UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Arbeitsgruppe Quanteninformatik

Classical Simulation of Quantum Circuits with Restricted Boltzmann Machines

Master's Thesis

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Jannes Stubbemann

submitted to:

Jun. Prof. Dr. Sevag Gharibian

and

Dr. Robert Schade

Paderborn, April 3, 2020

Declaration

(Translation from German)

I hereby declare that I prepared this thesis entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

Original Declaration Text in German:

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

City, Date

Signature

Contents

1 Notations **1**

2 Boltzmann Machines **3**
2.1 Overview 3
2.2 Restricted Boltzmann machines 5
2.3 Gibbs Sampling 6
2.4 Supervised Learning 8
2.5 Application to Quantum Computing 10
2.5.1 Diagonal gates 10
2.5.2 Non-diagonal gates 13

Bibliography **15**

1 Notations

- sets with capital letters as X, V, H
- vectors have an arrow as $\vec{v}$
- i might either be an index or the imaginary number from the context

2 Boltzmann Machines

The following chapter gives an introduction to Boltzmann machines and their applications to the classical simulation of quantum computing.

An overview of the architecture and mathematical properties of Boltzmann machines are given in the first section. The restricted Boltzmann machine is motivated as a special kind of Boltzmann machine with helpful mathematical properties in the second part of this chapter. Afterwards the concepts of Gibbs sampling and supervised learning are explained. In the last section, a constructive approach is given on how restricted Boltzmann machines can be applied to the classical simulation of quantum computing.

The introduction to Boltzmann machines and restricted Boltzmann machines as well as the introduction to Gibbs sampling are based on [19] and [9] which are also recommended as a more throughout introduction into the topic. The section on supervised learning and gradient descent methods is based on [21]. The reader who is already familiar with the concept of Boltzmann machines and how they can be trained in a supervised manner can safely skip to section 2.5 which is based on the work of Jónsson, Bauer and Carleo [16].

2.1 Overview

The concept of the Boltzmann machine has first been proposed in the 1980s as a model for parallel distributed computing [15]. Boltzmann machines are physically inspired by the Ising Spin model and can be interpreted as energy based recurrent neural networks representing probability distributions over vectors $\boldsymbol{d}_i \in \{0,1\}^n$ [1].

A Boltzmann machine is a network of stochastic units (or neurons) $X = V \cup H$ which are segmented into *visible* neurons $V = \{v_1, \ldots, v_n\}$ and *hidden* neurons $H = \{h_1, \ldots, h_m\}$. The joint state of the visible neurons $\boldsymbol{v} = (v_1 \ldots v_n) \in \{0,1\}^n$ represents n-dimensional data points $\boldsymbol{d}_i \in \{0,1\}^n$ while hidden neurons increase the expressiveness of the Boltzmann machine by acting as non-linear feature detectors to model dependencies between the visible neurons [12].

The neurons are connected to each other by weighted links W_{ij} and poss biases a_i (visible) or b_i (hidden) respectively. In the general case, the neurons of a Boltzmann machine are allowed to be fully connected with each other. A graphical representation of a fully connected Boltzmann machine is shown in figure 2.1.

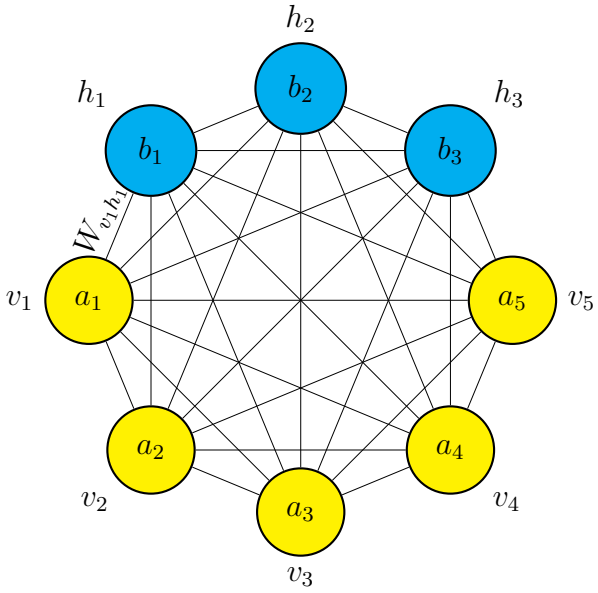

Figure 2.1: Graphical representation of a fully connected Boltzmann machine with 5 visible neurons (yellow) v_1 to v_5 and 3 hidden neurons (blue) h_1 to h_3. Each neuron posses a bias a_1 to a_5 and b_1 to b_3 respectively. The connection weight between two neurons i and j is given by W_{ij}.

