Utrecht University

MASTERS THESIS

Machine Learning Classical Spin Models

Author:
Ryan VAN MASTRIGT

Supervisors:
dr. L. FRITZ
dr. D. PANJA

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

in the

Institute for Theoretical Physics
Department of Physics and Astronomy

&

Centre for Complex Systems Studies

June 27, 2019

UTRECHT UNIVERSITY

Abstract

Faculty of Science
Department of Physics and Astronomy

Master of Science

Machine Learning Classical Spin Models

by Ryan VAN MASTRIGT

Machine learning has become increasingly popular as a computational tool in all aspects of science and the private sector, however its application as an additional computational physics tool has only recently began to gain traction. This thesis aims to study the application of supervised and unsupervised neural networks to computational classical statistical physics problems, focusing on what is learned and whether it is a useful tool for these applications. Supervised neural networks are used to distinguish between the different phases of four different models, containing a second-order phase-transition (PT), infinite-order PT, and both in close proximity to one another, as well as a model without a PT but with frustration. Different iterations of the restricted Boltzmann machine (RBM), a type of unsupervised neural network, are trained on the one- and two-dimensional Ising model. The neural network is able to differentiate between the different phases only if the PT is clearly discernible from the input configurations. It is too crude a tool to differentiate between two close subsequent PTs. It is concluded that application of neural networks to detect PTs in classical statistical physics models where an intuition for the PT exists, holds no advantage over alternative conventional computational methods. This thesis concludes that restrictions placed on the RBM, such that the RBM has translation invariance, still allow the restricted RBMs to learn the magnetisation really well, while two-spin correlations are learned less well. Surprisingly, the block Gibbs sampling of the restricted RBMs is better behaved than for the unrestricted RBM, as can be explained using an analysis of the trained weight-matrix.

Acknowledgements

This thesis would not have been completed if it were not for the support of all the wonderful people in my life. First and foremost I would like to thank my supervisors Lars Fritz and Deb Panja for the fruitful discussions and helpful advise over this past year. They made sure to give me direction whenever I felt lost, and I am grateful for their support.
I would also like to thank Mikael, Sonja, Kitinan, Georgia, Tycho, Gerwin and Michał for the always interesting weekly meetings, providing some welcome diversity in topics and subjecting me to more aspects of condensed matter unrelated to my thesis. Special shoutout to Sonja for organising the weekly pizza-seminar, they were always delicious.
My gratitude also goes out to the students of *De Ivoren Toren* for their wonderful company over the last year, as well as the shared sense of sorrow over these last few stressful months. Thank you for providing me with a sense of community through the sometimes lonely process of writing a thesis.
My parents and brother have my eternal gratitude for calmly listening to me complain whenever I felt stuck, and for tolerating me whenever I felt cantankerous from the stress. Thank you for letting me vent whenever I needed venting and for giving me advice whenever I needed advising.
Finally I would like to thank my amazing friends from the *Helaasheid der Pindakaas*. Thank you for being there for me whenever I needed someone to talk to, or whenever I really needed to take my mind off of something. You provided me with an healthy dose of distractions, reminding me that there is more to life than academia. I feel incredibly lucky to have had all of you as my friends for almost a decade now, and I would not have had it any other way. I love you all dearly.

Contents

Chapter 1

Introduction

Though machine learning has now ingrained itself in the public consciousness with the rise of its commercial success over the past decade, it has been an intensively studied academic subject since the 1950's. One of the first learning machines which captured the attention of the academic community was built as early as 1954 by Marvin Minsky[1]. Research really took of from there, propelled even further with the invention of the neuron-inspired perceptron[2] in 1958. This was the first real link between neuroscientific concepts and mathematical networks, sparking the initial inspiration for many more years of research to come. Over the next few years much progress was made, including making a machine which could play the game of tic-tac-toe[3]. Ironically it was Minsky himself who set the stage for the first artificial intelligence (A.I.) winter by the publication of his book about perceptrons[4], co-authored by Papert Seymour. The book provided a mathematical introduction to the field of perceptron-based learning, and presented a quite critical view of the limitations of these type of machine learning algorithms. This led to a decade-long recession in machine learning research, now referred to as the first A.I. winter. It was in this recession that Finnish computer scientist Seppo Linnainmaa[5] laid the foundation for one of the central algorithms in machine learning, the backpropagation algorithm. Unfortunately, the importance of his research went unnoticed for many years.

The interest in machine learning was rekindled in the 1980's, where the foundations for different kinds of machine learning algorithms still in use nowadays were laid. For example, convolutional neural networks, monumental in classifying pictures, are ultimately based on an initial design by Fukushima[6] in 1980. The revival of interest in neural networks by the mainstream was mainly due to the rediscovery of the backpropagation algorithm[7]. This allowed the neural network to learn much faster than before, which allowed for application of machine learning on personal computers. This paved the way for the commercialisation of machine learning, which led to the first machine learning package for commercial markets in 1989, evolve[8]. From that moment, machine learning really takes off. From beating the world champion in chess in 1997[9] to beating the world champion in Go in 2016[10], machine learning has made heaps of progress. It has now become a part of our everyday lives, from giving you recommendations on what to watch next[11] to automatically organising your pictures based on their contents[12].

1.1 Machine learning and physics

Given the success of machine learning in many different areas of science, it is natural to ask whether machine learning can be of any use in physics. Its application in classifying large amounts of data by finding the underlying structure is similar to

the goal of the theoretical physicist. Ultimately any physical theory should be able to describe and explain the results from experiments. Unfortunately the link between the two is not always as clear-cut. Theoretical derivations are limited by the analytic mathematical tools to our disposal, which often prove insufficient to describe reality exactly. The theorist is forced to make informed simplifications, filtering out small irrelevant effects, and to take limits.

To circumvent part of the limitations of analytic derivations, numerical techniques can be used. Sometimes the numerical technique simply consists of evaluating integrals unsolvable by hand to obtain estimates for physical observables. In condensed matter physics one often deals with complex models of many interacting particles. While the underlying theoretical structure of how each pair of particles should interact is usually known and well-studied, the up-scaling of this behaviour to a many-body problem can greatly temper analytic derivation. There are theoretical tools to deal with many-body problems, but exact solutions are usually limited to a select group of models. However, numerical techniques can be used to simulate these other groups of models. The Monte Carlo method is often used to simulate these kinds of systems[13], and has helped in confirming analytic predictions of certain models and providing insight in systems resistant to analytic techniques.

Just as the physics community was quick to embrace Monte Carlo techniques, machine learning has now garnered an interest within all parts of the community. It can be used a tool to discriminate between phases in quantum mechanical systems[14–17]. It is also possible to construct neural networks in such a manner that they can represent quantum-mechanical groundstates[18–23]. Machine learning has been applied to classical statistical physics models as well. It is possible to classify different phases of the model[24–29], or to learn to model the underlying structure of the physical system[30–33]. There are applications in active matter[34], quantum computing[35, 36], particle physics[37], and physics can even offer insights into the fundamental aspects of machine learning[38].

1.2 The goal of this thesis

The goal of this thesis is to provide a critical investigation into the use of machine learning tools in classical statistical physics models. Specifically, the use of neural networks to locate and learn phase transitions is analysed with regard to blind application and physical interpretation of the results. A generative neural network's ability to encode the physical properties of the Ising model is studied in detail as well. This serves to provide a critical picture of machine learning tools in physics, and to separate the hype of a novel numerical technique from the actual usefulness of the technique.

The application of neural networks as a tool to locate phase transitions is analysed by studying the application of feed-forward neural networks, convolutional neural networks and the confusion-scheme network to the two-dimensional Ising model on a square lattice, the two-dimensional xy-model, the two-dimensional anti-ferromagnetic Ising model on a triangular lattice, and the two-dimensional Coulomb gas on a square lattice. The neural networks are used to classify the different phases of the Ising model, the xy-model, and the Coulomb gas model. The anti-ferromagnetic

Ising model on a triangular lattice is a model without a phase transition and is studied as an example for dealing with degenerate groundstates and frustration.

Earlier studies have shown that neural networks are able to learn the phase transition in the Ising model[25, 28], and the xy-model[24]. These studies are repeated, including an in-depth analysis of different neural network topologies and how that affects the results of the neural network. The learned behaviour of the neural network when faced with a system with degenerate groundstates and frustration is studied by training the network as a thermometer. The application of the network will then show how well the network is able to pick up on subtle changes in the configurations, for example by training the convolution layer such that the subsequent hidden layer looks at a configuration directly corresponding to frustrated and unfrustrated triangles. The application of neural networks as a way to get an estimate for the temperature of a given configuration was used in earlier research[31], although it was not the main focus of the paper. Finally, different types of neural networks are applied to the two-dimensional Coulomb gas model, which has not been studied with neural networks before to the knowledge of the author at the moment of writing. This model is said to have both an Ising-like and KT-like phase transition, which should be detectable according to earlier research. Both the feed-forward and convolutional neural network are applied to the system. In addition, an unsupervised variation on the conventional neural networks, the confusion-scheme neural network, is applied to the model as well. The confusion-scheme neural network was shown to work on the Ising model[27], as well as on systems with quasi long range order (QLRO) and multiple phase transitions[29].

Another application of machine learning, namely to learn the underlying probability distribution of the physical system, is studied by training a restricted Boltzmann machine (RBM) on the one-dimensional and two-dimensional Ising model. Numerical studies training an RBM on configurations of the two-dimensional Ising model have been done before[31, 32]. Instead of repeating these results we instead focus on a more analytic derivation of the trained RBM by minimising the KL-divergence directly, as opposed to using conventional numerical techniques like contrastive divergence[39] as in the aforementioned papers. Specific restriction are placed on the RBM to enforce symmetries present in the Ising model, such as the global spin-flip symmetry and translation invariance. The behaviour of different iterations of the RBM are then analysed and compared to show how well the RBM is able to capture the Ising model, and what physical aspects it picks up on the most.

The layout of this thesis is as follows. Chapter 2 provides a short introduction to the concepts of machine learning necessary to understand the contents of this thesis. No prior experience with machine learning is expected. Chapter 3 gives a theoretical background to the physical models encountered in this thesis. Some derivations will be made to provide the reader with sufficient physical insight to understand the results of the machine learning. The results are included in chapter 4 and 5. Chapter 4 contains the results for the neural network analysis of phase transitions for the Ising model, xy-model, anti-ferromagnetic Ising model, and Coulomb gas model. Chapter 5 contains the results for training different RBM iterations on the Ising model. The results for both chapters are recapped and discussed in chapter 6.

Chapter 2

Introduction to Machine learning

In this chapter a brief introduction to the machine learning concepts relevant for this thesis is given. First the general concept of machine learning is discussed, after which follows a brief mathematical introduction to neural networks, continuing into adaptations to the basic neural network in the form of convolution layers and the confusion scheme. Another class of neural networks called Boltzmann machines are given a brief mathematical introduction, after which the more practical restricted Boltzmann machine is discussed.

2.1 A brief introduction to machine learning

The question of what machine learning is can best be answered by looking at the two constituents that make up the concept. A machine, while commonly associated with something physical, is in this context best understood as a function: an operation which takes a set of inputs and generates a single output. This function could for example be a probability distribution $g(\mathbf{x}|\mathbf{w})$, which is a function of the inputs $\mathbf{x}$, given the parameters $\mathbf{w}$. The learning aspect then comes in by relating this function to a desired outcome. Lets say we have a set of configurations $\mathbf{X}$ where each configuration has an associated value y' and we want the machine to predict the value y' when we put in a configuration of $\mathbf{X}$. How well the machine does this is measured by some cost function $\mathcal{C}(\mathbf{y}', g(\mathbf{X}|\mathbf{w}))$, which relates the outcome of the function g (and thus of the machine) to the desired output y'. The actual learning then comes in via the minimisation of this cost function, the values of the parameters $\mathbf{w}$ are changed such that the cost function is minimised. This minimisation process and updating of the parameters $\mathbf{w}$ is what is meant by learning.

In the field of machine learning there are generally considered to be two types of learning a machine can do: supervised and unsupervised.
Supervised learning means that the machine learns using labelled data, meaning that each configuration $\mathbf{x}$ put in the machine to learn needs to have an associated desired outcome y', i.e. label. The cost function of supervised machine learning models depends on these labels to correctly classify how well the machine is performing. An example of a supervised machine learning model is a feed-forward neural network.

On the other hand, unsupervised machine learning models do not need such an associated label for its inputs. The cost function is able to judge how well the machine is performing without needing any additional labels with the data. An example of an unsupervised machine learning model is the restricted Boltzmann machine.

2.2 Neural Networks

An artificial neural network is a nonlinear model for supervised machine learning, which was originally inspired by the neural networks seen in biological systems[40]. A neural network consists of a multitude of neurons connected to each other in a particular topology. The network itself is not a learning algorithm, but rather a framework for many different machine learning algorithms to work together and process data inputs.

The basic building block of neural networks is the neuron, referred to as a node in graphs. The neuron can be understood as a function which takes an input vector $\mathbf{z}$ and gives an output $a_i(\mathbf{z})$. This output is usually a scalar, although there are functions which give vectors as outputs. Figure 2.1 shows a schematic picture of such a neuron. The inputs of the neuron, $\mathbf{z}$, are represented by the arrows pointing towards the node. The form of the function a_i, referred to as the activation function, depends on the type of non-linearity used in the network. The inputs of neurons are generally weighted by their relevant importance using a linear transformation $\mathbf{z} = \mathbf{w}\mathbf{x} + b$, where the parameters $\mathbf{w}$ rescale the inputs $\mathbf{x}$, and b adds a bias-term. The output of the neuron can thereafter be used as new inputs for other neurons.

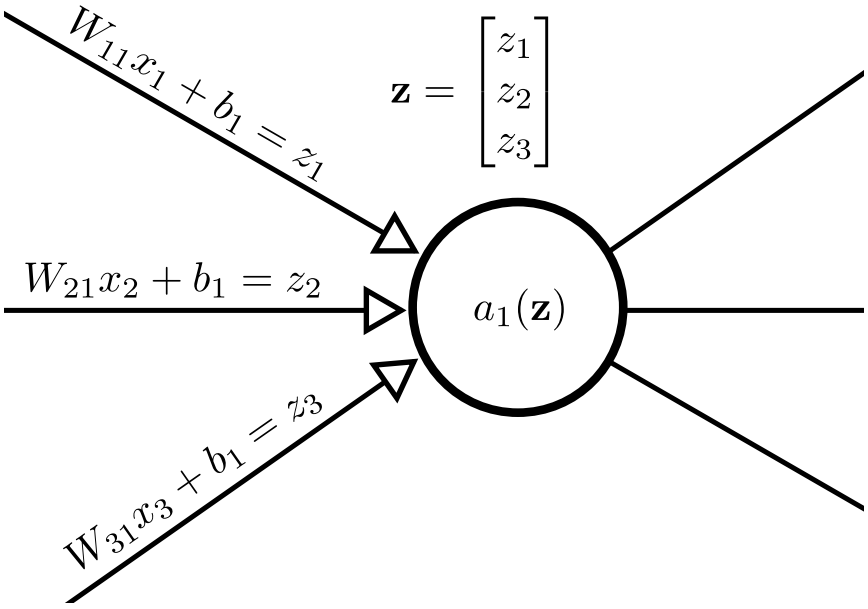


FIGURE 2.1: *Schematic picture of a neuron as used in neural networks.*

2.2.1 From neuron to network: the feed-forward Neural Network

One of the simplest and commonly used neural networks is the feed-forward neural network (FNN). It consists of just three layers of neurons, where a layer of neurons consists of a group of neurons which are not connected to other neurons in the same layer. All the neurons in the layer take their inputs from a single other layer and have their outputs connected to a single other layer. Since the network is directional, these layers are typically ordered from input to output. The first layer in the FNN is the input layer, which takes the form of the input configurations. Then comes the hidden layer, where each neuron takes all the linearly transformed inputs from the input layer and puts them through their activation functions. Finally we have the

output layer, where each output neuron also linearly transforms all the outputs of the hidden layer and puts them through their activation functions.

This can be made more clear in a mathematical form. We denote each layer by an upper index between brackets, for example the linear transformations fed into the activation functions of the i-th layer will be denoted as $\mathbf{z}^{(i)}$. If we have d-dimensional configuration $\mathbf{x}$ as input, we can write the input layer as a vector of the form

$$\mathbf{x}^{(1)} = (x_1, x_2, ..., x_d)^T. \tag{2.1}$$

The first linear transformation is denoted as

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{x}^{(1)} + \mathbf{b}^{(2)}, \tag{2.2}$$

where $\mathbf{W}$ is a $N^{(2)} \times N^{(1)}$ weight matrix with $N^{(i)}$ the number of neurons in the i-th layer. $\mathbf{b}^{(2)}$ is a $N^{(2)}$-dimensional bias vector. Each component of the $\mathbf{z}$ vector then gets acted on by the activation function of the corresponding neuron in the hidden layer:

$$\mathbf{a}^{(2)}(\mathbf{z}^{(2)}) = [(a_1(z_1), a_2(z_2), ..., a_{N^{(2)}}(z_{N^{(2)}}))^{(2)}]^T. \tag{2.3}$$

This gives us the output of the hidden layer

$$\mathbf{x}^{(2)} = \mathbf{a}^{(2)}(\mathbf{z}^{(\mathbf{2})}), \tag{2.4}$$

which then forms the input of the output layer. The output layer again transforms the input of the hidden layer

$$\mathbf{z}^{(3)} = \mathbf{W}^{(3)}\mathbf{x}^{(2)} + \mathbf{b}^{(3)}. \tag{2.5}$$

The vector $\mathbf{z}^{(3)}$ is the input for the activation functions of the output neurons so that we get the final output of the neural network

$$x^{(3)} = \mathbf{a}^{(3)}(\mathbf{z}^{(\mathbf{3})}) = \mathbf{y}. \tag{2.6}$$

We call the final output layer $\mathbf{y}$. This process is shown schematically in figure 2.2.

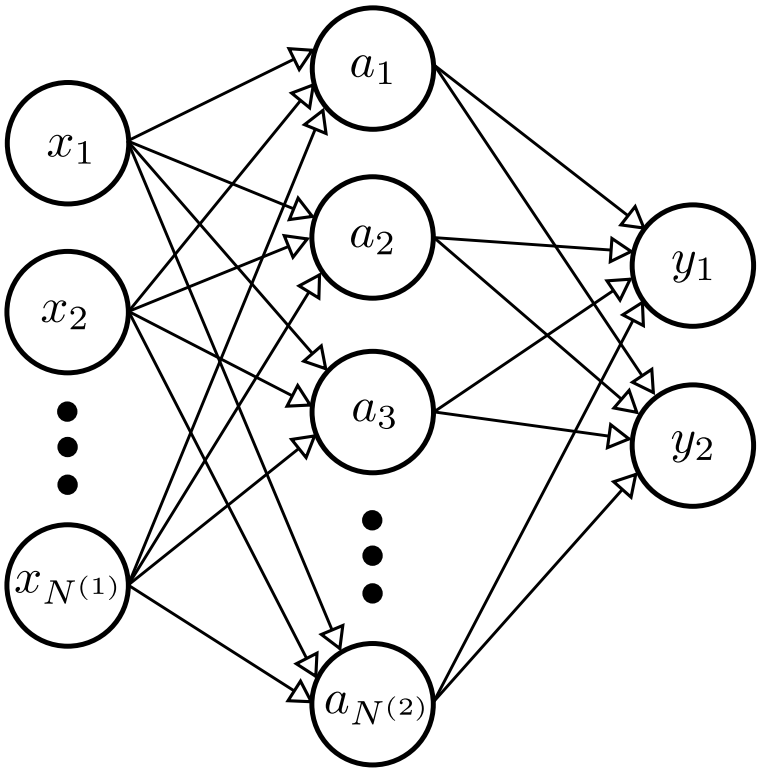


FIGURE 2.2: *Schematic graph of a feed-forward neural network. The arrows between nodes denote the linear transformations of the value of the nodes at the base of the arrows to be used as the inputs for the activation functions of the nodes at the head of the arrows.*

Training the feed-forward neural network

The learning procedure of a FNN is done through the backpropagation algorithm. The algorithm can be derived through simple partial differentiation and application of the chain rule. Here we make use of the same notation as before, but we denote the matrix multiplication more explicitly in terms of the individual components of the matrix. For example $w_{jk}^{(l)}$ denotes the weight for the k-th neuron in layer $l-1$ to the j-th neuron in layer l. In a general feed-forward neural network we can relate the outputs of the j-th neuron in the l-th layer to the outputs of the neurons in the $l-1$-th layer by

$$x_j^{(l)} = a_j^{(l)}(\sum_k w_{jk}^{(l)} x_k^{(l-1)} + b_j^{(l)}) = a_j^{(l)}(z_j^{(l)}). \tag{2.7}$$

The training of the neural then happens by minimising some predefined cost function $\mathcal{C}(\mathbf{y}, \mathbf{y}')$, which depends on the output layer $\mathbf{y}$ and the labels $\mathbf{y}'$ paired with the input data $\mathbf{x}$. The minimum of this function is found numerically by gradient descend, each learning step consists of moving through the neural network parameter space spanned by the $\mathbf{W}$-matrices and biases $\mathbf{b}$ towards a lower cost function. The direction of this step is determined by taking the derivative of each individual parameter with respect to the cost function.

