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, July 13, 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

List of Figures

List of Algorithms

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 Quantum Computing

This chapter gives an introduction to the field of *Quantum Computation*. It covers the basics of qubits, quantum gates and circuits and a high level overview of quantum complexity classes. This chapter is based on [21] which is recommended as a reference for a deeper introduction into the field.

2.1 Qubits

While classical computers harness classical physical phenomena like electrical current to perform calculations, quantum computers harness the properties of *quantum mechanics*. The simplest quantum mechanical system is a quantum bit, or qubit for short. The qubit is the fundamental building block of a quantum computer. Its properties will be explained in this section.

Mathematically, a qubit can be described by a two dimensional complex valued unit vector, also called the *state vector* or *wave function* of the qubit:

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{2.1}$$

with amplitudes $\alpha, \beta \in \mathbb{C}$ and $\alpha^2 + \beta^2 = 1$. Rewriting $|\psi\rangle$ as

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\rho}\sin\frac{\theta}{2}|1\rangle) \tag{2.2}$$

with $\theta, \rho, \in \mathbb{R}$ allows for an intuitive interpration of a qubit as a point on the surface of the three dimensional unit sphere, also called the *Bloch Sphere*, which is visualised in figure **??**. The factor $e^{i\gamma}$ in front is called a *global phase* and can be ignored:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\rho}\sin\frac{\theta}{2}|1\rangle \tag{2.3}$$

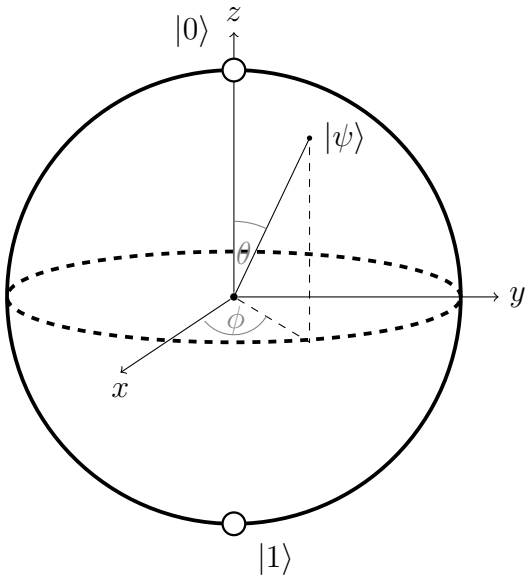


Figure 2.1: The Bloch Sphere in its full beauty.

The states on the north and south pole of the Bloch Sphere are called the *computational basis states* defined as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{2.4}$$

and

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.5}$$

These states correspond to the classical states 0 and 1 of a classical bit. Thus, yet another way of writing the wave function of a qubit, which is also the most practical one for quantum computation, is as a linear combination of the two computational basis states:

$$|\psi\rangle = \alpha \cdot |0\rangle + \beta \cdot |1\rangle. \tag{2.6}$$

A qubit whose amplitudes α and β are both non-zero is in a so called *superposition* of the two computational basis state. A qubit in a superposition can be interpreted as being in both states $|0\rangle$ and $|1\rangle$ at the same time. This property is one of the underlying reason why the classical simulation of quantum systems is computationally so hard and one of the sources of the potential computational power of quantum computers. It is not possible to read the values of the amplitudes α and β directly, though.

The qubit can only be in a superposition as long as it is completely isolated from its environment. Such a state is called a *coherent* state. As soon as the qubit interacts with its environment, as it is necessary in order to read its value,

it collapeses randomly into one of the two computational basis states. All the information stored in the amplitudes before gets lost in this process.

In a quantum computer the qubits are in a coherent state and potentially in a superposition during the computation. At the end of the computation, a so called *projective measurement* is performed to read the states of the individual qubits. A projective measurement is given by *operators*, in our case $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. The probablities to observe a qubit in the $|0\rangle$ or $|1\rangle$ state are then defined by the amplitudes

$$\begin{aligned} p(0) &= \langle\psi| M_0^\dagger M_0 |\psi\rangle && (2.7)\\ &= \langle\psi| M_0 |\psi\rangle && (2.8)\\ &= \alpha \langle 0|0\rangle \langle 0| \alpha |0\rangle + \beta \langle 1|0\rangle \langle 0| \beta |1\rangle && (2.9)\\ &= \alpha^2 \langle 0|0\rangle \langle 0|0\rangle + \beta^2 \langle 1|0\rangle \langle 0|1\rangle && (2.10)\\ &= \alpha^2 && (2.11) \end{aligned}$$

and

$$\begin{aligned} p(1) &= \langle\psi| M_1^\dagger M_1 |\psi\rangle && (2.12)\\ &= \langle\psi| M_1 |\psi\rangle && (2.13)\\ &= \alpha \langle 0|1\rangle \langle 1| \alpha |0\rangle + \beta \langle 1|1\rangle \langle 1| \beta |1\rangle && (2.14)\\ &= \alpha^2 \langle 0|1\rangle \langle 1|0\rangle + \beta^2 \langle 1|1\rangle \langle 1|1\rangle && (2.15)\\ &= \beta^2 && (2.16) \end{aligned}$$

as $M^\dagger M = M$ and $\langle i|j\rangle = \delta_{ij}$.

After the measurement, the state of the qubit will be either

$$|\psi\dagger\rangle = \frac{M_0 |\psi\rangle}{\sqrt{p(0)}} \tag{2.17}$$

or

$$|\psi\dagger\rangle = \frac{M_1 |\psi\rangle}{\sqrt{p(1)}} \tag{2.18}$$

respectively, implying that the state of the qubit collapsed into one of the two computational basis states. Thus, any succinct measurement will result in the same outcome. In order to perform multiple measurements on the same state, it has to be prepared multiple times.