Each configuration $\boldsymbol{c} = (v_1, \ldots, v_n, h_1, \ldots, h_m)$ of neuron states of the Boltzmann machine is associated with an energy $E(\boldsymbol{c})$ value which is defined by the parameters $\mathcal{W}$, consisting of its weights and biases $\mathcal{W} = \{a_i, b_i, W_{ij}\}$:

$$E(\boldsymbol{c}; \mathcal{W}) = -\sum_{v_i \in V} a_i v_i - \sum_{h_i \in H} b_i h_i - \sum_{x_i, x_j \in X} W_{x_i, x_j} x_i x_j \tag{2.1}$$

When sampling configurations from the Boltzmann machine (discussed in more detail in section 2.3) the Boltzmann machines prefers low energy states over states with a high energy. The stationary probability of a configuration $\boldsymbol{c}$ with energy $E(\boldsymbol{c}; \mathcal{W})$ is given by the so called Gibbs-Boltzmann distribution [10]:

$$p(\boldsymbol{c}; \mathcal{W}) = \frac{\mathrm{e}^{-E(\boldsymbol{c}; \mathcal{W})}}{Z(\mathcal{W})} \tag{2.2}$$

where $Z(\mathcal{W})$ is the normalizing partition function

$$Z(\mathcal{W}) = \sum_{\boldsymbol{c\prime} \in C} \mathrm{e}^{-E(\boldsymbol{c\prime}; \mathcal{W})} \tag{2.3}$$

In a training phase (discussed in section 2.4) the parameters of the Boltzmann machine can be adapted in such a way that the marginal probability distribution of the visible neurons

$$p(\boldsymbol{v}; \mathcal{W}) = \sum_{\boldsymbol{h}_k \in \{0,1\}^m} p(\boldsymbol{v}, \boldsymbol{h}_k; \mathcal{W}) \tag{2.4}$$

,which traces out the hidden unit states by summing over all possible configurations of them, resembles the probability distribution of data points $\boldsymbol{d}_i$ in a training set $D = \{d_1, \ldots, d_l\}$.

For a fully connected Boltzmann machine this representation consists of an exponential number of summands and thus cannot be calculated efficiently. So called Restricted Boltzmann machines (RBM) have a specific architecture with a restricted connectivity which allows the calculation of the marginal probability to be efficient. RBMs and their architectures will be explained in the next section.

2.2 Restricted Boltzmann machines

The so called Restricted Boltzmann machine (RBM) is an important type of Boltzmann machine with a specific architecture and properties [22]. Since their invention RBMs have been applied to variety of machine learning tasks and played a key role in the development of deep learning architectures as building blocks of so called Deep Belief networks [3, 14]. RBMs are also the kind of Boltzmann machines which are being used in this study for the simulation of quantum circuits.

In the restricted case the neurons of the Boltzmann machine are separated into two layers, one being the visible layer containing the visible neurons $v_i \in V$ and the other layer being the hidden layer containing the hidden neurons $h_j \in H$. Each neuron of the RBM is only allowed to be connected to the neurons from the other layer. Intra-layer connections are not allowed, making the graph of the RBM bipartite as shown in figure 2.2.

