Project 2: Solving the Ising model in 1 dimension using a Restricted Boltzmann machine

Jonny Aarstad Igeh
Department of Physics, University of Oslo, Norway

June 7, 2024

Abstract

In this project, we have implemented a Restricted Boltzmann machine (RBM) to solve the 1D Ising model. We have shown that the RBM can be used to approximate the ground state energy of the Ising model, and that the model is capable of learning the ground state energy of the system. We have also shown that the model is sensitive to its hyperparameters, and that it is important to have a large enough sample size for the Monte-Carlo sampler to properly traverse the spin-configuration landscape. For a 12 spin-chain lattice our model found the ground state energy $E_0 = -24$, for $J = \mu = 1.0$, with the spin-aligned ground state $\Psi = \bigotimes_i \sigma_\uparrow$. Our successful identification of the ground state of the simple 1D Ising model, as well as the more complex long-range interacting Ising model, shows that the RBM can be used to solve more complex systems, and that it is a promising tool for solving more complex systems in the future.

Contents

1 Introduction

Generative machine learning algorithms have taken both the scientific community, and the general public by storm in later years. The ability to generate new data at will, based on a set of training data, has proven to be a powerful tool in a wide range of applications, from image recognition to natural language processing, and even in art - where sophisticated models can produce music and video that even experts struggle to differentiate from human-made content. We are entering a new era of digital content, where the line between human and machine is becoming increasingly blurred.

In this report, we will build one of earliest algorithms in generative machine learning, the Restricted Boltzmann Machine (RBM), and use it to model the Ising model in one dimension. We will follow in the footsteps on some of the pioneers in the field, Carleo and Troyer [1], and attempt to recreate, in simpler terms, their revolutionary work.
The Ising model is a simple model of a ferromagnetic material, where the spins of the atoms are aligned in a lattice. The model is simple enough to be solved analytically, but complex enough to exhibit interesting phase transitions in higher dimensions. Our work will focus on the 1D Ising model, where the ground state can be found by brute force. However, we will use the RBM to model the wavefunction of the system, and use it to find the ground state in a more delicate manner. We will do MCMC sampling of various spin configurations in the lattice, and use the marginal probability of the RBM as a quantum wavefunction - and attempt to minimize the local energy to identify the ground state of the system.

This means, that our RBM will generate a probability distribution which will "mimic" the true distribution of the wavefunction that solves the Ising model in one dimension. We will use Metropolis sampling to sample the probability distribution generated by the RBM, and use the samples to calculate the local energy of the system. To optimize the RBM, a standard gradient descent method will be employed. The theoretical framework for our project will be presented in the theory section, and the methods and implementations will be sketched out in the method section. The results from our simulations will be presented in the results section, and analyzed in the discussion section. Finally, we will conclude our report with a summary of our findings and suggestions for further research.

2 Theory

2.1 The Variational Method

The variational method is a powerful tool in quantum mechanics, and is used to estimate the ground state energy of a system. This method is based on the variational principle, and is an effective way of approximating the ground state which is very often difficult to find, analytically but also numerically. We start with an initial ansatz wavefunction, and calculate the expectation value of our system Hamiltonian

$$E = \frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle} = \frac{\int d\sigma \Psi^*(\sigma,\theta) H \Psi(\sigma,\theta)}{\int d\sigma \Psi^*(\sigma,\theta)\Psi(\sigma,\theta)}. \tag{1}$$

where θ are the variational parameters, and σ is the spin configuration of the system. The expectation value of the Hamiltonian is then minimized with respect to the variational parameters, and the optimal parameters are used to estimate the ground state energy of the system.

2.1.1 The Variational Principle

The variational principle states that for any given Hamiltonian, the expectation value of the Hamiltonian in any state is always greater than or equal to the ground state energy of the Hamiltonian. This means that if we have a trial wavefunction, $|\Psi\rangle$, we can use this to estimate the ground state energy of the Hamiltonian.

$$E_0 \leq \frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle}. \tag{2}$$

As we can see, if we work to minimize the expectation value of the Hamiltonian, we will always find a "better" estimate to the ground state energy - while never going below it.

2.2 The Ising model

The Ising model is a well-studied model in statistical mechanics, and is used to describe the behavior of ferromagnetic materials. The model was first introduced by Wilhelm Lenz in 1920, and later solved by Ernst Ising in 1925. The model consists of a lattice of spins, where each spin can take on one of two values, $s_i = \pm 1$. The energy of the system is given by the Hamiltonian

$$H = -J \sum_{\langle ij \rangle} \sigma_i^z \sigma_j^z - \mu \sum_i \sigma_i^z, \tag{3}$$

where J is the coupling constant, σ_i is the spin at site i, and μ is the magnetic moment. The first sum runs over all nearest neighbor pairs of spins, and the second sum runs over all spins in the lattice. In our system, we will assume periodic boundary conditions - meaning we have a spin-chain lattice, with a constant non-zero temperature $T_0 = 1.0$.
The simplest 1D Ising model does not have a phase transition, and can be solved analytically.

