finally, we have

$$p(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{n_h} \sigma((2\mathbf{h} - 1) \odot (\mathbf{b} + \mathbf{W}^\top \mathbf{v}))_j. \quad (\text{B.5})$$

Analogously for  $p(\mathbf{v}|\mathbf{h})$  one finds

$$p(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{n_v} \sigma((2\mathbf{v} - 1) \odot (\mathbf{a} + \mathbf{W} \mathbf{h}))_i. \quad (\text{B.6})$$

## B.2 Log-Likelihood Derivative

For the data set distribution  $p_{\text{data}}$  and parameters  $\theta = (\mathbf{W}, \mathbf{a}, \mathbf{b})$  the log-likelihood is given by

$$\begin{aligned}
\ell(\theta) &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log p(\mathbf{v}) \\
&= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}) \\
&= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \left( \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \right) \\
&= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} - \log \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}.
\end{aligned} \tag{B.7}$$

Taking the partial derivative we find

$$\begin{aligned}
\partial_{\theta} \ell(\theta) &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \partial_{\theta}(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}} - \frac{\sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \partial_{\theta}(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}} \\
&= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \left\langle \partial_{\theta}(-E(\mathbf{v}, \mathbf{h})) \right\rangle_{p(\mathbf{h}|\mathbf{v})} - \left\langle \partial_{\theta}(-E(\mathbf{v}, \mathbf{h})) \right\rangle_{p(\mathbf{v}, \mathbf{h})}.
\end{aligned} \tag{B.8}$$

This gives us

$$\begin{aligned}
\partial_{w_{ij}} \ell(\theta) &= \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}, \\
\partial_{a_i} \ell(\theta) &= \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}, \\
\partial_{b_j} \ell(\theta) &= \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}},
\end{aligned} \tag{B.9}$$

where  $\langle \cdot \rangle_{\text{data}}$  denotes the expectation value with respect to the data set distribution, and  $\langle \cdot \rangle_{\text{model}}$  denotes the expectation value with respect to the model distribution.

## Appendix C

# Quantum Boltzmann Machine

### C.1 Log-Likelihood Derivative

This derivation follows along the lines of that laid out in [15]. We start with the log-likelihood

$$\begin{aligned}
 \ell(\theta) &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log p(\mathbf{v}) \\
 &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \frac{\text{tr}(\Lambda_{\mathbf{v}} e^{-H})}{\text{tr}(e^{-H})} \\
 &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \left[ \log \text{tr}(\Lambda_{\mathbf{v}} e^{-H}) - \log \text{tr}(e^{-H}) \right],
 \end{aligned} \tag{C.1}$$

where  $\sum_{\mathbf{v}}$  denotes the sum over all possible configurations of  $\mathbf{v}$ . Taking the partial derivative yields

$$\partial_{\theta} \ell(\theta) = \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \left[ \frac{\text{tr}(\Lambda_{\mathbf{v}} \partial_{\theta} e^{-H})}{\text{tr}(\Lambda_{\mathbf{v}} e^{-H})} - \frac{\text{tr}(\partial_{\theta} e^{-H})}{\text{tr}(e^{-H})} \right]. \tag{C.2}$$

Due to the noncommutativity of  $H$  and  $\partial_{\theta} H$ , we need to use the trick laid out in [15] where we take  $e^{-H} = (e^{-\delta\tau H})^n$  with  $\delta\tau \equiv 1/n$ , which allows one to write

$$\partial_{\theta} e^{-H} = - \sum_{m=1}^n e^{-m\delta\tau H} \delta\tau \partial_{\theta} H e^{-(n-m)\delta\tau H} + \mathcal{O}(\delta\tau^2). \tag{C.3}$$

Taking the limit as  $n \rightarrow \infty$  of both sides gives

$$\begin{aligned}
 \partial_{\theta} e^{-H} &= \lim_{n \rightarrow \infty} - \sum_{m=1}^n e^{-m\delta\tau H} \delta\tau \partial_{\theta} H e^{-(n-m)\delta\tau H} + \mathcal{O}(\delta\tau^2) \\
 &= - \int_0^1 d\tau e^{-\tau H} \partial_{\theta} H e^{(\tau-1)H}.
 \end{aligned} \tag{C.4}$$

