Quantum Boltzmann Machines and Auto-encoders

Henrik Modahl Breitenstein

January 15, 2024

Abstract

In this paper we

Contents

1 Introduction **3**

I Theory **4**

2 Machine Learning **5**
2.1 Basic Principles 5
2.1.1 Nodes and Layers 5
2.1.2 Activation function 7
2.1.3 Training a neural network 8
2.2 Boltzmann Machine 11
2.2.1 Structure and training 11
2.2.2 Restricted Boltzmann machine 14
2.3 Monte Carlo Methods 15
2.3.1 Importance sampling 15
2.3.2 Metropolis-Hastings algorithm 16
2.3.3 Gibbs Sampling 16
2.4 Controls and Errors 17

3 Quantum Mechanics **18**
3.1 Basic Principles 18
3.1.1 Wavefunction and superposition 18
3.1.2 Operators and the Schrödinger equation 19
3.1.3 Global and local energy 19
3.1.4 Pauli Exlusion Principle 20
3.2 Measurement in Quantum Mechanics 20
3.2.1 Measuring a Quantum State 20
3.2.2 Phase of a Quantum State 20
3.3 Entanglement 20

4 Many-Body Methods **21**
4.1 Basic Principles 21
4.1.1 Slater Determinants 21
4.2 Second quantization 22
4.2.1 Creation and annihilation operators 22
4.2.2 Operators in second quantization 22
4.2.3 Hamiltonian in second quantization 24
4.3 Methods and Algorithms 25

4.3.1 Full Configuration Interaction Theory . . . 25
4.4 The Lipkin-Meshkow-Glick model . . . 26
4.4.1 The model system . . . 26
4.4.2 Rewriting the Hamiltonian . . . 28
4.4.3 LMG Hamiltonian represented with Pauli matrices . . . 28
4.4.4 Analytical Solution . . . 30

5 Quantum Computing **33**
5.1 Basic Principles . . . 33
5.1.1 Bra and ket notation . . . 33
5.1.2 The Qubit . . . 34
5.1.3 Multi-Qubit States . . . 35
5.1.4 Unitary Operators . . . 36
5.2 Quantum programming . . . 38
5.2.1 Quantum Circuit . . . 38
5.2.2 Measurement of Output . . . 38
5.2.3 Transformation of basis . . . 39
5.2.4 Circuit measurement to expectation value . . . 40
5.3 Entangled Qubits and Bell states . . . 40
5.3.1 Entanglement . . . 40
5.3.2 Bell States . . . 41
5.4 Noisy intermediate-scale Quantum . . . 42
5.5 Variational Quantum Eigensolver . . . 43
5.5.1 Optimizing the choice of state . . . 43

6 Quantum Machine Learning **44**
6.1 Quantum Nodes . . . 44
6.2 Variational Quantum Boltzmann Machine . . . 44
6.2.1 Variational Quantum Imeginary Time Evolution . . . 44

II Implementation **45**

7 Qiskit **46**
7.1 Initialization and Gates . . . 46
7.2 Measurement and execution . . . 48
7.3 Copy and append . . . 48

8 Restricted Boltzmann Machine **50**
8.1 Calculation of the local energy . . . 50
8.1.1 The Lipkin model . . . 50

9 Variational Quantum Boltzmann Machine **53**
9.1 Initialization . . . 53
9.2 Optimization of the VarQBM . . . 53
9.3 Quantum Machine Structure . . . 53
9.4 Quantum Machine Parameter Optimization . . . 53

Chapter 1

Introduction

Machine learning is becoming more and more prevalent

Part I

Theory

Chapter 2

Machine Learning

Machine learning is a machine adapting the way it processes some information by looking at its own output from said processing. A neural network is such a machine.

2.1 Basic Principles

2.1.1 Nodes and Layers

The fundamental building block of neural networks are nodes, which often is analogously seen as an neuron from in a brain, hence the name neural network. In a node we have the bias and the activation function and a weight for each connected node.

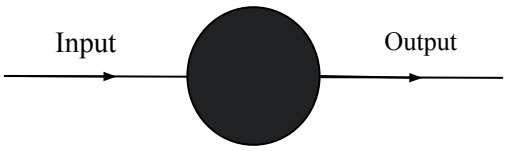


Figure 2.1: A singular node. Has an input which is affected by the weight, bias and activation function within the node.

In the figure above the node takes in a input from the left, x. The weight, w, and bias, b, create the intermediate output of the node accordingly:

$$n = wx + b\,. \tag{2.1}$$

Then n is passed through an activation function, σ and we get the output of the node

$$z = \sigma\left(n\right)\ . \tag{2.2}$$

Combining several nodes which takes the same input we get a layer. Layers are then connected where the previous layer's output is sent as input to the next via

these connections. This is the simplest network, called a feedforward network, as we pass the input through layer by layer without any inter-layer connections.

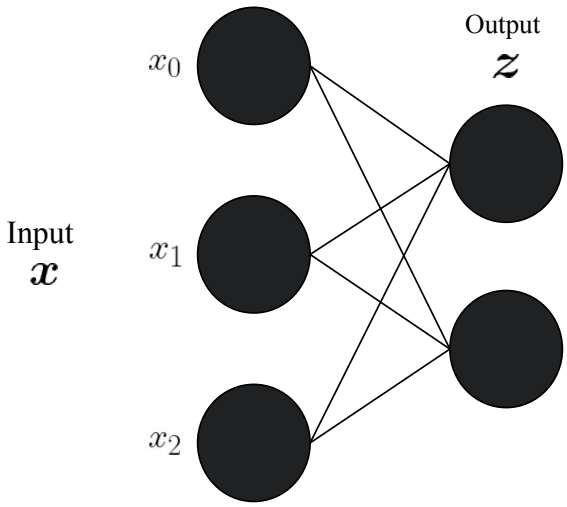


Figure 2.2: A simple feedforward neural network where we have an input layer, individual connections to the input, and a output layer. The output layer's nodes are all connected to each of the input layer's nodes.

Here the first layer is called an input layer as it has a one to one correspondence with a input data point. The input and output layer is fully connected however, where the output of each of the input layer's nodes are sent to each of the output layer's, where each connection has its own weight. Using vectors to represent the layers we have

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \tag{2.3}$$

as input, which is then inserted in the input layer's nodes as shown in 2.2. The weights now has an extra dimension as each nodes is connected to each node in the next layer, so we can represent it with a matrix:

$$W = \begin{bmatrix} w_{00} & w_{10} & w_{20} \\ w_{01} & w_{11} & w_{21} \end{bmatrix} \tag{2.4}$$

where the weight w_{kj} is the weight of the connection between node k of the input layer and node j of the output layer. Feeding the input forward we first get the intermediate output

$$\boldsymbol{n} = W\boldsymbol{x} + \boldsymbol{b} \tag{2.5}$$

where we use simple matrix multiplication. The output layer often has a unique activation function to make the output more applicable to a certain context. For example, in classification problems the softmax function

$$\sigma_s(\boldsymbol{z}')_i = \frac{e^{z'_i}}{\sum_{j=1}^{N} e^{z'_j}} \ , \tag{2.6}$$

which normalizes the output to a probability distribution, is often used. Using the softmax activation function, we have the final output

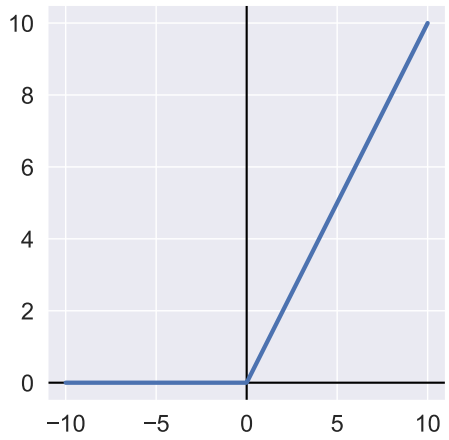
$$\boldsymbol{z} = \sigma_s \left(\boldsymbol{n} \right) \ . \tag{2.7}$$

The network model is easily expandable to more layers as one just sends the output of a layer into the next by following equation 2.5.

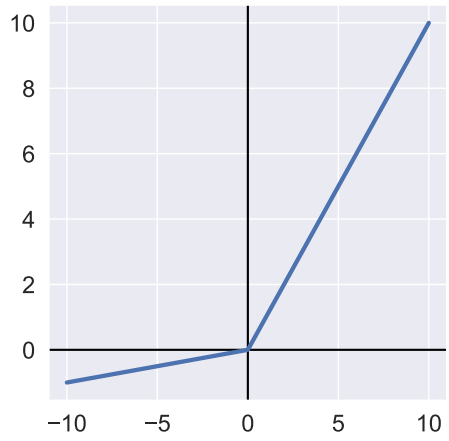
2.1.2 Activation function

The term activation function is inspired from the way biological neurons 'activate' after a certain signal threshold is met. The activation function chosen for a given network can have large impact on the finished trained model. Some common activation functions are

ReLU
$max\left(0, x\right)$



The Rectified Linear Unit activation function is one of the most basic activation functions. Since it does not have an upper bound, the activation function can have problems with values becoming too large. ReLU is however one of the most popular activation function because it has an derivative easy to calculate and avoids the vanishing gradient problem during learning.

Leaky ReLU $max\,(ax, x)\ \ a \in \langle 0, 1\rangle$	

Leaky ReLU adds the possibility for negative values to impact the output of the node.

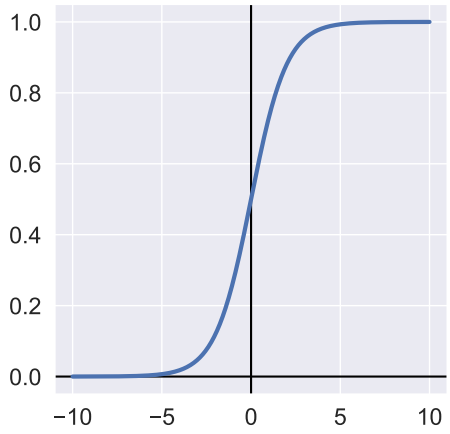
Sigmoid $\sigma\,(x) = \frac{1}{1+e^{-x}}$	

Table 2.1

The Sigmoid activation function has both a upper and lower bound, preventing the values from escalating out of control, but has an more computationally expensive derivative and can struggle with the vanishing gradient problem during learning.

2.1.3 Training a neural network

Training a neural network is done by changing the weights and biases such that the output comes closer to a desired goal. This goal is to minimize a cost function C, which can be any function. As an example, an common cost function for regression is the squared error

$$C = \sum_k (t_k - z_k)^2 \tag{2.8}$$

where t is a target value and z is the model output. We want to change the weights and biases in such a way that the cost function decreases. A natural way would be to use the gradient of the cost function in terms of the weights and biases, and decrease the variables by a proportional amount, such that we approach the minima of the cost function. Finding the gradient of the cost function in terms of the weights and biases is done by backpropagation, a method introduced by S. Linnainmaa in 1960 [1], completed by F. Rosenblatt in 1976 [2] and popularized by D. E. Rumelhart in 1982 [3]. The method is derived by application of the chain rule starting from the output and going layer by layer to get to the input.
We expand our simple input-output model 2.2 and define a subscript for the different layers and nodes as shown here:

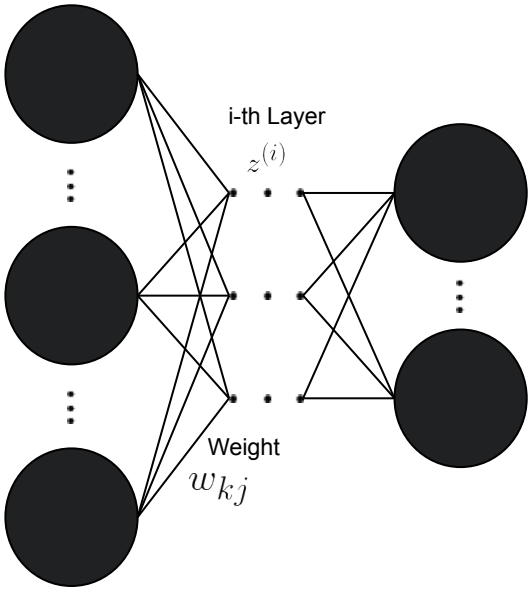


Figure 2.3: A more general network where we w_{kj} is the weight matrix

We have that:

$\boldsymbol{z}^{(i)}$ is the i-th layer vector output. (i) is the layer subscript, going from the input layer $z^{(0)}$ to the output layer $z^{(L)}$.