Two special qubit states which often occur in the domain of quantum computation are the $|-\rangle$ and $|+\rangle$ state defined as

$$|-\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle - \frac{1}{\sqrt{2}} \cdot |1\rangle \tag{2.19}$$

and

$$|+\rangle = \frac{1}{\sqrt{2}} \cdot |0\rangle + \frac{1}{\sqrt{2}} \cdot |1\rangle \,. \tag{2.20}$$

Those states differ in a *relative phase* and both have the same measurement statistics, lying between the states $|0\rangle$ and $|1\rangle$ on the Bloch Sphere:

$$\begin{aligned} p(0) &= \langle +| \, M_0 \, |+\rangle && (2.21)\\ &= \frac{1}{\sqrt{2}} \langle 0| \, |0\rangle \, \langle 0| \frac{1}{\sqrt{2}} \, |0\rangle + \frac{1}{\sqrt{2}} \langle 1| \, |0\rangle \, \langle 0| \frac{1}{\sqrt{2}} \, |1\rangle && (2.22)\\ &= \frac{1}{2} \langle 0| \, |0\rangle \, \langle 0| \, |0\rangle + \frac{1}{2} \langle 1| \, |0\rangle \, \langle 0| \, |1\rangle && (2.23)\\ &= \frac{1}{2} && (2.24) \end{aligned}$$

and

$$\begin{aligned} p(1) &= \langle +| \, M_1 \, |+\rangle && (2.25)\\ &= \frac{1}{\sqrt{2}} \langle 0| \, |1\rangle \, \langle 1| \frac{1}{\sqrt{2}} \, |0\rangle + \frac{1}{\sqrt{2}} \langle 1| \, |1\rangle \, \langle 1| \frac{1}{\sqrt{2}} \, |1\rangle && (2.26)\\ &= \frac{1}{2} \langle 0| \, |1\rangle \, \langle 1| \, |0\rangle + \frac{1}{2} \langle 1| \, |1\rangle \, \langle 1| \, |1\rangle && (2.27)\\ &= \frac{1}{2} && (2.28) \end{aligned}$$

for the $|+\rangle$ state. The same probablities hold for the $|-\rangle$ state. Again, the state will be destroyed on measurement and be either $|0\rangle$ or $|1\rangle$ afterwards, depending on the measurement outcome.

A qubit represents the simplest quantum system. The stochastic nature of quantum physics allows a qubit to be in a superposition of the two computational basist states $|0\rangle$ and $|1\rangle$. Nevertheless, it is not possible to access such a superposition state directly but only ever to observe a qubit in one of the two computational basis states.

The next section will describe how combining multiple qubits into a bigger quantum system gives raise to an exponential growth of the state vector of such systems and how multiple qubits can be *entangeled* with each other.

2.2 Multiple Qubits and Entanglement

The state vector of a single qubit lies in a two dimensional space, assigning a complex valued amplitude to each of the both possible measurement outcomes $|0\rangle$ and $|1\rangle$. The state vector of a multi qubit system is defined by the *tensor product* of the state vectors of the individual subsystems. For a two qubit system consisting of $|\psi_1\rangle = \alpha_1 |0\rangle + \beta_1 |1\rangle$ and $|\psi_2\rangle = \alpha_2 |0\rangle + \beta_2 |1\rangle$ the state space is given by:

$$
\begin{aligned}
|\psi_{1,2}\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle && (2.29)\\
&= (\alpha_1 |0\rangle + \beta_1 |1\rangle) \otimes (\alpha_2 |0\rangle + \beta_2 |1\rangle) && (2.30)\\
&= \alpha_1\alpha_2 |0\rangle |0\rangle + \alpha_1\beta_2 |0\rangle |1\rangle + \beta_1\alpha_2 |1\rangle |1\rangle + \beta_1\beta_2 |1\rangle |1\rangle && (2.31)\\
&= \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\alpha_2 |10\rangle + \beta_1\beta_2 |11\rangle && (2.32)\\
&= \alpha_1\alpha_2 |0\rangle + \alpha_1\beta_2 |1\rangle + \beta_1\alpha_2 |2\rangle + \beta_1\beta_2 |3\rangle && (2.33)
\end{aligned}
$$

where we define $|0\rangle = |00\rangle$, $|1\rangle = |01\rangle$, $|2\rangle = |10\rangle$ and $|3\rangle = |11\rangle$. In the general case, the wave function of a n-dimensional qudit system is therefore given by a 2^n dimensional vector:

$$|\psi\rangle = \sum_{i=0}^{2^d} (a_i |i\rangle). \tag{2.34}$$

A classical simulation of a d-dimensional qudit system therefore has to keep track of 2^d complex amplitudes. This makes it impossible to simulate quantum systems above a certain size. For the classical simulation of a system consisting of 500 qubits the dimension of the state vector is already larger than the number of atoms in the universe.

As in the single qubit case, the probability to observe state $|n\rangle$ is given by a_n^2. It is also possible to only measure a subset of the qubits, leaving the other qubits in the normalised state:

$$|\psi\prime\rangle = \frac{(M_m \otimes I) |\psi\rangle}{\sqrt{(p(m))}}. \tag{2.35}$$

An interesting property that can appear in multi qudit systems is *entanglement*. Consider for instance the two qudit state

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.36}$$

which can be created with a very simple quantum program as shown later in section **??**. Note that this state is a proper two qubit state as it fulfills the normalisation property $a_{00}^2 + a_{11}^2 = \frac{1}{\sqrt{2}}^2 + \frac{1}{\sqrt{2}}^2 = \frac{1}{2} + \frac{1}{2} = 1$.