2.2.1 Long-range interaction

Introducing long-range interactions between spins in the lattice will allow us to study how the ground state changes when more interactions are added. The Hamiltonian for the long-range Ising model is given by

$$H = -J \sum_{i \neq j} \frac{\sigma_i^z \sigma_j^z}{|i-j|^\alpha} - \mu \sum_i \sigma_i^z, \tag{4}$$

where the coupling constant now is also a function of the distance between the spins, and the first sum runs over all pairs of spins in the lattice. The parameter α determines the range of the interaction, and for $\alpha = 0$ we have the nearest neighbor Ising model. This means we have a splitwise-function to express the interaction between spins, as presented in the article by Martinez et al. [5].

$$J = \begin{cases} J_0 \frac{1}{|i-j|} & \text{if } |i-j| < n_v, \\ 0 & \text{otherwise} \end{cases}$$

where n_v is the number of neighbouring spins to include in the long-range interaction, i and j the indices of the spins, and J_0 is the coupling constant.

2.3 The Monte Carlo method

Initially equation (1) does not look like something we can use Markov-Chain Monte Carlo methods (MCMC) to evaluate, however with some clever rewriting it can be reformulated into a more useful expression. We begin by re-expressing the integrand in the numerator

$$\Psi_T^*(\sigma,\theta)H\Psi_T(\sigma,\theta) = \Psi_T^*(\sigma,\theta)\frac{\Psi_T(\sigma,\theta)}{\Psi_T(\sigma,\theta)}H\Psi_T(\sigma,\theta).$$

In this expression we recognise both the expression for the local energy E_L and the expression of the un-normalised probability density $\tilde{P} = \Psi_T^*\Psi_T$ formed from the wavefunction. Applying this to equation (1), and using the normalising factor in the denominator to normalise $\tilde{P}$, which we will denote by simply P, we get

$$E[H] = \int d\sigma P(\sigma,\theta)E_L(\sigma,\theta) \approx \frac{1}{N}\sum_i n_i(E_L)_i. \tag{5}$$

Where N is the total number of samples, n_i is the number of samples where the local energy has the value $(E_L)_i$ and

$$P(\sigma,\theta) = \frac{|\Psi_T(\sigma,\theta)|^2}{\int d\sigma \Psi_T^*(\sigma,\theta)\Psi_T(\sigma,\theta)}, \quad E_L = \frac{1}{\Psi_T}H\Psi_T.$$

This form of the local energy now allow us to solve the integral and get an estimate of the ground state energy of the system, for a given value of variational parameters using MCMC methods by sampling the local energy throughout spin-configuration space. The Monte Carlo method can be summarised into the following points

1: Initialise the system: Pick an initial spin-configuration $\sigma_0 = \sigma$ and a randonmly initialized variational parameter vector θ. Also fix the number N of MC samples the system should run over.
2: Select a proposed spin-configuration x_p, and let x_n be the current spin-configuration.
3: Check with some condition to see if the proposed move is accepted
4: **if** The proposed move is accepted **then**
5: Set the new spin-configuration as the proposed $\sigma_{n+1} = \sigma_p$
6: **else**
7: Set the new spin-configuration as the old spin-configuration $\sigma_{n+1} = \sigma_n$
8: **end if**
9: With the spin-configuration σ_{n+1} save or update any new values of importance and return to point (2) until we have N samples.

There are multiple ways of finding both a candidate for the new spin-configuration and accepting or rejecting the proposed move. We will use the Metropolis-Hastings algorithm, which is a widely used algorithm for MCMC methods.

2.3.1 Metropolis-Hastings

Here we will give a brief overview of the steps in the Metropolis-Hastings MCMC algorithm, but a more detailed explanation can be found in [4].

- *Proposal of spin-configuration*: The Metropolis algorithm is a Markov Chain Monte Carlo method, and as such we need to propose a new spin-configuration based on the current spin-configuration. This is done by "swapping" a random spin in the current lattice configuration, σ_i with it's opposite spin, $-\sigma_i$, i.e if the spin is up, we propose a new configuation where the spin is down.