$\boldsymbol{n}$ is the intermediate output of a layer before it is sent through an activation function.

σ is an activation function.

$W^{(i)}$ is a weight matrix with weights w_{kj}, which is the weight from k-th node of the i-th layer to node j of layer $i+1$.

Starting from the output we want to find

$$\Delta W^{(L-1)} = -\eta \frac{\partial C}{\partial W^{(L-1)}} \, , \tag{2.9}$$

where η is a chosen learning rate. The change in a single weight

$$\Delta w_{kj}^{(L-1)} = -\eta \frac{\partial C}{\partial w_{kj}^{(L-1)}} \ , \tag{2.10}$$

where we can add the dependency on the activation function and intermediate output

$$\Delta w_{kj}^{(L-1)} = -\eta \frac{\partial C}{\partial z_k^{(L)}} \frac{\partial z_k^{(L)}}{\partial n_k^{(L)}} \frac{\partial n_k^{(L)}}{\partial w_{kj}^{(L-1)}} \tag{2.11}$$

Where we have that

$$\frac{\partial C}{\partial z_k^{(L)}} = -2\left(t_k - z_k^{(L)}\right) \tag{2.12}$$

and using the sigmoid activation function, 2.1, we have

$$\frac{\partial z_k^{(L)}}{\partial n_k^{(L)}} = \frac{\partial\left(\frac{1}{1=e^{-n_k}}\right)}{\partial n_k} = \frac{e^{-n_k}}{\left(1+e^{-n_k}\right)^2} = n_k(1-n_k) \tag{2.13}$$

and then lastly the intermediate output

$$\frac{\partial n_k^{(L)}}{\partial w_{kj}^{(L-1)}} = \frac{\partial\left(w_{kj}^{(L-1)} z_j^{(L-1)} + b_j^{(L-1)}\right)}{\partial w_{kj}^{(L-1)}} = z_j^{(L-1)} \tag{2.14}$$

So we can write the change of weight as

$$\Delta w_{kj}^{(L-1)} = \eta\left(t_k - z_k^{(L)}\right) z_k^{(L)}\left(1 - z_k^{(L)}\right) z_j^{(L-1)} \ . \tag{2.15}$$

We simplify by defining

$$\delta_k^{(L)} = \left(t_k - z_k^{(L)}\right) z_k^{(L)}\left(1 - z_k^{(L)}\right) \tag{2.16}$$

and get

$$\Delta w_{kj}^{(L-1)} = \eta \delta_k^{(L)} z_j^{(L-1)} \ , \tag{2.17}$$

where we have shortened the -2 factor into η as well. For the next layer, $(L-2)$, we have that a individual node is connected to each of the previous layer's nodes. We then have the change

$$\Delta w_{jm}^{(L-2)} = \eta\left[\sum_k \frac{\partial C}{\partial z_k^{(L)}} \frac{\partial z_k^{(L)}}{\partial n_k^{(L)}} \frac{\partial n_k^{(L)}}{\partial z_j^{(L-1)}}\right] \frac{\partial z_j^{(L-1)}}{\partial n_j^{(L-1)}} \frac{\partial n_j^{(L-1)}}{\partial w_{jm}^{(L-2)}} \ , \tag{2.18}$$

where we have calculated the expression within the sum already

$$\frac{\partial C}{\partial z_k^{(L)}} \frac{\partial z_k^{(L)}}{\partial n_k^{(L)}} \frac{\partial n_k^{(L)}}{\partial z_j^{(L-1)}} = \left(t_k - z_k^{(L)}\right) z_k^{(L)}\left(1 - z_k^{(L)}\right) w_{kj}^{(L-1)} = \delta_k^{(L)} w_{kj}^{(L-1)} \ . \tag{2.19}$$

The expression outside the sum is calculated as in 2.13 and 2.14

$$\frac{\partial z_j^{(L-1)}}{\partial n_j^{(L-1)}} \frac{\partial n_j^{(L-1)}}{\partial w_{jm}^{(L-2)}} = z_j^{(L-1)} \left(1 - z_j^{(L-1)}\right) z_m^{(L-2)} \ . \tag{2.20}$$

We can then write the change in weight as

$$\Delta w_{jm}^{(L-2)} = \eta \left[\sum_k \delta_k^{(L)} w_{kj}^{(L-1)}\right] z_j^{(L-1)} \left(1 - z_j^{(L-1)}\right) z_m^{(L-2)} \ , \tag{2.21}$$

which we can further simplify by defining

$$\delta_j^{(L-1)} = \left[\sum_k \delta_k^{(L)} w_{kj}^{(L-1)}\right] z_j^{(L-1)} \left(1 - z_j^{(L-1)}\right) \tag{2.22}$$

and we get

$$\Delta w_{jm}^{(L-2)} = \eta \delta_j^{(L-1)} z_m^{(L-2)} \ . \tag{2.23}$$

This process can then be repeated easily with the fact that

$$\delta_k^{i-1} = \left[\sum_k \delta_k^{(i)} w_{kj}^{(i-1)}\right] z_j^{(i-1)} \left(1 - z_j^{(i-1)}\right) \tag{2.24}$$

for any layers further in.

2.2 Boltzmann Machine

In this thesis we are going to use a slightly different type of neural network, called a Boltzmann machine. First major contribution to Boltzmann machines comes from G. E Hinton and T.J. Sejnowski in 1983 [4]. The neural net is a generative model which learns by matching the probability distribution of its inputs.

2.2.1 Structure and training

A Boltzmann machine has interconnected nodes within a layer.

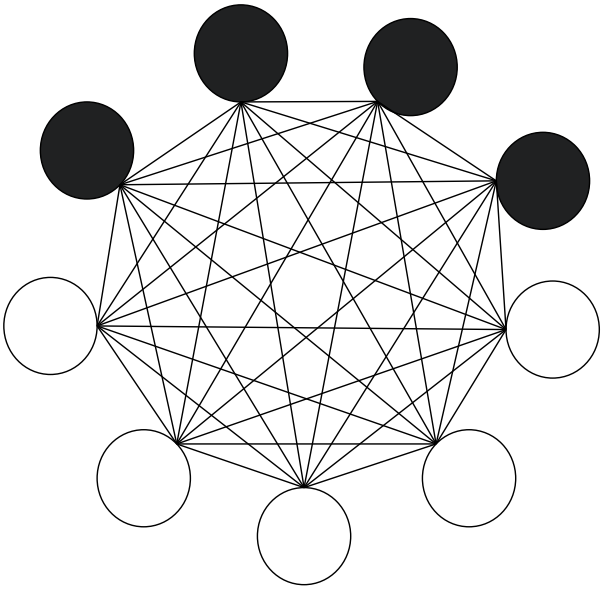


Figure 2.4: A unrestricted Boltzmann machine where every node is connected. Here the black nodes are the visible layer while the white nodes constitute the hidden layer.

The nodes are binary, being able to take the value 0 or 1, have a weights w_{ij} for connection strength between node v_i and h_j and biases a_i for the visible layer and b_j for the hidden layer. For a system of N_h hidden neurons and N_v visible neurons we have the probability distribution of nodes taking the value 1 defined as

$$P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})} \,, \tag{2.25}$$

where the energy of the model is given by

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i}^{N_v} a_i v_i - \sum_{j}^{N_h} b_j h_j - \sum_{i}^{N_v} \sum_{j}^{N_h} v_i w_{ij} h_j \tag{2.26}$$

and the normalization factor

$$Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \,, \tag{2.27}$$

where we sum over all possible states of the model, which increases exponentially as $2^{(N_v+N_h)}$. The marginal distribution over the visual layer can be written as

$$P(\boldsymbol{v}) = \sum_{\boldsymbol{h}} \frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{2.28}$$

To train a Boltzmann machine we need to have a cost function that compares the predicted distribution of the model and the actual distribution of the data set. As such we will use the Kullback-Leibler divergence as a cost function:

$$KL(\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}) = \sum_{(\boldsymbol{v},\boldsymbol{h})} R(\boldsymbol{v}) \log \frac{R(\boldsymbol{v})}{P(\boldsymbol{v})} \ , \tag{2.29}$$

where $R(\boldsymbol{v})$ is the distribution we want to approximate and $P(\boldsymbol{v})$ is the distribution of the neural network model. Following the derivations of A.L. Yuille [5] we have that

$$\frac{\partial KL(\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})}{\partial w_{ij}} = -\sum_{(\boldsymbol{v},\boldsymbol{h})} \frac{R(\boldsymbol{v})}{P(\boldsymbol{v})} \frac{\partial P(\boldsymbol{v})}{\partial w_{ij}} \ , \tag{2.30}$$

where we further have

$$\frac{\partial P(\boldsymbol{v})}{\partial w_{ij}} = \frac{1}{Z} \frac{\partial}{\partial w_{ij}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial \log Z}{\partial w_{ij}} \tag{2.31}$$

which we can express as

$$\frac{\partial P(\boldsymbol{v})}{\partial w_{ij}} = -\sum_{\boldsymbol{h}} v_i h_j P(\boldsymbol{v}, \boldsymbol{h}) + \sum_{\boldsymbol{h}} \left[P(\boldsymbol{v}, \boldsymbol{h}) \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j P(\boldsymbol{v}, \boldsymbol{h}) \right] \ . \tag{2.32}$$

Then

$$\frac{\partial P(\boldsymbol{v})}{\partial w_{ij}} = -\sum_{\boldsymbol{h}} v_i h_j P(\boldsymbol{v}, \boldsymbol{h}) + P(\boldsymbol{v}, \boldsymbol{h}) \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j P(\boldsymbol{v}, \boldsymbol{h}) \ . \tag{2.33}$$

Using the result of equation 2.33 in equation 2.30 we get that

$$\frac{\partial KL(\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})}{\partial w_{ij}} = \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j \frac{P(\boldsymbol{v}, \boldsymbol{h})}{P(\boldsymbol{v})} R(\boldsymbol{v}) - \left[\sum_{\boldsymbol{v},\boldsymbol{h}} R(\boldsymbol{v}) \right] \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j P(\boldsymbol{v}, \boldsymbol{h}) \ , \tag{2.34}$$

which we can simplify

$$\frac{\partial KL(\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})}{\partial w_{ij}} = \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j P(\boldsymbol{h}|\boldsymbol{v}) R(\boldsymbol{v}) - \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j P(\boldsymbol{v}, \boldsymbol{h}) \ , \tag{2.35}$$

where we require that

$$\frac{\partial \log Z}{\partial w_{ij}} = \sum_{\boldsymbol{v},\boldsymbol{h}} v_i h_j P(\boldsymbol{v}, \boldsymbol{h}) \ . \tag{2.36}$$

We then define the expectation, or correlation, values

$$\langle v_i h_j \rangle_{\text{data}} = P(\boldsymbol{h}|\boldsymbol{v}) R(\boldsymbol{v}) \tag{2.37}$$

and

$$\langle v_i h_j \rangle_{\text{model}} = P(\boldsymbol{v}, \boldsymbol{h}) \ . \tag{2.38}$$

This gives us the update rule

$$\Delta w_{ij} = -\eta \left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \right) \ . \tag{2.39}$$

2.2.2 Restricted Boltzmann machine

Estimating $\langle v_i h_j \rangle_{\text{data}}$ and $\langle v_i h_j \rangle$ is done by Gibbs sampling, which is explained in a later chapter, but can be inefficient and take a long time to converge for complex models. Removing the weights between nodes within the same layer we can alleviate much of the computational cost of training. This type of Boltzmann machine is called a restricted Boltzmann machine, or RBM:

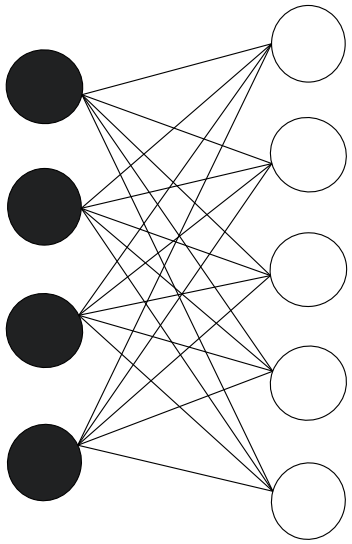


Figure 2.5: A restricted Boltzmann machine where there are no connections between nodes within the same layer. The white nodes are hidden while the black ones are the visible nodes.