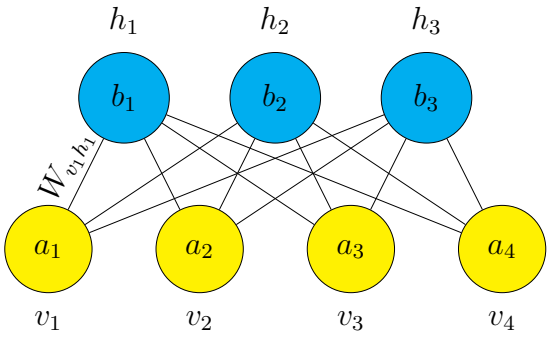

Figure 2.2: Graphical representation of a RBM with 5 visible neurons and 3 hidden ones. There are only connections between the two layers and no connection between two neurons from the same layer.

The marginal probability of the visible neuron states in a RBM has the form:

$$p(\boldsymbol{v};\mathcal{W}) = \sum_{\boldsymbol{h}_k\in\{0,1\}^m} p(\boldsymbol{v},\boldsymbol{h}_k;\mathcal{W}) \tag{2.5}$$

$$= \frac{1}{Z(\mathcal{W})} \sum_{\boldsymbol{h}_k\in\{0,1\}^m} \mathrm{e}^{-E(\boldsymbol{v},\boldsymbol{h}_k;\mathcal{W})} \tag{2.6}$$

$$= \frac{1}{Z(\mathcal{W})} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_m\in\{0,1\}} \mathrm{e}^{\sum_{v_i} b_i v_i} \prod_{j=1}^{m} \mathrm{e}^{h_j(b_j+\sum_{i=1}^{n} W_{ij}v_i)} \tag{2.7}$$

$$= \frac{\mathrm{e}^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \sum_{h_1\in\{0,1\}} \mathrm{e}^{h_1(b_1+\sum_{i=1}^{n} W_{i1}v_i)} \cdots \sum_{h_m\in\{0,1\}} \mathrm{e}^{h_m(b_m+\sum_{i=1}^{n} W_{im}v_i)} \tag{2.8}$$

$$= \frac{\mathrm{e}^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \prod_{i=1}^{m} \sum_{h_i\in\{0,1\}} \mathrm{e}^{h_i(b_i+\sum_{i=1}^{n} W_{ij}v_i)} \tag{2.9}$$

$$= \frac{\mathrm{e}^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \prod_{i=1}^{m} (1 + \mathrm{e}^{b_i+\sum_{i=1}^{n} W_{ij}v_i}). \tag{2.10}$$

This quantity consists of only a polynomial number of terms in the number of hidden units of the RBM and thus can be calculated efficiently. This makes the RBM a compact representation of a probability distribution over vectors $\boldsymbol{d}_i \in D$ inferred from a dataset D.

Even though the RBM has a restricted connectivity between its units, it is a universal function approximator [18]. It can model any distribution over $\{0,1\}^m$ arbitrary well with m visible and k+1 hidden units, where k denotes the cardinality of the support set of the target distribution, that is, the number of input elements from $\{0,1\}^m$ that have a non-zero probability of being observed. This also implies a worst case exponential number of hidden units for distributions with a large support set [18]. It has been shown though that even less units can be sufficient depending on the patterns in the support set [20].

2.3 Gibbs Sampling

Boltzmann machines are generative models representing probability distributions over their configurations. This means that it is possible to draw configurations from a Boltzmann machine according to their (marginal) probabilities given in equations 2.4 and 2.10.

Though it is required to calculate $Z(\mathcal{W})$ for the exact probabilities of each configuration, it is not necessary to calculate the energies for all the 2^{n+m} possible configurations of a Boltzmann machine to draw samples from it.

Instead, Boltzmann machines can be seen as *Markov chains* with a *stationary probability distribution*. With a stochastic process called *Gibbs sampling* the samples can be drawn efficiently according to this stationary distribution for RBMs.

This section gives a short introduction into Markov chains, Gibbs sampling and how it can be applied to draw configurations from a RBM.

Gibbs sampling belongs to the class of so called *Metropolis-Hastings* algorithms and by that is a *Monte Carlo Markov Chain* (MCMC) algorithm [11]. It is a simple algorithm to produce samples from the joint probability distribution of multiple random variables like in the case of neuron state configurations of a Boltzmann machine, which can be considered as a Markov chain.