To calculate expressions for all these derivatives we start with the derivative of the final layer, which we label layer L. Note that for the FNN $L = 3$, the derivation presented here is valid for deep neural networks with any number of layers. The error

$\Delta_j^{(L)}$ of the j-th neuron in the L-th layer is defined as the change in the cost function with respect to the weighted input $z_j^{(L)}$:

$$\Delta_j^{(L)} = \frac{\partial \mathcal{C}}{\partial z_j^{(L)}}. \tag{2.8}$$

Similarly, we can define the error $\Delta_j^{(l)}$ of neuron j in layer l with respect to the weighted input $z_j^{(l)}$ as

$$\Delta_j^{(l)} = \frac{\partial \mathcal{C}}{\partial z_j^{(l)}} = \frac{\partial \mathcal{C}}{\partial a_j^{(l)}} \dot{a}_j^{(l)}(z_j^{(l)}), \tag{2.9}$$

where $\dot{a}_j^{(l)}(x)$ denotes the derivative of the activation function of neuron j in the l-th layer with respect to its input evaluated at x. Since $\partial b_j^{(l)} / \partial z_j^{(l)} = 1$, equation (2.9) can also be interpreted as the derivative of the cost function with respect to the bias $b_j^{(l)}$.

Because the error of the neurons in layer l depends on the activation functions of neurons in the subsequent layer $l+1$, the chain rule can be used to write the error of neuron j in layer l in terms of the error of the neurons in layer $l+1$:

$$\begin{aligned}
\Delta_j^{(l)} &= \frac{\partial \mathcal{C}}{\partial z_j^{(l)}} \\
&= \sum_k \Delta_k^{(l+1)} \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} \\
&= \sum_k \Delta_k^{(l+1)} w_{kj}^{(l+1)} \dot{a}_j^{(l)}(z_j^{(l)}).
\end{aligned} \tag{2.10}$$

We need just one more equation to have expressions for the partial derivatives of all parameters. Differentiating the cost function with respect to the weight $w_{jk}^{(l)}$ gives

$$\frac{\partial \mathcal{C}}{\partial w_{jk}^{(l)}} = \frac{\partial \mathcal{C}}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{jk}^{(l)}} = \Delta_j^{(l)} a_k^{(l-1)}. \tag{2.11}$$

The equations (2.8), (2.9), (2.10) and (2.11) define the four backpropagation equations. These equations relate the parameters of different layers to each other and the cost function.

Because the updating of parameters of one layer is related to the subsequent layer, the updating of the parameters can be done layer by layer, starting from the outermost layer L. This is where the algorithm gets its name, backpropagation, from. The error is said to propagate backwards through the neural network. Because there are no intra-layer connections, the parameters can be updated in parallel per layer. This greatly speeds up the learning process, especially on a system with many cores or when the program runs on a GPU.

2.2.2 Encoding translation invariance: the convolutional neural network

While the feed-forward neural network is already capable of distinguishing between the ordered and disordered phases in the two-dimensional Ising model, as will be shown in this thesis, it fails to take advantage of certain symmetries in the system. The convolutional neural network (CNN) has a design aimed at taking advantage of local features and translation invariance present in the system. The core idea is that explicitly including these ideas in the topology of the neural network helps the network pick up on these symmetries faster while requiring less parameters to train.

The CNN builts upon the FNN by adding an additional kind of layer: a convolution layer. This layer consists of a set of filters typically followed by pooling layers which coarse-grain the input while maintaining locality and spatial structure. The size of the filters depends on the dimensionality of the input data. For typical two-dimensional data the filter is characterised by a height H, and width W given in number of neurons. The number of different filters, usually related to the amount of relevant local features one wishes to extract from the input data, is given by the depth D. The neurons corresponding to a particular filter all share the same weights and biases.

The convolution filters take a small spatial patch of the previous layer as inputs. This spatial patch is sometimes called a receptive field. Each filter then identifies the neurons of the receptive field one-to-one to the neurons of the filter (they are of equal size). The values of the neurons in the receptive field then get multiplied with the values of the corresponding neurons in the filter. The resulting values all get added together and form the value for a single neuron in a convolution layer belonging to that particular filter. The convolution then consists of 'running' this filter over all locations in the spatial plane. How the filter moves over this spatial plane is defined by the stride S. Usually the input data is padded with additional neurons at the edges to make sure the filter can run over all input data neurons. These padded neurons are conventionally given the value 0, however in this thesis periodic boundary conditions are used unless specified otherwise.

When the input layer has moved through the filters, it hits the pooling layer. In this layer the filtered data is coarse-grained via some pooling operation. One of the most common used pooling operations is the max pool. In max pool a small region of neurons is replaced by a single neuron whose output is the largest value of the output within the region. This reduces the dimension of outputs thus speeding up the learning process. This layer is then followed by an all-to-all connected layer and an output layer. This means that we can still use the backpropagation algorithm to train the CNN. Figure 2.3 shows a schematic illustration of a CNN.

By designing the network in such a way local properties and translation invariance are explicitly incorporated in the network topology. Each filter can be related to a single local property and is applied over the entire input data space. The final all-to-all connected layer can then take relative location and the amount of certain features into account to give a prediction. Because each filter is described by a single set of weights and biases the number of free parameters is reduced drastically compared to adding an additional all-to-all layer. This allows us to build much larger networks than would be possible with fully connected layers.

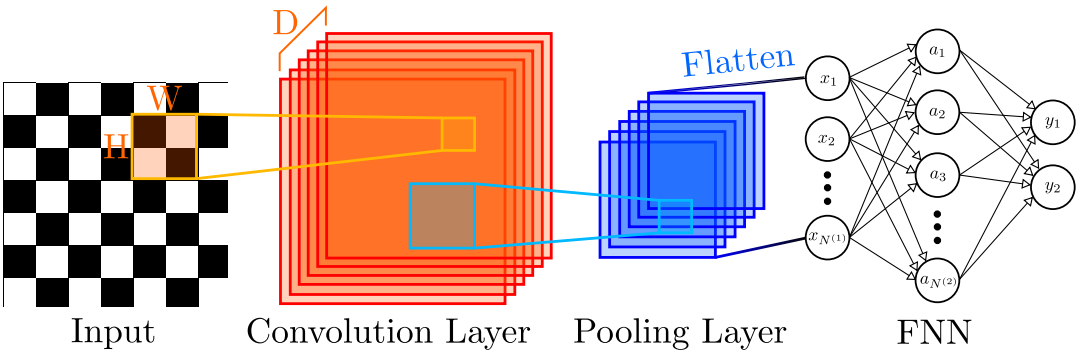


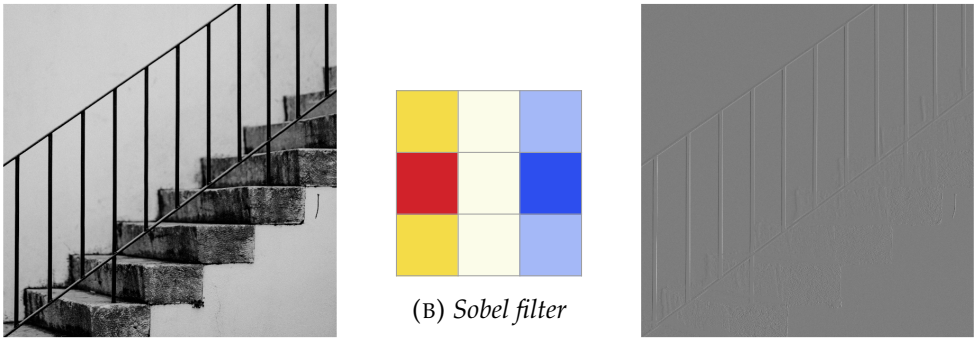
FIGURE 2.3: *Schematic illustration of a convolutional neural network. The network consists of a single convolution layer with D filters of size $H \times W$ followed by a pooling operation decreasing the size of the convolutional layer. The two-dimensional pooling layer is then flattened to a one-dimensional input vector for a feed-forward neural network (FNN).*

Interpreting filters through visual analysis

Interpreting the learned behaviour of a neural network is quite difficult. An advantage of CNNs is that they act on the original two-dimensional input configuration, rather than a flattened version as in the FNN. This means that the filters have a clear visual interpretation attached to them. The filters act on a part of the input configuration, adding the site (or pixel) values within the filter field with their associated filter values. It is common in CNN literature to show the convoluted image alongside the original image to show the effect of the filter[41]. By keeping track of the convoluted filters over the learning process it may become clear what the network is learning. As an example consider the filter

$$f_{\text{Sobel}} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}. \tag{2.12}$$

This filter is commonly used for edge detection in images, and is named after Irwin Sobel[42]. The effect of this filter is immediately clear when we apply it to an image, as is shown in figure 2.4. Clearly the bars of the staircase are highlighted out of the image by the filter. It may also be helpful to visualise the filter itself by transforming it to a heatmap picture. By thinking of these filters in this visual manner they can be given an interpretation of simple feature detectors. Multiple filters may then be correspond to different features, which are in turn fed into a FNN. Thinking of CNN in this way may help in interpreting the results, as well as help in the initial design of the network for a specific task.



(A) *original picture* (B) *Sobel filter* (C) *convoluted picture*

FIGURE 2.4: *Visualisation of the effect of the Sobel filter. Photo by Annie Spratt on Unsplash.*

2.2.3 Practical considerations

When building neural networks, several practical alterations are usually made. To prevent overfitting, it is common to introduce a L_2-regularisation term of the weight-matrices. The L_2-regularisation term of a parameter is nothing more then the squared magnitude of this parameter. Adding these terms to the cost function introduces a cost-penalty for increasing the absolute size of the parameters, preventing the learning procedure from increasing the parameters to decrease the cost function on the training set, while not increasing the accuracy on the validation set.

Another highly successful method to prevent overfitting, both in feed-forward neural networks and Boltzmann machines, is dropout[43]. In this method neurons can be randomly dropped during each learning step, where dropped means that all connections to and from this neuron are set to zero temporarily for this learning step. The idea is that this forces the other neurons to take over certain tasks previously performed by the dropped out neurons, which results in a better overall network which is less susceptible to noise.

In the previous discussion the activation functions where kept as general as they could be, however when actually making a neural network one has to chose the activation functions. There are a few popular choices. A very common one is called the sigmoid-function:

$$a_{\text{sigmoid}} = \sigma(z) = \frac{1}{1+\mathrm{e}^{-\mathbf{z}}}. \tag{2.13}$$

This function is commonly used for hidden neurons, and is also used a lot throughout this thesis. An alternative to the sigmoid-function is a simple tanh-function

$$a_{\tanh}(z) = \tanh z = \frac{\mathrm{e}^{z}+\mathrm{e}^{-z}}{\mathrm{e}^{z}-\mathrm{e}^{-z}} \tag{2.14}$$

which has similar behaviour as the sigmoid-function, with the main difference that the tanh-function has a range of $[-1, 1]$ and the sigmoid-function a range of $[0, 1]$.

A popular alternative to these functions is the rectifier-function, often called ReLu

for short, which is defined as

$$a_{\text{ReLu}}(z) = z^+ = \max(0, z). \tag{2.15}$$

This activation function has seen a lot of application in deep neural networks, and appears to allow for faster learning while retaining high accuracy compared to the non-linear activation functions.

For the output layer it is common to use a different kind of function, especially for classification problems. The function that will be used for the most part in this thesis is the softmax function:

$$a_{\text{softmax}}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_k^K e^{z_k}}. \tag{2.16}$$

This activation function is actually a vector, as indicated by the index i. Essentially this function takes the input vector $\mathbf{z}$ and normalises it into a probability distribution of K probabilities. For example when classifying two phases of matter, $K = 2$ and $\mathbf{z}$ corresponds to the input vectors for the two output neurons. The output of the output neurons then gives the normalised probability of the input configuration being in the phase corresponding to each neuron.

In this thesis the feed-forward and convolutional neural networks built for identifying phases in physical systems were all built in Google's TensorFlow API for Python [44].

2.2.4 Losing supervision by confusion

Neural networks like the FNN and CNN are generally used as supervised machine learning tools. The data is accompanied with the correct labels, and the neural network is able to minimise its cost function and will have learned something. However, there are situations where the labelling of the data might not be known. In such situations we can turn to unsupervised methods, such as Principle Component Analysis (PCA) or Boltzmann Machines, for example. However, there is also a way to transform the neural networks we have discussed in this section to the unsupervised kind.

A scheme which performs this task is the so-called confusion scheme[27]. This scheme is based on a very simple adaption of the regular neural network training procedure, where instead of providing the data with the correct labels, a wrong label is purposefully assigned to the label. For thermodynamic models it is possible to choose a false critical temperature T' and base the data labelling on that false critical temperature. The neural network is then trained on that falsely labelled data, and the accuracy of the network over the entire dataset is noted. One then repeats this procedure over a range of false critical temperatures, tracking the accuracy of the trained networks for each false critical temperature: $a(T')$.

The general idea is then that if the system truly exhibits a (or numerous) phase transitions in the mapped out T' range, this will show up in $a(T')$. The accuracy of the network will be large if T' is at the real critical point T_c, because then the data is correctly labelled and the network should be able to learn a correct identifier to distinguish the phases. For $T' \neq T_c$ some of the labelled data will be false, which will lead the neural network to be unable to classify part of the data. This leads to a drop

in accuracy. The neural network is said to be confused because of the false labelling, hence the name.

The true critical temperature can then be inferred by analysing the general shape of $a(T')$. It can be argued that this function should follow a rough W-shape. This argumentation is based on the assumption that the data has two different structures above and below T_c, and that the network is able to differentiate between them. Lets assume we have a dataset of configurations that run from T_i to T_f of a physical system with two phases, say phase 1 and phase 2, and a critical temperature $T_i < T_c < T_f$. If we then set $T' = T_i$ all the data will be labelled as phase 2, and the network will just ascribe phase 2 to every configuration. The accuracy will be 1. The same goes for $T' = T_f$, except the network now ascribes phase 1 to every configuration. When $T' = T_c$ the network is able to differentiate between the two structures, and again the accuracy will be 1. For $T_i < T' < T_c$ the network will see data with the same structure for T_i to T' and T' to T_c, but with different labels. We assume the network will choose to learn the label of the majority data, so that the accuracy in this range will be

$$a(T') = 1 - \frac{\min(T_c - T', T' - T_i)}{T_c - T_i}. \tag{2.17}$$

A similar argumentation holds for $T_c < T' < T_f$, so that the full function becomes

$$a(T') = \begin{cases} 1 - \frac{\min(T_c - T', T' - T_i)}{T_c - T_i} & T_i \leq T' < T_c \\ 1 & T' = T_c \\ 1 - \frac{\min(T' - T_c, T_f - T')}{T_f - T_c} & T_c < T' \leq T_f \end{cases}. \tag{2.18}$$

This function is plotted in figure 2.5 for $T_i = 0$, $T_f = 2$ and $T_c = 1$. Clearly this function has a W-shape, where the middle peak of the W corresponds to the critical temperature. The confusion scheme is then based on the assumption that the actual trained neural networks will show this same behaviour, so that the peak in the measured $a(T')$ can be identified with the critical temperature. It is thus a method to obtain the location of the phase transition for unlabelled data.

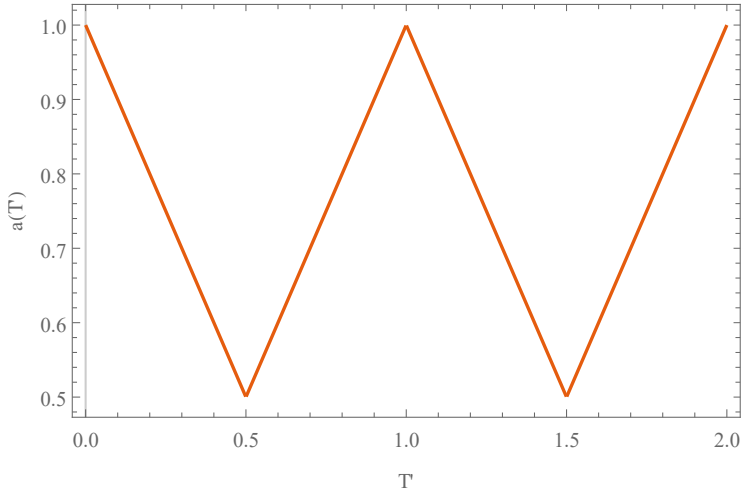


FIGURE 2.5: *Plot of equation* (2.18) *with* $T_i = 0$, $T_f = 2$ *and* $T_c = 1$.

2.3 Boltzmann machines

Neural networks like the ones discussed previously are classified under the denominator supervised machine learning. They are typically used to classify data into different classes. Unsupervised machine learning does not require any label to be given along with the training data and instead relies on some internal measure to determine how well it is performing. One such machine classified under the denominator unsupervised learning is the Boltzmann machine.

2.3.1 A mathematical introduction to Boltzmann machines

The Boltzmann machine is what is known as a generative machine. Its main objective is to model the underlying probability distribution of the training data during the training process, after which the machine can be used to generate new data given by its own probability distribution. The Boltzmann machine can be interpreted as a special kind of Markov random field or as a stochastic neural network. Here we follow a condensed derivation of Boltzmann machines starting from Markov random fields following Fisher and Igel[45].

To start we first need to consider Markov random fields. Markov random fields are a type of undirected graph. An undirected graph is defined in terms of labelled nodes $v_1, v_2, ...$ assembled together in a set V. Two nodes can be connected to each other via a line, or edge, which we denote by pairs of nodes. E.g. a connection between nodes v_1 and v_2 is denoted by $\{v_1, v_2\}$. The set of all edges together are labelled E. The entire topology of the graph G is then defined by combining the sets $G = (V, E)$. The neighbourhood $\mathcal{N}_v = \{w \in V : \{w, v\} \in E\}$ of v is the set of nodes connected to v. A clique is a subset of V in which all nodes are pairwise connected. The clique is maximal if no node can be added such that the clique is still a clique. An example of an undirected graph is shown in figure 2.6. In this example $\mathcal{N}_{v_1} = \{v_2, v_3\}$ and $\{v_1, v_2, v_3\}$ is a maximal clique.