Making the nodes independent of nodes in the same layer means we can write the conditional distributions as

$$P(\boldsymbol{v}|\boldsymbol{h}) = \prod_{i \in \boldsymbol{v}} P(v_i|\boldsymbol{h}) \, , \tag{2.40}$$

and

$$P(\boldsymbol{h}|\boldsymbol{v}) = \prod_{j \in \boldsymbol{h}} P(h_j|\boldsymbol{v}) \, . \tag{2.41}$$

In a RBM we first have a forward pass where we insert the input data into the visual layer, then we sample the hidden layer by the distribution:

$$p(h_j^{(0)} = 1|\boldsymbol{z}_v^{(0)}) = \sigma\left(\boldsymbol{z}_v^{(0)} \otimes \boldsymbol{W} + \boldsymbol{b}\right) \, , \tag{2.42}$$

where our activation function σ is the Sigmoid function 2.1. The index (0) indicate that it is the first pass-through the neural network. As the nodes are binary we then take a sample from $p(h_j^{(0)} = 1|\boldsymbol{z}_v^{(0)})$ as a Bernoulli distribution, which means each $z_{h,j}^{(0)}$ takes the value 1 with probability $h_j^{(0)}$. After the forward pass we have a backward pass where we sample from the hidden layer

$$p(v_j^{(1)} = 1|\boldsymbol{z}_h^{(0)}) = \sigma\left(\boldsymbol{z}_h^{(0)} \otimes \boldsymbol{W} + \boldsymbol{a}\right) \ , \tag{2.43}$$

where we then convert it to binary values as well. For a continuous valued output it is optional to let the last visual output to remain as a probability distribution. Estimating $\langle v_i h_j \rangle_{\text{data}}$ is done by sampling from $P(\boldsymbol{h}|\boldsymbol{v})$

$$\langle \boldsymbol{v}\boldsymbol{h} \rangle_{\text{data}} = \boldsymbol{z}_v^{(0)} \otimes p(\boldsymbol{h}^{(0)}|\boldsymbol{z}_v^{(0)}) \tag{2.44}$$

,

while estimating $\langle v_i h_j \rangle_{\text{model}}$ requires that we let the model sufficiently affect the output. To do this we do Gibbs sampling through k iterations of the forward and backward passes. Then we have

$$\langle \boldsymbol{v}\boldsymbol{h} \rangle_{\text{model}} = \boldsymbol{z}_v^{(k)} \otimes p(\boldsymbol{h}^{(k)}|\boldsymbol{z}_v^{(k)}) \tag{2.45}$$

.

And from 2.39 the change in weight becomes

$$\Delta \boldsymbol{W} = \boldsymbol{z}_v^{(0)} \otimes p(\boldsymbol{h}^{(0)}|\boldsymbol{z}_v^{(0)}) - \boldsymbol{z}_v^{(k)} \otimes p(\boldsymbol{h}^{(k)}|\boldsymbol{z}_v^{(k)}) \ . \tag{2.46}$$

2.3 Monte Carlo Methods

Monte Carlo methods is a way to gain insight into a probability distribution by using random samples from said distribution. As an example problem one is to calculate the integral

$$I(h, P) = \int h(x)P(x) \ dx,$$

where $h(x)$ is an arbitrary function and $P(x)$ is a probability distribution. Then by taking random samples from $P(x) \rightarrow x_s$, one could estimate the integral by

$$I(h, P) \approx \frac{1}{N} \sum_{s=1}^{N} h(x_s).$$

Which is simple enough when one can sample from the target distribution, here $P(x)$, something that is not always the case.

2.3.1 Importance sampling

When the distribution $P(x)$ is unknown we can instead draw our sample from a proposed distribution $Q(x)$ and then use importance weights to correct it. Then continuing with the example above we have

$$I(h, P) = \int \frac{h(x)P(x)}{Q(x)} Q(x) \ dx.$$

And our approximate becomes

$$I(h, P) \approx \frac{1}{N} \sum_{s=1}^{N} \frac{h(x_s)P(x_s)}{Q(x_s)}.$$

This requires a that $Q(x)$ is somewhat close to that of $P(x)$, which is not necessarily easy to construct. It is therefore better to use a adaptive proposal distribution instead.

2.3.2 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a Markov chain Monte Carlo method and introduces feature adaptive proposal distributions. For a desired distribution $P(x)$, a Markov chain describes the probability of a series of events where we have the transition probability $P(x'|x)$ of transitioning from state x to x'. Adapting the our proposed distribution $Q(x)$ by using Markov chains we move towards a stationary distribution where we then are at a equilibrium:

$$P(x'|x)P(x) = P(x|x')P(x') \ , \tag{2.47}$$

where the transition from x to x' is equally likely both ways. We can rewrite this as

$$\frac{P(x'|x)}{P(x|x')} = \frac{P(x')}{P(x)} \ . \tag{2.48}$$

The Metropolis-Hastings method is to set up a proposal $Q(x'|x)$ and then use a acceptance distribution $A(x', x)$ to either accept or reject samples from the proposed distribution. We can then rewrite the transition probability as

$$P(x'|x) = Q(x'|x)A(x', x) \ . \tag{2.49}$$

And then equilibrium equation 2.48 can then be written as

$$\frac{A(x', x)}{A(x|x')} = \frac{P(x')Q(x|x')}{P(x)Q(x'|x)} \ , \tag{2.50}$$

where we define the acceptance ratio

$$A(x', x) = \min\left(1, \frac{P(x')Q(x|x')}{P(x)Q(x'|x)}\right) \ . \tag{2.51}$$

For each iteration i we have a the state x' taken at random from $Q(x'|x_i)$ which then is either accepted by $x_{i+1} = x'$ or rejected by $x_{i+1} = x_i$. The distribution of $\{x_i\}$ will then approach our desired distribution $P(x)$ after a sufficient number of iterations.

2.3.3 Gibbs Sampling

Gibbs sampling is a special use case where we accept every suggested state. It is then necessary to make the proposed distribution as close to the actual distribution as possible. How we accomplish this is covered in the implementation section, see **??**.

2.4 Controls and Errors

Seeing how accurate a model is can be use full both during training, to see how well it optimizes, and for a trained model. For this we can simply look at the desired output of the model, a target solution $\boldsymbol{z_t}$, and compare to the output of the model $\boldsymbol{z_m}$ with the mean squared error for example:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{z}_{t,i} - \boldsymbol{z}_{m,i})^2 \ . \tag{2.52}$$

Chapter 3

Quantum Mechanics

Quantum mechanics is the physics and mathematics describing how things behave at the smallest of scales. Certainty becomes no more and everything devolves into probabilistic happenstance. The systems we plan to solve are of quantum mechanical nature, so we will start with a small introduction to the field.

3.1 Basic Principles

3.1.1 Wavefunction and superposition

A quantum state is described with the help of a wavefunction, which for one dimension we write as

$$\psi(x) : \mathbb{R} \to \mathbb{C} \, .$$

In and of itself the wavefunction does not have a good physical interpretation, but the squared absolute value becomes a probability distribution, such that

$$P(x) = |\psi(x)|^2 \, , \tag{3.1}$$

meaning that $|\psi(x)|^2$ is the probability of finding the particle at position x. The wavefunction therefor needs to be normalized, such that

$$\int_{-\inf}^{\inf} \mathrm{d}x \psi^* \psi = 1 \, . \tag{3.2}$$

It then also means the particle is in a undetermined position before measurement. The quantum state is a combination of all the possible positions it can be in, weighted by $\psi(x)$, in intervals for a continuum of eigenstates $|\phi\rangle$

$$|\psi\rangle = \int \mathrm{d}x \psi(x) \, |\phi\rangle \ . \tag{3.3}$$

And for a discrete set of eigenstates

$$|\psi\rangle = \sum_i \psi(x_i) \, |\phi\rangle \ . \tag{3.4}$$

Such a state $|\psi\rangle$ is called a superposition.

3.1.2 Operators and the Schrödinger equation

Affecting the wavefunction is done mathematically through operators. An operator maps a state to another, transforming the wavefunction. For a generalized operator $\hat{O}$ and the state $|\psi\rangle$ we have

$$\hat{O}\,|\psi\rangle = |\psi'\rangle \ , \tag{3.5}$$

where $|\psi'\rangle$ is a new state. When an operator acts on a eigenstate:

$$\hat{O}\,|\phi\rangle = \varepsilon\,|\phi\rangle \ , \tag{3.6}$$

the eigenvalue ε is a quantity of what $\hat{O}$ represents. For this quantity to be something physically measurable, the operator needs to be hermitian

$$\hat{O} = \hat{O}^{\dagger} \ , \tag{3.7}$$

such that $\varepsilon \in \mathbb{R}$. An important observable operator is the hamiltonian, representing the energy of the system. The hamiltonian maps a state to its energy distribution, indicating its time evolution which is dictated by the Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = \hat{H}|\Psi(t)\rangle \ , \tag{3.8}$$

where t is time. Eigenstates of the hamiltonian are the stable energy states of the system

$$H\,|\psi_n\rangle = E_n\,|\psi_n\rangle \ , \tag{3.9}$$

where E_n is the energy of the state.

3.1.3 Global and local energy

The global energy is the energy of the whole system, the possible are values of energy that can be measured. This is the energy eigenvalues of the hamiltonian:

$$E_n = \frac{\langle\psi_n|\,H\,|\psi_n\rangle}{\langle\psi_n|\psi_n\rangle} \ . \tag{3.10}$$

The local energy is, however, the energy associated with any particular basis state:

$$E_{\mathrm{loc}}(\phi) = \frac{\langle\phi|\,H\,|\psi\rangle}{\langle\phi|\psi\rangle} \ , \tag{3.11}$$

where then ϕ is the basis state, a specific point in the space of the system, and ψ is the state of the system.

3.1.4 Pauli Exlusion Principle

The Pauli Exclusion Principle says that, within a quantum system, identical fermions are limited to one per quantum state. This is expressed mathematically with a anti-symmetric wavefunction:

$$\Psi(\dots, x_i, \dots, x_j, \dots) = -\Psi(\dots, x_j, \dots, x_i, \dots) \,, \tag{3.12}$$

where x_i and x_j is two particles being switched.

3.2 Measurement in Quantum Mechanics

3.2.1 Measuring a Quantum State

3.2.2 Phase of a Quantum State

3.3 Entanglement

Chapter 4

Many-Body Methods

4.1 Basic Principles

4.1.1 Slater Determinants

Given a system with 2 fermions. The first particle particle has quantum numbers i and spin value α, while the second particle has quantum numbers k and spin value β. Since both particles are indistinguishable form each other, we have no way of knowing which particle is where. As a result the wavefunction needs to reflect this by being indifferent to particle permutation. A naive product wavefunction of the two fermions would look like

$$|\psi(x_1, x_2)\rangle = \varphi_{i\alpha}(x_1)\varphi_{k\beta}(x_2) \, . \tag{4.1}$$

If we try to switch the positions of x_1 and x_2 we get that

$$|\psi(x_2, x_1)\rangle = \varphi_{i\alpha}(x_2)\varphi_{k\beta}(x_1) \, , \tag{4.2}$$

which fails the anti-symmetric requirement for fermionic wavefunctions, which makes them distinguishable from each other. Instead we would want a wave function accounts for both scenarios in the first place

$$|\psi(\mathbf{x}_1, \mathbf{x}_2)\rangle = \frac{1}{\sqrt{2}} \left(\varphi_{i\alpha}(x_1)\varphi_{k\beta}(x_2) - \varphi_{i\alpha}(x_2)\varphi_{k\beta}(x_1)\right) \, , \tag{4.3}$$

where the wavefunction is a normalized combination of the two product wavefunctions. Here we have the antisymmetry

$$|\psi(\mathbf{x}_1, \mathbf{x}_2)\rangle = - |\psi(x_2, x_1)\rangle \, . \tag{4.4}$$

Which can be further generalized for a system with N fermions.

$$\psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \varphi_1(\mathbf{x}_1) & \varphi_2(\mathbf{x}_1) & \cdots & \varphi_N(\mathbf{x}_1) \\ \varphi_1(\mathbf{x}_2) & \varphi_2(\mathbf{x}_2) & \cdots & \varphi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{x}_N) & \varphi_2(\mathbf{x}_N) & \cdots & \varphi_N(\mathbf{x}_N) \end{vmatrix} \, . \tag{4.5}$$

where it gets its name from.

