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, March 29, 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

- 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

This 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. 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 is based on [9] and [3] which are also recommended as a more throughout introduction into the topic. The reader who is already familiar with the concept of Boltzmann machines can safely skip to section 2.5 which is based on the work of Jónsson, Bauer and Carleo [8].

2.1 Overview

The concept of the Boltzmann machine (BM) has first been proposed in the 1980s as a model for parallel distributed computing [7]. BMs are physically inspired by the Ising Spin model and can be interpreted as energy based recurrent neural networks representing probability distributions over vectors $\boldsymbol{d} \in \{0,1\}^n$ [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 data points $\boldsymbol{d}_i \in \{0,1\}^n$. The hidden neurons increase the expressiveness of the Boltzmann machine by acting as non-linear feature detectors to model dependencies between the visible neurons [5]. The neurons are connected to each other by weighted links W_{ij} and posses a bias a_i or b_i respectively. In the general case, Boltzmann machines are allowed to be fully connected. A graphical representation of a fully connected Boltzmann machine is shown in figure 2.1.

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 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

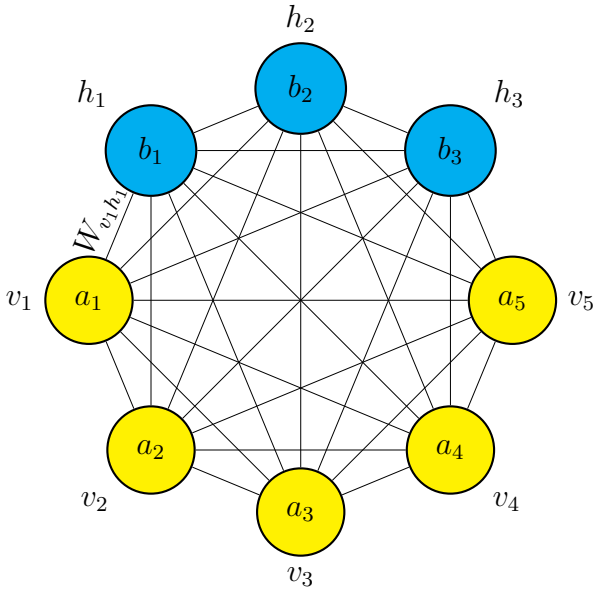

Figure 2.1: Graphical representation of a fully connected Boltzmann machine with 5 visible neurons (yellow) v_1 to v_5 and 3 hidden neurons (blue) h_1 to h_3. Each neuron posses a bias a_1 to a_5 and b_1 to b_3 respectively. The connection weight between two neurons i and j is given by W_{ij}.

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 [4]:

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

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 the parameters of the Boltzmann machine can be adapted in such a way that the marginal probability distribution of the visible neurons which traces out the hidden unit states by summing over all possible configurations of them:

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

resembles the probablity distribution of data points 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

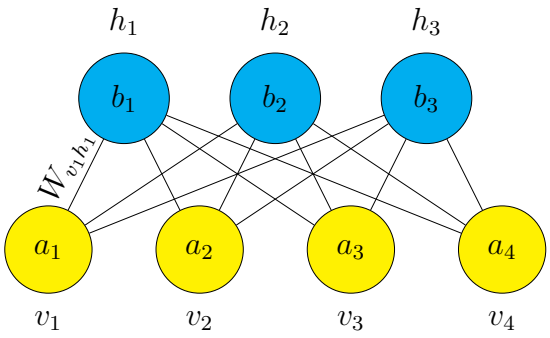

Figure 2.2: Graphical representation of a RBM with 5 visible neurons and 3 hidden ones. There are only connections between the two layers and no connection within one of the layers.

a restricted connectivity which makes the representation of the marginal probability compact as will be shown 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 [10]. Since their invention RBMs have been applied to variety of machine learning tasks. They also played a key role in the development of deep learning architectures as building blocks of so called Deep Belief networks [2, 6]. 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 seperated into two layers of visible and hidden neurons which form a bipartite graph. There are only connections allowed between the neurons from the two different layers and no intra-layer connections. The structure of an RBM is shown in figure 2.2.

The marginal probability of the visible neuron states in an RBM has a closed 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.