A Markov chain is a discrete stochastic process of configurations of random variables $C = \{\boldsymbol{c}^{(t)}\}$ at time steps $t = 1, \ldots, T$ which take values in a set Ω (for Boltzmann machines $\Omega = \{0,1\}^{m+n}$) and for which for all time steps t and for all configurations $\boldsymbol{c}_j, \boldsymbol{c}_i, \boldsymbol{c}_{i-1}, \ldots, \boldsymbol{c}_0 \in \Omega$ the *Markov property* holds:

$$p_{ij}^{(t)} := P(\boldsymbol{c}^{(t+1)} = c_j \mid \boldsymbol{c}^{(t)} = \boldsymbol{c}_i, \ldots, \boldsymbol{c}^{(0)} = c_0) \tag{2.11}$$

$$= P(\boldsymbol{c}^{(t+1)} = c_j \mid \boldsymbol{c}^{(t)} = \boldsymbol{c}_i) \tag{2.12}$$

meaning that the next state of the system only depends on the current state and not on the history of the system. A Markov chain can be represented as a (finite) graph as the one shown in figure 2.3.

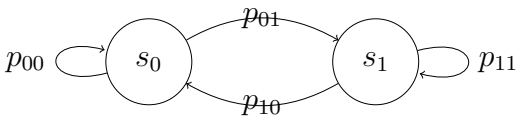

Figure 2.3: A Markov chain with two possible states c_0 and c_1. The Markov chain is described by the 2×2 matrix $\boldsymbol{P} = \ldots$

In the case of Boltzmann machines the transition probabilities p_{ij} are time independent and given by the ratios of configuration probabilities. For RBMs the transition probability p_{ij} from configuration c_i to c_j is given by:

$$p_{ij} = \frac{\mathrm{e}^{\sum_{v_i \in \boldsymbol{c_i}} b_i v_i} \prod_{i=1}^{m} (1 + \mathrm{e}^{b_i + \sum_{i=1}^{n} W_{ij} v_i})}{\mathrm{e}^{\sum_{v_i \in \boldsymbol{c_j}} b_i v_i} \prod_{i=1}^{m} (1 + \mathrm{e}^{b_i + \sum_{i=1}^{n} W_{ij} v_i})} \tag{2.13}$$

which can be calculated efficiently as the $Z(\mathcal{W})$ terms from equation 2.10 cancel out.

In each time step t during the Gibbs sampling, the state of a single randomly chosen neuron of the RBM is flipped so that the configurations $\boldsymbol{c}^{(t)}$ and $bmc^{(t+1)}$ only differ in the state of one neuron. With probability p_{ij} configuration $\boldsymbol{c}_j$ is kept as the new configuration and with probability $1 - p_{ij}$ the Boltzmann machine will stay in its current configuration bmc_i. This corresponds to a random walk on the Markov chain defined by the RBM transition probabilities. The algorithm is given in algorithm 2.3.

Algorithm 1 Gibbs Sampling

Require: $bm(c)$: function returning the energy of a Boltzmann machine for configuration c
Require: T: time steps
$t \leftarrow 0$
$c^{(0)} \leftarrow randomize(\{0,1\}^{m+n})$ (random initialization)
repeat
  $r \leftarrow random(m+n)$
  $E_i \leftarrow bm(c^{(t)})$
  $E_j \leftarrow bm(\{c_1, \ldots, \bar{c_r}, \ldots, c_{m+n}\})$
  if $random(1) < \frac{E_i}{E_j}$ **then**
    $c^{(t+1)} \leftarrow \{c_1, \ldots, \bar{c_r}, \ldots, c_{m+n}\}$
  $t \leftarrow t+1$
until $t = T$
return $c^{(T)}$

The Markov chain for configurations of a Boltzmann machine is known to converge to it so called *stationary distribution* π, that is

$$\pi^T = \pi^T \boldsymbol{P} \tag{2.14}$$

with $\boldsymbol{P} = (p_{ij})$ being the transition matrix of the Markov chain with the transition probabilities as its entries. Once the Markov chain reaches its stationary distribution, all subsequent states will be distributed according to this distribution.

This means running Gibbs sampling for sufficiently many time steps T will sample configurations according to the Gibbs-Boltzmann distribution given in equation 2.10.