- *Acceptance algorithm*: Given a new spin configuration, we need to decide whether to accept or reject the proposed re-configuration. This is done by calculating the ratio of the probability of the new spin-configuration and the old spin-configuration, and comparing this to a random number between 0 and 1. If the ratio is greater than the random number, we accept the configuration, if not we reject it. The acceptance probability is given by

$$P_{\text{accept}} = \min\left(1, \left|\frac{P(\sigma_{\text{new}})}{P(\sigma_{\text{old}})}\right|^2\right).$$

2.4 Restricted Boltzmann Machines

Solving for the ground state of our hamiltonian, it is important that we make a good wavefunction ansatz. In this ansatz, it is of the essence that we manage to properly capture the entanglement (or other physical properties) in the system - this is where the Restricted Boltzmann Machine (RBM) comes in. The generative nature of the probabilities produced by the RBM makes it a good candidate for modeling the wavefunction of a quantum system, and we will use this to our advantage in this project.

2.4.1 Architecture

The RBM is a type of neural network, and consists of two layers of nodes, a visible layer and a hidden layer. The nodes in the visible layer represent the physical degrees of freedom in the system, and the nodes in the hidden layer are used to model the correlations between the visible nodes. The nodes in the visible layer are connected to the nodes in the hidden layer, but there are no connections between nodes in the same layer. This is what makes the RBM a "restricted" Boltzmann machine. A visualization of the network is shown in figure 1. Working with our 1D Ising Hamiltonian, we require a discete spin

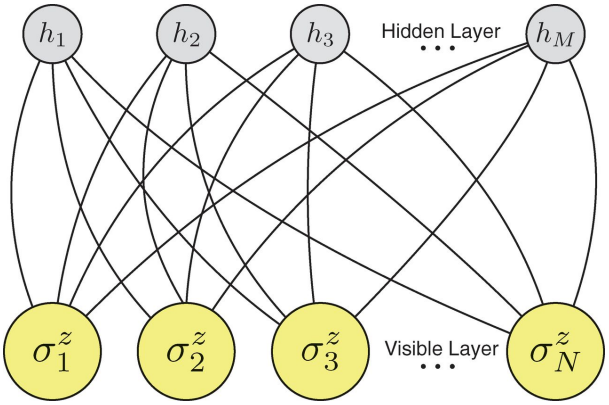

Figure 1: Visualization of the Restricted Boltzmann Machine.[1]

lattice model - and thus our RBM will be a *binary-binary* RBM. This means that the nodes in the visible layer and the hidden layer can only take on binary values, $\sigma_i, h_j \in \{0, 1\}$, and the RBM is described in detail by the Boltzmann distribution

$$P(\sigma, \mathbf{h}, \theta) = \frac{1}{Z(\theta)} e^{\frac{1}{T} - E(\sigma, \mathbf{h}, \theta)}, \tag{6}$$

where $Z(\theta)$ is the partition function, and θ are the variational parameters, T the temperature of the RBM (usually put to 1.0), and $E(\sigma, \mathbf{h}, \mathbf{a})$ is the energy-function of the RBM, expressed in a binary-binary RBM as

$$E(\sigma, \mathbf{h}, \theta) = -\sum_i a_i \sigma_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} \sigma_i h_j, \tag{7}$$

where σ is the visible layer, $\mathbf{h}$ is the hidden layer, and $\mathbf{a}, \mathbf{b}, \mathbf{w}$ are the variational parameters.

2.4.2 Initialization

As with any neural network architecture, the visible and hidden nodes are connected, as seen in figure (1), and these connections have their weights and biases. Furthermore, the RBM has a bias on both layers - and a "running" an RBM means to give an input to the visible layer, run it "through" the network to the hidden layer, and then take the trip back to the visible layer. The goal is then for the machine to "re-generate" a desired output, given some input.

For our purposes, we only will make use of the probability distribution supplied by the RBM - but the weights and biases play an important role nonetheless, and making a good initial starting point for the variational parameters is important.
Following the article [1], we will initialize the parameters as follows

$$W_{ij}, a_i, b_j \in (-0.1, 0.1) \tag{8}$$

which will be samples from the standard normal distribution with $\mu = 0$, $\sigma = 0.1$.

2.4.3 Marginal probability

In order to find the probability of any given configuartion of the visible units (the spin-configuration), we need to marginalize over the hidden units. This is done by summing over all possible hidden unit configurations, and the marginal probability is given by

$$P(\sigma) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} e^{-E(\sigma, \mathbf{h}, \mathbf{a})}, \tag{9}$$

In our binary-binary model, we will only allow the hidden units to take on values of -1 and 1. With that, and no intra-layer interactions, we can trace out the hidden units completely [1], and the marginal probability is then given by

$$P_\sigma = e^{\sum_j a_j \sigma_j} \times \prod_i^M F_i(\sigma), \tag{10}$$

$$F_i(\sigma) = 2 \cosh \left[b_i + W_{ij} \sigma_j \right]. \tag{11}$$

This marginal probability is what will constitute our wavefunction, and we will use this to calculate the local energy of the system. As we can see, this is not dependent on the hidden units - which gives us freedom in how we initialize our RBM. Meaning,

$$\Psi(\sigma, \theta) = e^{\sum_j a_j \sigma_j} \times \prod_i^M F_i(\sigma), \tag{12}$$

where we've excluded the partition function, since this is usually intractable - and will cancel in our expression for calculating acceptance probabilities (see (2.3.1)).