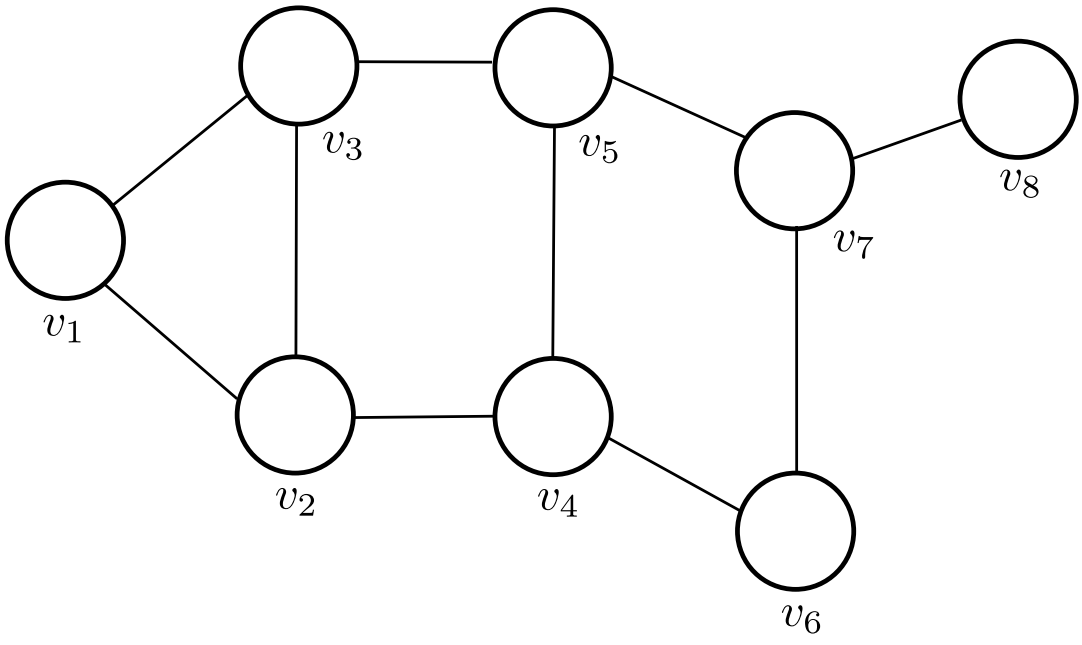


FIGURE 2.6: *Example of an undirected graph.*

To each node v is then assigned a random variable X_v taking values in a state space $\Lambda_v = \Lambda$. The set of all these random variables over all nodes $\mathbf{X} = (X_v)_{v \in V}$ is considered a Markov random field if the joint probability distribution p satisfies the (local) Markov property with relation to the graph. If for all $v \in V$ the random variable X_v is conditionally independent of all other variables given its neighbourhood, that is if

$$\forall v \in V,\ \forall \mathbf{x} \in \Lambda^{|V|} :\ p(x_v|(x_w)_{w \in V \setminus \{v\}}) = p(x_v|(x_w)_{w \in \mathcal{N}_v}) \tag{2.19}$$

holds, the local Markov property will be fulfilled. There are other Markov properties but for our purposes this one is sufficient.

If the probability distribution p satisfies this local Markov property, the Hammersley-Clifford theorem tells us the probability factorises with maximal cliques $\mathcal{C}$:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}). \tag{2.20}$$

Here Z is the partition function of the distribution, given by

$$Z = \sum_{\{\mathbf{x}\}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C). \tag{2.21}$$

The sum over $\{\mathbf{x}\}$ denotes taking the sum over all possible configurations of $\mathbf{x}$. If p is strictly positive, then so are the potential functions ψ_C, and we can write the probability function in a more suggestive form:

$$p(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{C \in \mathcal{C}} \ln \psi_C(\mathbf{x}_C)\right) = \frac{1}{Z} \exp\left(-E(\mathbf{x})\right). \tag{2.22}$$

We call E the energy function. This distribution is also called the Gibbs distribution.

This graph is then considered a (stochastic binary) Boltzmann machine if the energy function of the graph is given by

$$E(\mathbf{x}) = -\left(\sum_{i<j} w_{ij} v_i v_j + \sum_i b_i v_i\right), \tag{2.23}$$

where w_{ij} gives the weight between two connected binary nodes v_i and v_j, which can take the values $\{0, 1\}$. b_i is the bias associated with node v_i. The learning of this model then means altering the weights and biases such that the probability distribution $p(\mathbf{v})$ best fits the unknown underlying probability distribution $q(\mathbf{v})$ of the training data S. This is done by maximising the so-called likelihood

$$\mathcal{L}(\mathbf{W}, \mathbf{b}|S) = \prod_{i=1}^{l} p(\mathbf{v}_i|\mathbf{W}, \mathbf{b}), \tag{2.24}$$

with l the number data samples. This is generally not possible analytically and numerical techniques like gradient ascend are used.

Maximising the logarithm of the likelihood corresponds to minimising the Kullback-Leibler divergence (KL-divergence), which is a measure for the distance between

two distributions. The KL divergence of p and q for a finite state space Ω is

$$\mathrm{KL}(q||p) = \sum_{\{\mathbf{v}\}\in\Omega} q(\mathbf{v}) \ln \frac{q(\mathbf{v})}{p(\mathbf{v})} \tag{2.25}$$

$$= \sum_{\{\mathbf{v}\}\in\Omega} q(\mathbf{v}) \ln q(\mathbf{v}) - \sum_{\{\mathbf{v}\}\in\Omega} q(\mathbf{v}) \ln p(\mathbf{v}). \tag{2.26}$$

It can be understood as the difference between the entropy of q and p averaged over all $\mathbf{v}$ configurations weighted by q. The KL divergence is always positive and zero if and only if the two distributions are equal.

2.3.2 Gibbs sampling

Once the Boltzmann machine is trained it can be used to produce samples from the joint probability distribution. The way to this is by constructing a Markov chain by updating each variable based on its conditional distribution given the state of the other variables. The particular procedure to construct this Markov chain for a Boltzmann machine is called Gibbs sampling.

We again consider a Markov Random Field (MRF) $\mathbf{X} = (X_1, ..., X_N)$ in an undirected graph $G = (V, E)$ with $V = 1, ..., N$. The random variables X_i, $i \in V$ take values in a finite set Λ. The joint probability distribution of $\mathbf{X}$ is given by

$$\pi(\mathbf{x}) = \frac{1}{Z} \mathrm{e}^{-\mathcal{E}(\mathbf{x})}, \tag{2.27}$$

where $\mathcal{E}$ is some energy function. The Markov chain can then be constructed by assuming that the MRF changes its state over time, so that the Markov chain is

$$X = \{\mathbf{X}^{(k)} | k \in \mathbb{N}_0\}. \tag{2.28}$$

The chain takes values in $\Omega = \Lambda^N$. So $\mathbf{X}^{(k)} = (X_1^{(k)}, ..., X_N^{(k)})$ describes the state of the MRF at a time $k \geq 0$.

New states of the chain in between the timesteps are constructed by first randomly picking a variable X_i, $i \in V$, with probability $q(i)$. q is a positive definite probability distribution on V. The new state for this variable X_i is sampled based on its conditional probability distribution given the state of all other variables, which reduces to the conditional probability distribution given the state of the neighbourhood of X_i because of the local Markov property. The transition probability $p_{\mathbf{xy}}$ for two states $\mathbf{x}$, $\mathbf{y}$ of the MRF $\mathbf{X}$ with $\mathbf{x} \neq \mathbf{y}$ is

$$p_{\mathbf{xy}} = \begin{cases} q(i)\pi(y_i|(x_v)_{v\in V\setminus i}), & \text{if } \exists i \in V \text{ so that } \forall v \in V \text{ with } v \neq i : x_v = y_v \\ 0, & \text{else} \end{cases}. \tag{2.29}$$

The transition probability for the state to remain the same is

$$p_{\mathbf{xx}} = \sum_{i\in V} q(i)\pi(x_i|(x_v)_{v\in V\setminus i}). \tag{2.30}$$

The Markov chain defined by these transition probabilities is called the Gibbs chain.

Next we need to show that the Gibbs chain converges to the joint distribution π

of the MRF. We first show that the detailed balance condition holds:

$$\pi(i)p_{ij} = \pi(j)p_{ji} \qquad \forall i,j \in \Omega. \tag{2.31}$$

If this condition holds, π is a stationary distribution, meaning that

$$\pi^T = \pi^T \mathbf{P}, \tag{2.32}$$

where $\mathbf{P}$ is the transition matrix of the markov chain: $\mathbf{P} = (p_{ij})_{i,j\in\Omega}$.

For $\mathbf{x} = \mathbf{y}$ the detailed balance obviously holds. If $\mathbf{x} \neq \mathbf{y}$ the condition holds if more than one random variable differs between the two, since then $p_{\mathbf{xy}} = 0$. If $\mathbf{x}$ and $\mathbf{y}$ differ by exactly one random variable we can write

$$\begin{aligned}\pi(\mathbf{x})p_{\mathbf{xy}} &= \pi(\mathbf{x})q(i)\pi(y_i|(x_v)_{v\in V\setminus i}) \\ &= \pi(x_i,(x_v)_{v\in V\setminus i})q(i)\frac{\pi(y_i,(x_v)_{v\in V\setminus i})}{\pi((x_v)_{v\in V\setminus i})} \\ &= \pi(\mathbf{y})q(i)\pi(x_i|(x_v)_{v\in V\setminus i}) = \pi(\mathbf{y})p_{\mathbf{yx}}.\end{aligned} \tag{2.33}$$

So detailed balance holds and π is the stationary distribution.

To see that the Markov chain converges to the stationary distribution π, note that π is strictly positive. Since π is strictly positive, so will then the conditional probability distributions of the single variables. So every single variable X_i has a finite probability to take every state $x_i \in \Lambda$ in a single transition. This means that every state of the MRF can reach any other in the state space in a finite number of steps. The Markov chain is thus irreducible. Since $p_{\mathbf{xx}} > 0$ for all $\mathbf{x} \in \Lambda^N$, the chain is aperiodic. The aperiodicity and irreducibility of the Markov chain guarantee that the chain converges to the stationary distribution π.

2.3.3 Adding hidden nodes

In general the amount of nodes in the graph does not have to be equal to the dimension of the input data. The graph can be divided in a set of nodes which couple directly to the input nodes, which we call the visible nodes

$$\mathbf{V} = \{v_1, v_2, ..., v_m\}. \tag{2.34}$$

The nodes which do not directly couple to the input data are called the hidden (or latent) nodes

$$\mathbf{H} = \{h_1, h_2, ..., h_n\}. \tag{2.35}$$

Figure 2.7 shows a example of a Boltzmann machine with hidden nodes. Now the goal is for the marginal distribution of $\mathbf{V}$ to describe the data distribution q. So the visible variables correspond to the input data while the hidden variables introduce dependencies between the visible variables. The marginal distribution over $\mathbf{v}$ is thus given by summing over all configurations of $\mathbf{h}$:

$$p(\mathbf{v}) = \sum_{\{\mathbf{h}\}} p(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\sum_{\{\mathbf{h}\}} \exp\left(-E(\mathbf{v},\mathbf{h})\right). \tag{2.36}$$

The marginal distribution over $\mathbf{h}$ is

$$p(\mathbf{h}) = \sum_{\{\mathbf{v}\}} p(\mathbf{v}, \mathbf{h}). \tag{2.37}$$

Likewise, the conditional distributions are defined as

$$p(\mathbf{v}|\mathbf{h}) \equiv \frac{p(\mathbf{v})}{p(\mathbf{v}, \mathbf{h})} \tag{2.38}$$

$$p(\mathbf{h}|\mathbf{v}) \equiv \frac{p(\mathbf{h})}{p(\mathbf{v}, \mathbf{h})}. \tag{2.39}$$

The learning is then done through gradient ascend of the natural logarithm of the likelihood. In general the derivative with respect to some parameter θ can be written as

$$\begin{aligned}
\frac{\partial \ln \mathcal{L}(\theta|\mathbf{v})}{\partial \theta} &= \frac{1}{p(\mathbf{v}|\theta)} \frac{\partial p(\mathbf{v}|\theta)}{\partial \theta} \\
&= \frac{1}{p(\mathbf{v}|\theta)} \left[-\frac{1}{Z} \sum_{\{\mathbf{h}\}} \frac{\partial E}{\partial \theta} \mathrm{e}^{-E} + \frac{1}{Z^2} \left(\sum_{\{\mathbf{v}\}} \sum_{\{\mathbf{h}\}} \frac{\partial E}{\partial \theta} \mathrm{e}^{-E} \right) \sum_{\{\mathbf{h}\}} \mathrm{e}^{-E} \right] \\
&= -\sum_{\{\mathbf{h}\}} \frac{p(\mathbf{v}, \mathbf{h}|\theta)}{p(\mathbf{v}|\theta)} \frac{\partial E}{\partial \theta} + \frac{1}{p(\mathbf{v}|\theta)} \left(\sum_{\{\mathbf{h}\}} \sum_{\{\mathbf{h}\}} p(\mathbf{v}, \mathbf{h}|\theta) \frac{\partial E}{\partial \theta} \right) p(\mathbf{v}|\theta) \\
&= -\sum_{\{\mathbf{h}\}} p(\mathbf{h}|\mathbf{v}, \theta) \frac{\partial E(\mathbf{v}, \mathbf{h}|\theta)}{\partial \theta} + \sum_{\{\mathbf{v}\}} \sum_{\{\mathbf{h}\}} p(\mathbf{v}, \mathbf{h}|\theta) \frac{\partial E(\mathbf{v}, \mathbf{h}|\theta)}{\partial \theta}.
\end{aligned} \tag{2.40}$$

This can be interpreted as the difference between the expected value of the energy function under the model distribution and the expected value of the energy function under the conditional distribution of hidden variables given a training example. It turns out that directly calculating these sums is computationally expensive, so Monte Carlo Markov chain methods are generally used. This is possible because it can be shown that the Markov chain nicely converges to the expected stationary distribution. This is still quite computationally expensive, since in general any node can be connected to any other node, which makes computing a Markov step expensive. A solution to this is to restrict the Boltzmann Machine in such a way as to make the Monte Carlo sampling faster.

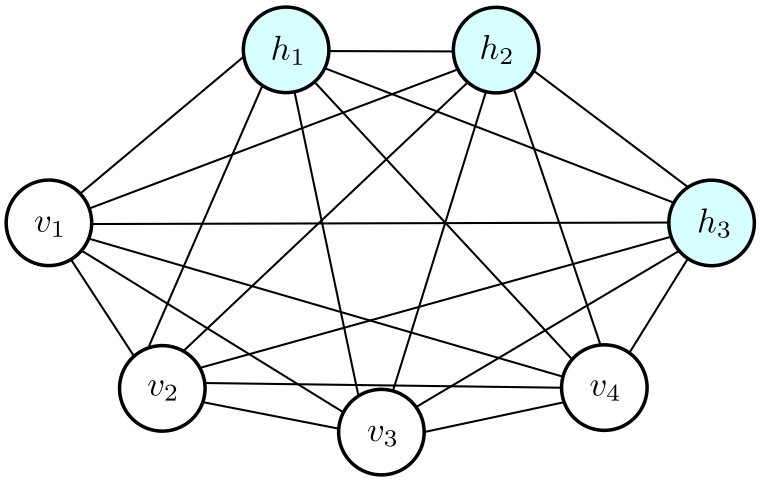


FIGURE 2.7: *Example graph of a Boltzmann machine with hidden nodes, denoted by* $H = \{h_1, h_2, h_3\}$.

2.3.4 Restricted Boltzmann Machines

The restricted Boltzmann Machine (RBM) is a restriction of the more general Boltzmann Machine. It restricts connections between nodes of the same type, so there are no visible-visible or hidden-hidden connections anymore. This divides the graph in a visible and hidden layer with no intra-layer connections. Every visible node is still connected to every hidden node and vice versa. The energy function for a $L \times N$ RBM can be rewritten in the form

$$E(\mathbf{v}, \mathbf{h}) = -\left(\sum_{i=1}^{L} \sum_{a=1}^{N} v_i W_{ia} h_a + \sum_{i=1}^{L} b_i v_i + \sum_{a=1}^{N} c_a h_a \right), \tag{2.41}$$

where c_j is the bias associated with the hidden node h_j. Figure 2.8 shows an example graph of a restricted Boltzmann machine.

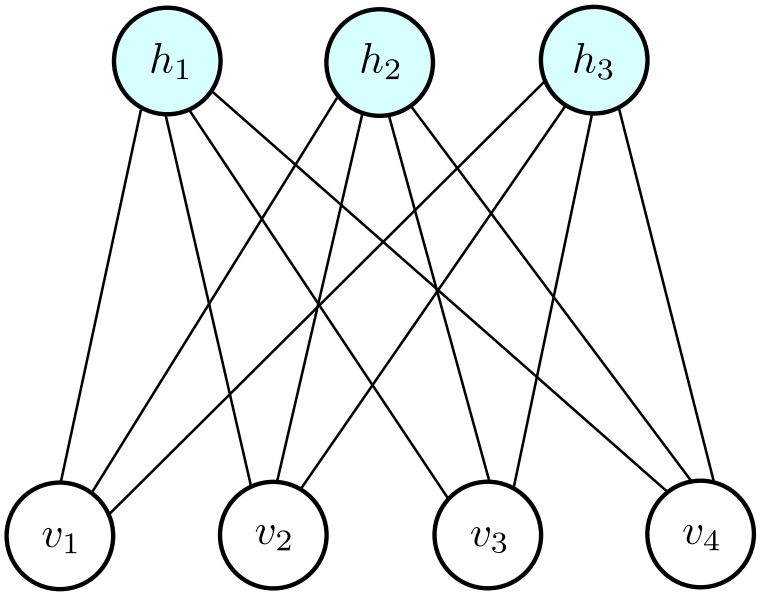


FIGURE 2.8: *Graph of a* 4×3 *restricted Boltzmann machine.*

With this new energy function we can rewrite the average of equation (2.40) over the dataset S with underlying distribution $q(\mathbf{v})$ for the weight parameter W_{ia} as

$$\begin{aligned}\frac{1}{l}\sum_{\{\mathbf{v}\}\in S}\frac{\partial \ln \mathcal{L}(\theta|\mathbf{v})}{\partial W_{ia}} &= \frac{1}{l}\sum_{\{\mathbf{v}\}\in S}\left[\sum_{\{\mathbf{h}\}} p(\mathbf{h}|\mathbf{v})v_i h_a - \sum_{\{\mathbf{v}\}}\sum_{\{\mathbf{h}\}} p(\mathbf{v},\mathbf{h})v_i h_a\right] \\ &= \frac{1}{l}\sum_{\{\mathbf{v}\}\in S}\left[\langle v_i h_a\rangle_{p(\mathbf{h}|\mathbf{v})} - \langle v_i h_a\rangle_{p(\mathbf{v},\mathbf{h})}\right] \\ &= \langle v_i h_a\rangle_{p(\mathbf{h}|\mathbf{v})q(\mathbf{v})} - \langle v_i h_a\rangle_{p(\mathbf{h},\mathbf{v})}, \end{aligned} \tag{2.42}$$

where l are the number of samples in the dataset S and the term in between the brackets $\langle ... \rangle$ indicates taking the average over the distribution in the subscript. This can be done for all parameters in the energy-function, so that the derivatives of the logarithm of the likelihood are

$$\frac{1}{l}\sum_{\{\mathbf{v}\}\in S}\frac{\partial \ln \mathcal{L}(\theta|\mathbf{v})}{\partial W_{ia}} = \langle v_i h_a\rangle_{p(\mathbf{h}|\mathbf{v})q(\mathbf{v})} - \langle v_i h_a\rangle_{p(\mathbf{v},\mathbf{h})}, \tag{2.43}$$

$$\frac{1}{l}\sum_{\{\mathbf{v}\}\in S}\frac{\partial \ln \mathcal{L}(\theta|\mathbf{v})}{\partial b_i} = \langle v_i\rangle_{p(\mathbf{h}|\mathbf{v})q(\mathbf{v})} - \langle v_i\rangle_{p(\mathbf{v},\mathbf{h})}, \qquad \text{and} \tag{2.44}$$

$$\frac{1}{l}\sum_{\{\mathbf{v}\}\in S}\frac{\partial \ln \mathcal{L}(\theta|\mathbf{v})}{\partial c_a} = \langle h_a\rangle_{p(\mathbf{h}|\mathbf{v})q(\mathbf{v})} - \langle h_a\rangle_{p(\mathbf{v},\mathbf{h})}. \tag{2.45}$$

In RBM literature it is common to call the average over $p(\mathbf{h}|\mathbf{v})q(\mathbf{v})$ the average over the data, since the visible layer, $\mathbf{v}$, comes from the data distribution $q(\mathbf{v})$ only. Likewise, the average over $p(\mathbf{v},\mathbf{h})$ is referred to as the average over the model since it is determined by the internal distribution of the RBM only.