From here one can take the trace of both sides to arrive at

$$\begin{aligned}
\text{tr}(\partial_\theta e^{-H}) &= -\text{tr}\left(\int_0^1 d\tau e^{-\tau H} \partial_\theta H e^{(\tau-1)H}\right) \\
&= -\int_0^1 d\tau \text{tr}(e^{-\tau H} \partial_\theta H e^{(\tau-1)H}) \\
&= -\int_0^1 d\tau \text{tr}(e^{(\tau-1)H} e^{-\tau H} \partial_\theta H) \\
&= -\int_0^1 d\tau \text{tr}(e^{-H} \partial_\theta H) \\
&= -\text{tr}(e^{-H} \partial_\theta H),
\end{aligned} \tag{C.5}$$

which gives

$$\begin{aligned}
\frac{\text{tr}(\partial_\theta e^{-H})}{\text{tr}(e^{-H})} &= -\frac{\text{tr}(e^{-H} \partial_\theta H)}{\text{tr}(e^{-H})} \\
&= -\text{tr}(\rho \partial_\theta H) \\
&= -\langle \partial_\theta H \rangle.
\end{aligned} \tag{C.6}$$

Unfortunately, due to the additional factor of  $\Lambda_{\mathbf{v}}$  in the first term of Eq. (C.2), one arrives at

$$\begin{aligned}
\text{tr}(\Lambda_{\mathbf{v}} \partial_\theta e^{-H}) &= -\text{tr}\left(\int_0^1 d\tau \Lambda_{\mathbf{v}} e^{-\tau H} \partial_\theta H e^{(\tau-1)H}\right) \\
&= -\int_0^1 d\tau \text{tr}(\Lambda_{\mathbf{v}} e^{-\tau H} \partial_\theta H e^{(\tau-1)H}),
\end{aligned} \tag{C.7}$$

which is nontrivial to compute in practice.

## C.2 Log-Likelihood Lower Bound

This derivation follows along the lines of that laid out in [15]. The Golden-Thompson inequality that  $\text{tr}(e^A e^B) \geq \text{tr}(e^{A+B})$  allows one to write (for small  $\epsilon > 0$ )

$$\text{tr}(e^{-H} e^{\log(\Lambda_{\mathbf{v}} + \epsilon)}) \geq \text{tr}(e^{-H + \log(\Lambda_{\mathbf{v}} + \epsilon)}). \tag{C.8}$$

Taking the limit  $\epsilon \rightarrow 0$  yields

$$\text{tr}(\Lambda_{\mathbf{v}} e^{-H}) \geq \text{tr}(e^{-H_{\mathbf{v}}}), \tag{C.9}$$

where

$$H_{\mathbf{v}} = \langle \mathbf{v} | H | \mathbf{v} \rangle \tag{C.10}$$

is the *clamped* Hamiltonian. This is called clamped because the visible qubits are held to the classical state of the visible vector  $\mathbf{v}$  due to an infinite energy penalty imposed by the  $\log(\Lambda_{\mathbf{v}} + \epsilon)$  term. Using this we can write the inequality

$$\begin{aligned}
p(\mathbf{v}) &= \frac{\text{tr}(\Lambda_{\mathbf{v}} e^{-H})}{\text{tr}(e^{-H})} \\
&\geq \frac{\text{tr}(e^{-H_{\mathbf{v}}})}{\text{tr}(e^{-H})},
\end{aligned} \tag{C.11}$$

which in turn allows for the log-likelihood to be bounded as

$$\ell(\theta) \geq \tilde{\ell}(\theta), \quad (\text{C.12})$$

where

$$\tilde{\ell}(\theta) = \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \log \frac{\text{tr}(e^{-H_{\mathbf{v}}})}{\text{tr}(e^{-H})}. \quad (\text{C.13})$$