2.5 Optimization

By use of the variational principle, we can find the optimal variational parameters by minimizing the expectation value of the local energy. The "optimal" variational parameters will then be the ones that yields the lowest expectation value, and by (2), this will be our closest estimate to the ground state energy (and ground state configuration). For our project, we will make use of steepest gradient descent for optimization - since our problem is a mild minimization problem, this simple method will still yield strong results.

2.5.1 Steepest Gradient Descent

The steepest gradient descent method is a simple optimization algorithm, where we iteratively update the variational parameters in the direction of the steepest gradient. The gradient descent algorithm is as follows:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \nabla_\theta \langle E_L \rangle, \tag{13}$$

where η is the learning rate, θ the variational parameters, and $\nabla_\theta \langle E_L \rangle$ is the gradient of the local enerrgry w.r.t. the variational parameters at the current iteration.
I.e we calculate the gradient of the parameter(s) we are using to minimize the expectation value of the Hamiltonian, and then update the parameters by taking a step in the direction of the negative gradient. Finding the gradient of the local energy is non-trivial, and will also have to be estimated using the monte-carlo sampler. Following the derivations found in [3], the gradients are found by

$$\frac{\partial \langle E_L \rangle}{(\partial \theta_i)} = 2\big(\langle E_L \frac{1}{\Psi} \frac{\partial \Psi}{\partial \theta_i} \rangle - \langle E_L \rangle \langle \frac{1}{\Psi} \frac{\partial \Psi}{\partial \theta_i} \rangle \big) \tag{14}$$

Calculating the gradients, $\frac{1}{\Psi}\frac{\partial \Psi}{\partial \theta_i}$, we will follow the results presented in app. C of [1], and use the expression for the gradient of the local energy w.r.t. the variational parameters as

$$\begin{aligned}\frac{1}{\Psi}\partial_{a_i}\Psi &= \sigma_i,\\ \frac{1}{\Psi}\partial_{b_j}\Psi &= \tanh[\gamma_j],\\ \frac{1}{\Psi}\partial_{W_{ij}}\Psi &= \sigma_i \tanh[\gamma_j],\end{aligned}$$

where $\gamma_j = b_j + \sum_i W_{ij}\sigma_i$. These derivatives will be evaluated for each spin configuration in the sampling process, and found the same way an estimate for the local energy is found, see (1), and used to update the variational parameters in the gradient descent algorithm.

3 Method & implementations

3.1 Ising model

As presented in the theory section (2.2), we will implement the Ising model, and evaluate this to get estimates of the local energy, that we are looking to minimize. The following shows how our simple 1D Ising model can be implemented in Python, with periodic boundary conditions

```
spin_configuration = ...
spin_configuration = np.append(spin_configuration, spin_configuration[0])
J = ...
mu = ...
energy = 0
for idx in range(len(spin_configuration) - 1):
    energy -= J * spin_configuration[idx] * spin_configuration[idx + 1] + mu * spin_configuration[idx]
```

and for the long-range interaction case

```
spin_configuration = ...
J = ...
mu = ...
alpha = ...
nv = ...
energy = 0
def Jij(i, j, alpha, nv):
    dist = np.abs(i - j) ** alpha
    if dist > nv:
        return 0
    return J / dist

for idx in range(len(spin_configuration) - 1):
    energy -= mu * spin_configuration[idx]
    for jdx in range(idx+1, len(spin_configuration))
        energy -= Jij(idx, jdx, alpha, nv) * spin_configuration[idx] * spin_configuration[jdx]
```

where the choice of J, μ, α and n_v are parameters that we can tune to our liking.

