Generative Learning of the two-dimensional Ising Model Near Criticality

CHE696 Final Project

Jacques Esterhuizen
Department of Chemical Engineering
University of Michigan
Ann Arbor, MI
esterhui@umich.edu

Frank Doherty
Department of Chemical Engineering
University of Michigan
Ann Arbor, MI
fdoherty@umich.edu

Nathan Ng
Department of Chemical Engineering
University of Michigan
Ann Arbor, MI
nsng@umich.edu

April 24, 2019

Project Summary

In this project, we aimed to recreate some of the results presented in Morningstar and Melko's paper *Deep Learning the Ising Model Near Criticality* [1]. In this paper, they pose the question of whether the unsupervised learning of generative neural networks can capture the physical probability distribution underlying the two-dimensional Ising spin model. They test this assumption by implementing a number of different neural network architectures, those being the restricted Boltzmann machine, the deep Boltzmann machine, the deep restricted Boltzmann machine, and deep belief networks. Herein, we reproduce a few of their results by implementing the Metropolis-Hastings Monte Carlo algorithm for generating training data, and then using said data to train restricted Boltzmann machines and deep belief networks. We compare the ground truth statistics generated from Monte Carlo to the statistics generated from restricted Boltzmann machines and deep belief networks. Our generated results generally support Morningstar and Melko's claim that shallow networks may be more efficient than deep networks at representing the Ising model's physical probability distribution, as our single layer restricted Boltzmann machine outperformed our two layer deep belief network.

***K*eywords** Deep learning · Statistical physics

1 Introduction

In the early days of the 20th century, the physics world was still hotly debating the nature of matter at its smallest size. Atomic theory was accepted by many, but the framework for relating microscopic behaviors of atoms to a material's observable macroscopic properties was still being actively developed. The role of statistical mechanics was to provide such a

framework, where thermodynamics could be explained through simplified models that relied solely on statistics and mechanics. Explaining the phenomenon of phase transitions was one debate that atomic theory could explain well, where the slight changes of atomic properties in a narrow temperature range could result in a drastic change of macroscopic phase. The Ising model is well known for achieving this behavior, where a lattice of "atomic" sites can exhibit a phase transition behavior relative to changing temperature.

The Ising model is robust in its ability to calculate a physical probability distribution, able to relate properties such as energy and heat capacity to temperature. An issue lies with its scalability, since the number of possible configurations grows as 2^N for a system of N binary lattice sites. Sampling this state space becomes increasingly difficult as it grows exponentially, and many statistical physicists today are examining the efficiency of machine learning to sample this physical probability distribution. Here we take inspiration from Morningstar and Melko to test out several generative neural network architectures for sampling the Ising model state space [1].

2 Methods

In this work, we used Metropolis-Hastings and Wolff Monte Carlo updates to simulate 10^4 realistic Ising spin configurations for 20 different temperatures. The resulting spin configurations for each temperature were used for training a restricted Boltzmann machine (RBM) and a deep belief network (DBN). The trained models were then used to generate 10^4 new Ising spin configurations. The spin configurations generated via Monte Carlo were compared to the spin configurations generated via our neural network models by comparing the distributions of expected energy ($\frac{\langle E \rangle}{N}$) and heat capacity ($\frac{\langle C \rangle}{N}$) per Ising spin for the generated samples.

2.1 The Ising Model

The Ising model consists of a n-dimensional lattice of discrete spin states, with spin values of either +1 or -1. In this work, we will consider the 2-dimensional Ising lattice of size 8×8. Positive spins will be represented as black pixels and negative spins as white pixels. A representative 8×8 2-D Ising model is shown in Figure 1.

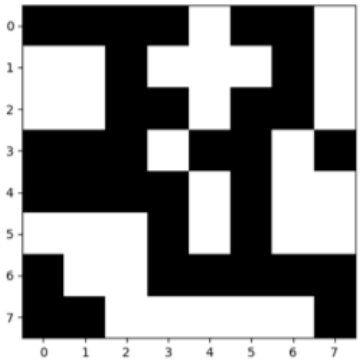

Figure 1: An example configuration of the 8×8 2-D Ising Lattice used in this work.

2.1.1 The Ising Hamiltonian

Consider N spins $s_i = \pm 1$ located on a $\sqrt{N} \times \sqrt{N}$ two-dimensional lattice. The Ising spin-Hamiltonian $\mathcal{H}$ for inter-spin interaction strength J and external magnetic field h acting on spins s_i is given by

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j - h \sum_{i=1}^{N} s_i \tag{1}$$

The first term in the equation describes the cumulative interaction strength between nearest neighbor spins, and the second term in the equation describes the interaction strength of every spin with the applied external magnetic field.