Although Gibbs sampling is a very simple algorithm it is an important algorithm in the context of Boltzmann machines. Different versions of Gibbs sampling have been developed for RBMs like (persistent) contrastive divergence [13, 23] or parallel tempering [7]. In this study, Gibbs sampling will be used in its simplest version as described in this chapter.

2.4 Supervised Learning

The probability distribution over vector spaces given by a Boltzmann machine can be trained to resemble the distribution of data points $\boldsymbol{d}_i$ in a dataset $D = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_d\}$. This can be done either in a *unsupervised* or in a *supervised* manner.

In both cases the parameters $\mathcal{W} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{W}\}$ of the Boltzmann machine are updated minimizing an objective function $O(\mathcal{W})$ which depends on the parameters of the Boltzmann machine and depicts the overlap of the current and the target distribution in an iterative process called *gradient descent*.

This section gives a brief introduction into supervised learning and the gradient descent method *Adamax* [17].

Before the training phase, the Boltzmann machine is initialized with random parameters and a training set $D = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_d\}$ is given. In the case of supervised learning of Boltzmann machines the training set consists of tuples of configurations and their corresponding target energy values $\boldsymbol{d}_i = (\boldsymbol{c}_i, t_i)$.

In the batch version of gradient descent, which is being used in this study, the training set D is split into subsets $D_1, \ldots, D_l$, also called *batches*.

For each such batch D_i, the average overlap $O(\boldsymbol{c}_j; \mathcal{W})$ for all $\boldsymbol{c}_j \in D_i$ is computed. Afterwards, the gradients ΔO_x are calculated and the parameters updated into the direction of the steepest descent:

$$\Theta_i = \Theta_i + \Delta O \ldots. \tag{2.15}$$

This process is repeated for a defined number of iterations to minimize the objective function $O(\mathcal{W})$. A graphical representation of this process is shown in figure 2.4.

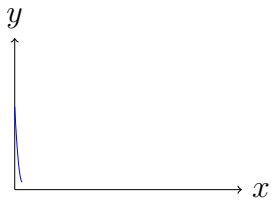

Figure 2.4: Iterations of gradient descent.

The simple update rule in 2.15 has several drawbacks though. First, The gradient and thus the step size will become smaller as closer the function approaches its minimum, leading to only small improvements and many steps necessary. Second, the shape of the function will usually have (many) local minima which should be avoided. Several variants of gradient descent have been developed to overcome these issues [21].

One successful such strategy which is used in this study is called *AdaMax*, a special version of *Adam* [17].It combines the advantages of AdaGrad [8] and RMSProp [2], two other popular gradient descent methods. It is computationally efficient, has little memory requirements and proofed to be well suited for problems with very noise and sparse gradients.

The algorithm updates exponential moving averages of the gradient (mt) and the squared gradient (vt) where the hyper-parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages. The moving averages themselves are estimates of the 1st moment (the mean) and the 2nd raw moment (the uncentered variance) of the gradient. The update rule for the parameters Θ_{ij} for Adamax are summarized in algorithm 2.4.

Algorithm 2 Adamax

Require: α: step size
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates
Require: $f(\Theta)$: Stochastic objective function with parameters $Theta$
Require: Θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector)
$u_0 \leftarrow 0$ (Initialize the exponential weighted infinity norm)
$t \leftarrow 0$ (Initialize time step)
while Θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\Theta f_t(\Theta_{t-1})$ (Get gradients w.r.t. objective function at time step t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $u_t \leftarrow max(\beta_2 \cdot u_{t-1}, |g_t)$ (Update the exponentially weighted infinity norm)
    $\Theta_t \leftarrow \Theta_{t-1} - (\alpha/(1 - \beta_1^t)) \cdot m_t/u_t$ (Update parameters)
return Θ_t (Resulting parameters)

2.5 Application to Quantum Computing

Machine learning techniques have become a successful approach for the classical simulation of quantum systems [5, 6, 4]. A compact presentation of the wavefunction of a many body quantum system with a RBM has first been given by Carleo and Troyer [5]. The same framework has later been used by Jónsson, Bauer and Carleo for the classical simulation of the quantum Fourier and Hadamard transform [16]. Their work gives a constructive approach on how the parameters of a complex valued RBM representing the quantum state

$$\Psi_{\mathcal{W}}(\mathcal{B}) = \frac{\mathrm{e}^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \prod_{i=1}^{m} (1 + \mathrm{e}^{b_i + \sum_{i=1}^{n} W_{ij} v_i}) \tag{2.16}$$

can be adapted to apply a universal set of quantum gates to the quantum state $\Psi_{\mathcal{W}}$.