3.2 Monte-Carlo sampler

One of the most important implementations is the Monte-Carlo sampler, this will allow us to both evaluate the local energy that we are hoping to minimize, as well as finding the gradients needed to achieve this minimization, by optimizing the network parameters. As an example, the following code snippet shows a simple MCMC sampling implementation

```
import numpy as np
n_steps = ...
curent_configuration = ...
grads = []
local_energies = []
for i in range(n_steps):
    old_probability = RBM.marginal_probability(curent_configuration) ** 2
    new_configuration = propose_new_configuration(curent_configuration)
    new_probability = RBM.marginal_probability(new_configuration) ** 2
    acceptance_ratio = min(1, (new_probability / old_probability) ** 2)
    if np.random.rand() < acceptance_ratio:
        curent_configuration = new_configuration
    grads.append(RBM.local_energy_gradients(curent_configuration))
    local_energies.append(RBM.local_energy(curent_configuration))
grads = np.mean(grads, axis=0)
local_energies = np.mean(local_energies)
```

This is the general structure we wish to implement for our project, following the steps introduced in the theory sections. We will use this to evaluate the local energy and the gradients of the local energy, which we will use to optimize the network parameters.

3.3 Restricted Boltzmann machine

As presented in section (2.4), we will set up our RBM neural network using a class architecture in Python. This is an efficient way for us to easily store and manipulate weights and biases of a given network instance. Initializing the weights and biases, as well as the two layers, are easily done using the Numpy library[2], and the following code snippet illustrates how this is done.

```
    import numpy as np
    n_visible = ...
    n_hidden = ...
    self.W = np.random.randn(n_visible, n_hidden) * 0.1
```

```
    self.a = np.random.randn(n_visible) * 0.1
    self.b = np.random.randn(n_hidden) * 0.1

    visible_layer = np.random.choice([-1, 1], size=n_visible, p=[0.5, 0.5])
    hidden_layer = np.random.choice([-1, 1], size=n_hidden, p=[0.5, 0.5])
```

With this, our network is initialized with random weights and biases, and the two layers are initialized with random values sampled from the standard normal distribution (numpy.randon.randn). The properties of the RBM (probabilities etc.) are calculated easily using the formulas from section (2.4).

3.4 Optimization

Finding the gradients of the local energy is crucial for the optimization of the network parameters. Extracting the gradients from the monte-carlo sampling as shown in previous sections, we can use these to update the network parameters following the steepest descent algorithm outlined in section (2.5). To following will outline the general structure of the optimization algorithm

1: Initialize the network parameters θ
2: Initialize the learning rate η
3: Initialize the number of epochs n_{epochs}
4: **while** not converged and not $> n_{\text{epochs}}$ **do**
5: Run the Monte-Carlo sampler to obtain estimates of the gradients of the local energy
6: Update the network parameters $\theta \leftarrow \theta - \eta \nabla_\theta E_{\text{local}}$
7: Calculate the local energy E_{local}, and check for convergence
8: **end while**

4 Results & discussion

In the following section, all results are produced with the following parameters for the RBM and the Hamiltonian:

$$N_{\text{visible}} = 12, \quad N_{\text{hidden}} = 12, \quad J = 1.0, \quad \mu = 1.0, \quad \text{learning rate} = 0.1, \quad \text{number of epochs} = 25$$

4.1 Ising model

As we know from the analytical solution of the 1D Ising model, the ground state is the configuration with all spins aligned in the same direction. This is the state with the lowest energy, and we can use this as a benchmark for our RBM implementation.

4.1.1 Importance of enough samples

The Monte-Carlo sampling is the bread and butter of our implementation, and the number of samples we use is crucial for the accuracy of our results. To illustrate the necessity of a large "enough" sample size, we have plotted the energy of the Ising model as a function of the number of samples in figure(2). In figure(2), we see that the energy of the Ising model converges to the analytical value *only* for the

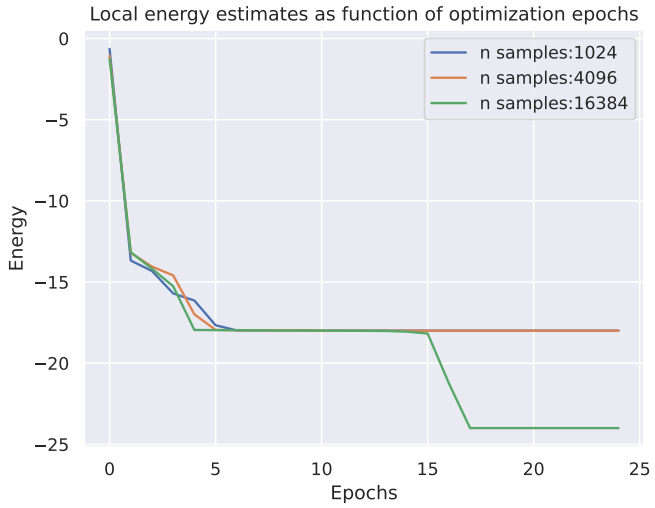

Figure 2: Energy of the Ising model as a function of the number of samples.

largest samplesize, $n = 2^14$, whereas it is stuck in a local minima for the smaller samplesizes. With too few samples, the Monte-Carlo sampler will not propely traverse the spin-configuration landscape well enough to properly identify the ground state, and subsequently it will yield the wrong results.
We can also do a statistical analysis of this problem, showcasing the variance and standard error, in the following plot From these two error plots, figure (4) and figure (3), we can see where the higher sample size "breaks out" of the local minima, there is a bump in the statistical errors - and this is where the model leaves the local minima, and progresses onwards to the true minima - the ground state of the 12 spin system with a ground state energy of -24 (with $J = 1.0, \mu = 1.0$).