### C.3 Log-Likelihood Lower Bound Derivative

This derivation follows along the lines of that laid out in [15]. Taking the partial derivative of the log-likelihood lower bound yields

$$\begin{aligned} \partial_{\theta} \tilde{\ell}(\theta) &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \left[ \frac{\text{tr}(\partial_{\theta} e^{-H_{\mathbf{v}}})}{\text{tr}(e^{-H_{\mathbf{v}}})} - \frac{\text{tr}(\partial_{\theta} e^{-H})}{\text{tr}(e^{-H})} \right] \\ &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) \left[ \frac{\text{tr}(-e^{-H_{\mathbf{v}}} \partial_{\theta} H_{\mathbf{v}})}{\text{tr}(e^{-H_{\mathbf{v}}})} - \frac{\text{tr}(-e^{-H} \partial_{\theta} H)}{\text{tr}(e^{-H})} \right] \\ &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) [\text{tr}(-\rho_{\mathbf{v}} \partial_{\theta} H_{\mathbf{v}}) - \text{tr}(-\rho \partial_{\theta} H)] \\ &= \sum_{\mathbf{v}} p_{\text{data}}(\mathbf{v}) [\langle -\partial_{\theta} H_{\mathbf{v}} \rangle_{\mathbf{v}} - \langle -\partial_{\theta} H \rangle] \\ &= \overline{\langle -\partial_{\theta} H_{\mathbf{v}} \rangle_{\mathbf{v}}} - \langle -\partial_{\theta} H \rangle. \end{aligned} \quad (\text{C.14})$$

Plugging in our parameters we get

$$\begin{aligned} \partial_{w_{ij}} \tilde{\ell}(\theta) &= \overline{\langle \sigma_i^z \sigma_j^z \rangle_{\mathbf{v}}} - \langle \sigma_i^z \sigma_j^z \rangle \\ &= \langle \sigma_i^z \sigma_j^z \rangle_{\text{data}} - \langle \sigma_i^z \sigma_j^z \rangle_{\text{model}}, \\ \partial_{b_i} \tilde{\ell}(\theta) &= \overline{\langle \sigma_i^z \rangle_{\mathbf{v}}} - \langle \sigma_i^z \rangle \\ &= \langle \sigma_i^z \rangle_{\text{data}} - \langle \sigma_i^z \rangle_{\text{model}}, \end{aligned} \quad (\text{C.15})$$

where  $\langle \cdot \rangle_{\text{data}}$  denotes the expectation value with respect to the data set distribution, and  $\langle \cdot \rangle_{\text{model}}$  denotes the expectation value with respect to the model distribution.

When restrictions are imposed on connections within the hidden layer, the clamped Hamiltonian reduces to

$$H_{\mathbf{v}} = - \sum_{i=1}^n (\Gamma_i \sigma_i^x + b'_i(\mathbf{v}) \sigma_i^z), \quad (\text{C.16})$$

where  $b'_i(\mathbf{v}) = b_i + (\mathbf{W}^{\top} \mathbf{v})_i$ . This allows one to rewrite the clamped density matrix as

$$\begin{aligned} \rho_{\mathbf{v}} &= \frac{1}{Z_{\mathbf{v}}} \exp \left( \sum_{i=1}^n (\Gamma_i \sigma_i^x + h'_i(\mathbf{v}) \sigma_i^z) \right) \\ &= \frac{1}{Z_{\mathbf{v}}} \prod_{i=1}^n \exp (\Gamma_i \sigma_i^x + b'_i(\mathbf{v}) \sigma_i^z) \\ &= \prod_{i=1}^n \rho_{\mathbf{v}}^{(i)}. \end{aligned} \quad (\text{C.17})$$