Nevertheless, there are no two single qubit states $|a\rangle$ and $|b\rangle$ such that $|\psi\rangle =$

$|a\rangle |b\rangle$. The two qubits of $|\psi\rangle$ are *entangled* with each other. Measuring one of the qubits immediately determines the state of the other qubit, i.e. measuring the first qubit as $|0\rangle$ happens with probability:

$$p_1(0) = \langle\psi| (M_0 \otimes I)^\dagger (M_o \otimes I) |\psi\rangle \tag{2.37}$$
$$= \frac{1}{\sqrt{2}} \langle 00| \frac{1}{\sqrt{2}} |00\rangle \tag{2.38}$$
$$= \frac{1}{2} \tag{2.39}$$

leaving the system in the post measurement state

$$|\psi\prime\rangle = \frac{(M_0 \otimes I)\psi}{\sqrt{p_1(0)}} \tag{2.40}$$
$$= \frac{\frac{1}{\sqrt{2}} |00\rangle}{\frac{1}{\sqrt{2}}} \tag{2.41}$$
$$= |00\rangle . \tag{2.42}$$

The same maths apply for the case of measuring the first qubit as $|1\rangle$.

Even if the state of the second qubit is not determined before, it will be either $|0\rangle$ or $|1\rangle$ from the moment on the *other* qubit has been measured. The exponential growth of the state space dimesion and entanglement are two important properties of quantum systems which make them hard to simulate classicaly. Known quantum algorithms with quantum speedups like Shor's algorithm create entanglement in clever ways to perform calculations in ways which seem to be impossible for classical computers.

The next section introduces the notion of *quantum gates* which allow to manipulate the state of a single or multiple qubits.

2.3 Quantum Gates

It is physically possible to modify the state vector of a quantum system even when it is in a coherent state. This opens the theoretical possibility to perform calculations with them and build quantum computers.

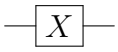
Mofifications happen by the application of so called *quantum gates* to the state. Quantum physics allows the gates only to be linear and reversable. Thus, quantum gates can be mathematically described as unitary matrices. Gates vary in the number of qubits on which they act as well as in the effects they have on the qubits' state.

This chapter will introduce single and multi qubit gates and gives an overview over some common quantum operations.

2.3.1 Single Qubit Gates

Single qubit gates are linear mappings that can be applied to the state vector of a single qubit. Mathematically, single qubit gates can be described by 2×2 unitary matrices. The fact that these matrices are unitary makes sure the state $|\phi\rangle = G|\psi\rangle$ after gate G has been applied to state $|\psi\rangle$ is a proper quantum state which preserves the normalisation constraint $\alpha_{|\phi\rangle}^2 + \beta_{|\phi\rangle}^2 = 1$.

An example of a single qubit gate is the NOT or X gate, also denoted as:



or

The X gate is the quantum analoge of the classical NOT gate. Similar to the classical version, the X gate swaps the states $|0\rangle$ and $|1\rangle$ when applied to them. It is even more general though, as it maps a single qubit state of the form

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.43}$$

to the state

$$|\phi\rangle = \beta|0\rangle + \alpha|1\rangle \tag{2.44}$$

thus swapping the amplitudes for $|0\rangle$ and $|1\rangle$. The X gate has the following matrix representation:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.45}$$

A single qubit gate is applied to a quantum state by matrix multiplication:

$$\begin{aligned} |\phi\rangle &= X|\psi\rangle && (2.46)\\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}|\psi\rangle && (2.47)\\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} && (2.48)\\ &= \begin{pmatrix} \beta \\ \alpha \end{pmatrix} && (2.49) \end{aligned}$$

Another example for a single qubit gate with no classical analoge is the *Hadamard gate* or H gate, described by the unitary:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.50}$$

Notice that the H gate is a unitary as it is reversable:

$$H^\dagger H = I \tag{2.51}$$

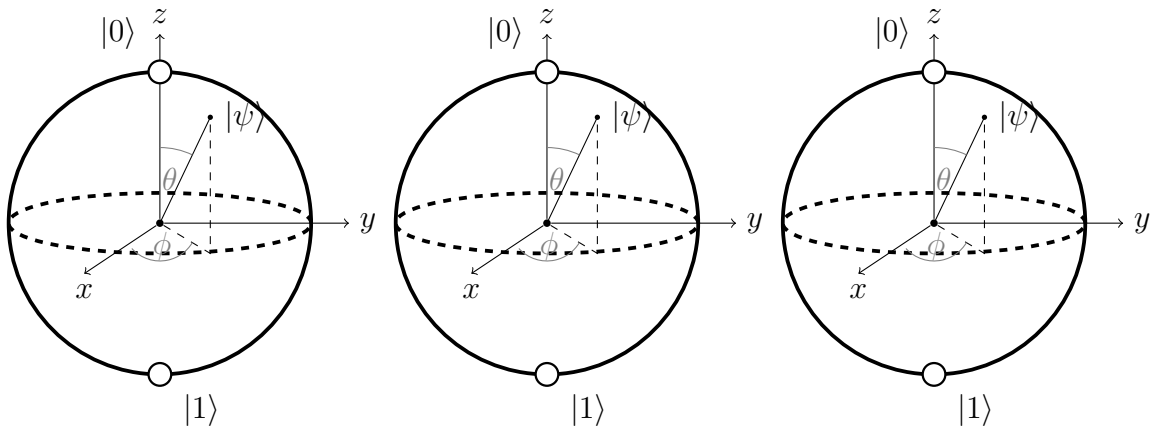
The Hadamard gate maps between the so called Z and X bases, mapping the states:

$$H\left|0\right\rangle = \left|+\right\rangle \tag{2.52}$$
$$H\left|1\right\rangle = \left|-\right\rangle \tag{2.53}$$

and back

$$H\left|+\right\rangle = \left|0\right\rangle \tag{2.54}$$
$$H\left|-\right\rangle = \left|1\right\rangle . \tag{2.55}$$

The Hadamard gate is also a good example of how the Bloch Sphere visualisation can help to understand the transformation of a qubit state. The application of the Hadamard gate to the $|+\rangle$ is shown in figure X.