The averages needed to update the variables are calculated through Monte Carlo Markov Chain methods, just as for the unrestricted Boltzmann machine. However, because there are no intra-layer connections the probability to get one layer given the other layer factorises over the individual values of the layer. I.e.

$$p(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{L} p(v_i|\mathbf{h}) \quad \text{and} \tag{2.46}$$

$$p(\mathbf{h}|\mathbf{v}) = \prod_{a=1}^{N} p(h_a|\mathbf{v}), \tag{2.47}$$

which greatly simplifies matters. From this perspective it can be useful to regard RBMs as a kind of stochastic neural network, with activation probabilities for the neurons

$$p(v_i = 1|\mathbf{h}) = \sigma(\sum_a W_{ia}h_a + b_i) \tag{2.48}$$

$$p(h_a = 1|\mathbf{v}) = \sigma(\sum_i W_{ia}v_i + c_a), \tag{2.49}$$

where σ is the sigmoid function (2.13). Note that the activation probabilities only take this form if we take $v \in \{0, 1\}$. Because the probability distribution factorises, the averages needed for learning can be calculated by iterative sampling from the conditional distributions (2.48) and (2.49) in a Markov Chain.

Block Gibbs sampling and the RBM flow

The procedure of Gibbs sampling as described for the unrestricted Boltzmann machine simplifies significantly for the restricted case. Since the variables in the same layer are conditionally independent, the states of all variables in one layer can be sampled jointly. Compared to the unrestricted case, where the Markov chain was defined by single-site transitions only this gives a significant increase in computational speed.

The Gibbs sampling can now be performed in just two steps. First a new hidden layer state $\mathbf{h}$ is sampled based on $p(\mathbf{h}|\mathbf{v})$. A new state for the visible layer can then be sampled from the new hidden layer with the conditional probability $p(\mathbf{v}|\mathbf{h})$. This essentially generates a new flow of visible configurations $\mathbf{v}$ as

$$\{\mathbf{v}\}^{(0)} \to \{\mathbf{h}\}^{(0)} \to \{\mathbf{v}\}^{(1)} \to ... \to \{\mathbf{h}\}^{(k-1)} \to \{\mathbf{v}\}^{(k)}. \tag{2.50}$$

This process is known as block Gibbs sampling. Since the units are conditionally independent this can be done in parallel, greatly speeding up the process in comparison to the unrestricted Boltzmann machine.

In theory this sampling should go on for infinite steps, because then the sample has truly converged to the equilibrium distribution of the model. In practice far fewer iterations are made. A commonly used alternative to proper Gibbs sampling is Contrastive Divergence (CD) introduced by Hinton[39]. This is a form of approximate Gibbs sampling. In CD-n, n iterations of block Gibbs sampling are performed (sometimes n is as small as 1) to obtain approximations to the averages over the model distribution. This comes at a cost of less sampling far from the datapoints in the mini-batch. So part of the ability to generalise gets sacrificed in order to improve the trainability of the model.

There are other methods to calculate the averages, like Persistent Contrastive Divergence[46] which builds upon CD. Sometimes completely other methods are used, like parallel tempering or other Metropolis schemes.

Chapter 3

Theoretical background

This chapter contains the relevant physical background needed to appreciate the results obtained from the neural network analysis as well as the connection to the restricted Boltzmann machine. First a general introduction to the phenomenon of phase transitions will be discussed. A very pedestrian view of the real-space renormalisation group will be given. After these general condensed matter subjects, the specific physical models used in this thesis will be discussed in detail. Some derivations will be made for the models which serve to provide a deeper understanding of the particular models. This deeper understanding is necessary to interpret the learned behaviour of the neural networks and restricted Boltzmann machine.

3.1 Introduction to phase transitions

Central in the field of condensed matter physics is the topic of phase transitions (PT). These are the transitions of a particular phase of a system to another phase of the same system. The typical classical example of a phase transition is the transition from the liquid phase of water to the gaseous phase. It turns out that many different kinds of systems display phase transitions and significant effort has gone into describing these transitions as precisely as possible.

Phase transitions are typically characterised by a jump in some thermodynamic variable. Universally this can be classified in terms of the free energy of a system, from which all thermodynamic quantities of a system can be derived[47]. According to the Ehrenfest classification of phase transitions, a phase transition where the first derivative of the free energy with respect to some thermodynamic variable exhibits a discontinuity is a first order phase transition. Likewise, a second order phase transition exhibits a discontinuity in the second derivative of the free energy. In more modern literature, the phase transitions are separated in two distinct classes: first order and continuous. First order transitions involve a latent heat and continuous phase transitions have a divergent susceptibility, an infinite correlation length (in the thermodynamic limit) and a power-law decay of correlations near criticality. Transitions which are continuous but break no symmetries are called infinite-order phase transitions.

While this description is useful for classification, for theoretical purposes one usually discusses so-called order parameters. An order parameter is usually defined as a quantity which has some non-zero value in one phase while it vanishes in the other phase. This could be a thermodynamic quantity, such as the magnetisation in the Ising model, but it does not have to be. Since this is mostly used as a theoretical tool to study phase transitions in mathematical models, the quantities used do not

need to be experimentally measurable but rather more emphasis is placed on the mathematical properties. As such these parameters are not necessarily physical, but rather very human in that they are explicitly chosen by the theorist such that they are the most useful to their calculation.

Phase transitions are usually accompanied by a break of symmetry, although this does not have to be the case. What is meant by this is that particular symmetries are present in one phase, while they are not present in the other phase. To stay with the example of water, the phase transition of ice to liquid water displays such a break of symmetry. Ice has a discrete rotational symmetry since it sits on some lattice. Liquid water has no such lattice but is said to have continuous rotational symmetry since it looks the same in any direction. Typically lower temperature phases have 'less symmetry' than higher temperature phases, meaning that more symmetry operations leave the phase unchanged in higher temperature phases. A system is said to display spontaneous symmetry breaking if the ground states of the system do not display the same symmetry as is present in the mathematical model of the system itself. The system is thus forced to choose one of the ground states, breaking the symmetry. The two dimensional Ising model is an example of this: the mathematical model is invariant under a global spin-flip transformation (a $\mathbb{Z}_2$ transformation), but the low-temperature ground states are states with either all spins up or all spins down. The ground states are evidently not invariant under the spin-flip transformation (they transform into one-another), but the system still has to choose one of these ground states. So the symmetry is said to be spontaneously broken.

As mentioned before, systems tend to have particular behaviours of physical quantities near criticality. These quantities tend to scale according to some power as a function of the reduced temperature (in classical systems). How they scale is described by particular exponents, called the critical exponents. They are very important for the classification of different models since the behaviour of these physical quantities is believed to be dependent on a few general features of the models, which allows for the grouping of models sharing the same critical exponents values. These models should thus display the same general features near criticality and are said to belong to the same universality class.

3.2 Introduction to the real-space renormalisation group

Renormalisation Group (RG) is an important part of modern physics, it is present in both high- and low-energy physics, albeit in slightly different forms. In this thesis only the low-energy approach will be considered, more specifically only the real-space renormalisation group. While the approach is different in different fields of physics, the main philosophy is the same. The idea is to step-by-step re-scale the system such that you only keep the important aspects while disregarding the irrelevant physics. It can be thought of as 'zooming out' on the physical system, rescaling the parameters such that the system still describes the same physics. In condensed matter physics it is commonly used to investigate phase transitions by studying how the physical parameters change close to the phase transition. At criticality a physical system is scale-invariant, meaning that changing the scale leaves the system unchanged. So a system at criticality is invariant under RG transformations.

We can make these statements more concrete. Lets say we have a discrete real-space system which can be described by the state variables $\{s_i\}$ and the coupling variables $\{J_k\}$ in some function which can describe all the physics, for example the Hamiltonian $\mathcal{H}(\{s_i\}, \{J_k\})$. A real-space RG transformation would then be a reduction and transformation of the state variables

$$\{s_i\} \to \{s_i'\},$$

such that there are less state variables than before. If we can transform the coupling variables

$$\{J_k\} \to \{J_k'\}$$

such that the Hamiltonian remains the same, i.e.

$$\mathcal{H}(\{s_i\}, \{J_k\}) = \mathcal{H}(\{s_i'\}, \{J_k'\}),$$

we say that we have a renormalisable theory. The change in the coupling variables can typically be described by a set of equations which give the mapping of the coupling variables to their next value. This change of coupling variables under each RG transformation is said to describe a flow of the coupling variables. This is what is meant with the RG flow. If the coupling variables remain unchanged under a RG transformation the system is at a fixed point. As mentioned before these points tend to correspond to critical points. An example of a RG flow diagram is shown in figure 3.1.

In RG a distinction is made between coupling parameters which continually decrease, increase or do something else under RG transformations. Coupling parameters which decrease under RG transformations are said to be irrelevant, meaning that they contribute less and less the more you rescale the system. Likewise, coupling parameters that increase under RG transformations are said to be relevant, clearly the process corresponding to this variable becomes more and more important under rescaling. Coupling parameters which do not fall under one of these classifications are called marginal, they require special inspection depending on the system.

This idea of relevant and irrelevant parameters also connects back to universality classes as discussed in the previous section. Just like some coupling parameters are irrelevant under RG transformations, so are certain observables. This means that

the apparent similarities of systems in the same universality class can be explained by the relevant and irrelevant variables. Systems in the same universality class will thus ultimately end up with the same relevant observables near criticality, while the irrelevant observables account for the differences on the microscopic level.

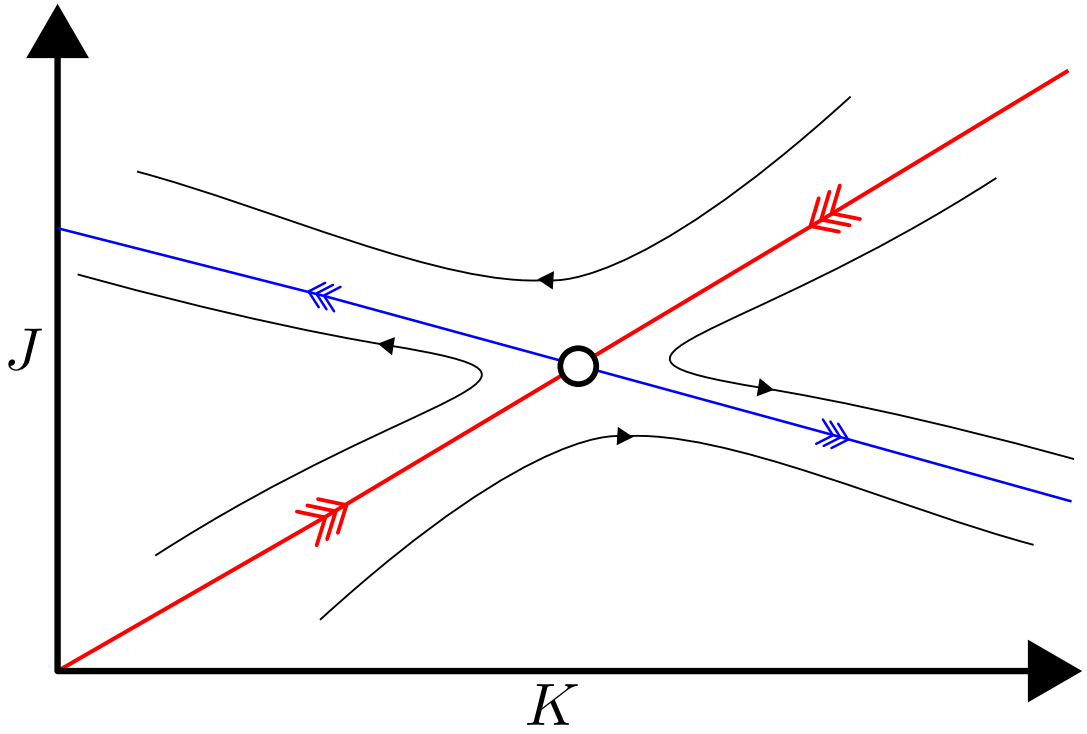


FIGURE 3.1: *Schematic picture of a RG flow as a function of the parameters K and J. The white circle is the fixed point of the flow, corresponding to the critical point. The red line is a critical line, on which the system flows towards the critical point under the RG transformations. The blue line flows away from the critical point. The areas separated by the critical line are the two phases the system can be in. The black lines give a schematic indication on how the variables on the line will flow under RG transformations.*

3.3 The Models

In this thesis many different well-known classical statistical physics models describing different condensed matter systems will be considered, either in the context of a numerical neural network analysis or in a restricted Boltzmann machine context. In this section the models will be introduced and the concepts relevant to understand the analysis following later will be explained and derived if needed.

3.3.1 The Ising chain

One of the most studied mathematical models in physics is the Ising model, conceptualised by Ising in 1925 to describe ferromagnetic systems[48]. The basic philosophy of the system is to describe the behaviour of a ferromagnetic system in terms of its most basic constituents only: the spins of the individual particles. In the standard Ising model these spins are binary, they can only be in the states 'up' or 'down'. The particles of the system are placed on a fixed lattice, where each lattice site contains a single particle. The form of the lattice has to be disclosed beforehand, since this can drastically impact the behaviour of the model. In this subsection we consider the one-dimensional Ising chain, where the lattice is just a line of lattice points. In the model the spin at lattice site i is represented by the symbol s_i. The spin can then take the values $s_i = \{-1, 1\}$, which correspond to spin down or up respectively.

The true physics of the Ising model lies in how these spins interact with one another. Typically only nearest-neighbour type interactions between spins are considered, although there are models which extend the range of the interactions. The interaction between two neighbouring spins at sites i and j is modelled by the interaction parameter $J_{i,j}$ and the product of the spins. The influence of an external magnetic field on the system can be modelled by including a hs_i term for every site, where h represents the magnetic field strength, or the strength of the coupling between the spin and the magnetic field. A Hamiltonian can then be defined for a system of L spins:

$$H = -\sum_{i=1}^{L} J_{i,i+1} s_i s_{i+1} - \sum_{i=1}^{L} h s_i. \tag{3.1}$$

In this thesis we will consider a constant interaction parameter for every spin couple that is greater than zero: $J_{i,j} = J > 0$. The external magnetic field strength will be zero unless mentioned otherwise: $h = 0$. The Hamiltonian then becomes

$$H = -J \sum_{i=1}^{L} s_i s_{i+1}. \tag{3.2}$$

The main strength of this model, apart from giving a conceptually clear model describing ferromagnetic systems, is that it is extremely well studied and understood. It is possible to derive expressions for physical quantities analytically. This makes this model an excellent candidate for performing novel numerical analyses since the results can be compared against the analytic results.

The main method to obtain expressions for physical quantities is to compute the partition sum Z of the system. Here periodic boundary conditions are assumed, thus one of the nearest neighbour sites of site $i = L$ is $L + 1 = 1$. The partition

function can then be calculated using the transfer-matrix method:

$$Z = \sum_{\{\mathbf{s}\}} \exp\left(\beta(J\sum_{i}^{L} s_i s_{i+1} - h\sum_{i}^{L} s_i)\right) \tag{3.3}$$

$$= \sum_{\{\mathbf{s}\}} \prod_{i}^{L} \exp\left(K s_i s_{i+1} + \beta h s_i\right) \tag{3.4}$$

$$= \mathrm{Tr}(\mathcal{T}^L), \tag{3.5}$$

where $\beta = \frac{1}{k_B T}$ with k_B the Boltzmann constant which we will set to 1, $K = \beta J$ and $\sum_{\{\mathbf{s}\}}$ represents the sum over all possible spin configurations. Here we included the magnetic field strength h to be able to calculate the magnetic susceptibility later on. The transfer-matrix $\mathcal{T}$ is then given by

$$\mathcal{T} = \begin{bmatrix} \mathrm{e}^{K+\beta h} & \mathrm{e}^{-K} \\ \mathrm{e}^{-K} & \mathrm{e}^{K-\beta h} \end{bmatrix}. \tag{3.6}$$

The trace of a product of the same matrices is equal to the sum of the product of the individual eigenvalues:

$$Z = \lambda_1^L + \lambda_2^L, \tag{3.7}$$

where λ_1 and λ_2 are the two eigenvalues of the transfer-matrix $\mathcal{T}$. These eigenvalues are

$$\lambda_1 = \mathrm{e}^K \cosh \beta h + \sqrt{\mathrm{e}^{2K}(\sinh \beta h)^2 + \mathrm{e}^{-2K}} \tag{3.8}$$

$$\lambda_2 = \mathrm{e}^K \cosh \beta h - \sqrt{\mathrm{e}^{2K}(\sinh \beta h)^2 + \mathrm{e}^{-2K}}. \tag{3.9}$$

Such that the partition function becomes

$$Z = (\mathrm{e}^K \cosh \beta h + \sqrt{\mathrm{e}^{2K}(\sinh \beta h)^2 + \mathrm{e}^{-2K}})^L + (\mathrm{e}^K \cosh \beta h - \sqrt{\mathrm{e}^{2K}(\sinh \beta h)^2 + \mathrm{e}^{-2K}})^L. \tag{3.10}$$

With this expression for the partition function it is possible to find expressions for certain physical quantities. The average energy of the system at zero magnetic field strength can be expressed as

$$\begin{aligned} \langle E \rangle\Big|_{h=0} &= -\frac{1}{L}\frac{\partial \ln Z}{\partial \beta}\Big|_{h=0} \\ &= -\frac{J(\cosh K)^{L-1}\sinh K + J(\sinh K)^{L-1}\cosh K}{(\cosh K)^L + (\sinh K)^L}. \end{aligned} \tag{3.11}$$

Likewise an expression for the magnetic field can be found, however for finite β the magnetisation vanishes:

$$\begin{aligned} \langle M \rangle\Big|_{h=0} &= -\frac{1}{L}\frac{\partial \ln Z}{\partial h}\Big|_{h=0} \\ &= 0. \end{aligned} \tag{3.12}$$

Another important quantity which will be used to evaluate the RBM results is the magnetic susceptibility, which can be expressed as

$$\begin{aligned}\chi\Big|_{h=0} i &= \frac{\partial\langle M\rangle}{\partial h}\Big|_{h=0} \\ &= \beta^2 \mathrm{e}^{2K} \frac{(\cosh K)^L - (\sinh K)^L}{(\cosh K)^L + (\sinh K)^L}\end{aligned} \tag{3.13}$$

for a vanishing magnetic field. Note that the one-dimensional Ising model has a free energy, defined as $f = \frac{1}{L}\ln Z$, which is analytic away from $T = 0$ even in the thermodynamic limit. This means that the one-dimensional Ising model displays no phase transitions.