With this we can compute the expectation values as

$$\begin{aligned}
\langle \sigma_i^z \rangle_{\mathbf{v}} &= \text{tr}(\rho_{\mathbf{v}}^{(i)} \sigma_i^z) \\
&= \frac{\text{tr} \left[ \exp(\Gamma_i \sigma_i^x + b'_i(\mathbf{v}) \sigma_i^z) \sigma_i^z \right]}{\text{tr} \left[ \exp(\Gamma_i \sigma_i^x + b'_i(\mathbf{v}) \sigma_i^z) \right]} \\
&= \frac{b'_i(\mathbf{v})}{D_i(\mathbf{v})} \tanh(D_i(\mathbf{v})),
\end{aligned} \tag{C.18}$$

where  $D_i(\mathbf{v}) = \sqrt{\Gamma_i^2 + b'_i(\mathbf{v})^2}$ .

The last equality above is obtained by using that for traceless  $A$  with  $\det A < 0$  we can write

$$\exp(A) = \cosh\left(\sqrt{|\det A|}\right) I + \frac{1}{\sqrt{|\det A|}} \sinh\left(\sqrt{|\det A|}\right) A. \tag{C.19}$$

This is obtained by using Cayley-Hamilton theorem along with the series expansion of the matrix exponential and grouping the terms.

## C.4 Effective $\beta$ as a Learnable Parameter

This derivation follows along the lines of that laid out in [22]. Suppose the D-Wave annealer samples according to a classical Boltzmann distribution  $p_{\text{DW}}$  of energies  $E_{\text{DW}} = E/\beta$ , i.e.,

$$\begin{aligned}
p_{\text{DW}} &= \frac{1}{Z_{\text{DW}}} e^{-E_{\text{DW}}} \\
&= \frac{1}{Z_{\text{DW}}} e^{-E/\beta}.
\end{aligned} \tag{C.20}$$

Then we can take the partial derivative of the corresponding log-likelihood

$$\partial_{\beta} \log p_{\text{DW}} = \frac{1}{\beta^2} (E - \langle E \rangle), \tag{C.21}$$

and after averaging over all configurations and multiplying by the learning rate  $\eta$  we get

$$\Delta\beta = \frac{\eta}{\beta^2} (\langle E \rangle_{\text{data}} - \langle E \rangle_{\text{model}}), \tag{C.22}$$

which we can use to treat the effective inverse temperature as a learnable parameter.

# Bibliography

- [1] D-Wave Systems Inc. *Leap<sup>TM</sup> Quantum Cloud Service*. URL: <https://cloud.dwavesys.com/leap/>.
- [2] Lisa Zyga. *D-wave sells first commercial quantum computer*. June 2011. URL: <https://phys.org/news/2011-06-d-wave-commercial-quantum.html>.
- [3] D-Wave Systems Inc. *The D-Wave Advantage System: An Overview*. URL: [https://www.dwavesys.com/media/s3qbjp3s/14-1049a-a\\_the\\_d-wave\\_advantage\\_system\\_an\\_overview.pdf](https://www.dwavesys.com/media/s3qbjp3s/14-1049a-a_the_d-wave_advantage_system_an_overview.pdf).
- [4] BIS. *Foreign Exchange turnover in April 2019*. Sept. 2019. URL: [https://www.bis.org/statistics/rpfx19\\_fx.htm](https://www.bis.org/statistics/rpfx19_fx.htm).
- [5] Alexei Kondratyev and Christian Schwarz. “The Market Generator”. In: *Risk* (May 2019). URL: <http://dx.doi.org/10.2139/ssrn.3384948>.
- [6] Cameron Perot. *qbm*. 2022. URL: <https://jugit.fz-juelich.de/qip/qbm>.
- [7] Dukascopy. *Historical data feed :: Dukascopy Bank Sa: Swiss Forex Bank: ECN broker: Managed accounts: Swiss FX Trading Platform*. URL: <https://www.dukascopy.com/swiss/english/marketwatch/historical/>.
- [8] *Why log returns*. Nov. 2012. URL: <https://quantivity.wordpress.com/2011/02/21/why-log-returns/>.
- [9] *EU referendum*. URL: <https://www.gov.uk/government/topical-events/eu-referendum>.
- [10] Martin Fackler. *In Japan, a robust yen undermines the markets*. Oct. 2008. URL: <https://www.nytimes.com/2008/10/28/business/worldbusiness/28yen.html>.
- [11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [12] Philip M. Long and Rocco A. Servedio. “Restricted Boltzmann Machines Are Hard to Approximately Evaluate or Simulate”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, 2010, pp. 703–710. ISBN: 9781605589077.
- [13] Geoffrey Hinton. *A Practical Guide to Training Restricted Boltzmann Machines (Version 1)*. Aug. 2010. DOI: 10.1007/978-3-642-35289-8\_32.
- [14] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [15] Mohammad H. Amin et al. “Quantum Boltzmann Machine”. In: *Phys. Rev. X* 8 (2 May 2018), p. 021050. DOI: 10.1103/PhysRevX.8.021050. URL: <https://link.aps.org/doi/10.1103/PhysRevX.8.021050>.
- [16] M. Born and V. Fock. “Beweis des Adiabatenatzes”. In: *Zeitschrift für Physik* 51.3-4 (1928), pp. 165–180. DOI: 10.1007/bf01343193.