4.2 Second quantization

4.2.1 Creation and annihilation operators

Starting with the vacuum state $|0\rangle$. The creation operator creates a single-particle state α_i:

$$\hat{a}^\dagger_{\alpha_1} |0\rangle = |\alpha_1\rangle \ , \tag{4.6}$$

which can be extended for more particles

$$\hat{a}^\dagger_{\alpha_1} \hat{a}^\dagger_{\alpha_2} \dots \hat{a}^\dagger_{\alpha_n} |0\rangle = |\alpha_1 \alpha_2 \dots \alpha_n\rangle \ . \tag{4.7}$$

The annihilation operator destroys the particle

$$\hat{a}_{\alpha_1} |\alpha_1\rangle = |0\rangle \ . \tag{4.8}$$

They are hermitian conjugate, which means

$$\hat{a}_{\alpha_i} = \left(\hat{a}^\dagger_{\alpha_i}\right)^\dagger \ . \tag{4.9}$$

Furthermore, the Pauli principle says we cannot have two particles in the same state, which gives us

$$\hat{a}^\dagger_{\alpha_i} \hat{a}^\dagger_{\alpha_i} = 0 \ . \tag{4.10}$$

Similarly we cannot annihilate a particle that does not exist in the first place

$$\alpha \neq \{\alpha_i\}$$

$$\hat{a}_\alpha |\alpha_1 \alpha_2 \dots \alpha_n\rangle = 0 \ . \tag{4.11}$$

For fermions the state wavefunction is anti symmetric, therefor switching position of two particles yields a factor of -1, and we get that

$$\hat{a}^\dagger_{\alpha_i} \hat{a}^\dagger_{\alpha_k} = -\hat{a}^\dagger_{\alpha_k} \hat{a}^\dagger_{\alpha_i} \ .$$

From this we have the commutation relations

$$\left\{\hat{a}^\dagger_\alpha \hat{a}^\dagger_\beta\right\} = 0 \tag{4.12}$$

$$\left\{\hat{a}_\alpha \hat{a}_\beta\right\} = 0 \tag{4.13}$$

$$\left\{\hat{a}^\dagger_\alpha \hat{a}_\beta\right\} = \delta_{\alpha\beta} \tag{4.14}$$

4.2.2 Operators in second quantization

`https://github.com/ManyBodyPhysics/FYS4480/blob/master/doc/pub/week36/ipynb/week36.ipynb`
For a one-body operator in coordinate space we have

$$\hat{H}_0 = \sum_i \hat{h}_0(x_i) \ , \tag{4.15}$$

and using the anti-symmetric Slater determinant from 4.5, which we can write as:

$$\Phi(x_1, x_2, \ldots, x_n, \alpha_1, \alpha_2, \ldots, \alpha_n) = \frac{1}{\sqrt{n!}} \sum_p (-1)^p \hat{P} \psi_{\alpha_1}(x_1) \psi_{\alpha_2}(x_2) \ldots \psi_{\alpha_n}(x_n) \ , \tag{4.16}$$

we can define

$$\hat{h}_0(x_i)\psi_{\alpha_i}(x_i) = \sum_{\alpha'_k} \psi_{\alpha'_k}(x_i) \langle \alpha'_k | \hat{h}_0 | \alpha_k \rangle \ . \tag{4.17}$$

So for each one-particle function in the Slater determinant we gain a contribution to $\hat{H}_0 |\Phi\rangle$. Our Slater determinant can be written in second quantization as simply

$$|\Phi\rangle = |\alpha_1, \alpha_2, \ldots, \alpha_n\rangle \ . \tag{4.18}$$

With this we have

$$\begin{aligned} \hat{H}_0 |\alpha_1, \alpha_2, \ldots, \alpha_n\rangle = & \sum_{\alpha'_1} \langle \alpha'_1 | \hat{h}_0 | \alpha_1 \rangle | \alpha'_1 \alpha_2 \ldots \alpha_n \rangle \\ & + \sum_{\alpha'_2} \langle \alpha'_2 | \hat{h}_0 | \alpha_2 \rangle | \alpha_1 \alpha'_2 \ldots \alpha_n \rangle \\ & + \ldots \qquad (4.19) \\ & + \sum_{\alpha'_n} \langle \alpha'_n | \hat{h}_0 | \alpha_n \rangle | \alpha_1 \alpha_2 \ldots \alpha'_n \rangle \ , \qquad (4.20) \end{aligned}$$

where we go over each possible permutation of each particle. Furthermore, if we use the fact that we can write

$$|\alpha_1 \alpha_2 \ldots \alpha'_k \ldots \alpha_n\rangle = a^\dagger_{\alpha'_k} a_{\alpha_k} |\alpha_1 \alpha_2 \ldots \alpha_k \ldots \alpha_n\rangle \ , \tag{4.21}$$

we can then shorten the expression to

$$\hat{H}_0 = \sum_{\alpha\beta} \langle \alpha | \hat{h}_0 | \beta \rangle a^\dagger_\alpha a_\beta \ . \tag{4.22}$$

As a generalized one-body operator, that preserves the number of particles, in second quantization. A two-body operator in coordinate space can be written as

$$\hat{H}_I = \sum_{i<j} V(x_i, x_j) \ , \tag{4.23}$$

where V is some interaction force between two particles. Using our Slater determinant again we have that

$$V(x_i, x_j)\psi_{\alpha_k}(x_i)\psi_{\alpha_l}(x_j) = \sum_{\alpha'_k \alpha'_l} \psi'_{\alpha_k}(x_i)\psi'_{\alpha_l}(x_j) \langle \alpha'_k \alpha'_l | \hat{v} | \alpha_k \alpha_l \rangle \tag{4.24}$$

Once again summing over all possible permutations of each combination of two-particle pairs, we get

$$\begin{aligned} H_I|\alpha_1\alpha_2\dots\alpha_n\rangle = &\sum_{\alpha_1',\alpha_2'}\langle\alpha_1'\alpha_2'|\hat{v}|\alpha_1\alpha_2\rangle|\alpha_1'\alpha_2'\dots\alpha_n\rangle \\ &+\dots \\ &+\sum_{\alpha_1',\alpha_n'}\langle\alpha_1'\alpha_n'|\hat{v}|\alpha_1\alpha_n\rangle|\alpha_1'\alpha_2\dots\alpha_n'\rangle \\ &+\dots \\ &+\sum_{\alpha_2',\alpha_n'}\langle\alpha_2'\alpha_n'|\hat{v}|\alpha_2\alpha_n\rangle|\alpha_1\alpha_2'\dots\alpha_n'\rangle \\ &+\dots \\ &+\sum_{\alpha_{n-1}',\alpha_n'}\langle\alpha_{n-1}'\alpha_n'|\hat{v}|\alpha_{n-1}\alpha_n\rangle|\alpha_1\alpha_{n-1}'\dots\alpha_n'\rangle\ . \end{aligned} \tag{4.25}$$

Then, using the fact that

$$a^\dagger_{\alpha_k'}a^\dagger_{\alpha_l'}a_{\alpha_l}a_{\alpha_k}|\alpha_1\alpha_2\dots\alpha_k\dots\alpha_l\dots\alpha_n\rangle == |\alpha_1\alpha_2\dots\alpha_k'\dots\alpha_l'\dots\alpha_n\rangle\ , \tag{4.26}$$

we end up with

$$\begin{aligned} H_I|\alpha_1\alpha_2\dots\alpha_n\rangle &= \sum_{\alpha_1',\alpha_2'}\langle\alpha_1'\alpha_2'|\hat{v}|\alpha_1\alpha_2\rangle a^\dagger_{\alpha_1'}a^\dagger_{\alpha_2'}a_{\alpha_2}a_{\alpha_1}|\alpha_1\alpha_2\dots\alpha_n\rangle \\ &= {\sum_{\alpha,\beta,\gamma,\delta}}'\langle\alpha\beta|\hat{v}|\gamma\delta\rangle a^\dagger_\alpha a^\dagger_\beta a_\delta a_\gamma|\alpha_1\alpha_2\dots\alpha_n\rangle\ , \end{aligned} \tag{4.27}$$

where $\alpha,\ \beta$ are single-particle states while $\gamma,\ \delta$ are pairs of single-particle states. We can remove this distinction with the fact that

$$\langle\alpha\beta|\hat{v}|\gamma\delta\rangle = \langle\beta\alpha|\hat{v}|\delta\gamma\rangle\ . \tag{4.28}$$

And we end up with

$$\hat{H}_I = \frac{1}{2}\sum_{\alpha\beta\gamma\delta}\langle\alpha\beta|\hat{v}|\gamma\delta\rangle a^\dagger_\alpha a^\dagger_\beta a_\delta a_\gamma\ , \tag{4.29}$$

where all indices are summed over single-particle states only.

4.2.3 Hamiltonian in second quantization

The way we calculate the local energy in practicality is connected to the hamiltonian's effect on a state. As such we will go over a concrete example to make our implementation exelanation, see **??** simpler to understand. For an example hamiltonian

$$H = \sum_{p,\sigma}\varepsilon_{p,\sigma}\hat{a}^\dagger_{p,\sigma}\hat{a}_{p,\sigma}\ , \tag{4.30}$$

where p is the number of fermions, here 2, and σ is the up or down spin. We can then apply the hamiltonian to each of the basis states.

$$H\left|00\right\rangle = \left(\varepsilon_{0,+}\hat{a}^{\dagger}_{0,+}\hat{a}_{0,+} + \varepsilon_{0,-}\hat{a}^{\dagger}_{0,-}\hat{a}_{0,-} + \varepsilon_{1,+}\hat{a}^{\dagger}_{1,+}\hat{a}_{1,+}\varepsilon_{1,-}\hat{a}^{\dagger}_{1,-}\hat{a}_{1,-}\right)\left|00\right\rangle \tag{4.31}$$

$$H\left|01\right\rangle = \left(\varepsilon_{0,+}\hat{a}^{\dagger}_{0,+}\hat{a}_{0,+} + \varepsilon_{0,-}\hat{a}^{\dagger}_{0,-}\hat{a}_{0,-} + \varepsilon_{1,+}\hat{a}^{\dagger}_{1,+}\hat{a}_{1,+}\varepsilon_{1,-}\hat{a}^{\dagger}_{1,-}\hat{a}_{1,-}\right)\left|01\right\rangle \tag{4.32}$$

$$H\left|10\right\rangle = \left(\varepsilon_{0,+}\hat{a}^{\dagger}_{0,+}\hat{a}_{0,+} + \varepsilon_{0,-}\hat{a}^{\dagger}_{0,-}\hat{a}_{0,-} + \varepsilon_{1,+}\hat{a}^{\dagger}_{1,+}\hat{a}_{1,+}\varepsilon_{1,-}\hat{a}^{\dagger}_{1,-}\hat{a}_{1,-}\right)\left|10\right\rangle \tag{4.33}$$

$$H\left|11\right\rangle = \left(\varepsilon_{0,+}\hat{a}^{\dagger}_{0,+}\hat{a}_{0,+} + \varepsilon_{0,-}\hat{a}^{\dagger}_{0,-}\hat{a}_{0,-} + \varepsilon_{1,+}\hat{a}^{\dagger}_{1,+}\hat{a}_{1,+}\varepsilon_{1,-}\hat{a}^{\dagger}_{1,-}\hat{a}_{1,-}\right)\left|11\right\rangle \ . \tag{4.34}$$