When analysing what a particular machine learning algorithm has learned, physical quantities such as the ones mentioned before can be useful to ascribe some physical meaning to the learned result. Apart from these thermodynamic quantities, statistical quantities such as correlations and variances can be useful. In turn, these quantities tend to have some connection to other physical quantities. A particular quantity which is useful in analysing the behaviour of a probability distribution is the two-point correlation function. For the one-dimensional Ising model this quantity can be calculated exactly. We start with the definition of the two-point correlation function of two spins located r-sites apart:

$$\langle s_i s_{i+r}\rangle\Big|_{h=0} = \frac{1}{Z}\sum_{\{v\}} s_i s_{i+r} \exp\left(K\sum_i s_i s_{i+1}\right).$$

Note that this expression can be written in a transfer-matrix form, with the extra matrix

$$D = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3.14}$$

This matrix then represents the ± 1 values the factors s_i and s_{i+1} can individually take in the partition sum, which is then represented in the transfer-matrix form by inserting D before the transfer-matrix for site i and one before the transfer-matrix for site $i + r$. This gives the expression

$$\langle s_i s_{i+r}\rangle = \frac{1}{Z}\mathrm{Tr}(\mathcal{T}^{i-1} D \mathcal{T}^r D \mathcal{T}^{L-(i-1+r)}). \tag{3.15}$$

The matrix product squashed between the two D matrices has the same eigenvalues as a product of L $\mathcal{T}$ matrices, but the eigenvalues correspond to the opposite eigenvectors as for the $\mathcal{T}$ matrix. This means that the two-spin correlation function can be expressed as

$$\begin{aligned}\langle s_i s_{i+r}\rangle &= \frac{1}{Z}(\lambda_1^{L-r}\lambda_2^r + \lambda_2^{L-r}\lambda_1^r) \\ &= \frac{(\cosh K)^{L-r}(\sinh K)^r + (\sinh K)^{L-r}(\cosh K)^r}{(\cosh K)^L + (\sinh K)^L}.\end{aligned} \tag{3.16}$$

Taking the thermodynamic limit, where $L \to \infty$, one gets the simple expression

$$\lim_{L\to\infty}\langle s_i s_{i+r}\rangle = (\tanh K)^r. \tag{3.17}$$

From this one can easily find an expression for the correlation length ξ, defined as $\langle s_i s_{i+r} \rangle \propto \exp(-r/\xi)$, for the one-dimensional Ising model in the thermodynamic limit:

$$\xi = -(\ln \tanh K)^{-1}. \tag{3.18}$$

Note that this quantity only diverges in the limit $K \to \infty$, which further strengthens the argument that the one-dimensional Ising model does not experience a phase transition at any finite temperature.

Renormalisation Group equations

The argument that there is no phase transition in the one-dimensional Ising model can be made even more strongly by applying RG to the model. If we consider an Ising chain with an even number of sites L, the Hamiltonian will be given by equation (3.2). The RG then consists of summing over the even sites, so that we are left with an effective Hamiltonian H_{eff} only over the odd sites:

$$\sum_{\{s\}_{\text{even}}} \mathrm{e}^{-\beta H(s)} = \mathrm{e}^{-H_{\text{eff}}(s_{\text{odd}})}. \tag{3.19}$$

The odd sites are now a distance 2 apart from one another and there are half as many of them as in the original Hamiltonian, so that we have effectively 'zoomed in' by a factor of 2.

For the one-dimensional case this rescaling can be calculated exactly by explicitly performing the sum over the even sites. For a single even site this gives:

$$\sum_{s_2=\pm 1} \exp(K s_2(s_1 + s_3)) = 2\cosh K(s_1 + s_3) \equiv \Delta \mathrm{e}^{K' s_1 s_3}, \tag{3.20}$$

where Δ and K' are new constants defined by the equation above. The sum over all the even spins then gives

$$\begin{aligned} \sum_{\{s\}_{\text{even}}} \mathrm{e}^{-\beta H} &= \Delta^{N/2} \exp\Big(K' \sum_{i=1}^{L/2} s_{2i-1} s_{2i+1}\Big) \\ &= \mathrm{e}^{-H_{\text{eff}}}, \end{aligned} \tag{3.21}$$

where the effective Hamiltonian can be written as

$$H_{\text{eff}}(s') = \sum_{i \in \mathbb{Z}_{\text{odd}}} K' s_i s_{i+2} + \frac{N}{2} \ln \Delta. \tag{3.22}$$

The $\ln \Delta$-term just adds a constant to the Hamiltonian, which can be removed from the Hamiltonian without changing any of the physics.

The new parameters of the effective Hamiltonian can be expressed in terms of the old parameters by considering the values s_1 and s_3 can take in equation (3.20). There are only two distinct cases, $s_1 = s_3$, and $s_1 = -s_3$. Those cases give

$$2\cosh 2K = \Delta \mathrm{e}^{K'} \qquad \text{if} \quad s_1 = s_3 \tag{3.23}$$

$$2 = \Delta \mathrm{e}^{-K'} \qquad \text{if} \quad s_1 = -s_3. \tag{3.24}$$

Taking the product of the two cases gives an expression for Δ:

$$\Delta^2 = 4\cosh 2K \quad \rightarrow \quad \Delta = 2\sqrt{\cosh 2K}. \tag{3.25}$$

Likewise the ratio of the two cases gives an expression for K':

$$e^{2K'} = \cosh 2K. \tag{3.26}$$

The equation is typically rewritten in terms of $v = \tanh K$:

$$\begin{aligned} e^{2K'} &= 2\cosh^2 K - 1 \\ \frac{2}{e^{2K'}+1} &= \frac{1}{\cosh^2 K} \\ -\frac{e^{2K'} - 1 - (e^{2K'}+1)}{e^{2K'}+1} &= \frac{1}{\cosh^2 K} \\ \frac{e^{2K'}-1}{e^{2K'}+1} &= \frac{\cosh^2 K - 1}{\cosh^2 K} \\ \tanh K' &= \tanh^2 K. \end{aligned} \tag{3.27}$$

This can be written more cleanly as

$$v' = v^2, \tag{3.28}$$

where $v \in [0, 1]$. Equation (3.28) then provides a RG map where the degrees of freedom are integrated out over each step.

This map has two fixed point: $v_0 = 0$ and $v_1 = 1$. The point v_0 corresponds to that of $K = 0$, where the temperature is infinite or where there are no interactions between spins. The spins are then fully uncorrelated and the model is in its paramagnetic phase. The point v_1 corresponds to $K = \infty$, where the temperature vanishes or where the interaction strength becomes infinitely large. The spins will all align in this case, so this fixed point corresponds to the ferromagnetic phase. Since there is no fixed point for any finite temperature, there is no phase transition where the correlation length diverges and the system becomes scale-invariant. The stability of the fixed points can be easily checked by taking the derivative of the map: $\partial(v^2) = 2v$, which is 2 for v_1 and is thus unstable. The fixed point v_0 has derivative 0, which means that it is stable. Since $0 \leq v \leq 1$ and $v = 1$ is unstable, the RG flow of the system runs from the v_1 fixed point in the direction of v_0. This can also be easily seen by noting that $v' < v \; \forall \, 0 < v < 1$. A schematic illustration of the RG flow is shown in figure 3.2. Thus it should be clear that the one-dimensional Ising model does not contain any phase transitions at finite temperatures.

FIGURE 3.2: *Schematic illustration of the RG flow for the one-dimensional Ising model.*

3.3.2 Two-dimensional Ising Model on a square lattice

The one-dimensional Ising chain discussed in the previous section can be easily generalised to two dimensions. Rather than having a single line of spins, the spins now occupy a two-dimensional lattice and are connected to their nearest neighbours. There are a multitude of different two-dimensional lattices, which can change the behaviour of the Ising model. In this section we will consider a square lattice with periodic boundary conditions, where every spin is connected to four nearest neighbours. We will refer to this model as the two-dimensional square Ising model. The spins are now labelled by two indices: $s_{i,j}$ where i and j denote the horizontal and vertical positions on the lattice respectively. The Hamiltonian can then be formulated as

$$H = -J \sum_{i,j} s_{i,j}(s_{i+1,j} + s_{i,j+1}). \tag{3.29}$$

While this models appears to be a straightforward generalisation of the one-dimensional Ising chain to two dimensions, the analytic results as were obtained for the one-dimensional Ising model are not so simple to derive.

Fortunately, Onsager showed nearly twenty years after the introduction of the model by Ising that this model can be solved in the thermodynamic limit[49]. The very lengthy calculations will not be repeated in this thesis, rather the results will be quoted as given. Onsager showed that the free energy in the thermodynamic limit can be written as

$$-\beta f = \ln 2 + \frac{1}{8\pi^2} \int_0^{2\pi} \mathrm{d}\theta_1 \int_0^{2\pi} \mathrm{d}\theta_2 \ln\Big(\cosh^2 2K - \sinh 2K(\cos\theta_1 + \cos\theta_2)\Big). \tag{3.30}$$

This function is not analytic for all values of K and in fact displays a phase transition. The point where this phase transition occurs was postulated by Onsager, and later proved by Yang[50] to be:

$$K_c = \frac{1}{2} \ln(1 + \sqrt{2}), \tag{3.31}$$

where K_c is the critical value for K where the phase transition occurs. At this critical point the system is said to go from an ordered to a disordered state. By an ordered state is meant a state where all the spins all either pointing up or down, while in the disordered state the spin directions are distributed randomly. The ordered phase is characterised by constant two-spin correlations, while the disordered phase is characterised by an exponential decay with distance between the spins in the two-spin correlations. This phase transition is then characterised by an order parameter: the magnetisation M. In the ordered state M will be ± 1, while in the disordered state it will be 0. The system is then said to undergo a spontaneous symmetry breaking when it crosses the critical point, since the ground state is invariant under a global spin flip transformation in the disordered state just like the Hamiltonian, while in the ordered state performing a global spin flip transformation changes the sign of M. The ground state of the system thus no longer contains the same symmetries as the Hamiltonian and is then said to have broken the symmetry.

Finite-size scaling

Since the numerical analysis of this thesis will be concerned with finite systems, some finite-size corrections need to be made to the analytic results in the thermodynamic limit. For a second-order phase transition the finite-size effects of critical

phenomena can be extracted from the size dependence of the free energy. This scaling of the free energy is described by the scaling ansatz[51]:

$$F(L,T) = L^{-(2-\alpha)/\nu}\mathcal{F}(tL^{1/\nu}), \tag{3.32}$$

where α and ν are the usual critical exponents, L is the size of the system, t is the reduced temperature $T/T_c - 1$ and $\mathcal{F}$ is a function of the scaled variable $tL^{1/\nu}$. Since the various thermodynamic quantities of a system can be expressed in terms of derivatives of F, one can get the scaling forms for the thermodynamic quantities:

$$M = L^{-\beta/\nu}\mathcal{M}^0(tL^{1/\nu}), \tag{3.33}$$

$$\chi = L^{\gamma/\nu}\chi^0(tL^{1/\nu}), \tag{3.34}$$

$$C = L^{\alpha/\nu}\mathcal{C}^0(tL^{1/\nu}). \tag{3.35}$$

Here $\mathcal{M}^0$, χ^0 and $\mathcal{C}^0$ are the scaling functions and again β and γ are the usual critical exponents. These functions are what actually will be measured in finite-size calculations, such as Monte Carlo simulations. This scaling ansatz is only taken to be valid for sufficiently large system sizes and temperatures close to the critical temperature T_c.

Likewise for systems with periodic boundary conditions undergoing a second-order phase transition (where the correlation length diverges), a simple relation between the critical temperature at the thermodynamic limit T_c and the apparent finite-size critical temperature $T_c(L)$ exists:[52]

$$T_c(L) = aL^{-1/\nu} + T_c. \tag{3.36}$$

Here a is a scaling parameter. This particular equation will be useful in determining T_c from the neural network analysis as described in chapter 4, where the neural network is only capable of determining $T_c(L)$.

Renormalisation Group equations

A renormalisation similar to the even-odd spin renormalisation that was applied to the Ising chain can be applied to the two-dimensional Ising model on a square lattice[53]. Because we are now dealing with a two-dimensional problem, the 'even' spins now lie on alternate diagonal rows in the square lattice. This means that the remaining 'odd' spins form a $\pi/2$-rotated square lattice with a lattice spacing increased by a factor $\sqrt{2}$. The lattice transformation is shown schematically in figure 3.3. Since none of the 'even' spins that are summed over couple to any of the other 'even' spins being summed over (the Ising model has nearest-neighbour interactions only), the sum over these 'even' spins for every 'odd' site can be performed independently. For 'odd' site (i,j) the sum is

$$\begin{aligned}\sum_{s_{i,j}=\pm 1} &\exp\left[Ks_{i,j}(s_{i-1,j} + s_{i+1,j} + s_{i,j-1} + s_{i,j+1}))\right] \\ &= 2\cosh\left[K((s_{i-1,j} + s_{i+1,j} + s_{i,j-1} + s_{i,j+1})\right].\end{aligned} \tag{3.37}$$

While the form of the function is similar, there is a major difference between the one-dimensional case and this case. In the one-dimensional case there were only two cases, all spins aligned or anti-aligned. This allowed us to write the result in terms of a single interaction parameter K'.

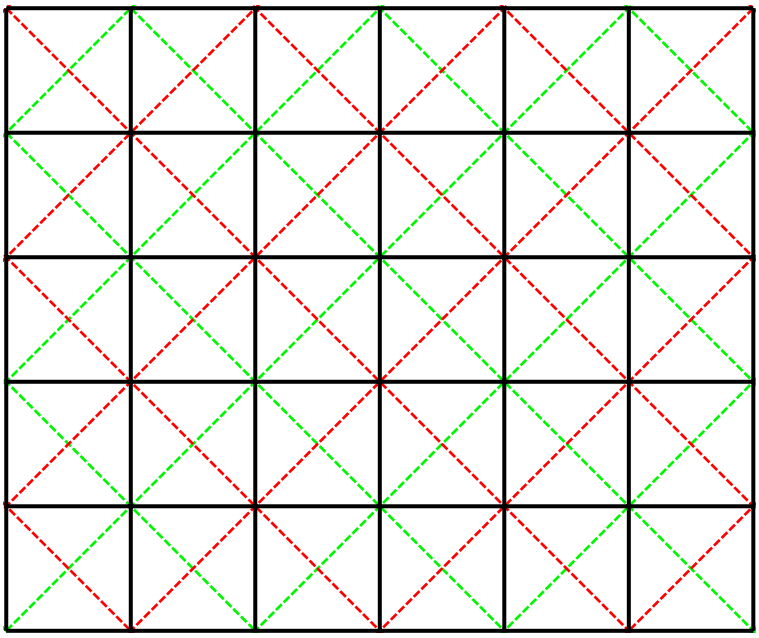


FIGURE 3.3: *Schematic picture of the lattice transformation under a RG step. The initial lattice is shown in black. The green lattice is the new lattice after the RG step, corresponding to the odd-numbered spins on the original lattice. The (even-numbered) spins which are integrated out lie on the red lattice.*

Now there are three cases: all spins aligned, one anti-aligned pair and zero magnetisation. Since the original Hamiltonian only has the single interaction parameter K, it is not going to be possible to rewrite the renormalised Hamiltonian in its original functional form. Instead we take a trial form with three renormalised interaction strengths:

$$\begin{aligned} &2\cosh\left[K(s_{i-1,j}+s_{i+1,j}+s_{i,j-1}+s_{i,j+1})\right] \\ &= \Delta(K)\exp[K_1'(s_{i-1,j}s_{i,j-1}+s_{i+1,j}s_{i,j-1}+s_{i-1,j}s_{i,j+1}+s_{i,j+1}s_{i+1,j}) \\ &\quad +K_2'(s_{i-1,j}s_{i+1,j}+s_{i,j-1}s_{i,j+1})+K_3's_{i-1,j}s_{i+1,j}s_{i,j-1}s_{i,j+1}]. \end{aligned} \tag{3.38}$$

The new interactions can be understood as the renormalised nearest-neighbour coupling (K_1'), a new interaction between next-nearest neighbours (K_2') and a new interaction between the four spins at the vertices of any square of the lattice (K_3'). A relation between K and the new couplings can be derived in the same manner as for the Ising chain. We consider the now four distinct cases:

$$2\cosh 4K = \Delta(K)\exp\left[4K_1'+2K_2'+K_3'\right] \tag{3.39}$$
$$2\cosh 2K = \Delta(K)\exp\left[-K_3'\right] \tag{3.40}$$
$$2 = \Delta(K)\exp\left[-2K_2'+K_3'\right] \tag{3.41}$$
$$2 = \Delta(K)\exp\left[-4K_1'+2K_2'+K_3'\right], \tag{3.42}$$

since the case with zero magnetisation is now split into two cases: one with two anti-aligned neighbouring pairs and one with four anti-aligned neighbouring pairs. By solving this set of equations we can get solutions for the renormalised coupling

constant:

$$K_1' = \frac{1}{4}\ln(\cosh 4K) \tag{3.43}$$

$$K_2' = \frac{1}{8}\ln(\cosh 4K) \tag{3.44}$$

$$K_3' = \frac{1}{8}\ln(\cosh 4K) - \frac{1}{2}\ln(\cosh 2K) \tag{3.45}$$

$$\Delta(K) = 2\cosh^{1/2}(2K)\cosh^{1/8}(4K). \tag{3.46}$$

Our renormalised Hamiltonian is now of another functional form, which means that the renormalisation procedure outlined above can not be iterated again and again, as was done for the Ising chain. It is possible to derive a flow if we make the assumption that K_3' will not matter in our calculations, so that it can be ignored. We will do the same to the next-nearest neighbour interactions K_2', but since K_2' is positive definite and will thus strife for aligned spins, its presence is approximated by including it in a new nearest neighbour coupling parameter

$$K' = K_1' + K_2'. \tag{3.47}$$

The model is then again of the same functional form as the original Hamiltonian, so that the process can be repeated. This gives us a recursion relation for the new renormalised coupling constant:

$$K' = \frac{3}{8}\ln(\cosh 4K). \tag{3.48}$$

$\Delta(K)$ is kept the same.

The recursion relation (3.48) has three fixed points:

$$K_0 = 0, \tag{3.49}$$

$$K_* = \infty \qquad \text{and} \tag{3.50}$$

$$K_c \approx 0.506981. \tag{3.51}$$

The stability of these points to linear order can again be checked by taking the derivative of the map, $\partial K'$:

$$\partial K' = \frac{3}{2}\tanh 4K. \tag{3.52}$$

For K_0 the derivative is $\partial K' = 0$, so K_0 is a stable fixed point. For K_* the derivative is $\partial K' = 3/2 > 1$, so the point should be unstable, however it is nonsensical to consider deviations around infinity and from the form of equation (3.48) it is clear that K' grows for each new iteration with sufficiently large K. Since K' just grows larger and larger in this limit, K_* is said to be stable. K_c has $\partial K' \approx 1.449 > 1$, so the fixed point is unstable. This means that the RG flow will flow towards K_0 for $K_0 < K < K_c$ and towards K_* for $K_c < K < K_*$. To check that the assumption to ignore K_3' was permitted, we can calculate K_3' at the critical point K_c. We get $K_3' \approx -0.053$, which suggests that the four-spin coupling is small at the critical point and can be dealt with in perturbation theory. Figure 3.4 shows a schematic illustration of the RG flow.