RBMs allow only an exact application of unitary gates diagonal in the computational basis. For non-diagonal gates more hidden layers would be necessary, making the calculation of the quantum state intractable [4]. Nevertheless, it is possible to train a Boltzmann machine in a supervised fashion so its parameters are adapted to approximately apply a non-diagonal gate to its state.

The rules for the applications of diagonal and non-diagonal gates to a RBM state from [16] detailed in following two sections.

2.5.1 Diagonal gates

Diagonal gates can be applied to a RBM quantum state by solving a set of linear equations. This section describes the rules for the applications of single qubit Z

rotations, controlled Z rotations as well as the Pauli X, Y and Z gates.

Single-Qubit Z rotations

The action of the single Z rotation of angle θ is given by the 2×2 unitary matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\theta} \end{pmatrix}. \tag{2.17}$$

Its action on qubit l yields $\langle \mathcal{B} \mid R_l^z(\theta) \mid \Psi_W \rangle = \mathrm{e}^{i\theta B_l} \Psi_W(\mathcal{B})$. Considering a RBM machine with weights $\mathcal{W}\prime = \{\alpha, \beta, W\}$, the action of the $R^Z\theta$ gate is exactly reproduced if $\mathrm{e}^{B_l a_l} \mathrm{e}^{i\theta B_l} = \mathrm{e}^{B_l a\prime_l}$ is satisfied, which is the case for:

$$a\prime_j = a_j + \delta_{jl} i\theta \tag{2.18}$$

The action of the Z rotation thus simply modifies the bias of the visible neuron l of the RBM.

Controlled Z rotations

The action of a controlled Z rotations acting on two given qubits l and m is determined by the 4×4 unitary matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{i\theta} \end{pmatrix} \tag{2.19}$$

where θ is a given rotation angle. This gate is diagonal and can compactly be written as an effective two-body interaction:

$$\langle \mathcal{B} \mid CZ(\theta) \mid \Psi_W \rangle = \mathrm{e}^{i\theta B_l B_m} \Psi_W(Z_1 \dots Z_N). \tag{2.20}$$

As the RBM architecture does not allow direct interaction between visible neurons, the CZ gate requires the insertion of a dedicated extra hidden unit h_c which is connected to the qubits l and m:

$$\langle \mathcal{B} \mid CZ(\theta) \mid \Psi_W \rangle = \mathrm{e}^{\Delta a_l B_l + \Delta a_m Z_m} \sum_{h_c} \mathrm{e}^{W_{lc} B_l h_c + W_{mc} B_m h_c} \tag{2.21}$$

$$= \mathrm{e}^{\Delta a_l B_l + \Delta a_m B_m} \times (1 + \mathrm{e}^{W_{lc} B_l + W_{mc}} B_m) \Psi_W(\mathcal{B}), \tag{2.22}$$

where the new weights W_{lc} and W_{mc} and visible units biases $a_l\prime = a_l + \Delta a_l$, $a_m\prime = a_m + \Delta a_m$ are determined by the equation:

$$\mathrm{e}^{\Delta a_l B_l + \Delta a_m B_m} (1 + \mathrm{e}^{W_{lc} B_l + W_{mc} B_m}) = C \times \mathrm{e}^{i\theta B_l B_m} \tag{2.23}$$

for all the four possible values of the qubits values $B_l, B_m = \{0, 1\}$ and where C is an arbitrary (finite) normalization. A possible solution for this system is:

$$W_{lc} = -2\mathrm{A}(\theta) \tag{2.24}$$

$$W_{mc} = 2\mathrm{A}(\theta) \tag{2.25}$$

$$\Delta a_l = i\frac{\theta}{2} + \mathrm{A}(\theta) \tag{2.26}$$

$$\Delta a_m = i\frac{\theta}{2} - \mathrm{A}(\theta) \tag{2.27}$$

where $\mathrm{A}(\theta) = arccosh(\mathrm{e}^{-i\frac{\theta}{2}})$

Pauli X gate

The X gate just flips the qubit l and the RBM amplitudes are:
$\langle \mathcal{B} \mid X_l \mid \Psi_W \rangle = \langle B_1 \dots \bar{B}_l \dots B_N \mid \Psi_W \rangle$.
Since $\bar{B}_l = (1 - B_l)$, it must be satisfied that

$$(1 - B_l)W_{lk} + b_k = B_l W_{lk}\prime + b_k\prime \tag{2.28}$$

and

$$(1 - B_l)a_l = B_l a_l\prime + C \tag{2.29}$$

hold for all the (two) possible values of $B_l = \{0, 1\}$. The solution is simply:

$$W_{lk}\prime = -W_{lk} \tag{2.30}$$

$$b_k\prime = b_k + W_{lk} \tag{2.31}$$

$$a_l\prime = -a_l \tag{2.32}$$

$$C = a_l \tag{2.33}$$

whereas all the a_j and the other weights W_{jk} with $j \neq i$ are unchanged.

Pauli Y gate

A similar solution is found also for the Y gate, with the noticeable addition of extra phases with respect to the X gate:

$$W_{lk}\prime = -W_{lk} \tag{2.34}$$

$$b_k\prime = b_k + W_{lk} \tag{2.35}$$

$$a_l\prime = -a_l + i\pi \tag{2.36}$$

$$C = a_l + \frac{i\pi}{2} \tag{2.37}$$

whereas all the a_j and other weights W_{jk} with $j \neq l$ are unchanged.

Pauli Z gate

The Pauli Z gate is a special case of the Z rotation with $\theta = \pi$. Thus the only change necessary is to set the bias a_l of the visible neuron l to

$$a_l\prime = a_l + i\pi. \tag{2.38}$$

2.5.2 Non-diagonal gates

Exact applications of non-diagonal gates to Boltzmann machine quantum states would require the introduction of a second hidden layer [4]. In contrast to a RBM with just one hidden layer, this would make the calculation of the represented quantum state inefficient.

Thus there are no rules for the application of non-diagonal gates to a RBM state similar to those for diagonal gates. Nevertheless, the parameters of the RBM can be adapted in a supervised learning process instead.

The training set consists of samples drawn from the RBM with its current set of parameters. In the sampling process, the energy function of the RBM can be adapted to resemble the energy states after a non-diagonal unitary gate G has been applied to its state.

For any non-diagonal unitary gate

$$G = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{2.39}$$

applied to qubit l samples from the state $|\Phi(\mathcal{B})\rangle = G_l |\Psi(\mathcal{B})\rangle$ can be drawn according to

$$||\Phi(\mathcal{B}_{\mathcal{B}_l=0})\rangle|^2 = |a \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=0})\rangle + c \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=1})\rangle|^2 \tag{2.40}$$

when the state of qubit l is sampled to be 0 and

$$||\Phi(\mathcal{B}_{\mathcal{B}_l=1})\rangle|^2 = |b \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=0})\rangle + d \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=1})\rangle|^2 \tag{2.41}$$

when the state is sampled to be 1. The sampling happens according to the squared norm $|\Phi|^2$ to draw the samples according to the probability distribution

of the corresponding quantum state.