Before it will be shown how those probabilities can be used sample from the configurations of the neuron states of the Boltzmann machine in section 2.3, a short excursion on the Representational power of BMs and RBMs is made in the next section.

Even though the RBM has a restricted connectivity between its units, it is a universal function approximator. 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. It has been shown though that even less units can be sufficient depending on the patterns in the support set.

2.3 Gibbs Sampling

Boltzmann machines are generative models which represent probability distributions over their configurations $\boldsymbol{c} = \{v_1,\ldots,v_n,h_1,\ldots,h_m\}$. The probability for a configuration c_i is given in equation X and X for fully connected and restricted Boltzmann machines respectively.

Drawing samples from a Boltzmann machine according to their probabilities does not require to compute the energy function of all the 2^{n+m} possible configurations. Instead, samples can be drawn in a stochastic process called *Gibbs sampling*.

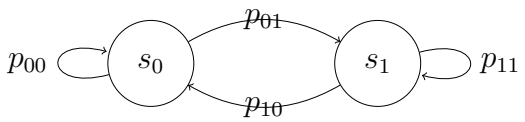

Figure 2.3: A Markov chain with two possible states c_0 and c_1. The Markov chain is desribed by the 2×2 matrix $\boldsymbol{P} = \ldots$

Gibbs sampling belongs to the class of so called *Metropolis-Hastings* algorithms and by that is a *Monte Carlo Markov Chain* (MCMC) method. 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. The possible configurations can be considered as a *Markov chain*.

A Markov chain is a discrete stochastic process of configurations of random variables $C = \{c^{(t)}\}$ at timesteps $t = 1, \ldots, T$ 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 $c_j, c_i, c_{i-1}, \ldots, c_0 \in \Omega$ the *Markov property* holds:

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

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

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 corresponds to a random walk on a finite graph as the one shown in figure 2.3.

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 again can be calculated efficiently.

In each timestep during the Gibbs sampling, the state of a single randomly chosen unit is flipped so that the configurations $c^{(t)}$ and $c^{(t+1)}$ only differ in the state of one neuron. With probability p_{ij} configuration c_j is kept as the new configuration and with probability $1 - p_{ij}$ the Boltzmann machine will stay in configuration c_i.

This process is repeated for a predefined number of time steps T. The algorithm is given in algorithm 2.3.

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

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

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 initialisation)
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 $c^{(T)}$

with $\boldsymbol{P} = (p_{ij})$ being the transition matrix 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 that running the Gibbs sampling algorithm from X for sufficiently many time steps T will sample configurations from a Boltzmann machine according to the Gibbs-Boltzmann distribution given in equations X and X.

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 in a dataset. 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 typically depicts the overlap of the current and the target distribution in an iterative process called *gradient descent*.

Before the training phase, the Boltzmann machine is initialized with random parameters. Afterwards, in each iteration, the overlap for a subset of the training data, also called batch, is computed. Then, the gradient of this value is calculated to update the parameters in the direction of the steepest descent of the target function. This process is depicted in figure X.

In case of supervised learning for Boltzmann machine, the training set consists of tuples of configurations and target energy values. For each batch, the difference between the current and target energy state can be calculated and derived for the parameters of the Boltzmann machine. Afterwards, the parameters will be updated according to:

... (2.15)

As seen in fig X, the shape of the objective function typically contains many (local) minimas. There are several variations of stochastic gradient descent methods which try to avoid the local minimas and reach the global one. One successful such strategy that is also being used in this study is *AdaMax*, a special version using the infinity norm of *Adam*. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradient. It d is designed to combine the advantages of two recently popular methods: AdaGrad (Duchi et al., 2011), which works well with sparse gradients, and RMSProp.

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 2 nd raw moment (the uncentered variance) of the gradient

In Adamax the parameters are updated according to:

... (2.16)

which is computationally efficient, has little memory requirements and prooved to be well suited for problems with very noise and sparse gradients.