(a) $y = x$ (b) $y = 3sinx$ (c) $y = 5/x$

Figure 2.2: Application of the Hadamard gate visualised in the Bloch Sphere.

in figure **??**.

Operator	Gate	Matrix
X		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
H		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
S		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
T		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{pmatrix}$

Though there are indefinitely many possible quantum gates in theory, a universal finite gate set consisting of only a few gates is sufficient to construct arbitrary unitary transformations. This is the same as in the classical case where NAND gates suffice to build up arbitrary classical computation circuits and good news for the facrication of quantum computers with reusable components.

However, single qubit gates are not sufficient to build all possible unitary transformations on multi qubit systems. For this purpose, multi qubit gates are a necessity.

2.3.2 Multi Qubit Gates

Single qubit gates are not enough to create universal gate sets for quantum computation. In order to run arbitrary quantum programs *controlled* gates are needed. Controlled quantum gates are simple extensions of single qubit gates. Every single qubit gate can be implemented as a controlled version of it with one *control* and one *target* qubit. Only if the control qubit is in the $|1\rangle$ state, the gate is applied to the target qubit. As the control qubit can be in a superposition state, it is possible to apply and not apply the gate to the target qubit at the same time, so to say.

The propotypical two qubit gate is the controlled NOT or CNOT gate. It is the controlled version of the X gate described before. As the CNOT gate acts on a two qubit state, it can be described by a 4×4 matrix. Each column describes

the mapping of one of the four base state $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. The matrix representation of the CNOT gate is the following:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.56}$$

The matrix can be read as follows: The first two colums describe that the vectors $|00\rangle$ and $|01\rangle$ won't change when the gate is applied to these states. Only when the first qubit is in the $|1\rangle$ state, the state of the second qubit will be swapped, thus $|10\rangle$ will be mapped to $|11\rangle$ and $|11\rangle$ to $|10\rangle$.

This also shows how the matrix representation of generalised two qubit gates can be derived: As long as the control qubit is off, the gate has no effect on the state. Thus, the matrix for the controlled gate is the same as the identity with 1s on the diagonal and 0s elsewhere in the upper left corner. Only when the first qubit is in the $|1\rangle$ state, the matrix differs from the identity. For example, the controlled version of the Z gate

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.57}$$

is given by:

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{2.58}$$

Controlled gates are represented with a vertical line ending in a black dot indicating the control qubit:

With two qubit operations at hand, it is possible to build a universal gate set. A commonly used gate set is the so called *Clifford + T* set, consists of the gates $CNOT$, H, S and T. The Solovay-Kitaev theorem guarantees that this set can efficiently approximate any unitary operation. In this study, another universal gate set will be used though, composed of the CZ, $\sqrt{X}$, $\sqrt{Y}$ and T gates.

The next sections will demonstrate how gates can be composed into *quantum circuits*, a notation for quantum programs.

2.4 Quantum Circuits

The standard model for computation in theoretical computer science is the Turing Machine. Inventend by Alan Turing in 1936 during World War Two, it is a powerful tool to understand the limits of (classical) computation. The theoretical nature of the Turing machine gives it unlimited computational resources like memory though, which do not reflect the properties of realisable computer architectures.

Another model of computation which does not suffer from this gap between theory and practical devices is the *circuit model*. In the classical circuit model, each bit is represented by a wire and operations (or gates) are represented by different shapes acting on those wires. The circuit model is as powerful as a Turing Machine and the model of choice in the field of quantum computing.

Quantum programs are described by quantum circuits. As in the classical case, each qubit is represented by a wire and each gate by a rectangular on those wires. Figure **??** describes a very simple quantum circuit which flips a single qubit starting in the $|0\rangle$ state by applying a X gate to it before measuring it.

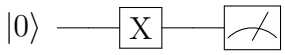


Figure 2.3: A simple quantum circuit.

The circuit is being read from left to right. Important to note though is that for the calcuation of the resulting state, the matrix representation of the gates are multiplied from the left side to the current state by simply following the rules of linear algebra. In the example above the final state of the program represented at the end of the circuit is:

$$
\begin{aligned}
|\phi\rangle &= X\,|\psi\rangle && (2.59)\\
&= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} |\psi\rangle && (2.60)\\
&= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} && (2.61)\\
&= \begin{pmatrix} \beta \\ \alpha \end{pmatrix}. && (2.62)
\end{aligned}
$$

In practice, circuits will consist of several qubits and multiple gates applied to them. It is possible to apply gates to different qubits sequentially as well as in parallel:

Note that when calculating the state of the quantum program above, one has to build the tensor product of all qubit states to calculate the state of the system. For the circuit from figure **??**, the initial state before any gate has been applied is:

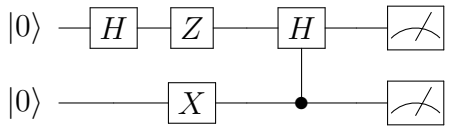


Figure 2.4: A slightly more complex quantum circuit.

$$|\psi_{init}\rangle = |0\rangle \otimes |0\rangle \tag{2.63}$$
$$= \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} \tag{2.64}$$
$$= \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \tag{2.65}$$
$$= |00\rangle \tag{2.66}$$