The local energy of a state $\left|s\right\rangle$ is:

$$E_{local} = \frac{\left\langle s\right| H\left|\psi\right\rangle}{\left\langle s|\psi\right\rangle} \ , \tag{4.35}$$

4.3 Methods and Algorithms

4.3.1 Full Configuration Interaction Theory

We want to show that diagonalizing the Hamiltonian matrix yields the energies of the different levels. Starting of with the time-independent Schrodinger equation

$$H\left|\psi\right\rangle = E\left|\psi\right\rangle \ . \tag{4.36}$$

Assuming we can express $\left|\psi\right\rangle$ in terms of an orthonormal basis $\{\left|n\right\rangle\}$:

$$\left|\psi\right\rangle = \sum_{n} c_n \left|n\right\rangle \ , \tag{4.37}$$

we can then insert this into the time-independent Schrodinger equation and get

$$\sum_{n} c_n H\left|n\right\rangle = E\sum_{n} c_n \left|n\right\rangle \ . \tag{4.38}$$

Since the basis $\{\left|n\right\rangle\}$ is orthonormal we that the identity matrix can be expressed as

$$I = \sum_{k} \left|k\right\rangle\left\langle k\right| \ ,$$

where $\left|k\right\rangle \in \{\left|n\right\rangle\}$. Inserting this into equation 4.38 we get that

$$\sum_{k}\sum_{n} \left\langle k\right| H\left|n\right\rangle c_n \left|k\right\rangle = E\sum_{n} c_n \left|n\right\rangle \ . \tag{4.39}$$

Multiplying from the left side by $\left|m\right\rangle \in \{\left|n\right\rangle\}$ we get

$$\sum_n \langle k| H |n\rangle c_n \langle m|k\rangle = E \sum_n c_n \langle m|n\rangle \ , \tag{4.40}$$

where

$$\langle m|k\rangle = \delta_{mk} \langle m|n\rangle \qquad\qquad = \delta_{mn} \ ,$$

such that

$$\sum_n \langle m| H |n\rangle c_n = E c_m \ . \tag{4.41}$$

$\langle m| H |n\rangle$ is the element m, n of the Hamiltonian matrix, so 4.41 can be expressed as

$$\sum_n H_{mn} c_n = E c_m \ . \tag{4.42}$$

And in matrix form this becomes

$$HC = EC \ , \tag{4.43}$$

where

$$C = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} \ .$$

Diagonalizing H will then give us the eigenvalues E as the energy of the different levels of the system.

4.4 The Lipkin-Meshkow-Glick model

`https://arxiv.org/pdf/1805.12442.pdf`
One of problem that many-body physics encounter in complex, real world, system is the fact that the many-body Shrödinger equation isn't exactly solvable. This may require either to approximat the Shrodinger equation or limit the number of particles in a system. Therefor it is more common to test many-body methods on simplified models where the exact solution is available without approximation. One of these models is the Lipkin-Meshkow-Glick model, abbreviated LMG in this thesis, first built in the 1960's [6].

4.4.1 The model system

The idea behind the LMG model is to have two energy levels separated by an energy-value ε, one just below the Fermi level and one just above. In our case we fill up the base energy level with any number of particles, which then can be excited up to the second energy level. 4.1 shows a visualized example with two particles:

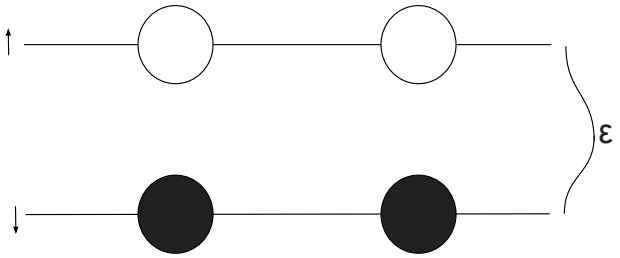


Figure 4.1: The LMG model with two particles and two holes where the particles can move two. The two levels are separated by a constant ε.

For a N-fermion system the levels are N-fold degenerate, represented by the different positions the particles can be in in 4.1, with two characteristic quantum numbers associated with each particle. The σ quantum number is assumed to be -1 in the lower level and $+1$ in the higher level, which can be seen as particle spin. We will use p to denote the degenerate state in which a particle resides in. So σ denotes which level the particle is on and p denotes where in that level it resides. The Hamiltonian proposed by Lipkin, Glick and Meshkow is as follows:

$$H = \sum_{p\sigma}\left(\frac{1}{2}\sigma\varepsilon\right)\hat{a}^{\dagger}_{p\sigma}\hat{a}_{p\sigma} - \frac{V}{2}\sum_{pp'\sigma}\hat{a}^{\dagger}_{p\sigma}\hat{a}^{\dagger}_{p'\sigma}\hat{a}_{p'-\sigma}\hat{a}_{p-\sigma} - \frac{W}{2}\sum_{pp'\sigma}\hat{a}^{\dagger}_{p\sigma}\hat{a}^{\dagger}_{p'-\sigma}\hat{a}_{p'\sigma}\hat{a}_{p-\sigma}\ , \tag{4.44}$$

where the operators $\hat{a}^{\dagger}_{p\sigma}$ creates and $\hat{a}_{p\sigma}$ destroys a particle in the energy level associated with σ and in the p position within that level. As an example, if we start out with two particles in the lower level:

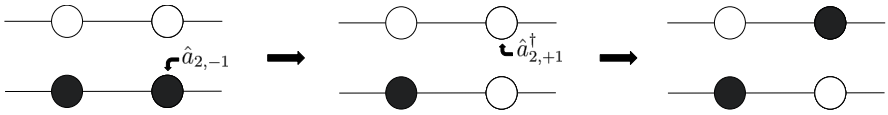


Figure 4.2: Two particles in the $\sigma = -1$ level filling out the $p = 1$ and $p = 2$ state of the LMG model. The particle at $p = 2$ and $\sigma = -1$ is destroyed and then a particle is created at $p = 2$, $\sigma = +1$.

Here we excite one of the particles in the lower layer up to the $\sigma = +1$ layer by destroying and then creating a particle. The Hamiltonian has three parts. Firstly we have the single-particle energy of each particle in the system. Then the a term that moves pairs of particle from one level to another, with an interaction strength proportional to V. Lastly we have a term that splits pairs of particles, with an interaction strength proportional to W, as in the example above.
In the $N = 2$ fermions example above we have in total four possible states, since two fermions cannot have the same quantum numbers. Essentially we want to look at each of these states and determine their contribution to the ground state energy by how likely the system is to be measured in that state. In our case we will simplify the Hamiltonian at 4.44 by defining $W = 0$ such that we are left with:

$$H = \sum_{p\sigma} \left(\frac{1}{2}\sigma\varepsilon \right) \hat{a}^{\dagger}_{p\sigma}\hat{a}_{p\sigma} - \frac{V}{2} \sum_{pp'\sigma} \hat{a}^{\dagger}_{p\sigma}\hat{a}^{\dagger}_{p'\sigma}\hat{a}_{p'-\sigma}\hat{a}_{p-\sigma} \tag{4.45}$$

4.4.2 Rewriting the Hamiltonian

An advantage of the LMG model is the fact that the two-body interaction does not change the value of p. Together with the two-valued σ, each particle can only exist in two possible states. This suggests to use the quasi-spin operators to rewrite the Hamiltonian. These quasi-spin operators are defined as follows:

$$\begin{aligned} \hat{J}_{+} &= \sum_{p}^{N} \hat{a}^{\dagger}_{p,+}\hat{a}_{p,-} \\ \hat{J}_{-} &= \sum_{p}^{N} \hat{a}^{\dagger}_{p,-}\hat{a}_{p,+} \\ \hat{J}_{z} &= \frac{1}{2}\sum_{p}^{N} \left(\hat{a}^{\dagger}_{p,+}\hat{a}_{p,+} - \hat{a}^{\dagger}_{p,-}\hat{a}_{p,-} \right) \end{aligned} \tag{4.46}$$

Where the $+,-$ indicates the quantum number $\sigma = \{+1, -1\}$ as associated with spin up and spin down respectively. The $\hat{J}_{+}$ accounts for the energy of a particle being excited up a level, the $\hat{J}_{-}$ accounting for a particle falling down a level. While the $\hat{J}_{z}$ takes into account the difference in single-particle energy of the two energy levels. All for each degenerate states p. Using these quasi-spin operators we can rewrite the Hamiltonian as:

$$H = \varepsilon\hat{J}_{z} - \frac{V}{2}\left(\hat{J}_{+}\hat{J}_{+} + \hat{J}_{-}\hat{J}_{-} \right) - \frac{W}{2}\left(\hat{J}_{+}\hat{J}_{-} + \hat{J}_{-}\hat{J}_{+} \right) \tag{4.47}$$

4.4.3 LMG Hamiltonian represented with Pauli matrices

To represent the LMG Hamiltonian on a quantum computer we need to first write it in terms of Pauli matrices. From the quasi-spin form 4.47 we need to find a conversion to what Pauli gates are to be applied to which qubits. By defining $W = 0$ we have that

$$H = \varepsilon\hat{J}_{z} + \frac{1}{2}V\left(\hat{j}^{2}_{+} + \hat{J}^{2}_{-} \right) \ . \tag{4.48}$$

Following the lecture notes of Hjort-Jensen [7], we have the mappings:

$$\hat{j}^{(n)}_{z} = \frac{1}{2}\sum_{\sigma} \sigma a^{\dagger}_{n\sigma}a_{n\sigma} \tag{4.49}$$

$$\hat{j}^{(n)}_{\pm} = \hat{a}^{\dagger}_{n\pm}\hat{a}_{n\pm} \ , \tag{4.50}$$

and the commutation relations:

$$\left[\hat{J}_{+}, \hat{J}_{-} \right] = 2\hat{J}_{z} \tag{4.51}$$

and

$$\left[\hat{J}_z, \hat{J}_\pm\right] = \pm\hat{J}_\pm \ . \tag{4.52}$$

With the ladder operators being

$$\hat{J}_\pm = \hat{J}_x \pm i\hat{J}_y \ , \tag{4.53}$$

we then have the total spin operator

$$\hat{J}^2 = \hat{J}_x^2 + \hat{J}_y^2 + \hat{J}_z^2 = \frac{1}{2}\left\{\hat{J}_+, \hat{J}_-\right\} + \hat{J}_z^2 \ , \tag{4.54}$$

where n represents which particle the operator is applied to. Defining the number operators as

$$\hat{N}_+ = \sum -n\hat{a}_{n+}^\dagger \hat{a}_{n+} \tag{4.55}$$

$$\hat{N}_- = \sum -n\hat{a}_{n-}^\dagger \hat{a}_{n-} \tag{4.56}$$

$$\hat{N} = \sum -n\sigma\hat{a}_{n\sigma}^\dagger \hat{a}_{n\sigma} \ , \tag{4.57}$$

we can see that the operator $\hat{J}_z$ counts half the difference of particles in the upper and lower levels. Since the Hamiltonian preserves the number of particles and particles can only be moved in pairs, $\hat{J}_z$ can take the values:

$$\hat{J}_z \in \left\{-\frac{N}{2}, -\frac{N}{2}+1, \ldots, \frac{N}{2}-1, \frac{N}{2}\right\} \tag{4.58}$$