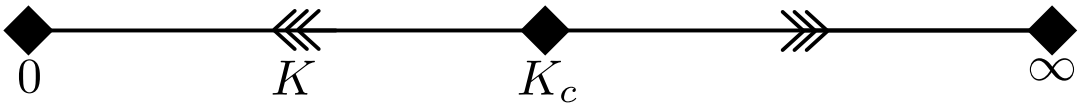


FIGURE 3.4: *Schematic illustration of the RG flow for the two-dimensional Ising model.*

Scaling relations

From the RG equations derived above it is also possible to derive a scaling form of the free energy $f \equiv \frac{1}{L} \ln Z$ near the critical point. This is particularly useful for deriving critical exponents. From the expressions for the renormalised coupling parameters we can derive the renormalised free energy:

$$f(K') = 2f(K') - \ln \Delta(K). \tag{3.53}$$

Since we are interested in the region near the critical point and $\ln \Delta(K)$ is analytic in K, we can neglect this term. A more general recursion relation between the singular part of the free energy can be written as

$$f_s(K) = b^{-d} f_s(K'), \tag{3.54}$$

where in our example $b = \sqrt{2}$, which represents the change in nearest-neighbour distance, and $d = 2$ for dimensionality. If we are close to the critical point K_c we can express K' in terms of a linearised recursion relation:

$$K' = K_c + \frac{\mathrm{d}R}{\mathrm{d}K}\Big|_{K=K_c} (K - K_c) + \ldots, \tag{3.55}$$

where R is the map between K' and K: $K' = R(K)$. This can be rewritten in terms of $\delta K' = K' - K_c$ and $\delta K = K - K_c$

$$\delta K' = \lambda \delta K = b^y \delta K. \tag{3.56}$$

Here $\lambda = \frac{\mathrm{d}R}{\mathrm{d}K}\Big|_{K=K_c}$. This is written as b^y, since subsequent application of the RG transformation yields $\lambda(b)\lambda(b) = \lambda(b^2)$. The singular free energy can now be expressed as

$$f_s(K_c + \delta K) = b^{-d} f_s(K_c + b^y \delta K). \tag{3.57}$$

Since K_c is a constant this is equivalent to

$$f_s(\delta K) = b^{-d} f_s(b^y \delta K). \tag{3.58}$$

We can rewrite δK in terms of the reduced temperature t:

$$\delta K = Kt. \tag{3.59}$$

Since K is finite at the critical point, the singular free energy is written as

$$f_s(t) = b^{-d} f_s(b^y t). \tag{3.60}$$

This equation should hold for any b, so we can set b equal to $b = |t|^{-1/y}$, so that

$$f_s(t) = |t|^{d/y} f_s(t/|t|). \tag{3.61}$$

With this form of the free energy we are able to derive critical exponents from our RG transformations. For example, the specific heat is defined as

$$C_V = -T\frac{\partial^2 F}{\partial T^2}\Big|_V, \tag{3.62}$$

where $F = Lf = \ln Z$. In a second-order phase transition, such as the one present in the two-dimensional Ising model, second derivatives of the free energy will have a singularity. For the specific heat this singularity is usually characterised by $C_V \propto |t|^{-\alpha}$, where α is the critical exponent related to C_V. This means that the singular free energy has to be proportional to $f_s \sim |t|^{2-\alpha}$, so $d/y = 2 - \alpha$ and

$$f_s(t) = |t|^{2-\alpha} f_s(t/|t|). \tag{3.63}$$

Since we set $b^y = \lambda$ we can find y:

$$y = \frac{\ln \lambda}{\ln b} \tag{3.64}$$

$$= \frac{\ln(3/2 \tanh 4K_c)}{\ln \sqrt{2}} \tag{3.65}$$

$$\approx 1.070, \tag{3.66}$$

so that

$$\alpha = 2 - \frac{d}{y} \approx 0.131. \tag{3.67}$$

The exact result from Onsager's solution is $\alpha = 0$, so we are still a ways off. More involved RG calculations of the two-dimensional Ising model are able to get much more accurate results. Simple RG calculations such as this one are able to provide insights to the critical behaviour of the model however, and was included in this thesis to compare to the flow of configurations of restricted Boltzmann machines.

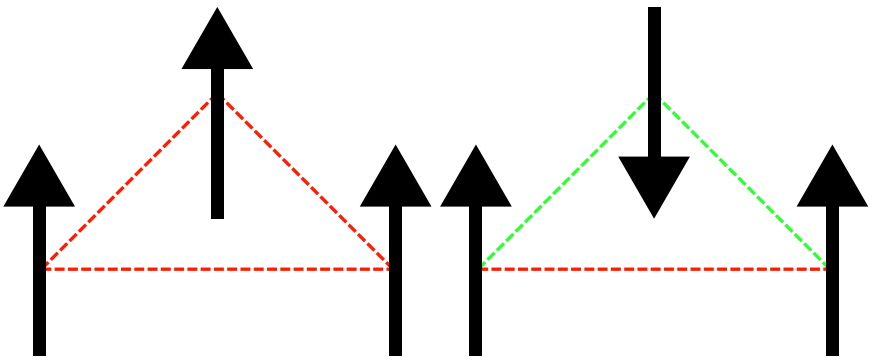
3.3.3 Two-dimensional AF Ising Model on a triangular lattice

Another model which appears very similar to the two-dimensional Ising model on a square lattice is the two-dimensional anti-ferromagnetic Ising model on a triangular lattice (referred to in short as TIAF). The Hamiltonian looks similar to that of the two-dimensional square Ising model:

$$H = J \sum_{(i,j)} s_i s_j, \tag{3.68}$$

where (i, j) indicates the sum over all nearest neighbours of site i. Note that the minus-sign in front of the interaction strength J has disappeared and that $J > 0$ still holds. This change means that the system will prefer to anti-align spins rather than align them as for the Ising model on a square lattice. Since the system is on a triangular lattice the number of nearest neighbours has increased to 6 compared to 4 on a square lattice.

However, these seemingly simple changes lead to drastically different behaviour of the model. To see how the system behaves in the low-temperature limit we first consider a single triangle lattice, such as the one shown in figure 3.5. There are $2^3 = 8$ different spin-configurations possible on this lattice. Of those 8, 2 are the states with all three spins pointing either all up or all down, as in figure 3.5a and its spin-flipped state. Obviously these states have the highest energy of all possible states since none of the spins anti-align. The 6 other states will constitute of states with either two spins up and one down or one spin up and two down, one of these configurations is shown in figure 3.5b. These states all have two anti-aligned spin pairs and thus all have the same energy. On this simple triangle lattice the system thus has a sixfold degenerate groundstate. Another way to look at this is to start the triangle configuration with an anti-aligned spin pair. The third (and last) spin can be either pointing up or down, it will not change the energy of the configuration. This ambiguity of not having a preferred spin direction for a particular spin in a configuration is called having a frustrated spin. This phenomenon of having frustrated spins is due to the geometric frustration induced by the triangular lattice: the model is said to not 'fit' on the lattice.



(A) *Not frustrated* (B) *Frustrated*

FIGURE 3.5: *Spin configurations on a triangle lattice. Aligned spins have a red line in between them, while anti-aligned spins have a green line in between them.*

This phenomenon stretches further than just on single triangle lattices, if one considers the zero temperature ground state of this model on a triangular lattice

this frustration stretches to the entire lattice. The degeneracy of the groundstate is quantified in terms of the residual entropy: the entropy of the system at zero temperature. A lower bound for this system can be found quite easily. One possible groundstate for the TIAF is shown in figure 3.6, where hexagonal spin configurations with frustrated spins in the middle are shown. In this configuration a third of all spins are frustrated, so that this configuration has a degeneracy of $2^{L/3}$. This gives us a lower bound on the residual entropy:

$$\frac{S(0)}{L} \geq \frac{1}{3} \ln 2 \approx 0.210. \tag{3.69}$$

It is actually possible to analytically calculate the residual entropy for the TIAF, as was first done by Wannier[54] in 1950. The residual entropy in the thermodynamic limit is equal to

$$\frac{S(0)}{L} = \frac{3}{\pi} \int_0^{\pi/6} \mathrm{d}\omega \, \ln\left(2 \cos \omega\right) \approx 0.338314. \tag{3.70}$$

The TIAF actually does not undergo a phase transition, unlike the anti-ferromagnetic Ising model on a square lattice. That being said, there is a transition from being near frustrated ground states to pure temperature-induced disorder. There is no phase transition associated with this change since there does not appear to be any discontinuity in the free energy in the thermodynamic limit.

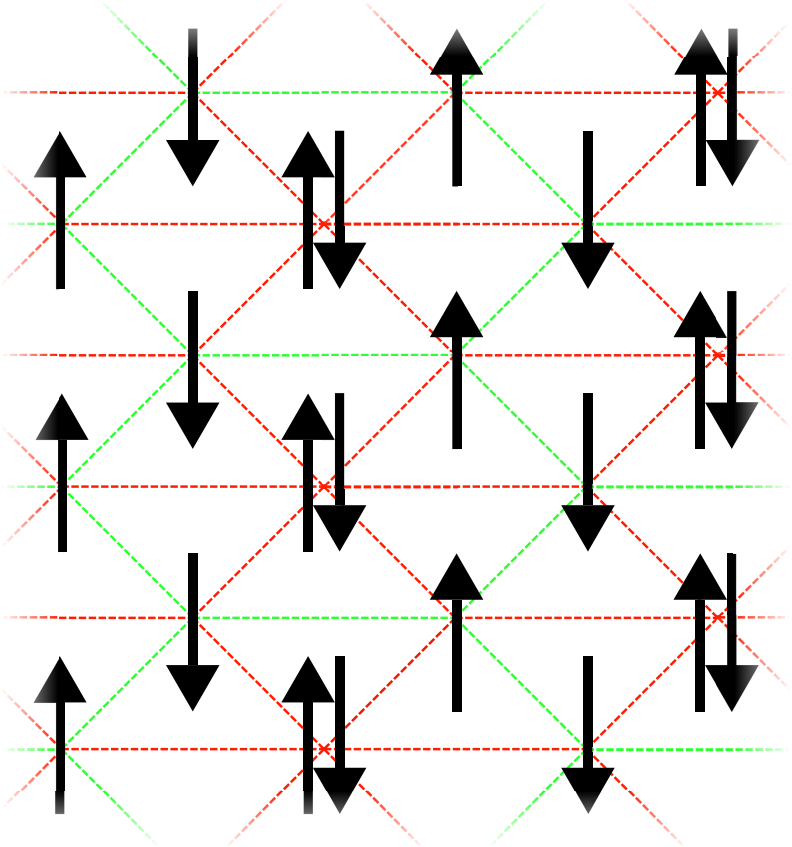


FIGURE 3.6: *Example of frustrated ground state of the anti-ferromagnetic Ising model on a triangular lattice. Sites with spins pointing in both directions have frustrated spins.*

The transition from order to disorder

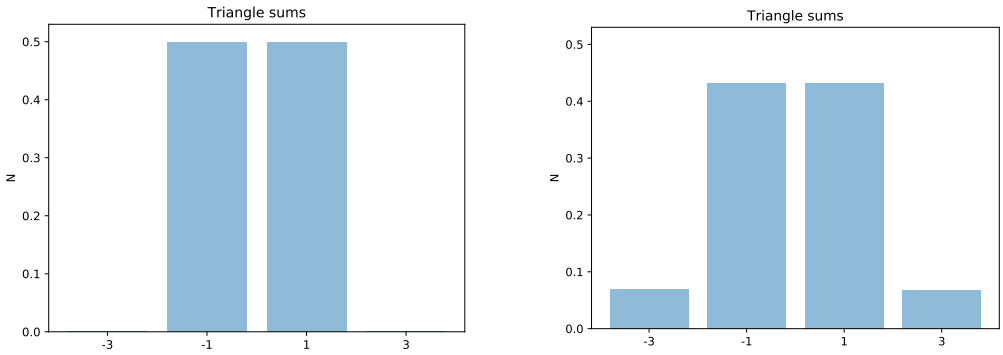
Even though there is no phase transition, there are still indicators tracking the transition from frustrated groundstates to temperature-induced disorder. For example, one could imagine that counting the amount of frustrated triangles versus not frustrated triangles would be an indication for the type of phase the system is in. This can be quantified simply by summing over the spins on each triangle vertex. If we sum over the upper triangle $\triangle$ and lower triangle $\triangledown$ for each site, where the site is always the left-most corner of the triangle, we sum over all triangles in the triangular lattice. The individual summations of the triangles can give 4 possible outcomes:

$$\sum_{\triangle_i} s_i = \begin{cases} -3, & \text{all spins down} \\ -1, & \text{frustrated} \\ 1, & \text{frustrated} \\ 3, & \text{all spins up} \end{cases}, \qquad \sum_{\triangledown_i} s_i = \begin{cases} -3, & \text{all spins down} \\ -1, & \text{frustrated} \\ 1, & \text{frustrated} \\ 3, & \text{all spins up} \end{cases}. \tag{3.71}$$

We can then obtain a normalised triangle-histogram by summing over all sites and dividing by the number of sites L and 2 for counting each site double. The results of such a calculation is shown in figure 3.7, using Monte Carlo simulations at low temperature ($T = 0.1$) and high temperature ($T = 4.5$). The figure shows that this triangle sum distribution behaves as we would expect.

For low temperatures we would expect nearly all triangles to be in a frustrated state, as in the ground state configuration of figure 3.6. There is no preference for any spin direction, so we expect the -1 and 1 sums to occur at the same frequency. In terms of the histogram we would expect values of around ~ 0.5 of -1 and $+1$, as we see in the histogram of the low temperature simulation.

For high temperatures we expect total temperature-induced disorder, so that essentially only the combinatorics of filling the triangular lattice plays a role. As was discussed at the start of this section, for an individual triangle $1/4$ of the possible configurations are not frustrated configurations. These are the configurations with all spins up or down, which correspond to the triangle sums $+3$ and -3 respectively. The addition of multiple connected triangles does not change the probability of each triangle configuration occurring. This can be seen by first considering an empty triangular lattice. Creating a single triangle configuration anywhere on the lattice then follows the combinatorics of the single triangle lattice. There is a probability of $3/4$ to get $\sum_{\triangle} s_i = \pm 1$ and $1/4$ to get $\sum_{\triangle} s_i = \pm 3$. If we build the lattice from here, by placing spins in adjacent sites to the triangle, the new created triangles will have the same sum probabilities. We can thus consider the individual triangle sum probability for each triangle on the lattice, so that we expect triangle sums with ± 1 with chance $3/4$, and triangle sums with ± 3 with chance $1/4$. Since there is again no preference for spins up or down, the chances will split evenly between the $\pm$ sums.



(A) $T = 0.1$ (B) $T = 4.5$

FIGURE 3.7: *Histograms of the normalised triangle sums obtained from Monte Carlo simulations of the TIAF of size* $L = 36 \times 36$ *at two different temperatures.*

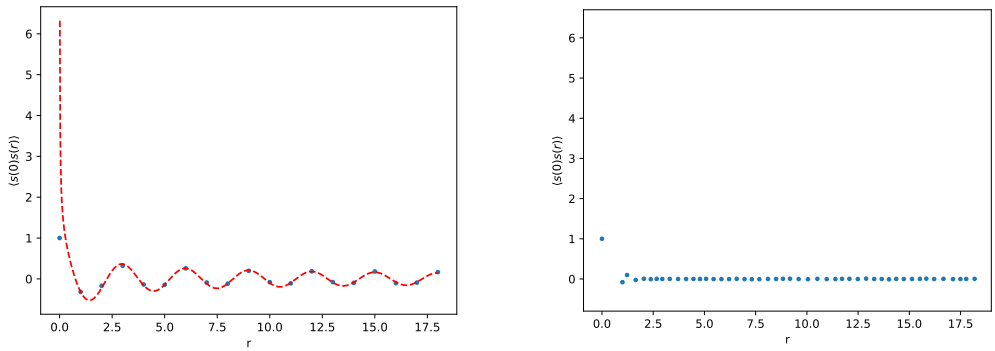
The two-spin correlation function $\langle s(0)s(r)\rangle$ could also be used as an indicator for the phase the system is in. For low temperatures there is clearly a sense of order, so one would expect a non-vanishing correlation function. For zero temperature in the thermodynamic limit this correlation function can be calculated exactly, and shown to behave as[55]

$$\langle s(0)s(r)\rangle \propto \epsilon_0 r^{-\frac{1}{2}} \cos\left(\frac{2}{3}\pi r\right), \tag{3.72}$$

where ϵ_0 is a constant defined in terms of E_0^T, which is the decay amplitude of the pair correlation at the critical point of an isotropic ferromagnetic triangular lattice:

$$\epsilon_0 \equiv \sqrt{2}(E_0^T)^2. \tag{3.73}$$

For high temperatures we expect the correlations to vanish, since the spins are completely independent of each other. A comparison between the two-spin correlations at low and high temperatures from Monte Carlo simulations is shown in figure 3.8, where equation (3.72) is plotted with the low temperature case.



(A) $T = 0.100$ (B) $T = 4.100$

FIGURE 3.8: *Two-spin correlation* $\langle s(0)s(r)\rangle$ *as a function of the spin-spin distance r from Monte Carlo simulations at different temperatures (blue dots) of a* 36×36 *lattice. The dashed red line is equation* (3.72).

3.3.4 *xy*-Model

Another classical model of spins is the xy-model. In this model the spins are no longer restricted to the binary up-down states, but are modelled as two-dimensional vectors free to point anywhere in the lattice plane (which we will call the xy-plane for obvious reasons). The interaction between the spins is modulated by the dot product between neighbouring vectors. The Hamiltonian of the system then takes the following form:

$$H = -J \sum_{(i,j)} \mathbf{s}_i \cdot \mathbf{s}_j \tag{3.74}$$

$$= -J \sum_{(i,j)} \cos(\theta_i - \theta_j). \tag{3.75}$$

Here the sum over (i,j) indicates a sum over all nearest neighbour pairs, $\mathbf{s}_i$ is the spin-vector at site i and $\theta_i \in [0, 2\pi)$ is the angle of the spin-vector at site i with regards to the x-direction. We consider the xy-model on a square lattice.

While the spins are now two-dimensional vectors, the Hamiltonian is still only dependent on one degree of freedom per spin: the angle $\theta(\mathbf{x})$, defined at each site $\mathbf{x}$ on the lattice. Since the spin-interaction only depends on the difference between the angles of neighbouring spins, it is invariant under $O(2)$ symmetry transformations of $\mathbf{s}$. An order parameter that breaks this symmetry is the average over all spins, the two-dimensional vector $\langle \mathbf{s} \rangle = s(\cos\phi, \sin\phi)$, whose direction and magnitude are specified by the angle ϕ and the scalar s respectively. In an ordered phase where all the spins are pointing in a similar direction $\langle \mathbf{s} \rangle$ will point in the average direction of all the spins with some finite s. In a disordered phase, where the spins are pointing in random directions, $\langle \mathbf{s} \rangle$ will vanish. For low finite temperatures the system will actually be in a quasi-long-range order (QLRO) phase, where the two-spin correlations decay algebraically, as will be shown. The average spin $\langle \mathbf{s} \rangle$ will actually vanish in this QLRO phase in the thermodynamic limit.

Free energy in the low-temperature region

It is straightforward to see that having all spins pointing in the same direction will minimise the energy. The ground state energy is invariant under spatially uniform changes of $\theta(\mathbf{x})$, as this will just rotate all the spins but will not change their interactions. This corresponds to a change of $\langle \mathbf{s} \rangle$ in ϕ only. However, spatially non-uniform changes, such as twists, splays and bends, will change the energy. This change in energy can be formalised in terms of the elastic free energy[56]

$$F_{el} = F[\theta(\mathbf{x})] - F[\theta = \text{const}], \tag{3.76}$$

where F is the free energy. This expression is expected to be analytic in $\nabla\theta$ and can be thought of as an expansion around the minimum free energy. Since any spatially non-uniform deformation of $\theta(\mathbf{x})$ will increase the free energy, there will be no linear term in $\nabla\theta$. The simplest form of F_{el} is then

$$F_{el} = \frac{1}{2} \int \mathrm{d}^2x \, \rho_s [\nabla\theta(\mathbf{x})]^2, \tag{3.77}$$

where ρ_s is called the spin-wave stiffness or helicity modulus. In two dimensions ρ_s has the units of energy. Note that F_{el} is invariant under uniform displacements

of θ and to uniform displacements and rotations of space. The helicity modulus can be defined in terms of the difference between the free energy $F[\theta_0]$, with boundary conditions $\theta = 0$ along the plane $z = 0$ and $\theta = \theta_0$ at $z = L$, and the free energy $F[0]$, with boundary conditions $\theta = 0$ at $z = 0$ and $z = L$:

$$\rho_s = \lim_{L\to\infty} 2(F[\theta_0] - F[0])/\theta_0^2. \tag{3.78}$$

This quantity can then be interpreted as the inherent response or resistance to a force that tries to impose a twist at the boundaries of the system.