When a gate gets applied to only a subset of the qubits as in the case of the first H gate in the circuit above, the unitary applied to the multi qubit state is implicitely the tensor product of the gate on that qubit and identity matrices on the remaining qubits. For the given circuit, the state $|\psi_{H_1}\rangle$ after the first H gate on the first qubit is calculated as:

$$|\psi_{H_1}\rangle = (H \otimes I)\,|00\rangle \tag{2.67}$$
$$= (\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix} \otimes \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix})\,|00\rangle \tag{2.68}$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0 & 1 & 0\\0 & 1 & 0 & 1\\1 & 0 & -1 & 0\\0 & 1 & 0 & -1\end{pmatrix}|00\rangle \tag{2.69}$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0 & 1 & 0\\0 & 1 & 0 & 1\\1 & 0 & -1 & 0\\0 & 1 & 0 & -1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \tag{2.70}$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\1\\0\end{pmatrix} \tag{2.71}$$
$$= |+0\rangle \tag{2.72}$$

This corresponds to the intuition that after applying a H gate to the first qubit,

it should be in the $|+\rangle$ state while the second qubit remains in the $|0\rangle$ state.

Similar as for the first gate, the unitary applied to the state when multiple gates are applied to different qubits at the same time, is constructed by the tensor product of those gates. The state $|\psi_{Z_1X_2}\rangle$ after the Z gate is applied to first and the X gate is applied to the second qubit is calculated as:

$$
\begin{aligned}
|\psi_{Z_1X_2}\rangle &= (Z \otimes X)\,|+0\rangle && (2.73)\\
&= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}) \,|+0\rangle && (2.74)\\
&= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix} |+0\rangle && (2.75)\\
&= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} && (2.76)\\
&= \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix} && (2.77)\\
&= |-1\rangle && (2.78)
\end{aligned}
$$

This again matches with the intuition that applying a Z gate to the $|+\rangle$ state leaves the first qubit in the $|-\rangle$ state and the X gate on the second qubit moves it from the $|0\rangle$ to the $|1\rangle$ state.

The final state $|\psi_{final}\rangle$ of the circuit is calculated by applying the 4×4 matrix representation of the controlled H gate to $|-1\rangle$. Notice that the second qubit is the controlled qubit in this case:

$$|\psi_{final}\rangle = CH\,|-1\rangle \tag{2.79}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} |-1\rangle \tag{2.80}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{2.81}$$

$$= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.82}$$

$$= |11\rangle \tag{2.83}$$

The first qubit has been swaped from the $|-\rangle$ to the $|1\rangle$ state as the second qubis is in the $|1\rangle$ state and thus the H gate has been applied. The circuit **??** thus maps the inital state $|00\rangle$ to $|11\rangle$.

Another simple yet interesting quantum circuit is shown in figure **??**.

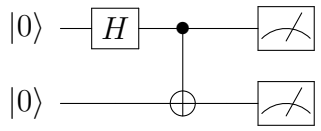


Figure 2.5: Bell creation circuit.

The Hadamard gate on the first qubit maps the system from the initial $|00\rangle$ into the $|+0\rangle$ state. Afterwards, the first qubit, which is currently between the $|0\rangle$ and $|1\rangle$ state on the Bloch Sphere, is used as the control qubit in the $CNOT$ gate. This has an interesting effect on the second qubit as the X gate is applied and not applied at the same time to it, entangling the two qubits with each other. The final state before measurement is then $\frac{|00\rangle+|11\rangle}{2}$ which has already been discussed in section X. The state is also known as one of the for *Bell states* which represent maximally entangled two qubit states and which play an important role in the analysis of *quantum communication*.

This simple quantum circuit demonstrates how entanglement can be created by the application of controlled gates with qubits in superposition states. Entanglement plays an important role in the construction of so called random quantum circuits in recent quantum supremacy exeperiments. These kind of circuits will be discussed in chapter X after a short overview of the theoretical computational power of quantum computers.

2.5 Quantum Computational Complexity

It is important to understand the theoretical capabilities and limitations of quantum computers to understand for which kind of problems they provide advantages over classical computers. For many years, it has been assumed that the extended Church Turing thesis, stating that a probablistic Turing machine can efficiently simulate any realistic model of computation, holds. This thesis is challenged by quantum computers which potentially can solve specific problems exponentially faster than classical computers.

Though there is no hard proof yet, there is complexity theoretical evidence that the extended Church Turing thesis might not hold and that quantum computers can be realised which can solve specific problems proofable faster than classical computers. The class of problems which can be solve efficiently by a quantum computer is called **BQP**, shorthand for *bounded-error quantum polynomial time*. A decision problem is in BQP if there exists a quantum program that solves the decision problem in 2/3 of the cases and runs in polynomial time. This is the quantum analoge of the *bounded-error probablistic polynomial time* or **BPP** which is decidable by a probablistic Turing machine in polynomial time and believed to be the same as **P**.

Currently, there are only a few problems known to be in BQP which are suspected not to be in P, providing evidence for the superiority of quantum computers. One of these problems is *factorisation*, the problem to decompose a composite integer into its prime factors. This problem has been known to be in *NP* before. It is also known, though, that factorisation is not an NP hard problem, indicating that quantum computers are probably not able to provide an exponential speedup for every problem which is not efficiently solvable on a classical computer. Indeed, classical algorithms might even exist which can solve factorisation on a classical computer efficiently that have not been discovered yet.

While BQP seems to include P and intersect with NP it can be shown that it is strictly included in PSPACE. The relationship of these complexity classes is visualised in figure 2.6.