Looking at the rotation operator

$$\hat{R} = e^{i\phi\hat{J}_z} \ ,$$

we have that the possible eigenvalues r of the signature operator would be

$$r = +1 \ , \ \hat{j}_z = 2n \tag{4.59}$$

$$r = +i \ , \ \hat{j}_z = 2n + \frac{1}{2} \tag{4.60}$$

$$r = -1 \ , \ \hat{j}_z = 2n + 1 \tag{4.61}$$

$$r = -i \ , \ \hat{j}_z = 2n + \frac{3}{2} \ . \tag{4.62}$$

Though since we can only move particles in pairs we only encounter the values $r = +1$ and $r = -1$. This also means that $\hat{j}_z$ can only change as follows:

$$\hat{j}_z \to \frac{1}{2}[(\hat{N}_+ \pm 2n) - (\hat{N}_- \mp 2n)] \ .$$

Which can be written as

$$\hat{j}_z \to \hat{J}_z \pm 2n \ ,$$

So for each particle we a state doublet $\{(n,+1),(n,-1)\}$. To map this to a quantum circuit we would need to have all the possible outcomes represented by the $|0\rangle$ and $|1\rangle$ measurements of the qubits in the circuit. One way to do this would be to have a qubit for each state (n,σ), then $|0\rangle$ represents an unoccupied state and $|1\rangle$ an occupied state. Such a mapping would require a qubit for each possible state, which is $2N$ since we start out with filling up the lower energy level with particles.
Another mapping, better in this case, would be to use the fact that both degeneracy of the states are never occupied at the same time. We would then use a qubit for each degeneracy, in total $\frac{N}{2}$ qubits, where $|0\rangle$ would represent $(n,+1)$ being occupied and $|1\rangle$ would be interpreted as $(n,-1)$ being occupied. This reduces the number of qubits needed by a half.
Using the mapping to one-body operators in 4.49 we get:

$$H = \varepsilon \sum_n \hat{j}_z^{(n)} + \frac{1}{2} V \left[\left(\sum_n \hat{j}_+^{(n)} \right)^2 + \left(\sum_n \hat{j}_-^{(n)} \right)^2 \right] . \tag{4.63}$$

$$= \varepsilon \sum_n \hat{j}_z^{(n)} + \frac{1}{2} V \sum_{n,m} \left(\hat{j}_+^{(n)} \hat{j}_+^{(m)} + \hat{j}_-^{(n)} \hat{j}_-^{(m)} \right) . \tag{4.64}$$

Using $\hat{j}_\pm = \hat{j}_x \pm i\hat{j}_y$ (4.53) we get

$$H = \varepsilon \sum_n \hat{j}_z^{(n)} + \frac{1}{2} V \sum_{n<m} \left(\hat{j}_x^{(n)} \hat{j}_x^{(m)} + \hat{j}_z^{(n)} \hat{j}_z^{(m)} \right) . \tag{4.65}$$

Then by converting the quasi-spin operators with the relations

$$j_x^{(n)} = \frac{1}{2} X_n \tag{4.66}$$

$$j_y^{(n)} = \frac{1}{2} Y_n \tag{4.67}$$

$$j_z^{(n)} = \frac{1}{2} Z_n , \tag{4.68}$$

we get the Hamiltonian in Pauli matrix form

$$H = \frac{1}{2}\epsilon \sum_n Z_n + \frac{1}{2} V \sum_{n<m} \left(X_n X_m - Y_n Y_m \right) . \tag{4.69}$$

4.4.4 Analytical Solution

For an exact solution of the LMG model one can use the full configuration interaction theory described in section 4.3.1. However, for a given total spin J the spin of a state can overlap with systems with fewer particles. For example, a system with $J = 2$ and another with $J = 1$ can both have the same spin projection $J_z = -1$ as seen below.

(a) $J = 1$, $J_z = -1$ (b) $J = 2$, $J_z = -1$

Figure 4.3: Two systems with different total spin but both are in a state where $J_z = -1$.

Therefor a more complete Hamiltonian would differentiate between these states. For a four-particle system we would then have

$$H_4 = \begin{bmatrix} H_{J=2} & & 0 \\ & H_{J=1} & \\ 0 & & H_{J=0} \end{bmatrix} . \tag{4.70}$$

Since the Hamiltonian commutes with J^2,

$$\left[H, J^2\right] = 0 \, , \tag{4.71}$$

J is a good quantum number and all other elements in the H_4 Hamiltonian becomes zero. As a diagonal block matrix we can then instead diagonalize the J-specific Hamiltonians separately. For $J = 2$ we have:

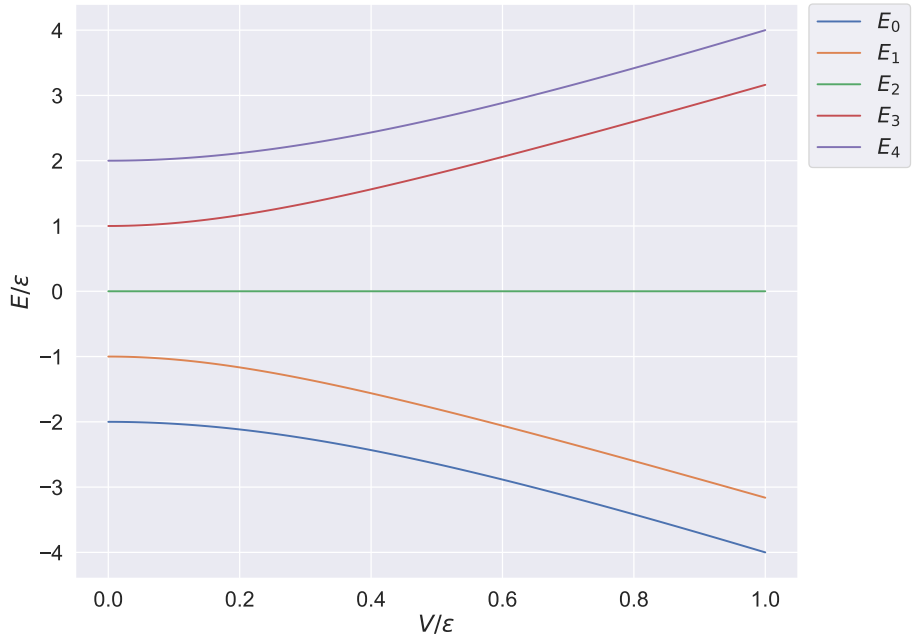


Figure 4.4: The analytical solution for the LMG model with total spin $J = 2$, $\varepsilon = 1$ and $W = 0$.

And for $J = 1$:

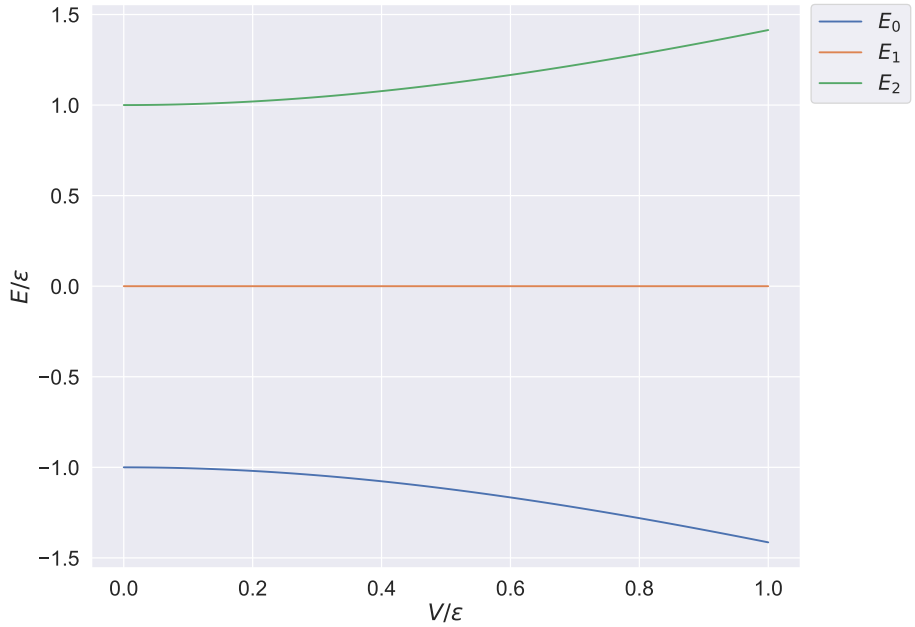


Figure 4.5: The analytical solution for the LMG model with total spin $J = 1$, $\varepsilon = 1$ and $W = 0$.

Chapter 5

Quantum Computing

Quantum computing is to process quantum information by applying operations to it. As opposed to classical computers, quantum computers can make use of quantum phenomena, such as entanglement and superposition, to expedite calculations in some cases. To understand how to process information through a quantum computer it is essential to understand the mathematics of quantum computing.

5.1 Basic Principles

5.1.1 Bra and ket notation

A state vector

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} , \tag{5.1}$$

is represented with the Dirac bra-ket notation as

$$\boldsymbol{x} = |x\rangle = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ \dots \\ x_{n-1} \end{bmatrix} , \tag{5.2}$$

and the conjugate transpose

$$\boldsymbol{x}^\dagger = \langle x| = \begin{bmatrix} x_0^* & x_1^* & x_2^* & \dots & \dots & x_{n-1}^* \end{bmatrix} . \tag{5.3}$$

With another state vector $|y\rangle$ we have the definition of the inner product

$$\langle y|x\rangle = \sum_{i=0}^{n-1} y_i^* x_i = y_0^* x_0 + y_1^* x_1 + \cdots + y_{n-1}^* x_{n-1} . \tag{5.4}$$

and the outer product

$$|y\rangle\langle x| = \begin{bmatrix} y_0^* x_0 & y_1^* x_0 & \cdots & y_{n-1}^* x_0 & y_n^* x_0 \\ y_0^* x_1 & y_1^* x_1 & \cdots & y_{n-1}^* x_1 & y_n^* x_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_0^* x_{n-1} & y_1^* x_{n-1} & \cdots & y_{n-1}^* x_{n-1} & y_n^* x_{n-1} \\ y_0^* x_n & y_1^* x_n & \cdots & y_{n-1}^* x_n & y_n^* x_n \end{bmatrix} \tag{5.5}$$

5.1.2 The Qubit

The most basic piece of information in quantum computing is the qubit, a quantum state with two different states. The standard orthonormal basis states of a qubit is as follows:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

This means that the state can only be measured to be in state $|0\rangle$ or $|1\rangle$. Before measurement, however, the state of a qubit can be a superposition of the computational basis states. Which can be described as a linear combination of the two basis states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} ,$$

where both $\alpha, \beta \in \mathbb{C}$ and with the normalization condition $|\alpha| + |\beta| = 1$. It is usefull to visualize the qubit as a sphere of possible states it can be in, called the Bloch sphere.

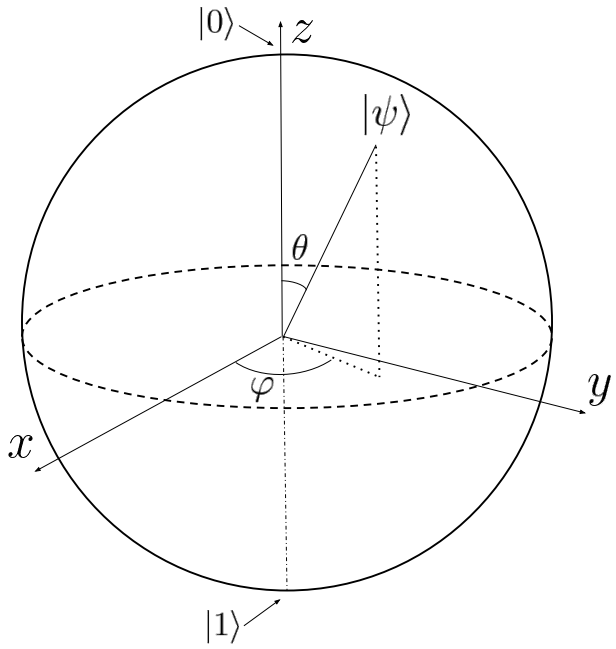


Figure 5.1: The possible states of a single qubit represented by a sphere with the apexes being the computational basis states.

Since we have the restriction $|\alpha| + |\beta| = 1$, the two complex values α and β only contribute to two degrees of freedom, resulting in the surface of the Bloch sphere, 5.1, as the state space of the single qubit.