Spin correlations and QLRO

To study more concretely the type of phase transition from an ordered to disordered phase one might naively expect, we consider the spin correlation function:

$$\begin{aligned}\langle \mathbf{s}(\mathbf{x}) \cdot \mathbf{s}(0)\rangle &= \langle \cos[\theta(\mathbf{x}) - \theta(0)]\rangle \\ &= \mathrm{Re}\langle \exp\left(\mathrm{i}[\theta(\mathbf{x}) - \theta(0)]\right)\rangle.\end{aligned} \tag{3.79}$$

This expression can be evaluated explicitly in the low-temperature regime where we approximate the Hamiltonian by F_{el}. The Fourier transforms of the fields fields are:

$$\theta(\mathbf{x}) = \int \frac{\mathrm{d}^2k}{(2\pi)^2}\hat{\theta}(\mathbf{k})\mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\mathbf{x}}, \tag{3.80}$$

$$\theta(0) = \int \frac{\mathrm{d}^2k}{(2\pi)^2}\hat{\theta}(\mathbf{k}), \qquad \text{and} \tag{3.81}$$

$$\int \mathrm{d}^2x\,[\nabla\theta(\mathbf{x})]^2 = \int \frac{\mathrm{d}^2k}{(2\pi)^2}k^2\hat{\theta}(\mathbf{k})\hat{\theta}(-\mathbf{k}). \tag{3.82}$$

Note that because $\theta(\mathbf{x})$ is real, $\hat{\theta}(-\mathbf{k}) = \bar{\hat{\theta}}(\mathbf{k})$, where the bar denotes taking the complex conjugate. This means that we can write the spin correlation function as

$$\langle \mathbf{s}(\mathbf{x}) \cdot \mathbf{s}(0)\rangle = \frac{\mathrm{Re}}{Z}\int \mathcal{D}\theta \exp\left[\int \frac{\mathrm{d}^2k}{(2\pi)^2}(\mathrm{i}\hat{\theta}_{\mathbf{k}}(\mathrm{e}^{-\mathrm{i}k\cdot x} - 1) - \frac{\beta\rho_s k^2}{2}\hat{\theta}_{\mathbf{k}}\hat{\theta}_{-\mathbf{k}})\right] \tag{3.83}$$

$$= \frac{1}{Z}\int \mathcal{D}\theta \exp\left[\int \frac{\mathrm{d}^2k}{(2\pi)^2}(-\frac{\beta\rho_s k^2}{2}\hat{\theta}_{\mathbf{k}}\hat{\theta}_{-\mathbf{k}} + \frac{(1-\mathrm{e}^{-\mathrm{i}k\cdot x})}{k^2\beta\rho_s})\right] \tag{3.84}$$

$$= \exp\left[-\int \frac{\mathrm{d}^2k}{(2\pi)^2}\frac{(1-\mathrm{e}^{\mathrm{i}k\cdot x})}{k^2\beta\rho_s}\right], \tag{3.85}$$

where $\mathcal{D}$ is a short-hand notation for the product of integration as well as containing any normalisation factors. The precise values of these factors are irrelevant in this calculations since they cancel with the partition function fraction. To obtain the second line we take the affine transformation $\hat{\theta}_{\mathbf{k}} \to \hat{\theta}_{\mathbf{k}} - \frac{\mathrm{i}}{k^2\beta\rho_s}(\mathrm{e}^{\mathrm{i}k\cdot x} - 1)$ and its complex conjugate for $\hat{\theta}_{-\mathbf{k}} = \bar{\hat{\theta}}_{\mathbf{k}}$. The integration left over θ is the same as the partition function, so only equation (3.85) is left.

Unfortunately the integral over k can not be evaluated analytically. In the limit $|\mathbf{x}| \to \infty$ this integral evaluates to[56]

$$\lim_{|\mathbf{x}|\to\infty}\int \frac{\mathrm{d}^2k}{(2\pi)^2}\frac{(1-\mathrm{e}^{\mathrm{i}k\cdot x})}{k^2\beta\rho_s} = \frac{1}{\beta 2\pi\rho_s}(\ln\Lambda|\mathbf{x}| + \gamma_E + \frac{1}{2}\ln 8 + \mathcal{O}((\Lambda|\mathbf{x}|)^{-3/2}), \tag{3.86}$$

where γ_E is the Euler-Mascheroni constant and Λ is a high wave-number cutoff. So the spin correlation function scales as

$$\langle \mathbf{s}(\mathbf{x}) \cdot \mathbf{s}(0) \rangle \approx \left(\tilde{\Lambda} |\mathbf{x}| \right)^{-\frac{1}{\beta 2\pi \rho_s}}, \tag{3.87}$$

where $\tilde{\Lambda} = \Lambda e^{\gamma_E + 1/2 \ln 8}$. The spin correlation function thus decays algebraically to zero with a temperature dependent exponent. The system is said to have quasi-long-range order, since the order-parameter correlation function decays with a power-law, provided T/ρ_s is not zero or infinite. If the helicity modulus tends to zero, the spin correlation function will decay faster than algebraically. In this case the system will transition from QLRO to disorder when $\rho_s \to 0$.

Vortex corrections to the free energy

The helicity modulus actually can decrease due to vortex corrections to the free energy. In the xy-model topological defects known as vortices can appear. These defects appear in order parameter space as singularities of the order parameter $\langle \mathbf{s}(\mathbf{x}) \rangle = s(\cos\theta(\mathbf{x}), \sin\theta(\mathbf{x}))$. These points are characterised by integrating over a closed contour C in real space enclosing these points:

$$v = \oint_C \nabla\theta \cdot \mathrm{d}\vec{\ell} = 2\pi k, \quad k = 0, \pm 1, \pm 2, ... \tag{3.88}$$

The integer k is called the winding number of the vortices enclosed by the contour. A nonzero winding number indicates the existence of a vortex somewhere enclosed in the contour.

An energy associated with the vortex can be calculated by introducing two non-intersecting cuts Σ^+ and Σ^- along the line on which θ undergoes a discontinuity of $2\pi k$. The vortex energy can be calculated using equation (3.77):[56]

$$\begin{aligned} E_{el} &= \frac{1}{2} \int \mathrm{d}^2 x \rho_s (\nabla\theta)^2 \\ &= \frac{1}{2} \rho_s \left(\int \theta (\nabla\theta) \cdot \mathrm{d}\mathbf{\Sigma} - \int \mathrm{d}^2 x \, \theta \nabla^2 \theta \right) \qquad (3.89) \\ &= \frac{1}{2} \rho_s \left(\int \theta^+ (\nabla\theta) \cdot \mathrm{d}\Sigma^+ + \int \theta^- (\nabla\theta) \cdot \mathrm{d}\Sigma^- \right) \\ &= \frac{1}{2} \rho_s (\theta^+ - \theta^-) \int_a^R \mathrm{d}r \, |\nabla\theta| \\ &= \pi k^2 \rho_s \ln(R/a), \qquad (3.90) \end{aligned}$$

where the observation that the configuration obtained by minimising the free energy means that it will obey $\rho_s \nabla^2 \theta = 0$ was used. a is the lattice constant and R the linear dimension of the system. Note that this energy appears to diverge for large system sizes. However, if another vortex is added the energy of the two vortices with winding numbers k_1 and k_2 is

$$\begin{aligned} E_{el} &= E_1 + E_2 + 2\pi \rho_s k_1 k_2 \ln(R/r) \qquad (3.91) \\ &= \rho_s \pi (k_1 + k_2)^2 \ln(R/a) + 2\pi \rho_s k_1 k_2 \ln(a/r), \qquad (3.92) \end{aligned}$$

where E_1 and E_2 are the energies of the isolated vortices and r is the distance between the two vortices. Note that the divergent lnR will disappear if $k_1 = -k_2$, so when the

vortex is accompanied by another vortex with an opposite winding number, which we will call its anti-vortex. Generally, if the sum over all winding numbers in the system is zero the divergent term will vanish.

It can be shown[56] that including thermally excited vortex pairs lead to a reduction in the macroscopic spin-wave stiffness ρ_s^R:

$$\rho_s^R = \rho_s - \frac{\rho_s^2}{(d-1)T} \int \mathrm{d}^d x \langle \mathbf{v}_s^\perp(\mathbf{x}) \cdot \mathbf{v}_s^\perp(0) \rangle. \tag{3.93}$$

To get to this equation one has to split the gradient $\mathbf{v}_s = \nabla\theta(\mathbf{x})$ of the microscopic angle θ in an analytic and singular part arising from vortices. The velocity can then be split in a longitudinal (analytic) and transverse (vortex) part $\mathbf{v}^\perp = \nabla\theta_{\text{sing}}$. The spin-correlation function is then also controlled by the renormalised ρ_s^R. The phase transition can be understood as a transition from an ordered phase with an increasing number of vortex pairs at increasing temperatures to a disordered phase where the vortex pairs are said to unbind. This type of phase transition does not have a diverging derivative of the free energy to any order and is referred to as a phase transition of infinite order. In literature this phase transition is often called a Kosterlitz-Thouless (KT) transition, named after the scientists who first described this kind of phase transition[57].

Critical temperature

The location of the phase transition can be determined by analysing the RG equations near the critical point[58]. The critical temperature then occurs when the equation

$$T_c = \frac{\rho_s^R \pi}{2} \tag{3.94}$$

holds. By these same RG equations it is also possible to infer the finite-size scaling behaviour of the critical temperature:

$$T_c(L) = \frac{\pi^2}{4c \ln^2(L)} + T_c. \tag{3.95}$$

where c is a constant to be fitted in the finite-size scaling analysis. The critical temperature as determined by simulations is $T_c \approx 0.893$ [59–61].

3.3.5 Coulomb Gas

The next model is technically not a spin-model, although it can be cast to one. The two-dimensional Coulomb gas on a square lattice is a model of charges interacting with the discrete Coulomb interaction in two dimensions. The model can be formulated simply as

$$H = \frac{1}{2}\sum_{i,j} q_i V(\mathbf{r}_i - \mathbf{r}_j) q_j, \tag{3.96}$$

where q_i is the charge on site i, $\mathbf{r}_i$ is the distance from the origin to site i, and $V(\mathbf{r})$ is the lattice Coulomb potential in two dimensions. The lattice Coulomb potential can be derived from

$$\Delta^2 V(\mathbf{r}) = -2\pi\delta_{\mathbf{r},0}, \tag{3.97}$$

where Δ^2 is the discrete form of the Laplacian, defined as

$$\Delta^2 f(\mathbf{r}) = \sum_{\hat{\mu}} [f(\mathbf{r}+\hat{\mu}) - 2f(\mathbf{r}) + f(\mathbf{r}-\hat{\mu})]. \tag{3.98}$$

The $\hat{\mu}$ are unit vectors pointing from the site at $\mathbf{r}$ to its nearest neighbours. For the square lattice this is just

$$\{\hat{\mu}\} = \{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}, \tag{3.99}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the usual unit vectors in the x- and y-directions respectively. The square lattice is thus taken to lie in the xy-plane. An expression for the Coulomb potential can be found by substituting the Fourier transforms

$$V(\mathbf{r}) = \frac{1}{L}\sum_k V_k \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{r}} \tag{3.100}$$

$$\delta_{\mathbf{r},0} = \frac{1}{L}\sum_k \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{r}} \tag{3.101}$$

into equation (3.97):

$$\frac{1}{L}\sum_{\{\hat{\mu}\}}\sum_k V_k \left[\mathrm{e}^{\mathrm{i}\mathbf{k}\cdot(\mathrm{r}+\hat{\mu})} - 2\mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathbf{r}} + \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot(\mathbf{r}-\hat{\mu})}\right] = -\frac{2\pi}{L}\sum_k \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\mathrm{r}}. \tag{3.102}$$

Performing the sum over $\{\hat{\mu}\}$ and equating the terms of the k-summation gives:

$$\begin{aligned} V_k &= \frac{-2\pi}{\mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\hat{\mathbf{x}}} + \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\hat{\mathbf{x}}} + \mathrm{e}^{\mathrm{i}\mathbf{k}\cdot\hat{\mathbf{y}}} + \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\hat{\mathbf{y}}} - 4} \\ &= \frac{\pi}{2 - \cos\mathbf{k}\cdot\hat{\mathbf{x}} - \cos\mathbf{k}\cdot\hat{\mathbf{y}}} \end{aligned} \tag{3.103}$$

This equation diverges if $\mathbf{k} \to 0$. This problem can be averted by defining a non-diverting Coulomb potential:

$$V'(\mathbf{r}) \equiv V(\mathbf{r}) - V(\mathbf{r}=0) = \frac{1}{L}\sum_k V_k(\exp(\mathrm{i}\mathbf{k}\cdot\mathbf{r}) - 1). \tag{3.104}$$

We can rewrite the Hamiltonian in terms of this non-diverting potential:

$$\begin{aligned} H &= \frac{1}{2}\sum_{i,j} q_i V'(\mathbf{r}_i - \mathbf{r}_j) q_j + \frac{1}{2} V(\mathbf{r}=0) \sum_{i,j} q_i q_j \\ &= \frac{1}{2}\sum_{i,j} q_i V'(\mathbf{r}_i - \mathbf{r}_j) q_j + \frac{1}{2} V(\mathbf{r}=0) (\sum_{i} q_i)^2. \end{aligned} \tag{3.105}$$

Since $V(\mathbf{r}=0)$ diverges, configurations with $\sum_i q_i \neq 0$ will have infinite energy and will thus not contribute to the partition sum and physics of the model. This means that the final Hamiltonian for the system can be written as

$$H_{\mathrm{CG}} = \frac{1}{2}\sum_{i,j} q_i V'(\mathbf{r}_i - \mathbf{r}_j) q_j, \tag{3.106}$$

with the extra constraint that

$$\sum_i q_i = 0, \tag{3.107}$$

so that the system is charge neutral and charge is conserved. The charges can take the values $q \in \{-1, 1\}$, the model can then be thought of as spins interacting with an interaction strength $V'(\mathbf{r}) \propto \ln \mathbf{r}$.

Phase transitions and order parameters

It turns out that this model can be cast to a special case of the xy-model with alternating interaction strengths ± 1 between even-odd and odd-even nearest neighbour sites at low temperatures[62]. This connection is used as an argument that the Coulomb model contains a KT-like phase transition. Conceptually this phase transition can be understood as an unbinding of neutral charge-pairs to a mixture of free charges and bounded neutral charge-pairs. This is conceptually similar to the vortex-binding and unbinding of the regular xy-model. However, a derivation of RG equations of the Coulomb Gas model[63] has shown a few differences between the usual KT RG equations, which are believed to be obeyed for systems undergoing a KT transition. This derivation, while approximate, provides an indication that the system can undergo both a continuous (KT-like) phase transition and a discontinuous (Ising-like) phase transition. This prediction is also supported by Monte Carlo simulations[64], which measured two phase transitions in very close proximity to one-another. They found

$$T_{\mathrm{KT}} \approx 0.516 \pm 0.008, \quad \text{and} \qquad T_{\mathrm{I}} \approx 0.532 \pm 0.004, \tag{3.108}$$

where the subscripts KT and I denote the KT-like and Ising-like phase transitions respectively.

The different phase transitions are characterised by different order parameters. The order parameter for the Ising-like phase transition is the same as the one typically used for the anti-ferromagnetic Ising model on a square lattice: the staggered magnetisation. The staggered magnetisation is defined as

$$M_S = \sum_i^L (-1)^i q_i. \tag{3.109}$$

At low temperatures the system is most likely in the ground state, which consists of a chequerboard pattern of positive and negative charges. In this state the staggered magnetisation will be either $+1$ or -1, depending on the charge of the first site. The ground state is thus considered doubly degenerate and they are separated by any global odd-numbered translation of the charges.

The order parameter for the KT-like phase transition is similarly analogous to the one used for the classical xy-model: the spin-wave stiffness. The inverse dielectric constant serves the same role as the spin-wave stiffness in the xy-model. From linear response theory one can derive an expression for the inverse dielectric constant[65]:

$$\epsilon^{-1}(\mathbf{k}) = \lim_{\mathbf{k}\to 0}\left[1 - \frac{2\pi\beta}{k^2 L}\langle q_{\mathbf{k}} q_{-\mathbf{k}}\rangle\right], \tag{3.110}$$

where q_k is the Fourier transform of q:

$$q_{\mathbf{k}} \equiv \sum_i q_i \mathrm{e}^{-\mathrm{i}\mathbf{k}\cdot\mathbf{r}_i}. \tag{3.111}$$

This quantity goes from 1 in the ordered (chequerboard) phase to 0 when the neutral charge pairs decouple.

Finite-size scaling

Since we are interested in simulating the Coulomb Gas model at finite sizes, we will need finite-size scaling relations to obtain information about physical quantities in the thermodynamic limit. The scaling relations of physical quantities related to the Ising-like transition (magnetisation, magnetic susceptibility) are expected to behave according to equations (3.33)-(3.35).

Similarly, ϵ^{-1} is expected to scale as ρ_s^R in the xy-model. For a system of size L it is expected to scale as[66]

$$\epsilon^{-1}(T_c, L) = \epsilon_\infty^{-1}\left[1 + \frac{1}{2\ln L + c}\right], \tag{3.112}$$

where $\epsilon_\infty^{-1} \equiv \epsilon^{-1}(T_c, \infty) = 4T_c$ and c is a fitting parameter. The scaling of the critical temperature for the KT transition is then the same as for the xy-model and is given by equation (3.95).

Chapter 4

Neural Network analysis of phase transitions

The goal of this numerical analysis using neural networks is to probe the usefulness of neural networks in numerical condensed matter physics tasks. This chapter will focus specifically on the detection of phase transitions in classical statistical physics systems using neural networks, and what the neural network is actually learning. We would expect the neural network to learn markers discriminating between phases related to the order parameter that is used for theoretical work. The hope is then that the neural networks is also capable of distinguishing between multiple phases in matter where there might not be a clear order parameter present, or where the phase separation is not so clearly defined.

To this end, we study a selection of classical statistical physics systems with neural networks. Each physical system displays different difficulties that can be used to test the capabilities of the neural network. A brief theoretical introduction to each of the treated physical models is given in chapter 3. An introduction to neural networks was given in chapter 2. Here we first consider the two-dimensional Ising model on a square lattice. We then move on to the xy-model, followed by the antiferromagnetic Ising model on a triangular lattice. We close the discussion with the two-dimensional Coulomb Gas model. More detail regarding the Monte Carlo simulations that were done to obtain configurations to train the neural networks can be found in appendix A.

4.1 Two-dimensional Ising model on a square lattice

The archetypal classical model to study any novel numerical technique on is the two-dimensional Ising model on a square lattice, as described in chapter 3. This is because it is an extremely well studied model where analytic results are known in the thermodynamic limit and finite-size effects are well documented in numerical studies. In this paper the two-dimensional Ising model is used to train a feed-forward neural network as described in chapter 2 to distinguish between the ordered and disordered phases of the system.

4.1.1 Training the neural network

To train the feed-forward neural network, Monte Carlo simulations of the two dimensional Ising model on a square $L \times L$ lattice were performed for multiple lattice sizes $L = \{20, 30, 40, 50, 60, 70\}$ at temperatures ranging from $T = 1$ to $T = 3.5$ with temperature steps $\Delta T = 0.1$ using the Wolff algorithm[67] which is described in more detail in appendix A. At each temperature 2000 configuration snapshots are taken which will be used to feed the neural network.

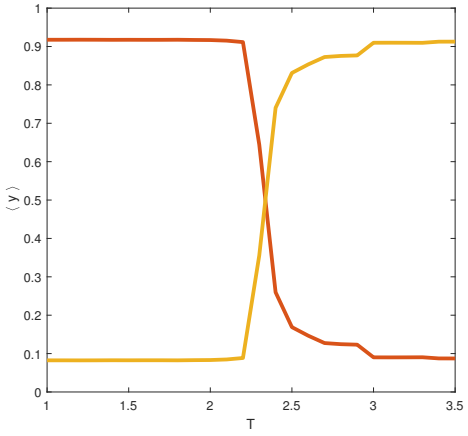
The feed-forward neural network consists of an input layer of $L \times L$ neurons, which is connected to a hidden layer of just 3 sigmoid neurons, which is sufficient to classify the different phases. The hidden layer then in turn connects to an output layer of two softmax neurons, which correspond to the probability of the system being in the ordered or disordered phase. The neural network is then fed a batch of configuration snapshots with accompanying label. During the training phase the neural network is only fed configurations sufficiently far from the critical point so that the phase of the configuration is well established. The neural network measures its success by calculating the cross entropy plus L_2-regularisation terms of the weight matrix $\mathbf{W}$ and biases $\mathbf{b}$ associated with each layer to make sure the values are kept sufficiently small. The neural network then learns by minimising this value using the Adam optimiser algorithm[68], which is a more sophisticated algorithm based on the basic stochastic descent algorithm.