In this work, we consider a ferromagnetic system such that $J > 0$, which means that spins will align at low temperatures. Additionally, we do not consider the presence of an external magnetic field (i.e., $h = 0$), and thus $\mathcal{H}$ reduces to

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} s_i s_j \tag{2}$$

It naturally follows that the canonical partition function $\mathcal{Q}$ for the Ising spin Hamiltonian is given by

$$\mathcal{Q} = \sum_{\{s_k\}} e^{-\beta \mathcal{H}} = \sum_{\{s_k\}} e^{\beta J \sum_{\langle i,j \rangle} s_i s_j} \tag{3}$$

where $\beta = \frac{1}{k_B T}$ and $\{s_k\}$ denotes the summation over all possible lattice configurations (2^N terms). This summation becomes intractable very rapidly, making predicting macroscopically observable properties via expectation values difficult. For example, the size of the set $\{s_k\}$ is $\sim 10^{19}$ for an 8×8 lattice. One way to efficiently estimate the macroscopically observable properties of the system without evaluating the partition function is to use Monte-Carlo simulation.

2.1.2 Metropolis-Hastings Algorithm

We implemented the Metropolis-Hastings algorithm, named after Nicholas Metropolis and W.K. Hastings [2, 3]. The configuration of the two-dimensional Ising spin-lattice can be modeled as a Markov chain. Markov chains are stochastic processes in which the probability of attaining a given state is only dependent on the immediately previous state and the transition probabilities. For simulating the spin lattice, we define the transition probability $W_{t,t+1}$ from the current state t to the next state $t+1$ as

$$W_{t,t+1} = \begin{cases} 1 & \Delta E \leq 0 \\ e^{-\beta \Delta E} & \Delta E > 0 \end{cases} \tag{4}$$

where $\Delta E = \mathcal{H}_{t+1} - \mathcal{H}_t$. Then, the Metropolis-Hastings algorithm can be run as follows:

1. Choose one spin at random, and flip it.
2. Evaluate the change in energy ΔE associated with flipping the spin.
3. Accept the change according to the Markov transition probabilities. That is, if $\Delta E \leq 0$ always accept the change. If $\Delta E > 0$, generate a random number. If the random number is less than $e^{-\beta \Delta E}$, accept the change.
4. Repeat steps 2-4.

A schematic representation of the Metropolis-Hastings update is shown in Figure 2.

Figure 2: Schematic depiction of the Metropolis-Hastings update step.

2.1.3 Wolff algorithm

We implemented the Wolff algorithm, introduced by Ulli Wolff in 1989 [4]. One can imagine that at low temperatures, moves with $\Delta E > 0$ become highly improbable as the transition probability trends to 0. Therefore, at low temperatures Metropolis-Hastings Monte Carlo can get stuck in local configurations where only entirely spin-up or spin-down configurations are sampled. In reality, spin-up and spin-down configurations are equally as likely to occur when there is no external magnetic field applied. The Wolff algorithm allows us to sample non-local configurations by identifying large clusters of like spins, and flipping them simultaneously. The Wolff algorithm can be run as follows:

1. Choose one spin at random.
2. Study all 4 of the neighbouring spins, s_j. If $s_j = s_i$, then combine spin s_j with s_i to form a cluster with probability $p = 1 - e^{-2\beta}$.
3. Repeat 2 for all of the terminal spins in the cluster, causing the cluster to grow in size.
4. Once the cluster can no longer grow, flip the spins for the entire cluster.

A schematic representation of the Wolff update is shown in Figure 3.

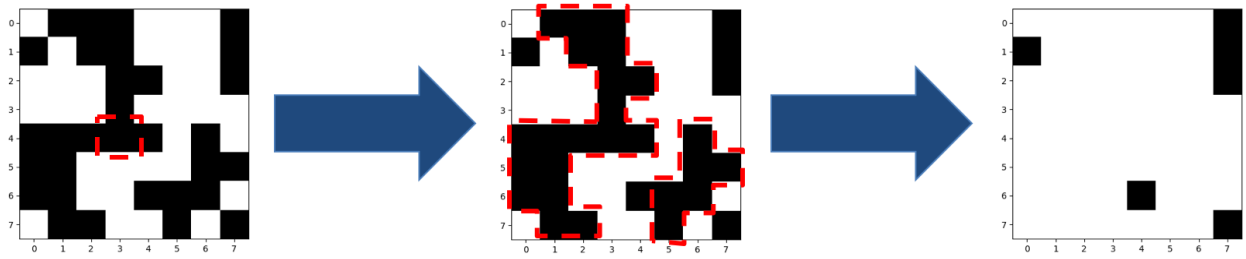

Figure 3: Schematic depiction of the Wolff update step, where a single site is chosen, grows into a cluster, and has its spin flipped.