- [17] Madita Willsch, Dennis Willsch, and Kristel Michielsen. *Lecture Notes: Programming Quantum Computers*. 2022. arXiv: 2201.02051 [quant-ph].
- [18] D-Wave Systems Inc. *Annealing Implementation and Controls*. URL: [https://docs.dwavesys.com/docs/latest/c\\_qpu\\_annealing.html](https://docs.dwavesys.com/docs/latest/c_qpu_annealing.html).
- [19] D-Wave Systems Inc. *QPU Architecture: Topologies*. URL: [https://docs.dwavesys.com/docs/latest/c\\_gs\\_4.html](https://docs.dwavesys.com/docs/latest/c_gs_4.html).
- [20] D-Wave Systems Inc. *QPU-Specific Characteristics*. URL: [https://docs.dwavesys.com/docs/latest/doc\\_physical\\_properties.html](https://docs.dwavesys.com/docs/latest/doc_physical_properties.html).
- [21] Mohammad H. Amin. “Searching for quantum speedup in quasistatic quantum annealers”. In: *Phys. Rev. A* 92 (5 Nov. 2015), p. 052323. DOI: 10.1103/PhysRevA.92.052323. URL: <https://link.aps.org/doi/10.1103/PhysRevA.92.052323>.
- [22] Guanglei Xu and William Oates. “Adaptive hyperparameter updating for training restricted Boltzmann machines on quantum annealers”. In: *Scientific Reports* 11 (Feb. 2021), p. 2727. DOI: 10.1038/s41598-021-82197-1.
- [23] D-Wave Systems Inc. *Ocean Developer Tools*. URL: <https://www.dwavesys.com/solutions-and-products/ocean/>.
- [24] D-Wave Systems Inc. *Solver Parameters*. URL: [https://docs.dwavesys.com/docs/latest/c\\_solver\\_parameters.html](https://docs.dwavesys.com/docs/latest/c_solver_parameters.html).
- [25] D-Wave Systems Inc. *Solver Properties*. URL: [https://docs.dwavesys.com/docs/latest/c\\_solver\\_properties.html](https://docs.dwavesys.com/docs/latest/c_solver_properties.html).
- [26] Steven H. Adachi and Maxwell P. Henderson. *Application of Quantum Annealing to Training of Deep Neural Networks*. 2015. arXiv: 1510.06356 [quant-ph].
- [27] Marcello Benedetti et al. *Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning*. 2016. arXiv: 1510.07611 [quant-ph].
- [28] Eric R. Anschuetz and Cristian Zanoci. “Near-term quantum-classical associative adversarial networks”. In: *Physical Review A* 100.5 (Nov. 2019). ISSN: 2469-9934. DOI: 10.1103/physreva.100.052327. URL: <http://dx.doi.org/10.1103/PhysRevA.100.052327>.
- [29] Nathan Wiebe and Leonard Wossnig. *Generative training of quantum Boltzmann machines with hidden units*. 2019. arXiv: 1905.09902 [quant-ph].
- [30] Lorenzo Rocutto, Claudio Destri, and Enrico Prati. *Quantum Semantic Learning by Reverse Annealing an Adiabatic Quantum Computer*. 2020. arXiv: 2003.11945 [quant-ph].
- [31] Vivek Dixit et al. “Training Restricted Boltzmann Machines With a D-Wave Quantum Annealer”. In: *Frontiers in Physics* 9 (2021), p. 374. ISSN: 2296-424X. DOI: 10.3389/fphy.2021.589626. URL: <https://www.frontiersin.org/article/10.3389/fphy.2021.589626>.
- [32] Salmenperä Ilmo. *Training Quantum Restricted Boltzmann Machines Using Dropout Method*. 2021.
- [33] Max Wilson et al. “Quantum-assisted associative adversarial network: applying quantum annealing in deep learning”. In: *Quantum Machine Intelligence* 3.1 (June 2021). ISSN: 2524-4914. DOI: 10.1007/s42484-021-00047-9. URL: <http://dx.doi.org/10.1007/s42484-021-00047-9>.
- [34] Jeffrey Marshall et al. “Power of Pausing: Advancing Understanding of Thermalization in Experimental Quantum Annealers”. In: *Physical Review Applied* (2019).