The samples are then being used to minimize the log likelihood of the overlap of the two quantum states Ψ and Φ

$$L(\mathcal{W}) = -\log O(\mathcal{W}) \tag{2.42}$$

$$= -\log \sqrt{\frac{|\langle \Psi_{\mathcal{W}} \mid \Phi \rangle|^2}{\langle \Psi_{\mathcal{W}} \mid \Psi_{\mathcal{W}} \rangle \langle \Phi \mid \Phi \rangle}} \tag{2.43}$$

$$= -\log \sqrt{\left\langle \frac{\Phi(\mathcal{B})}{\Psi_{\mathcal{W}}(\mathcal{B})} \right\rangle_{\Psi} \left\langle \frac{\Psi_{\mathcal{W}}(\mathcal{B})}{\Phi(\mathcal{B})} \right\rangle_{\Phi}^{*}} \tag{2.44}$$

with

$$\langle F(\mathcal{B}) \rangle_A := \frac{\sum_{\mathcal{B}} F(\mathcal{B}) |A(\mathcal{B})|^2}{\sum_{\mathcal{B}} |A(\mathcal{B})|^2} \tag{2.45}$$

by a gradient descent method. The gradients of this function with respect to the parameters of the Boltzmann machine have the following form:

$$\partial_{p_k} L(\mathcal{W}) = \langle \mathcal{O}_k^*(\mathcal{B}) \rangle_{\Psi} - \frac{\langle \frac{\Phi(\mathcal{B})}{\Psi(\mathcal{B})} \mathcal{O}_k^*(\mathcal{B}) \rangle_{\Psi}}{\langle \frac{\Phi(\mathcal{B})}{\Psi(\mathcal{B})} \rangle_{\Psi}} \tag{2.46}$$

with $\mathcal{O}_k(\mathcal{B}) = \partial_{p_k} \log \Psi_{\mathcal{W}}(\mathcal{B})$ being the variational derivatives of the RBMs wavefunction with respect to its parameters. These are simple to compute as well:

$$\partial_{a_i} \log \Psi_{\mathcal{W}}(\mathcal{B}) = v_i \tag{2.47}$$

$$\partial_{b_i} \log \Psi_{\mathcal{W}}(\mathcal{B}) = h_i \tag{2.48}$$

$$\partial_{W_{ij}} \log \Psi_{\mathcal{W}}(\mathcal{B}) = v_i h_j \tag{2.49}$$

After the training which happens for a predefined number of iterations on batches of the training set, the quantum state $\Psi_{\mathcal{W}}$ represented by the RBM approximates the target state Φ. Using this approach Jónsson et al. reported a per gate error of 10^{-3} [16].

Bibliography

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines*. *Cognitive Science*, 9(1):147–169, 1985.

[2] Yoshua Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *corr abs/1502.04390*, 2015.

[3] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[4] Giuseppe Carleo, Yusuke Nomura, and Masatoshi Imada. Constructing exact representations of quantum many-body systems with deep neural networks. *Nature communications*, 9(1):1–11, 2018.

[5] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[6] Dong-Ling Deng, Xiaopeng Li, and S Das Sarma. Quantum entanglement in neural network states. *Physical Review X*, 7(2):021021, 2017.

[7] Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Parallel tempering for training of restricted boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 145–152. MIT Press Cambridge, MA, 2010.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[9] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. pages 14–36, 01 2012.

[10] Josiah Willard Gibbs. *Elementary Principles in Statistical Mechanics: Developed with Especial Reference to the Rational Foundation of Thermodynamics*. Cambridge Library Collection - Mathematics. Cambridge University Press, 2010.

[11] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[12] Geoffrey Hinton. *Boltzmann Machines*, pages 132–136. Springer US, Boston, MA, 2010.

[13] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[14] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[15] Geoffrey E. Hinton and Terrence J. Sejnowski. Analyzing cooperative computation. In *Proceedings of the Fifth Annual Conference of the Cognitive Science Society, Rochester NY*, 1983.

[16] Bjarni Jónsson, Bela Bauer, and Giuseppe Carleo. Neural-network states for the classical simulation of quantum computing, 2018.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, 20(6):1631–1649, 2008.

[19] Guido Montufar. Restricted boltzmann machines: Introduction and review, 2018.

[20] Guido Montufar and Nihat Ay. Refinements of universal approximation results for deep belief networks and restricted boltzmann machines. *Neural computation*, 23(5):1306–1319, 2011.

[21] Sebastian Ruder. An overview of gradient descent optimization algorithms., 2016. cite arxiv:1609.04747Comment: Added derivations of AdaMax and Nadam.

[22] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[23] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.