2.1.4 Generating training data

We generally followed the same procedure as Morningstar and Melko for using Monte-Carlo simulation to produce training data, working with an 8×8 lattice. One Monte Carlo step consists of one Wolff update followed by 64 Metropolis-Hastings steps. Upon randomly initializing the lattice, 262,144 Monte Carlo steps were run to allow the spins to equilibrate before sampling began. Between samples, 64 Monte Carlo steps were run.

2.1.5 Calculating Physical Properties

We defined physical properties, namely the average energy and the heat capacity for use in comparing our data. In this work, the energy is given by

$$\langle E \rangle = Z^{-1} \sum_{x} q(x)\mathcal{H}(x) \tag{5}$$

Solving and simplifying heat capacity also gives us that

$$\langle C \rangle = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2} \tag{6}$$

2.2 Generative Models

Having set up Monte Carlo simulations to produce samples, we used two distinct generative neural network architectures to predict the underlying probabilities. We hypothesized that these neural networks would be able to generate samples with similar macroscopically observable properties, specifically focusing on the expected energy ($\frac{\langle E \rangle}{N}$) and heat capacity($\frac{\langle C \rangle}{N}$) per Ising spin. The neural network architectures we used were the restricted Boltzmann machine and the deep belief network. We chose these algorithms to enable direct comparison to Morningstar and Melko's results. However, it is worth noting that both of these neural network architectures have somewhat fallen out of favor, in large part due to the successes of generative adversarial networks. Implementation of both learning algorithms were programmed in Python using NumPy. For hyperparameter tuning, we simply chose the hyperparameters given by Morningstar and Melko.

2.2.1 Restricted Boltzmann Machine

The restricted Boltzmann machine (RBM) is an unsupervised learning algorithm that belongs to the class of neural network architectures known as generative models. As seen on the left in Figure 4, the RBM architecture consists of two layers of stochastic binary nodes, linked by symmetric weights. One layer of the nodes is called the visible layer, which corresponds to the observable properties which the model generates (i.e., the Ising lattice configuration). The second layer of nodes is called the hidden layer, which, together with the weights, represents the probability distribution over the observable properties. The two layers are fully inter-connected, and there are no connections within each layer. Each node is binary, with a Bernoulli probability of activating given by a normalized sigmoid activation function.

The goal of training is to learn the join probability distribution over the hidden and visible nodes, $P(h, v)$. In order to accomplish this, we used 100 epochs of the contrastive divergence algorithm, developed by Hinton, for training our RBM [5]. New samples were generated by initializing the visible nodes to one of the training configurations, and then performing iterative Gibbs sampling 10^4 times, which corresponds to passing the data forwards and backwards through the neural network in a Markov chain (i.e., $v_0 \rightarrow h_0 \rightarrow v_1 \rightarrow h_1 \rightarrow ...$).

2.2.2 Deep Belief Network

A deep belief network (DBN) can be considered to be a deep extension of the RBM. Therefore, the DBN is similarly an unsupervised learning algorithm which belongs to the class of neural network architectures known as generative models. As seen on the right in Figure 4, the first two layers (on the hidden side) operate as a restricted Boltzmann machine, with symmetric connections. After these first two layers, the remaining layers operate as a normal feed-forward neural network. Similar to the RBM, each node is binary, with a Bernoulli probability of activating given by a normalized sigmoid activation function. For our study, we worked with one additional hidden layer, leading to the 64, 24, 24 architecture observed in Figure 4.