While it is hard to proof some fundamental relationships between complexity classes like the famous $P = NP$ problem, it is believed that quantum computers can solve specific problems with practical applications like integer factorisation or the simulation of quantum systems up to exponentially faster than classical computers. The moment in time a physical quantum computer is able to outperform a classical computer on a specific problem for the first time has been coined by John Preskill in 2012 as *quantum supremacy*. Recently, Google announced their quantum supremacy results with a 54 qubit quantum computer, further challenging the extended Church Turing thesis and providing the first physical evedince that it might be possible to build quantum computers with advantages over classical computers in the future.

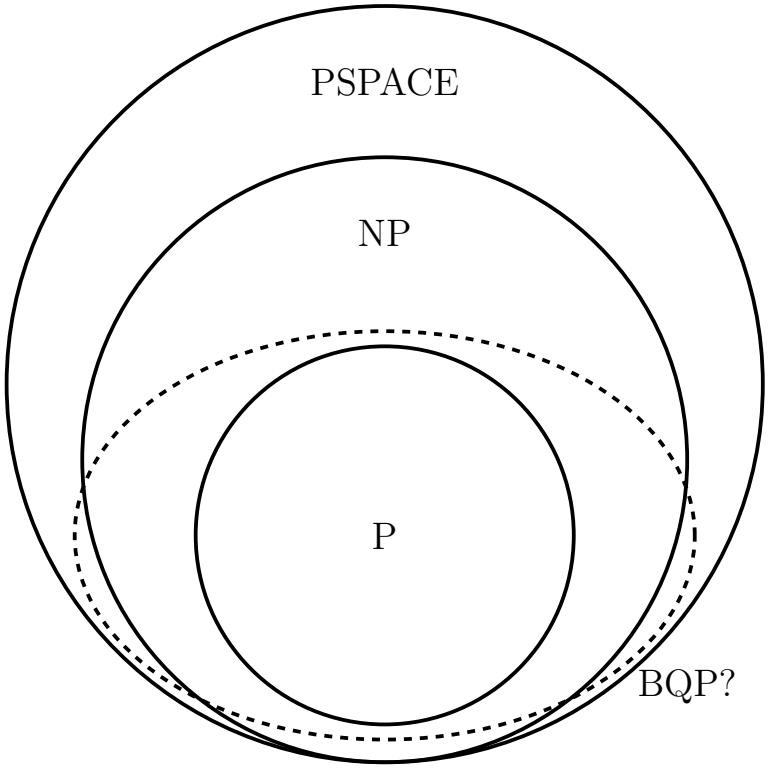


Figure 2.6: Relation of Complexity Classes to each other.

3 Random Circuit Sampling

Despite four decades of research in this field, there was no real proof whether it would be possible to build quantum computers which are truly more powerful than classical competers. This only changed recently, when Google announced the results of their quantum supremacy experiments on a 54 qubit quantum computer. To demonstrate the supremacy, they chose the problem of random circuit sampling which has been specically tailored to this purpose.

4 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 [22]. 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 4.5 which is based on the work of Jónsson, Bauer and Carleo [16].

4.1 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 **??**.

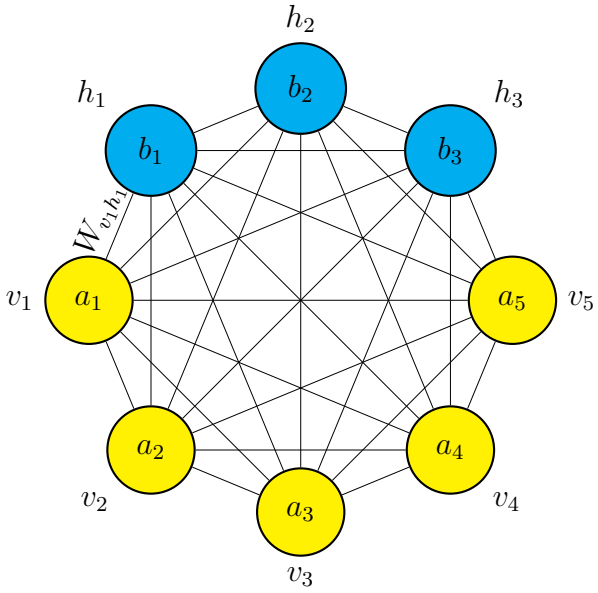


Figure 4.1: Graphical representation of a fully connected Boltzmann machine with 5 visible neurons (yellow) v_1 to v_5 and 3 hidden neurons (blue) h_1 to h_3. Each neuron posses a bias a_1 to a_5 and b_1 to b_3 respectively. The connection weight between two neurons i and j is given by W_{ij}.

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 $\mathcal{W} = \{a_i, b_i, W_{ij}\}$:

$$E(\boldsymbol{c}; \mathcal{W}) = -\sum_{v_i \in V} a_i v_i - \sum_{h_i \in H} b_i h_i - \sum_{x_i, x_j \in X} W_{x_i, x_j} x_i x_j \tag{4.1}$$

When sampling configurations from the Boltzmann machine (discussed in more detail in section 4.3) 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]:

$$p(\boldsymbol{c}; \mathcal{W}) = \frac{\mathrm{e}^{-E(\boldsymbol{c}; \mathcal{W})}}{Z(\mathcal{W})} \tag{4.2}$$

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

$$Z(\mathcal{W}) = \sum_{\boldsymbol{c\prime} \in C} \mathrm{e}^{-E(\boldsymbol{c\prime}; \mathcal{W})} \tag{4.3}$$

In a training phase (discussed in section 4.4) the parameters of the Boltzmann machine can be adapted in such a way that the marginal probability distribution of the visible neurons

$$p(\boldsymbol{v}; \mathcal{W}) = \sum_{\boldsymbol{h}_k \in \{0,1\}^m} p(\boldsymbol{v}, \boldsymbol{h}_k; \mathcal{W}) \tag{4.4}$$

,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.