- [35] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. MIT Press, 2012.
- [36] Brian Coyle et al. *Quantum versus Classical Generative Modelling in Finance*. 2020. arXiv: 2008.00691 [quant-ph].
- [37] Thomas Pochart, Paulin Jacquot, and Joseph Mikael. *On the challenges of using D-Wave computers to sample Boltzmann Random Variables*. 2021. arXiv: 2111.15295 [quant-ph].
- [38] Lev Barash et al. “Estimating the density of states of frustrated spin systems”. In: *New Journal of Physics* 21.7 (July 2019), p. 073065. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ab2e39. URL: <http://dx.doi.org/10.1088/1367-2630/ab2e39>.
- [39] D-Wave Systems Inc. *Error Sources for Problem Representation*. URL: [https://docs.dwavesys.com/docs/latest/c\\_qpu\\_ice.html](https://docs.dwavesys.com/docs/latest/c_qpu_ice.html).
- [40] D-Wave Systems Inc. *Other Error Sources*. URL: [https://docs.dwavesys.com/docs/latest/c\\_qpu\\_errors.html](https://docs.dwavesys.com/docs/latest/c_qpu_errors.html).
- [41] Jerome L. Myers, A. Well, and Robert Frederick Lorch. *Research Design and Statistical Analysis*. Routledge, 2010.
- [42] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.
- [43] Box George E P., Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, 1994.
- [44] Skipper Seabold and Josef Perktold. “statsmodels: Econometric and statistical modeling with python”. In: *9th Python in Science Conference*. 2010.
- [45] D. Foreman-Mackey et al. “emcee: The MCMC Hammer”. In: *PASP* 125 (2013), pp. 306–312. DOI: 10.1086/670067. eprint: 1202.3665.
- [46] Jonathan Goodman and Jonathan Weare. “Ensemble samplers with affine invariance”. In: *Communications in Applied Mathematics and Computational Science* 5.1 (2010), pp. 65–80. DOI: 10.2140/camcos.2010.5.65.
- [47] S. Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: <https://doi.org/10.1214/aoms/1177729694>.
- [48] Jiawei Han. *Kullback-Leibler Divergence*. URL: <http://hanj.cs.illinois.edu/cs412/bk3/KL-divergence.pdf>.
- [49] Gary Venter. “Tails of Copulas”. In: *Proceedings of the Casualty Actuarial Society* 89 (Jan. 2002).
- [50] Arthur Charpentier. *Copulas and tail dependence, part 1*. Sept. 2012. URL: <https://freakonometrics.hypotheses.org/2435>.