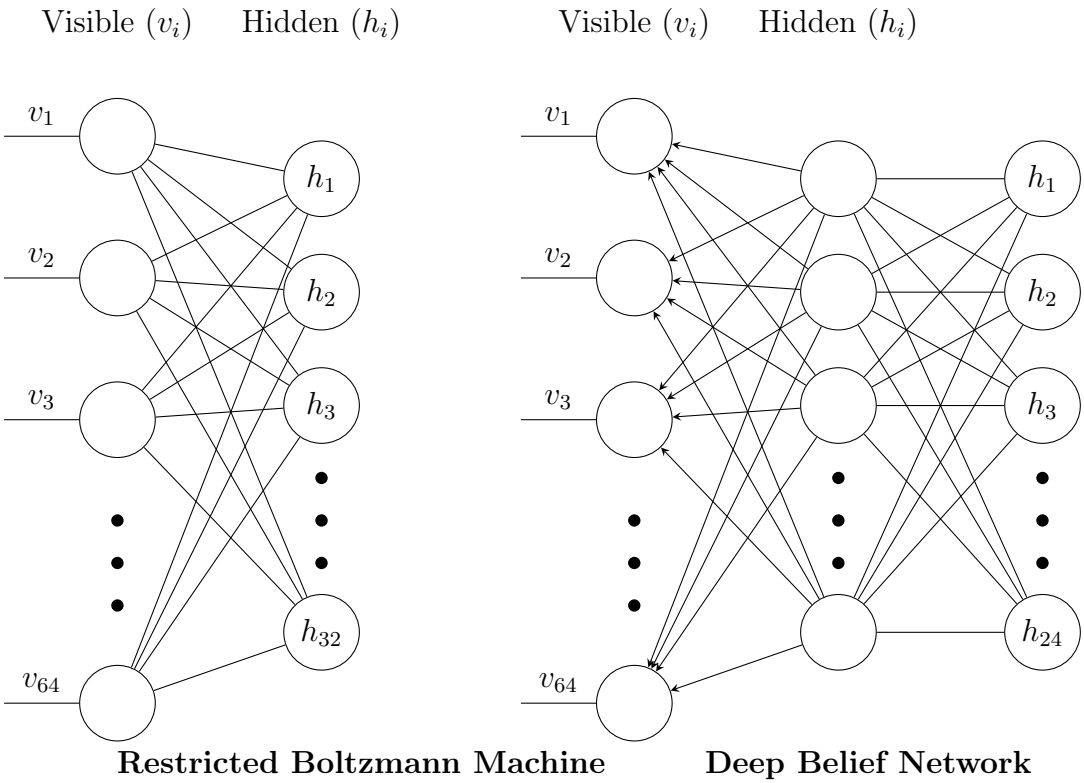

Figure 4: The two neural network architectures used in this work. On the left, the 64, 32 RBM used is shown. On the right, the 64, 24, 24 DBN is shown. h corresponds to hidden nodes and v corresponds to visible nodes.

Greedy layer-wise training was used for each DBN, with each pair of layers treated as a restricted Boltzmann machine. Each layer was trained with 100 epochs of the contrastive divergence algorithm [5]. The overall training procedure for DBN's was developed by Hinton *et. al.* [6]. New samples were generated by initializing the hidden nodes to a random configuration. Twenty forward and backward passes were then performed in the bidirectional nodes, at which point the resulting hidden node configuration was propagated forward through the network to create a sample.

2.3 Results

Upon training our two neural network architectures using the lattice configurations sampled from our Monte Carlo simulations at each temperature, we then wanted to compare the macroscopically observable properties for the samples derived from each neural network to the samples derived via Monte Carlo simulation.

2.3.1 Average Energy

On the whole, both the RBM and DBN performed well in predicting the average energy as shown in Figure 5. Both algorithms struggled as temperature increased, but each captured the general trend. Notably, there were deviations in both algorithms observed at the inflection point at the critical temperature $T_c = \frac{2}{log(1+\sqrt{2})} \approx 2.27$.

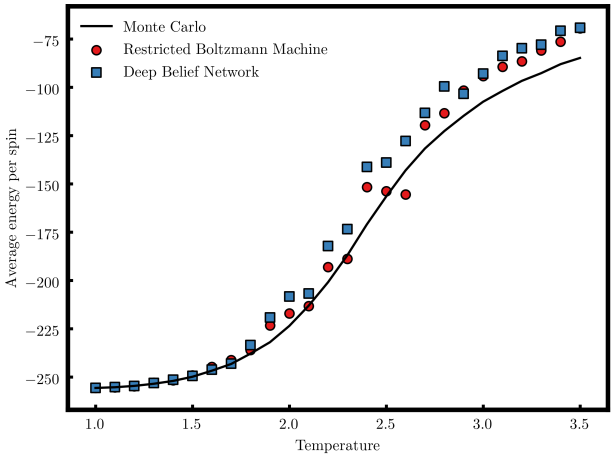

Figure 5: Plot qualitatively showing the agreement between the energies calculated from the configurations sampled from each method. Both the RBM and DBN appear to perform fairly well at most temperatures, with some slight perturbations near the critical temperature ($T_C \approx 2.27$), and a systematic overestimation at higher temperatures.