5.1.3 Multi-Qubit States

Bringing in another qubit will give us more possible measured outcomes. Therefor our computational basis needs to be expanded, encompassing every possible combination. This is done by tensor product of the possible single qubit measurements. For a two-qubit system we then have

$$
\begin{aligned}
|0\rangle \otimes |0\rangle &= \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \\
|1\rangle \otimes |0\rangle &= \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \\
|0\rangle \otimes |1\rangle &= \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \\
|1\rangle \otimes |1\rangle &= \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \ ,
\end{aligned}
\tag{5.6}
$$

which then is a orthonormal basis for our new two-qubit system. To simplify writing, the tensorproduct between states are often written as

$$
\begin{aligned}
|0\rangle \otimes |0\rangle &= |00\rangle \\
|1\rangle \otimes |0\rangle &= |10\rangle \\
|0\rangle \otimes |1\rangle &= |01\rangle \\
|1\rangle \otimes |1\rangle &= |11\rangle \ .
\end{aligned}
\tag{5.7}
$$

Adding more qubits is simple. For each additonal qubit one tensormulitply the basis with the single-qubit basis, resulting in a 2^N possible measured states of the system.

5.1.4 Unitary Operators

Unitary operators are used to transform a state to another. They are called unitary because of the condition:

$$
UU^* = U^*U = I \ , \tag{5.8}
$$

where I is the identity matrix of the same size as U. This condition is necessary such that the transformations doesn't change the size of the state vector. The unitary matrices applied to a state are often referred to as quantum gates, or just gates, as taken from logical gates in classical computing. For completeness we will give a explanation of all the gates used in this thesis:
The identity matrix does not transform a qubit, but it is an unitary matrix.

Identity:
$$
I = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \ . \tag{5.9}
$$

The Pauli spin matrices have applications outside of quantum computing and go by several different names, some which explain better what they do to a quantum bit. The first one is the

$$\textbf{NOT Gate}: \qquad \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} , \tag{5.10}$$

which rotates the state π radians around the x-axis, also called bit flip gate because it flips $|0\rangle$ to $|1\rangle$. The

$$\textbf{Pauli Y}: \qquad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{5.11}$$

rotates around the y-axis π radians. While the

$$\textbf{Phase Flip}: \qquad \sigma_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5.12}$$

rotates around the z-axis π radians. For transforming a state into a equal superposition of its opposite on the Bloch sphere we have the:

$$\textbf{Hadamard}: \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} , \tag{5.13}$$

which characteristically creates the equal superpositions from the base states:

$$H\,|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \quad H\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} . \tag{5.14}$$

We also have the

$$\textbf{Phase Gate}: \qquad S = \sqrt{\sigma_z} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} , \tag{5.15}$$

which rotates the state $\frac{\pi}{2}$ radians around the z-axis. There are also gates which require a control, an additional qubit which changes the effect of the gate. An example is the

$$\textbf{Controlled NOT}: \qquad \textbf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} , \tag{5.16}$$

which is essentially the identity when the control qubit is in state $|0\rangle$ and a **NOT** gate when the control qubit is in state $|1\rangle$. Then we have the

$$\textbf{SWAP Gate}: \qquad \textbf{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \tag{5.17}$$

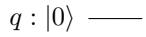
which switches the values of two target qubit.

5.2 Quantum programming

Quantum programming is setting up operations on quantum states in a specific order.

5.2.1 Quantum Circuit

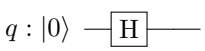
To represent the operations done, their order and target states, one uses a quantum circuit. A single state, without any operations, in quantum circuit notation looks like the following



q indicates in which bit of the registry the state resides and the line indicates what operations are done on the state, starting from left to right. So writing

$$H\,|0\rangle$$

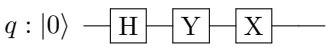
in quantum circuit notation we have



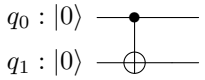
Doing operations in a row, for example

$$\sigma_x \sigma_y H\,|0\rangle \ ,$$

would be added to the right of the first Hadamard gate:

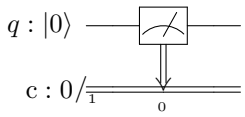


Some operations require a control. For the **CNOT** gate we have the control marked by a $\cdot$ and the target qubit marked by $\oplus$ as shown here:



5.2.2 Measurement of Output

In quantum mechanics we have that measuring a state changes it, or in our case the qubits collapses to either one of our base vectors $|0\rangle$ or $|1\rangle$. The measurement of a qubit on a quantum circuit is represented as the following:

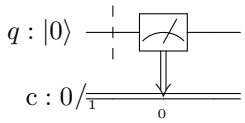


Measuring here will either give us $|0\rangle$ or $|1\rangle$ with an equal 50% chance. The information we want is most often the probabilities, and not the final state itself, since the probabilities ties to what state the qubit was in before it was measured. To extract the probabilities one utilizes the law of large numbers by computing the circuit a N number of times and calculate the average number of $|0\rangle$ measurements and $|1\rangle$ measurements.

5.2.3 Transformation of basis

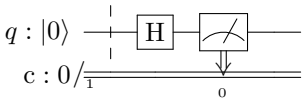
When measuring the qubits of a quantum circuit we are be often limited to doing the actual measuring in the z-basis. To overcome this limitation we simply transform the basis of the qubits to the basis we want to measure them in, or rather shift them back to the z-basis from the desired basis of measurement. The different basis transformations are as follows.

Z-basis:



Since we are already in the z-basis this one requires no transformation. The dashed line is to indicate that there are some sequence preceding it.

X-basis:

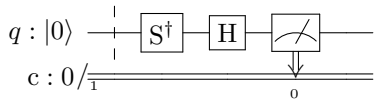


which comes from the fact that

$$HXH = Z\ . \tag{5.18}$$

And similarly

Y-basis:



where

$$S^\dagger HYHS^\dagger = Z \ . \tag{5.19}$$

5.2.4 Circuit measurement to expectation value

In the context of measurement the energy of a state for a given Hamiltonian we need to take into account the eigenvalues of the basis we measure in, here the z-basis. For the z-basis we have that

$$Z\left|0\right\rangle = 1\left|0\right\rangle \qquad Z\left|1\right\rangle = -1\left|1\right\rangle \ , \tag{5.20}$$

which means that for an example Hamiltonian

$$H_\psi = \varepsilon\sigma_x \tag{5.21}$$

will have the expectation value

$$\langle H_\psi\rangle = \varepsilon\left\langle\sigma_x\right\rangle \ , \tag{5.22}$$

where

$$\langle\sigma_x\rangle = n_0 - n_1 \ , \tag{5.23}$$

since we have matched it against the z-basis we measure in. Here n_0 and n_1 is the number of $|0\rangle$ and $|1\rangle$ measured respectively. For more than one qubit the eigenvalues accumulate. If we change our Hamiltonian to

$$H_\psi = \varepsilon\sigma_x \otimes \sigma_y \ , \tag{5.24}$$

we end up with the calculation of the expectation value

$$\langle\sigma_x \otimes \sigma_y\rangle = n_{00} - n_{01} - n_{10} + n_{11} \ , \tag{5.25}$$

which can be generalized for $d \cdot n_s$ where s is a basis state:

$$d = (-1)^{N_1} \ , \tag{5.26}$$

where N_1 is the number of 1's in the basis state s.

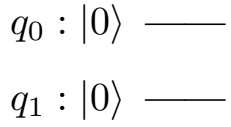
5.3 Entangled Qubits and Bell states

5.3.1 Entanglement

To make two qubits entangled we need to make them dependent on each other, meaning that measuring one of the qubits gives us information about the output of the other. An intuitive approach would be to use a controlled gate, since the

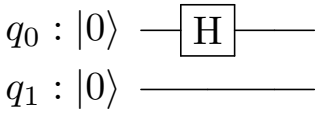
effect of the gate on the target qubit changes based on the state of the control qubit. We start with two qubits in the $|0\rangle$ state.

Quantum Circuit: **State after applied gates:**



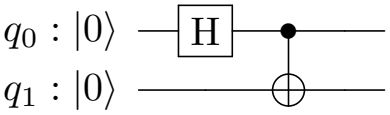
$$|q_1 q_2\rangle = |00\rangle \ . \tag{5.27}$$

Then to put both qubit in a superposition we apply the Hadamard gate to each of them:



$$|q_1 q_2\rangle = \tfrac{\sqrt{2}}{2}|00\rangle + \tfrac{\sqrt{2}}{2}|10\rangle \ . \tag{5.28}$$

In doing so the first qubit becomes maximally mixed, meaning the first qubit is equally likely to be measured either $|0\rangle$ or $|1\rangle$, as seen in 5.28. Still, the second qubit stays unchanged at $|0\rangle$, so measuring one of the qubits in 5.28 would not give us any information about the other. We apply then a **CNOT** gate:

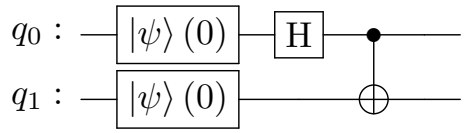


$$|q_1 q_2\rangle = \tfrac{\sqrt{2}}{2}|00\rangle + \tfrac{\sqrt{2}}{2}|11\rangle \ . \tag{5.29}$$

The **CNOT** gate makes it so that measuring $q_0 = |0\rangle$ tells us that $q_1 = |0\rangle$, while measuring $q_0 = |1\rangle$ makes it guaranteed that $q_1 = |1\rangle$, and vice versa. These linked measured values is what constitutes an entanglement of qubits.

5.3.2 Bell States

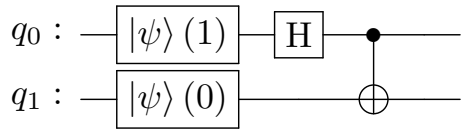
The resolving entangled state 5.29 is one of the four maximally entangled states of a two-qubit system called the Bell states. There are four of them because of the four combination of initial states our two-qubit system can be in. We have the first one with initial state $|00\rangle$:



$$|\Phi^+\rangle = \tfrac{\sqrt{2}}{2}|00\rangle + \tfrac{\sqrt{2}}{2}|11\rangle \, .$$

(5.30)

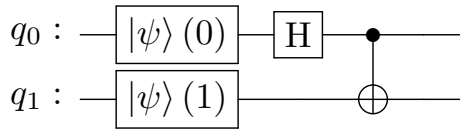
For initial state $|10\rangle$:



$$|\Phi^-\rangle = \tfrac{\sqrt{2}}{2}|00\rangle - \tfrac{\sqrt{2}}{2}|11\rangle \, .$$

(5.31)

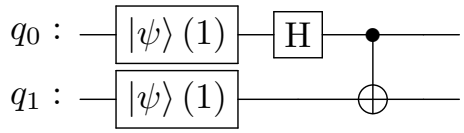
And for $|01\rangle$



$$|\Psi^+\rangle = \tfrac{\sqrt{2}}{2}|10\rangle + \tfrac{\sqrt{2}}{2}|01\rangle \, .$$

(5.32)

And lastly $|11\rangle$:



$$|\Psi^-\rangle = \tfrac{\sqrt{2}}{2}|10\rangle - \tfrac{\sqrt{2}}{2}|01\rangle \, .$$

(5.33)

5.4 Noisy intermediate-scale Quantum

The state of modern day quantum computers are in the so called 'noisy intermediate-scale quantum', meaning that the computers are not quite so accurate and with relatively few qubits. Taking into account, and minimizing, noise in our output is relevant when using real world quantum computers, opposed to simulated ones.

5.5 Variational Quantum Eigensolver

Essentially the Varitonal Quantum Eigensolver, abrivated VQE, is a method to find the ground state of a Hamiltonian by variation of the state the system is in. Starting with a Hamiltonian in Pauli form:

$$\hat{H} = \sum_i c_i \hat{P}_i \tag{5.34}$$

where c_i is the coefficient of the Pauli operator $\hat{P}_i$. We then have a general state $|\psi\rangle$ which is dependent on the angle to the z-axis, see 5.1, for each qubit in the system.

$$|\psi(\theta_1, \ldots, \theta_N)\rangle = |\psi(\boldsymbol{\theta})\rangle \ . \tag{5.35}$$

Then the energy of that state is the expectation value

$$E(\boldsymbol{\theta}) = \left\langle \hat{H} \right\rangle = \langle \psi(\boldsymbol{\theta})| \, c_i \hat{P}_i \, |\psi(\boldsymbol{\theta})\rangle = \sum_i c_i \langle \psi(\boldsymbol{\theta})| \, \hat{P}_i \, |\psi(\boldsymbol{\theta})\rangle \ . \tag{5.36}$$

By variating the angles of $\boldsymbol{\theta}$ and checking the energy of each state we can then find the lowest energy state, which would then be the ground state of the system. Going through an exhaustive list of states would be extremely taxing computationally, especially with higher number of qubits. And in this case we shall see that it is unnecessary.