4.2 Restricted Boltzmann machines

The so called Restricted Boltzmann machine (RBM) is an important type of Boltzmann machine with a specific architecture and properties [23]. 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 **??**.

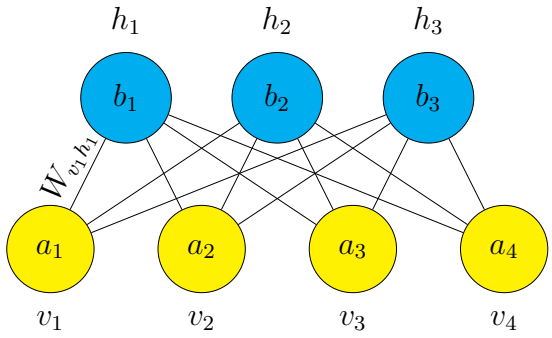


Figure 4.2: Graphical representation of a RBM with 5 visible neurons and 3 hidden ones. There are only connections between the two layers and no connection between two neurons from the same layer.

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

$$p(\boldsymbol{v};\mathcal{W}) = \sum_{\boldsymbol{h}_k\in\{0,1\}^m} p(\boldsymbol{v},\boldsymbol{h}_k;\mathcal{W}) \tag{4.5}$$

$$= \frac{1}{Z(\mathcal{W})} \sum_{\boldsymbol{h}_k\in\{0,1\}^m} \mathrm{e}^{-E(\boldsymbol{v},\boldsymbol{h}_k;\mathcal{W})} \tag{4.6}$$

$$= \frac{1}{Z(\mathcal{W})} \sum_{h_1\in\{0,1\}} \cdots \sum_{h_m\in\{0,1\}} \mathrm{e}^{\sum_{v_i} b_i v_i} \prod_{j=1}^{m} \mathrm{e}^{h_j(b_j+\sum_{i=1}^{n} W_{ij}v_i)} \tag{4.7}$$

$$= \frac{\mathrm{e}^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \sum_{h_1\in\{0,1\}} \mathrm{e}^{h_1(b_1+\sum_{i=1}^{n} W_{i1}v_i)} \cdots \sum_{h_m\in\{0,1\}} \mathrm{e}^{h_m(b_m+\sum_{i=1}^{n} W_{im}v_i)} \tag{4.8}$$

$$= \frac{\mathrm{e}^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \prod_{i=1}^{m} \sum_{h_i\in\{0,1\}} \mathrm{e}^{h_i(b_i+\sum_{i=1}^{n} W_{ij}v_i)} \tag{4.9}$$

$$= \frac{\mathrm{e}^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \prod_{i=1}^{m} (1 + \mathrm{e}^{b_i+\sum_{i=1}^{n} W_{ij}v_i}). \tag{4.10}$$

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].

4.3 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 4.4 and 4.10.

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:

$$p_{ij}^{(t)} := P(\boldsymbol{c}^{(t+1)} = c_j \mid \boldsymbol{c}^{(t)} = \boldsymbol{c}_i, \ldots, \boldsymbol{c}^{(0)} = c_0) \tag{4.11}$$

$$= P(\boldsymbol{c}^{(t+1)} = c_j \mid \boldsymbol{c}^{(t)} = \boldsymbol{c}_i) \tag{4.12}$$

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 **??**.

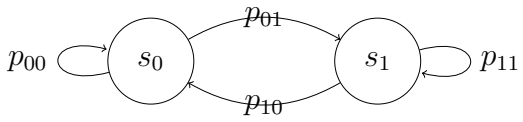


Figure 4.3: A Markov chain with two possible states c_0 and c_1. The Markov chain is described by the 2×2 matrix $\boldsymbol{P} = \ldots$

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:

$$p_{ij} = \frac{\mathrm{e}^{\sum_{v_i \in \boldsymbol{c_i}} b_i v_i} \prod_{i=1}^{m} (1 + \mathrm{e}^{b_i + \sum_{i=1}^{n} W_{ij} v_i})}{\mathrm{e}^{\sum_{v_i \in \boldsymbol{c_j}} b_i v_i} \prod_{i=1}^{m} (1 + \mathrm{e}^{b_i + \sum_{i=1}^{n} W_{ij} v_i})} \tag{4.13}$$

which can be calculated efficiently as the $Z(\mathcal{W})$ terms from equation 4.10 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 **??**.

Algorithm 1 Gibbs Sampling

Require: $bm(c)$: function returning the energy of a Boltzmann machine for configuration c
Require: T: time steps
1: $t \leftarrow 0$
2: $c^{(0)} \leftarrow randomize(\{0,1\}^{m+n})$ (random initialization)
3: **repeat**
4: $\quad r \leftarrow random(m+n)$
5: $\quad E_i \leftarrow bm(c^{(t)})$
6: $\quad E_j \leftarrow bm(\{c_1, \ldots, \bar{c_r}, \ldots, c_{m+n}\})$
7: $\quad$ **if** $random(1) < \frac{E_i}{E_j}$ **then**
8: $\quad\quad c^{(t+1)} \leftarrow \{c_1, \ldots, \bar{c_r}, \ldots, c_{m+n}\}$
9: $\quad t \leftarrow t+1$
10: **until** $t = T$
11: **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

$$\pi^T = \pi^T \boldsymbol{P} \tag{4.14}$$

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 4.10.

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, 24] or parallel tempering [7]. In this study, Gibbs sampling will be used in its simplest version as described in this chapter.

4.4 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:

$$\Theta_i = \Theta_i + \Delta O \ldots. \tag{4.15}$$

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 **??**.

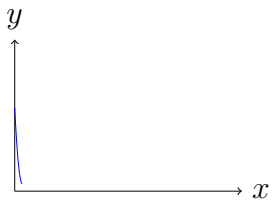


Figure 4.4: Iterations of gradient descent.