There is also a clear link between the number of samples and the number of visible units (spins) in our system. With too many spins, and not enough samples - we get the wrong results shown in figure(2) for $n = 2^{10}, 2^{12}$.

For illustrative purposes, we can show the convergence of the system with a larger spin-chain of 20 spins, with the same parameters as before, using 2^{14} samples. The results are shown in figure (5).

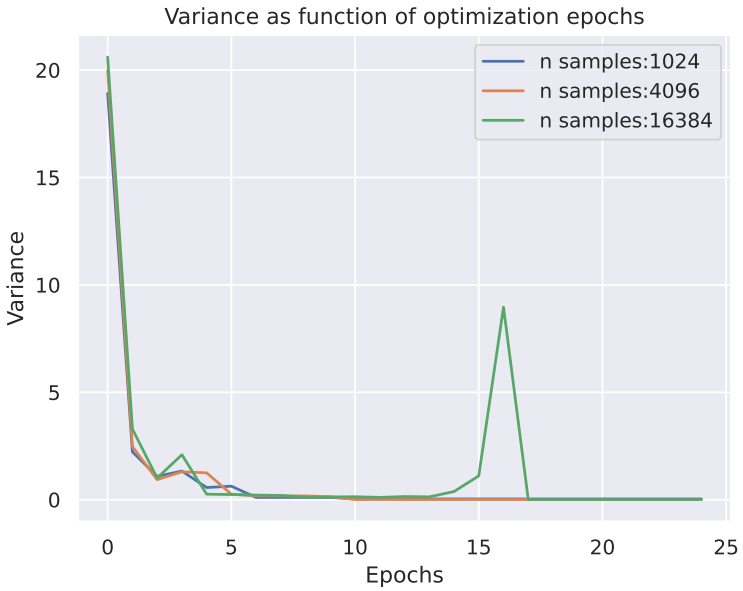

Figure 3: Variance of the Ising model as a function of the number of samples.

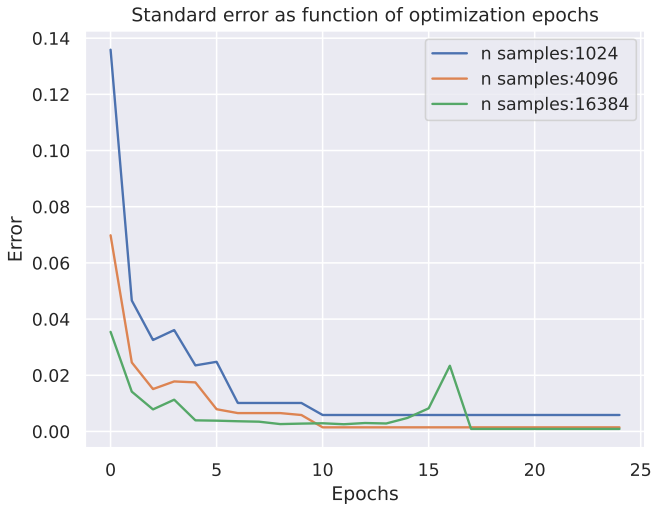

Figure 4: Standard error of the Ising model as a function of the number of samples.

4.1.2 RBM complexity

The complexity of the RBM is also a crucial parameter to consider, even if the hidden layers are not directly visible in the probability distributions, due to use tracing them out - but they are still important for the learning process. More hidden units will introduce more parameters to optimize, and the learning process will be more complex. In figure(6), we see the energy of the Ising model as a function of the number of hidden units in the RBM. Here we see an interesting result. The RBM performs much better for a simpler model, with fewer hidden units. This is likely due to the fact that the model is too complex, and the problem itself is quite simple - and the model is trying to overfit the data. We see that both 4 and 12 hidden units converge to the ground state energy, but with 12 hidden units the convergence is much quicker. This seems to hit the ”sweet spot” for model complexity - but this is a tradeoff - as the model complexity increases, the learning becomes more computationally demanding.