The method combines the advantages of two recently popular optimization methods: the ability of AdaGrad to deal with sparse gradients, and the ability of RMSProp to deal with non-stationary objectives. The method is straightforward to implement and requires little memory

2.5 Application to Quantum Computing

Boltzmann machine have been shown to be a good model for quantum physics. Carleo used RBMs to predict the wave functions of many body quantum states in X. Xiao could show that while General Boltzmann machines can represent the wave functions of many body systems directly, sampling becomes the P Sharp problem mentioned above. Restricted Boltzmann machines in contrast are not able to represent the exact states but can approach them reasonable well with a worst case inefficient representation but with an efficient sampling process. Carleo et al later used restriced Boltzmann machines for the classical simulation of quantum computing. This is the framework this study builds on top and will be explained in greater detail in this section.

With the concept of the RBM at hand the question remains how it can be used to represent the states of indivual qubits. The link is that as the wavefunction assigns an energy value to each state so does the Boltzmann machine assign an energy value to each state of visible units by formular X.

The state that is represented by the Boltzmann machine is therefore given by the superposition of possible state and their corresponding energy values:

FORMULAR

For the representation of quantum states the weights and biases of the RBM will be complex valued.

All gates which are diagonal in the computational basis can be applied by following rules to update the parameters of the RBM in order to satisfy the equations for the RBM. Non-diagonal gates can be approximated by training the Boltzmann machine to learn the state after the gate has been applied to the currently represented state of the RBM.

2.5.1 Diagonal 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 $\langle \mathfrak{B} \mid R_l^z(\theta) \mid \Psi_W \rangle = \mathrm{e}^{i\theta B_l} \Psi_W(\mathfrak{B})$. Considering a RBM machine with weights $W\prime = \{\alpha, \beta, W\}$, the action of the $R^Z\theta$ gate is exactly reproduced if we satisfy $\mathrm{e}^{B_l a_l} \mathrm{e}^{i\theta B_l} = \mathrm{e}^{B_l a\prime_l}$, which has the simple solution:

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

The action of this gate then simply modifies the local visible bias 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 we can compactly write it as an effective two-body interaction:

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

Since in the RBM architecture there is no direct interaction between visible spins, this CZ interaction can be mediated through the insertion of a dedicated extra hidden unit h_c which is coupled only to the qubits l and m:

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

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

, 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

We then consider a X gate acting on some given qubit l. In this case the gate just flips the qubit and the RBM amplitudes are:
$\langle \mathfrak{B} \mid X_l \mid \Psi_W \rangle = \langle B_1 \dots B_l . \bar{} . B_N \mid \Psi_W$,
therefore since $B_l \bar{=} (1 - B_l)$, we must satisfy

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

,

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

,
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

For a Z gate acting on qubit l we have:

$$\langle \mathfrak{B} \mid Z_l \mid \Psi_W \rangle = (-1)^{B_l} \langle \mathfrak{B} \mid \Psi_W \rangle \tag{2.38}$$

therefore we must satisfy $\mathrm{e}^{B_l a_l}(-1)^{B_l} = \mathrm{e}^{B_l a_l\prime}$, which has the simple solution:

$$a_l\prime = a_l + i\pi \tag{2.39}$$

,

whereas all the other weights and biases are unchanged.

2.5.2 Non-diagonal gates

While diagonal gates can be applied as shown for the gates above by solving the corresponding sysem of linear equations, there are no such rules for non-diagonal gates (why not?). Instead, the effect of non-diagonal gates on a qubit l have to be learned with the following framework.

Any non diagonal unitary of the form

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{2.40}$$

has the effect that when applied to qubit l currently in state 0: ... and ... when qubit l is currently in state 1. This means with the current RBM at hand we can sample from the state after the gate has been applied by sampling from the current state and adapting the wave function accordingly. This can be done efficiently as it is a simple addition of two wave function values which in turn can be calculated efficiently as well. This approach allows us to generate a training set with samples of the after gate state and the corresponding target wave function values. This training set can be used to minimize the overlap of the current and the target state:

For numerical reasons, it is a common trick to not minimize the bare overlap but the log of the overlap:

FORMULAR

which attains a minimum at .... The derivate of the log likelihood with respect to paramter k of the Boltzmann machine then reads as:

FORMULAR

which can be calculated efficently. Using a gradient descent the parameters can be updated in such a way to minimize the log overlap and approach the desired state.

In their work, Carleo et al tested the framework on a fast fourier transform by applying ... . They could find a per gate error of 10 to the minus three, which is similar to the one currently archivable by physical quantum computers [].

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

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

[4] 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.

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

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

[7] 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.

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

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

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