2.3.2 Heat Capacity

With regards to the heat capacity, both the RBM and DBN appeared to perform well at high and low temperatures far from the critical temperature, as seen in Figure 6. Near the critical temperature, the performance of both neural network architectures breaks down. The RBM had slight issues, and the DBN fared even worse in its predictions, tending to over-predict.

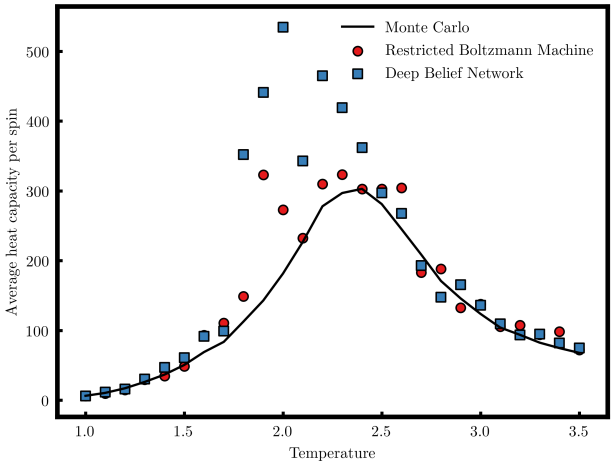

Figure 6: Plot qualitatively showing the agreement between the heat capacities calculated from the configurations sampled from each method. While both the RBM and DBN appear to perform fairly well at both high and low temperatures, the heat capacities becomes erratic near the critical temperature ($T_C \approx 2.27$). The DBN performs particularly poorly, and appears to be worse than the RBM.

To diagnose the poor performance of the RBM and DBN for recreating the heat capacity, we plotted the energy distribution as a histogram. These results confirm our theory that both the RBM and DBN created incorrect energy distributions, despite the correct average. While we would expect a Boltzmann distribution as in the Monte Carlo simulations, the

RBM and DBN both produced larger energy tails. Since the heat capacity is proportional to the variance, the larger energy spread explains the high heat capacities produced.

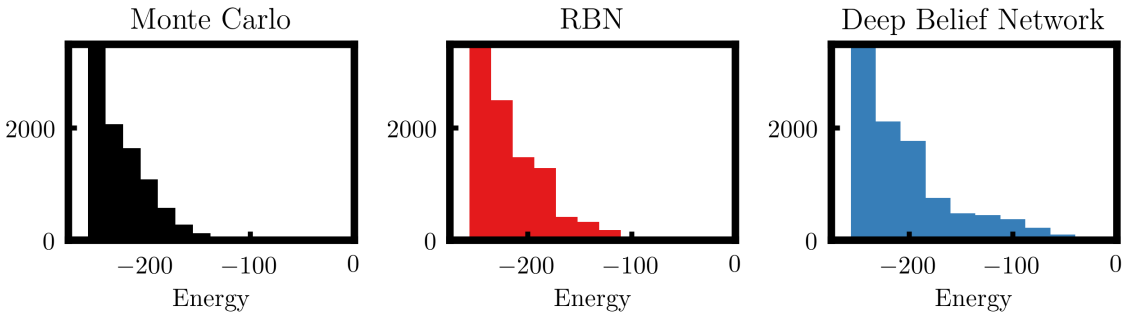

Figure 7: Histograms showing the energy distribution for the configurations sampled from each method. The samples generated from the Monte Carlo simulation form a tight Boltzmann distribution. The RBM and DBN samples also generally form a Boltzmann distribution but with more high energy states, with the DBN having a much larger fraction of high energy states.

3 Conclusions

The main conclusion of Morningstar and Melko was that shallow RBM's are better for reproducing the macroscopically observable properties of the Ising model, and our results appear to corroborate their finding. The RBM has comparable performance to the DBN for energy prediction, but much better accuracy for heat capacity prediction. In addition to its improved accuracy, the RBM trains much faster than the DBN.

Potential avenues for future investigation include examining the effect of training time on the prediction of properties, and potentially using the RBM to decorrelate samples in a hybrid RBM-Monte Carlo framework.

Code Availability

Code is available at https://github.com/jesterhui/ising_nn.

References

[1] Alan Morningstar and Roger G. Melko. Deep Learning the Ising Model Near Criticality. *arXiv:1708.04622 [cond-mat, stat]*, August 2017. arXiv: 1708.04622.

[2] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[3] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[4] Ulli Wolff. Collective Monte Carlo Updating for Spin Systems. *Physical Review Letters*, 62(4):361–364, January 1989.

[5] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Comput.*, 14(8):1771–1800, August 2002.

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