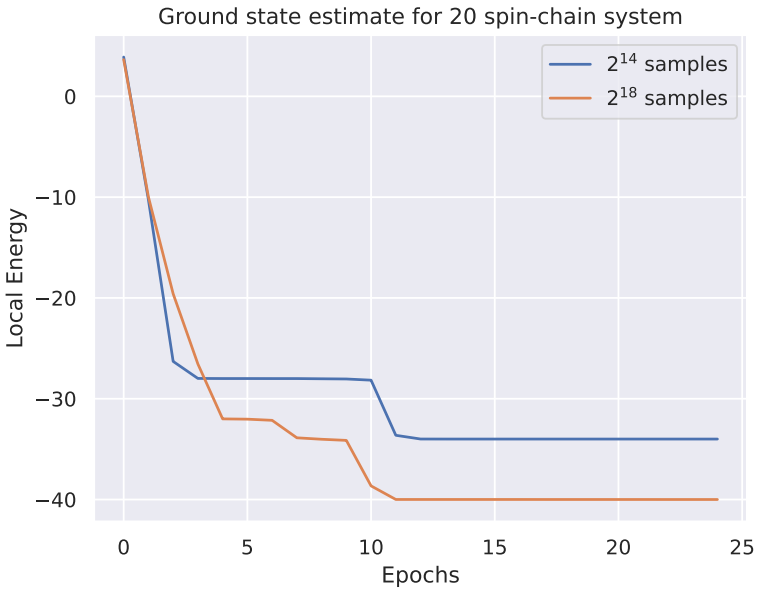

Figure 5: Energy of the Ising model as a function of the number of samples for a larger spin-chain.

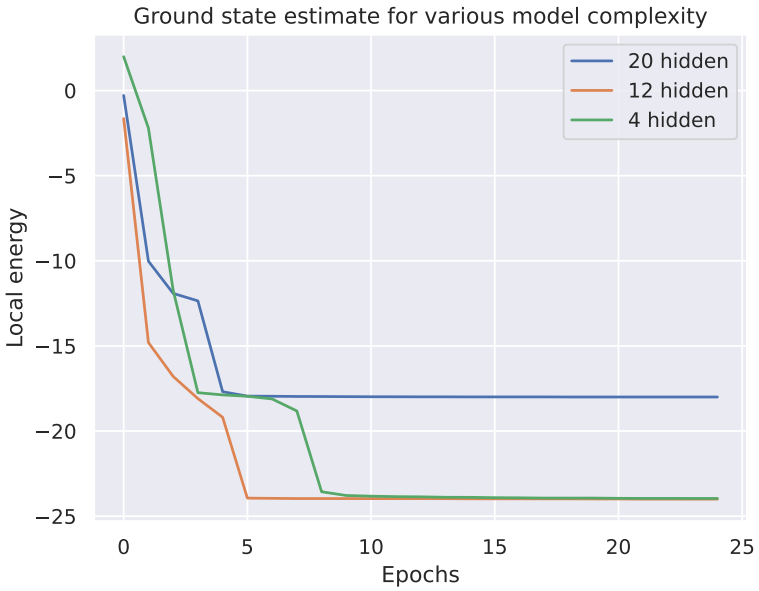

Figure 6: Energy of the Ising model as a function of the number of hidden units in the RBM.

4.1.3 Varying the temperature

The temperature of the system is also an important parameter to consider. We stated that we usually ignore this (see (2.2)), but it is still interesting to see how the RBM performs for different temperatures. In figure (7), we see the energy-convergence of the Ising model as a function of the temperature, for 50 epochs. Here we see that the RBM performs well for lower temperatures, but that the convergence *rate* is affected more and more as the temperature rises. At $T = 10.0K$, the model does not converge to the ground state within the 50 epochs, but it does not seem that it is stuck in a local minima, rather, it seems that the rate of convergence is too slow. This is likely due to the fact that the system is too "hot" to properly converge to the ground state, and needs more time to properly "find" the ground state.

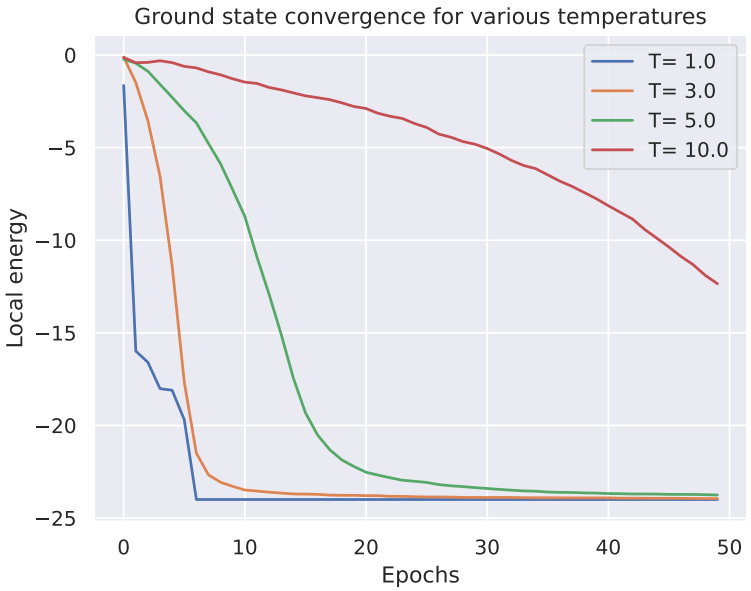

Figure 7: Energy of the Ising model as a function of the temperature.