The simple update rule in 4.15 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 [22].

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 **??**.

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)

4.5 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

$$\Psi_{\mathcal{W}}(\mathcal{B}) = \frac{e^{\sum_{v_i} b_i v_i}}{Z(\mathcal{W})} \prod_{i=1}^{m} (1 + e^{b_i + \sum_{i=1}^{n} W_{ij} v_i}) \tag{4.16}$$

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.

4.5.1 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

$$\begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\theta} \end{pmatrix}. \tag{4.17}$$

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:

$$a\prime_j = a_j + \delta_{jl} i\theta \tag{4.18}$$

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:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{e}^{i\theta} \end{pmatrix} \tag{4.19}$$

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

$$\langle \mathcal{B} \mid CZ(\theta) \mid \Psi_W \rangle = \mathrm{e}^{i\theta B_l B_m} \Psi_W(Z_1 \dots Z_N). \tag{4.20}$$

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:

$$\langle \mathcal{B} \mid CZ(\theta) \mid \Psi_W \rangle = \mathrm{e}^{\Delta a_l B_l + \Delta a_m Z_m} \sum_{h_c} \mathrm{e}^{W_{lc} B_l h_c + W_{mc} B_m h_c} \tag{4.21}$$

$$= \mathrm{e}^{\Delta a_l B_l + \Delta a_m B_m} \times (1 + \mathrm{e}^{W_{lc} B_l + W_{mc}} B_m) \Psi_W(\mathcal{B}), \tag{4.22}$$

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:

$$\mathrm{e}^{\Delta a_l B_l + \Delta a_m B_m} (1 + \mathrm{e}^{W_{lc} B_l + W_{mc} B_m}) = C \times \mathrm{e}^{i\theta B_l B_m} \tag{4.23}$$

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:

$$W_{lc} = -2\mathrm{A}(\theta) \tag{4.24}$$

$$W_{mc} = 2\mathrm{A}(\theta) \tag{4.25}$$

$$\Delta a_l = i\frac{\theta}{2} + \mathrm{A}(\theta) \tag{4.26}$$

$$\Delta a_m = i\frac{\theta}{2} - \mathrm{A}(\theta) \tag{4.27}$$

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

$$(1 - B_l)W_{lk} + b_k = B_l W_{lk}\prime + b_k\prime \tag{4.28}$$

and

$$(1 - B_l)a_l = B_l a_l\prime + C \tag{4.29}$$

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

$$W_{lk}\prime = -W_{lk} \tag{4.30}$$

$$b_k\prime = b_k + W_{lk} \tag{4.31}$$

$$a_l\prime = -a_l \tag{4.32}$$

$$C = a_l \tag{4.33}$$

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:

$$W_{lk}\prime = -W_{lk} \tag{4.34}$$

$$b_k\prime = b_k + W_{lk} \tag{4.35}$$

$$a_l\prime = -a_l + i\pi \tag{4.36}$$

$$C = a_l + \frac{i\pi}{2} \tag{4.37}$$

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

$$a_l\prime = a_l + i\pi. \tag{4.38}$$

4.5.2 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

$$G = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{4.39}$$

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

$$||\Phi(\mathcal{B}_{\mathcal{B}_l=0})\rangle|^2 = |a \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=0})\rangle + c \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=1})\rangle|^2 \tag{4.40}$$

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

$$||\Phi(\mathcal{B}_{\mathcal{B}_l=1})\rangle|^2 = |b \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=0})\rangle + d \cdot |\Psi(\mathcal{B}_{\mathcal{B}_l=1})\rangle|^2 \tag{4.41}$$

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

$$L(\mathcal{W}) = -\log O(\mathcal{W}) \tag{4.42}$$

$$= -\log \sqrt{\frac{|\langle \Psi_{\mathcal{W}} \mid \Phi \rangle|^2}{\langle \Psi_{\mathcal{W}} \mid \Psi_{\mathcal{W}} \rangle \langle \Phi \mid \Phi \rangle}} \tag{4.43}$$

$$= -\log \sqrt{\left\langle \frac{\Phi(\mathcal{B})}{\Psi_{\mathcal{W}}(\mathcal{B})} \right\rangle_{\Psi} \left\langle \frac{\Psi_{\mathcal{W}}(\mathcal{B})}{\Phi(\mathcal{B})} \right\rangle_{\Phi}^{*}} \tag{4.44}$$

with

$$\langle F(\mathcal{B}) \rangle_A := \frac{\sum_{\mathcal{B}} F(\mathcal{B}) |A(\mathcal{B})|^2}{\sum_{\mathcal{B}} |A(\mathcal{B})|^2} \tag{4.45}$$

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

$$\partial_{p_k} L(\mathcal{W}) = \langle \mathcal{O}_k^*(\mathcal{B}) \rangle_{\Psi} - \frac{\langle \frac{\Phi(\mathcal{B})}{\Psi(\mathcal{B})} \mathcal{O}_k^*(\mathcal{B}) \rangle_{\Psi}}{\langle \frac{\Phi(\mathcal{B})}{\Psi(\mathcal{B})} \rangle_{\Psi}} \tag{4.46}$$

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:

$$\partial_{a_i} \log \Psi_{\mathcal{W}}(\mathcal{B}) = v_i \tag{4.47}$$

$$\partial_{b_i} \log \Psi_{\mathcal{W}}(\mathcal{B}) = h_i \tag{4.48}$$

$$\partial_{W_{ij}} \log \Psi_{\mathcal{W}}(\mathcal{B}) = v_i h_j \tag{4.49}$$

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] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

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

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

[24] 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.