5.5.1 Optimizing the choice of state

Chapter 6

Quantum Machine Learning

6.1 Quantum Nodes

6.2 Variational Quantum Boltzmann Machine

6.2.1 Variational Quantum Imeginary Time Evolution

Part II

Implementation

Chapter 7

Qiskit

Qiskit is a toolkit for implementing quantum circuits, applying gates to quantum states and measurement of them, in our programs. In this section we will go over the most relevant parts.

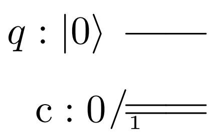
7.1 Initialization and Gates

To create an empty circuit we first make the quantum registries where the states will reside and the classical registries for the measurement results.

```
import qiskit as qk

quantum_reg = qk.QuantumRegistry(1)
classical_reg = qk.ClassicalRegistry(1)
circuit = qk.QuantumCircuit(quantum_reg, classical_reg)
```

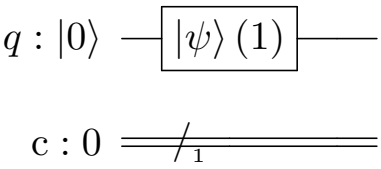
We then have the circuit:



The default initial state is $|0\rangle$, but we can change initialize it to something else.

```
circuit.initialize(1, 0)
```

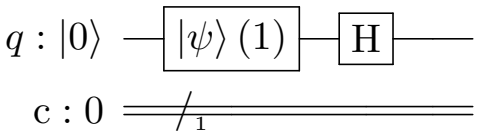
Resulting in



We apply gates by specifying target qubit or qubits.

```
circuit.h(0) #Hadamard gate
```

Which gives us



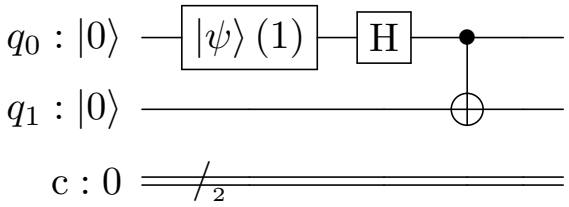
For controlled gates we add another qubit to the circuit:

```
quantum_reg = qk.QuantumRegistry(2)
classical_reg = qk.ClassicalRegistry(2)
circuit = qk.QuantumCircuit(quantum_reg, classical_reg)
circuit.initialize(1, 0)
circuit.h(0)
```

Then a controlled gate is applied by specifying the control and target:

```
circuit.cx(0, 1) # Controlled Not gate
```

And the result is

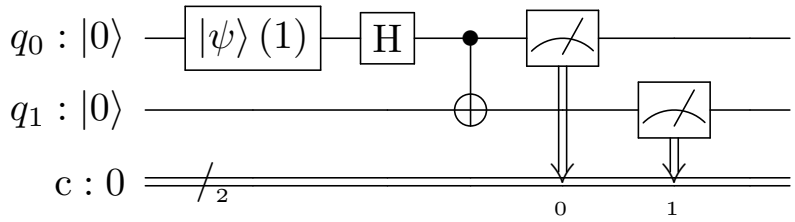


7.2 Measurement and execution

The measurement of a qubit is done by specifying the qubit to measure and which classical bit to store it into.

```
circuit.measure(0, 0)
circuit.measure(1, 1)
```

The circuit is then:



To run the code, or execute the circuit, we need to set up the backend. The backend is either a simulated quantum computer or a way for the program to communicate with a real one. Here we use the following simulator.

```
backend = qk.Aer.get_backend('qasm.simulator')
```

Then we execute the circuit above.

```
job = backend.run(circuit, shots = 10_000)
```

Where **shots** is how many times the circuit is to be run, as it is the average that is of interest. The results are then easily extracted.

```
counts = job.result().get_counts()
```

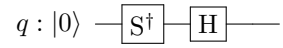
Which is returned in the form of a dictionary.

```
counts = {
    '00' : n_00 #Number of |00> measured,
    '01' : n_01 #Number of |01> measured,
    '10' : n_10 #Number of |10> measured,
    '11' : n_11 #Number of |11> measured
}
```

7.3 Copy and append

The copy and append methods can speed up our code by reduce the number of times we need to add gates to a circuit. Say we have the circuit for the Y-basis transformation

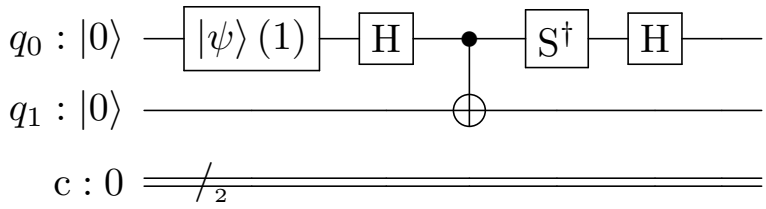
```
quantum_reg = qk.QuantumRegistry(1)
y_basis = qk.QuantumCircuit(quantum_reg)
y_basis.sdg(0)
y_basis.h(0)
```



and we want to append it to our previous circuit before measurements was added. We can then use qiskit's 'compose' method.

```
circuit.compose(y_basis, qubits = 0)
```

And we get



.

If we rather want to keep the base circuit for reuse, we can first create a copy and append to that one instead.

```
circuit_copy = circuit.copy()
circuit_copy.append(y_basis, qubits = 0)
```

Chapter 8

Restricted Boltzmann Machine

8.1 Calculation of the local energy

We know that the local energy of a particular sample state $|s\rangle$ can be calculated as

$$E_{loc} = \frac{\langle s| H |\psi_{rbm}\rangle}{\langle s|\psi_{rbm}\rangle} , \tag{8.1}$$

but here we will explain in more detail how this is done for the each of the different systems we are looking at. First of all it is important to remember the structure of the input to our function calculating the local energy. We want to vectorize the calculations as much as possible so the input will be all our samples, taken with the Gibbs or Metropolis-Hastings algorithm, together in one array:
Our operations will then be done on all the samples at once, decreasing computation time by vectorization as well as opening up for use of the GPU. For reference we will call the input array of all the samples $|s\rangle$ as $\mathbf{S}$. To use the definition of the local energy, 8.1, we need to extract the machine state from our set of samples. As described in **??** we check the distribution by extracting all unique states in our set of samples and setting the amplitudes as the square root of their relative occurrence in the set, presuming that all are positive definite.
...

We then have a approximation of our machine state $|\psi\rangle$. The next step of using each state in the sample set is varies with the hamiltonian of the system.

8.1.1 The Lipkin model

The Lipkin model hamiltonian is defined as

$$H = \sum_{p\sigma} \left(\frac{1}{2}\sigma\varepsilon\right) \hat{a}^\dagger_{p\sigma}\hat{a}_{p\sigma} - \frac{V}{2}\sum_{pp'\sigma} \hat{a}^\dagger_{p\sigma}\hat{a}^\dagger_{p'\sigma}\hat{a}_{p'-\sigma}\hat{a}_{p-\sigma} - \frac{W}{2}\sum_{pp'\sigma} \hat{a}^\dagger_{p\sigma}\hat{a}^\dagger_{p'-\sigma}\hat{a}_{p'\sigma}\hat{a}_{p-\sigma} \ , \tag{8.2}$$

For a more structured explanation we separate the hamiltonian in three parts:

$$H_\varepsilon = \sum_{p\sigma} \left(\frac{1}{2}\sigma\varepsilon\right) \hat{a}^\dagger_{p\sigma}\hat{a}_{p\sigma} \tag{8.3}$$

$$H_V = \frac{V}{2}\sum_{pp'\sigma} \hat{a}^\dagger_{p\sigma}\hat{a}^\dagger_{p'\sigma}\hat{a}_{p'-\sigma}\hat{a}_{p-\sigma} \tag{8.4}$$

$$H_W = \frac{W}{2}\sum_{pp'\sigma} \hat{a}^\dagger_{p\sigma}\hat{a}^\dagger_{p'-\sigma}\hat{a}_{p'\sigma}\hat{a}_{p-\sigma} \ . \tag{8.5}$$

A state $|b\rangle$ contributes for the different parts as follows:

- H_ε - the same state $|b\rangle$.
- H_V - states that are a excited or deexcited pair away from the state $|b\rangle$.
- H_W - states one excited or deexcited particle away from $|b\rangle$.

For the H_ε contribution we determine which samples matches which parts of the machine state $|\psi_{rbm}\rangle$. We create a mask showing which basis state each sample matches with, named $\vec{mask}$.
We then have an array of the machine states amplitudes multiplied by ε. Applying $\vec{mask}$ to this array then selects the correct value for each sample without having to calculate duplicates more than once. We then have the H_ε contribution for each sample directly:
...

For the H_V contributions we first need to find each possible one-pair-different basis states that matches with a part of the machine state. Since the machine state is defined as the sample distribution we can check what states are one-pair-different from each other by doing the following:
...

We check each of the unique states, $|b_i\rangle$, with all the other unique states, $|b_j\rangle$, and see if there are a one-pair-difference between them, setting the element (i, j) to 1 if that is the case. As a result we end up with a $N \times N$ matrix, $\mathbf{V}$, where N is the number of unique states. Each pair represents the element

$$\frac{\langle b_i| H_V |b_j\rangle}{\langle b_i|\psi_{rbm}\rangle} = \frac{V\alpha_j}{\alpha_i} \ , \tag{8.6}$$

where α_i and α_j is respectively the amplitude of the basis state $|b_i\rangle$ and $|b_j\rangle$ in the machine state ψ_{rbm}. This means we need to scale our matrix $\mathbf{V}$ with the interaction strength V. Then scale each row i with $\frac{1}{\alpha_i}$ and each column j with the corresponding states amplitude α_j. Each row then represents the elements of

$$E_{loc,V}(|b_i\rangle) = \frac{\langle b_i| H_V |\psi_{rbm}\rangle}{\langle b_i|\psi_{rbm}\rangle} ,$$

and since the amplitudes are already assumed to be positive definite, we can then calculate the total contribution of the basis state $|b_i\rangle$:

$$E_{loc,V}(|b_i\rangle) = \sum_j \mathbf{V}_{i,j} , \tag{8.7}$$

which we can then apply $\vec{mask}$ to and get the H_V contribution for each sample:
...

Chapter 9

Variational Quantum Boltzmann Machine

9.1 Initialization

9.2 Optimization of the VarQBM

9.3 Quantum Machine Structure

9.4 Quantum Machine Parameter Optimization

Bibliography

[1] F. Rosenblatt, *Principles of neurodynamics; perceptrons and the theory of brain mechanisms.* Washington: Spartan Books, 1962, bibliography : p. 609-616. [Online]. Available: http://hdl.handle.net/2027/mdp.39015039846566

[2] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, Jun 1976. [Online]. Available: https://doi.org/10.1007/BF01931367

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct 1986. [Online]. Available: https://doi.org/10.1038/323533a0

[4] G. E. Hinton and T. J. Sejnowski, "Analyzing cooperative computation," *Fifth annual connference of the Cogniitive Science Society*, 1983.

[5] A. Yuille, "Boltzmann machine," *JHU*, 2016.

[6] H. Lipkin, N. Meshkov, and A. Glick, "Validity of many-body approximation methods for a solvable model: (i). exact solutions and perturbation theory," *Nuclear Physics*, vol. 62, no. 2, pp. 188–198, 1965. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002955826590862X

[7] M. Hjort-Jensen. (2023) Quantum computing and solving the eigenvalue problem for the lipkin model. [Online]. Available: https://github.com/CompPhysics/QuantumComputingMachineLearning/blob/gh-pages/doc/pub/week9/ipynb/week9.ipynb