4.2 Long-range interactions

Adding long-range interactions, it is interesting to see how this affect the ground state of the system. Following the implementations in section (3.1), we can see how the energy of the system changes as a function of the range of the interaction, n_v, for a constant $\alpha = 1.0$. The result of adding more complex interactions are presented in figure (8).

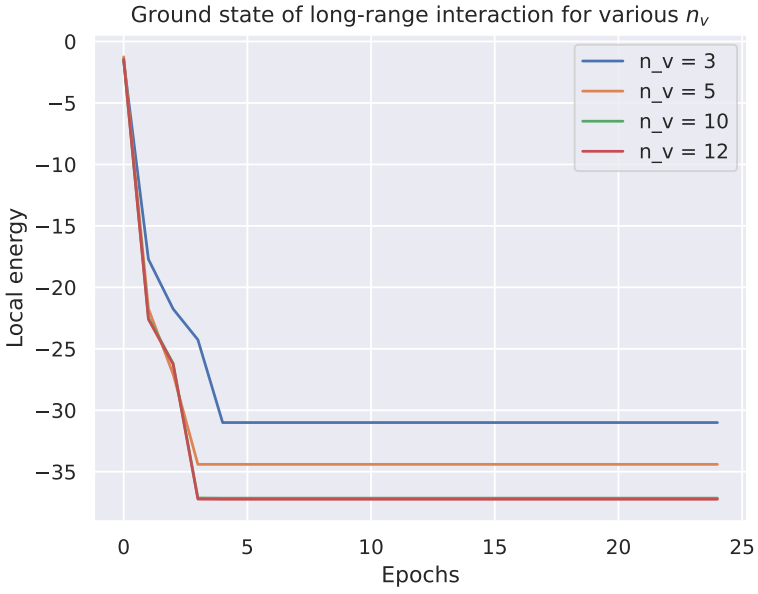

Figure 8: Energy of the Ising model as a function of the range of interaction.

5 Conclusion

We've successfully implemented a Restricted Boltzmann machine to solve the 1D Ising model, by clever usage of the NumPy library and Monte-Carlo sampling. We've shown that the RBM can be used to approximate the ground state energy of the Ising model, and that the model is capable of learning the ground state energy of the system. In our analysis, we found that the model is sensitive to it's hyperparameters, which is to be expected for any neural network - but we have also shown how important it is to have a large enough sample size for the Monte-Carlo sampler to properly traverse the spin-configuration landscape. The number of hidden units also play a crucial role in the learning process, where a too complex model will leave our RBM stuck trying to learn patterns that may not exist. We've also shown that the temperature of the system is an important parameter to consider, as the rate of convergence is affected by the temperature, and this further provides proof for why the temperature is usually ignored in RBM's for the Ising model (set to $T = 1$).

Adding long-range interactions changes the ground state energy, but the RBM is still capable of learning the ground state energy of the system. This is a promising result, as it shows that the RBM can be used to solve more complex systems than the simple Ising model. The 1D Ising model ground state is still the aligned spin-configuration, and even with the more complex Hamiltonian, our machine was successful in identifying this.

Further work could include implementing a 2D Ising model, where phase transitions are present, and the ground state is not as trivial to find. This would be a more challanging problem, but with our current results, there does not seem to be any reason for why the RBM would not be able to solve this problem as well. Adding even more dimensions would be an interesting problem to consider, as the complexity of the system increases, and the RBM would have to learn more complex patterns, and there might be the need for even more layers - extending the RBM to a Deep Belief Network.

References

[1] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325 (2017), pp. 602–606.

[2] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[3] Morten H. Jensen. "Computational physics II: Lecture notes spring 2024". In: 2024. Chap. 11.

[4] Morten H. Jensen. "Computational physics: Lecture notes fall 2015". In: 2015. Chap. 12.

[5] JG Martınez-Herrera, Omar Abel Rodrıguez-López, and MA Solıs. "Critical temperature of one-dimensional Ising model with long-range interaction revisited". In: *Physica A: Statistical Mechanics and its Applications* 596 (2022), p. 127136.