The system is trained using mini-batches of 10 configurations, with one epoch of training consisting of 11500 mini-batches. The training continues until the loss-function no longer seems to decrease and start to fluctuate around a fixed value, which tends to happen after about 10 epochs of training. Once the training is complete the system is fed a test set of configurations it has not seen in the training phase to obtain an unbiased estimate of the true accuracy of the training. To analyse what the neural network has learned it is fed all the configurations for each temperature step at a time, where the average of the output neurons as well as the average of the hidden layer neurons is calculated and saved for analysis.

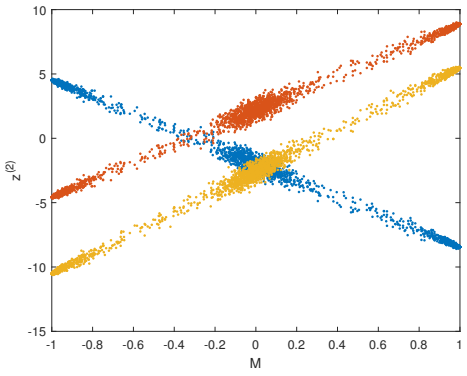
4.1.2 Neural network results for a single system size

As an example of the kind of results one gets for a trained feed-forward neural network we will discuss the results of the 50×50 model in depth. Figure 4.1a shows the typical average output values of the output layer neurons $\langle \mathbf{y} \rangle$ of a trained feed-forward neural network fed configurations of the 50×50 Ising model at different temperatures T. We then take the value where the system is most uncertain of which phase it is in (when $y_1 = y_2 = 0.5$) as the critical temperature $T_c(L)$. Via simple linear

interpolation we find $T_c(L) \approx 2.3486$. To find an approximation of the actual critical temperature T_c, the finite-size critical temperature needs to be determined for multiple system sizes. Note that this would only be true if the neural network's output depends on some physical quantity which scales according to the scaling ansatz.



(A) *The average output values of the output layer* $\mathbf{y}$ *(blue and red lines) at different temperatures* T.



(B) *The input values of the hidden layer* $\mathbf{z}^{(2)}$ *for configurations at magnetisation* M. *Each colour corresponds to a different hidden neuron.*

FIGURE 4.1: *Graphs of the neural values of a trained feed-forward neural network with 3 hidden neurons of a* 50×50 *square Ising model.*

To investigate this further we consider the input values of the activation function of the hidden layer $\mathbf{z}^{(2)}$ as a function of M. The results of feeding the trained neural network a multitude of configurations with corresponding magnetisation M on $\mathbf{z}^{(2)}$ is shown in figure 4.1b. Clearly there exists a linear relation between the output of the hidden layer and the magnetisation, which leads us to believe the neural network as effectively learned the order parameter of the system, without having any feature-engineering done explicitly beforehand. This follows the same argument in the appendix of the paper by Carrasquilla et al[25], where a toy model of a neural network with only 3 hidden neurons was created which could separate the phases of the two-dimensional Ising model.

4.1.3 A toy model to explain learned behaviour

To summarise Carrasquilla et al's[25] argumentation, they started with a neural network with 3 hidden perceptrons: neurons with a Heaviside step activation function. The first perceptron activates when the system is in a state with mostly up spins, the second perceptron activates when the system is in a state with mostly down spins and the third perceptron activates when the system is unpolarised or in a state with mostly up spins. They then define the first weight matrix and bias connecting the input layer to the hidden layer as

$$\mathbf{W}^{(2)} = \frac{1}{N(1+\epsilon)} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ -1 & -1 & \cdots & -1 \\ 1 & 1 & \cdots & 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{b}^{(2)} = \frac{\epsilon}{\epsilon+1} \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, \tag{4.1}$$

where $0 < \epsilon < 1$ is a parameter left free for now. In terms of an Ising configuration $\mathbf{x} = (s_1, s_2, ..., s_{L\times L})$ the input for the activation function of the hidden layer becomes

$$\mathbf{W}^{(2)}\mathbf{x} + \mathbf{b} = \frac{1}{1+\epsilon}\begin{pmatrix} M-\epsilon \\ -M-\epsilon \\ M+\epsilon \end{pmatrix}. \tag{4.2}$$

So the first perceptron activates when $M > \epsilon$, the second perceptron activates when $M < -\epsilon$ and the third perceptron activates when $M > -\epsilon$. The parameter ϵ thus controls the size of the region of M where two perceptrons are active to distinguish between the ordered and disordered state and is taken to be $0 < \epsilon \ll 1$. The output layer can then be rather arbitrarily chosen to be

$$\mathbf{W}^{(3)} = \begin{pmatrix} 2 & 1 & -1 \\ -2 & -2 & 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{b}^{(3)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{4.3}$$

This particular parametrisation guaranties that the output neuron for the ordered state will be active if only the first or second perceptron are active. If the third perceptron is active but the first perceptron is not, the output neuron for the disordered state will be active. This simple toy neural network is then able to distinguish between the ordered and disordered state for a given Ising configuration by effectively having learned the magnetisation. We state that this argumentation then also holds for the numerically trained neural network, which showed a clear linear dependence in the hidden layer on the magnetisation of the given Ising configuration. Thus we say that our neural network has effectively learned the magnetisation, and so the order parameter, of the system by itself.

4.1.4 Extrapolating to the thermodynamic limit

Since the system has learned to distinguish the different phases using a physical quantity of the system, finite-size scaling can be used to extrapolate the predicted critical temperature. By training 10 neural networks on systems of different sizes and obtaining the average of the finite-size critical temperature of these systems of different sizes, one is able to fit the finite-size scaling equation (3.36) with $\nu = 1$ to the data. We obtain using a weighted least-squares method the parameters $a = 2.7 \pm 1.3$ and $T_c = 2.254 \pm 0.029$, where the uncertainty corresponds to a 2σ deviation. As can be seen in figure 4.2, the error for the 20×20 system is quite large, a lot larger than the other system sizes. This is due to some of the neural networks settling into unfavourable minima of the cost function, never reaching an acceptable accuracy. This is a consequence of training the 10 neural networks from different initial parameter values, but keeping the same learning rate. Nevertheless the obtained critical temperature T_c after just single training of the neural network for relatively few different system sizes is remarkably accurate. Figure 4.2 shows the least-squares fit of equation (3.36) to the measured $T_c(L)$.

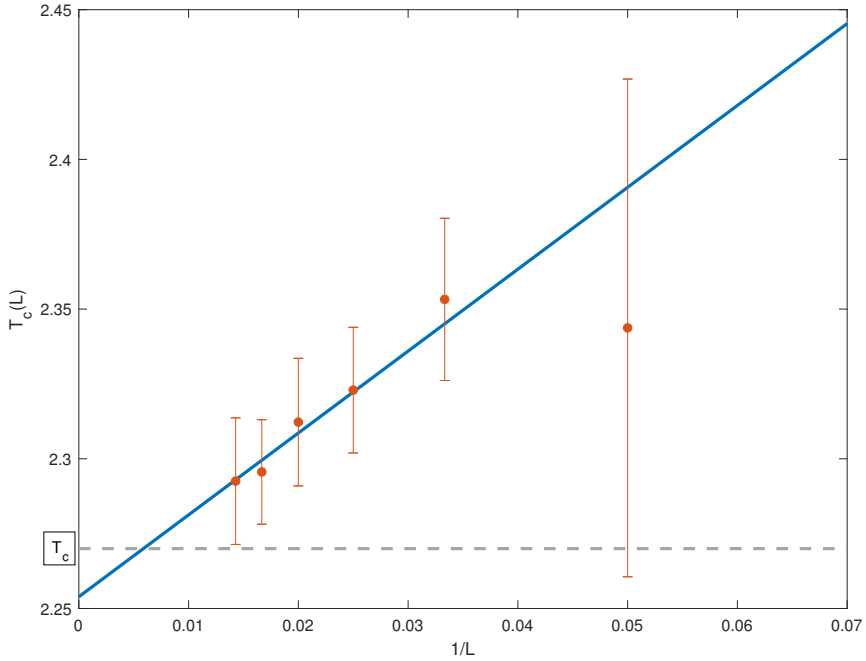


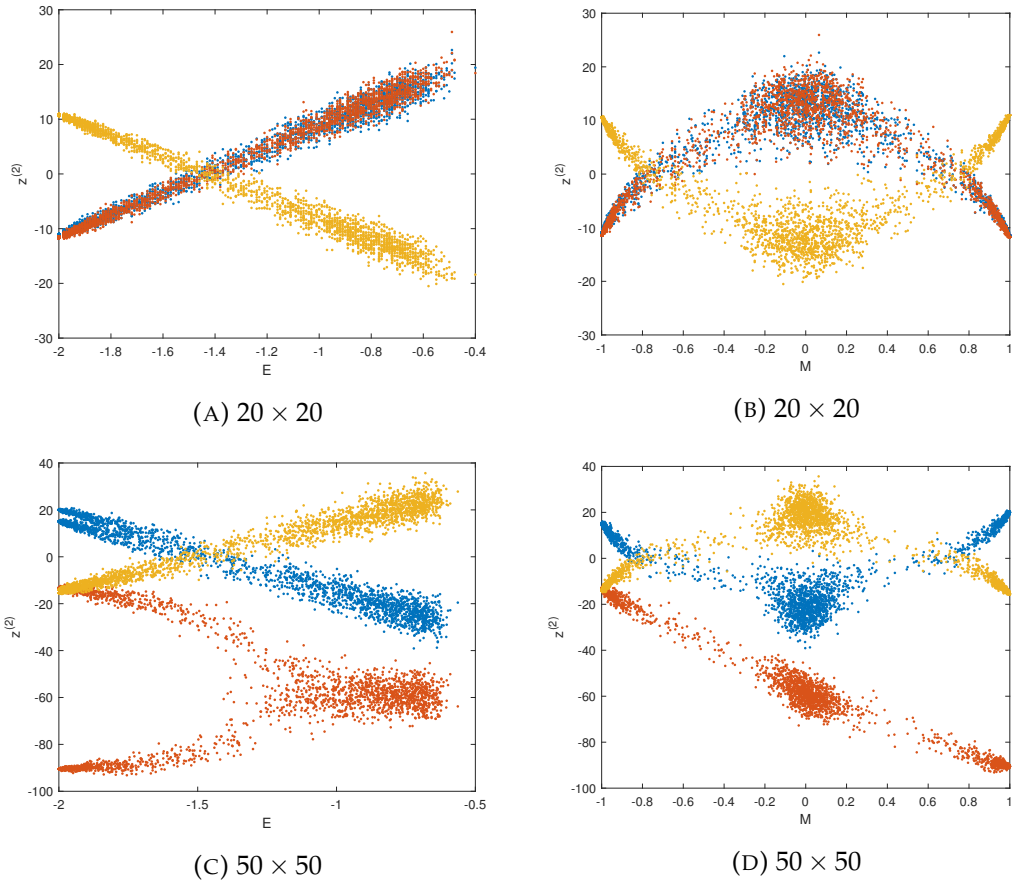
FIGURE 4.2: *Measured critical temperatures $T_c(L)$ of a two-dimensional Ising model on a square lattice at different system sizes $L \times L$ obtained from a feed-forward neural network plotted against the inverse system size $1/L$. The finite-sized critical temperatures averaged over 10 FNN iterations are displayed as orange dots. The errorbars show the standard deviation of the average. The blue line is a weighted least squares fit of equation* (3.36) *with $\nu = 1$ to the data. The grey dashed line is the analytic critical temperature in the thermodynamic limit T_c.*

4.1.5 Adding a convolutional layer

While the results for the feed-forward neural network are already enough to differentiate between the different phases in the Ising model, that might not be the case for more complicated models. That is why this subsection will focus briefly on the application of a convolutional neural network (CNN) as described in chapter 2 on the Ising model. We start with a relatively simple convolutional neural network of 4 filters of size 2×2 moving with unitary stride over the configurations. The configurations are padded such that periodic boundary conditions apply. The input layer then connects to the convolutional layers consisting of these 4 filters. The convolutional layer is then flattened and connects to a hidden layer of 3 sigmoid neurons, which connects to the output layer of 2 softmax neurons. Note that no pooling layer was used.

Training the convolutional neural network in the same way as was done for the feed-forward neural network gives excellent numeric results. The accuracy rate is near the 99% for the test set, with a cross entropy of order $\mathcal{O}(10^{-3})$. However, we are more interested in what the neural network has learned now that it has added a convolutional layer. By comparing the input arguments of the hidden layer $\mathbf{z}^{(2)}$ to the magnetisation M one can see that the linear relation of the feed-forward neural network has disappeared. If we compare $\mathbf{z}^{(2)}$ to the energy E however, a linear relation is once again visible. The $\mathbf{z}^{(2)}$-outputs for a trained convolutional neural network trained on a 20×20 Ising model as a function of the configuration energy E and magnetisation M is shown in figures 4.3a and 4.3b respectively. While the

trained neural network clearly shows a linear relation between the hidden layer inputs and the configuration energy E, this is not the case for CNN trained on larger Ising models. Figures 4.3c and 4.3d show $\mathbf{z}^{(2)}$ as functions of E and M respectively for a 50×50 Ising model. The system appears to now have learned both the magnetisation and the energy of the system. Further analysis of other system sizes shows differences in which hidden neurons learns what. Sometimes two hidden neurons have learned the magnetisation, while only one neuron is concerned with the energy of the system. The results seems dependent on the learning rate and batch size that are used to train the system.



(A) 20×20 (B) 20×20

(C) 50×50 (D) 50×50

FIGURE 4.3: *Hidden layer input arguments* $\mathbf{z}^{(2)}$ *of a trained convolutional neural network as functions of the magnetisation* M *and energy* E *for Ising models of sizes* 20×20 *and* 50×50. *The different colours correspond to the different hidden neurons.*

While at first glimpse it may seem useful to use both the magnetisation and energy as indicators for the phase transition, an finite-size scaling analysis of the final trained systems shows no improvement in accuracy. In fact, the scaling seems worse, giving an underestimation of the expected critical temperature. The results of the finite-size scaling analysis using equation (3.36) is shown in figure 4.4. The estimated critical temperature is $T_c = 2.224 \pm 0.0021$, where the uncertainty has been underestimated since no uncertainty in the finite-size critical temperatures were included. This leads us to conclude that while a convolutional neural network can be a great asset to learn different physical quantities, blind application of the network might not always give

the optimal results. Because of the sheer amount of free parameters the neural network is unlikely to end up in its global minimum of the loss function. Adding a convolutional layer to the neural network increases the complexity of the network (and thus the amount of local minima in the cost function), causing the predictability of the final trained network to decrease. For finite-size analyses this predictability is necessary to ensure the network always learns the right parameters for every system size such that the finite-size scaling is consistent. This predictability problem may be partially circumvented by proper feature-engineering based on the system. This problem of making sure the finite-size scaling is correct is also prevalent in the xy-model, as is evident from Beach et al[24] and will be discussed later in this chapter.

The neural network analysis of the two-dimensional Ising model on a square lattice has shown us that even a simple feed-forward neural network is capable of learning, without any feature-engineering, the order parameter of the system to be able to distinguish between the two phases. Precisely because it has learned a physical quantity we are able to perform standard extrapolating procedures, such as fitting the finite-size scaling equations one would use for Monte Carlo simulations to the data. This has shown us that neural networks can be a viable tool to study phase transitions numerically while still being able to get physically meaningful results from it, at least for relatively simple models. The rest of this chapter will be concerned with further developing and studying the results of neural network analysis on more complicated physical systems.

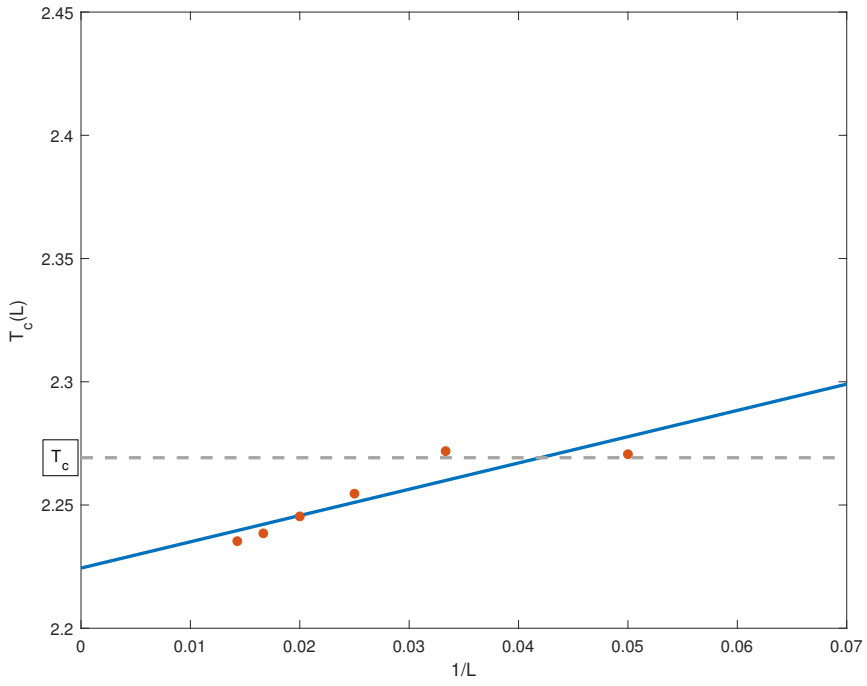


FIGURE 4.4: *Measured critical temperatures $T_c(L)$ of the two-dimensional Ising model on a square lattice at different system sizes $L \times L$ obtained from a convolutional neural network plotted against the inverse system size $1/L$. The finite-size critical temperatures are plotted as orange dots. The blue line is a least squares fit of equation (3.36) with $\nu = 1$ to the data. The grey dashed line is the expected critical temperature.*

4.2 xy-model

A system which displays an entirely different phase transition than the Ising model is the xy-model. While the system can still transition from an ordered (spins tend to point in the same direction locally) to a disordered state, like the Ising model, the $O(2)$ symmetry is actually preserved during the phase transition. So the xy-model does not experience spontaneous symmetry breaking as the Ising model did. The xy-model undergoes what is called a Kosterlitz-Thouless (KT) transition, as explained in section 3. Here we try to learn the KT-transition using neural networks.

4.2.1 Applying a feed-forward neural network

Naively we might start with a simple feed-forward neural network (FNN). Applying a neural network to configurations of the xy-model (obtained using Monte Carlo simulations with Wolff's algorithm[67]) gives reasonably acceptable accuracy rates of the FNN, around 88% over the test set within 20 epochs. A further analysis of the results for the different lattice sizes shows that the scaling of the critical temperature $T_c(L)$ for different lattice sizes $L \times L$ does not scale as expected. Figure (4.5) shows measured critical temperatures as a function of the lattice size. The figure clearly shows that the scaling of the critical temperature does not follow the expected scaling behaviour from equation (3.95). This is an indication that the neural network has not learned any physical observable, at least not consistently the same on for the different system sizes.

An analysis with just 3 hidden neurons similar to the one done in the previous section shows no indication that the system has learned the magnetisation or energy to distinguish between the two phases. This ambiguity in what it has learned is likely a result of the system settling into a local minimum of the loss-function, which does not appear to correspond to any obvious physical quantities. This is a common problem in neural network literature and the consensus seems to be that as long as the local minimum is performing well enough, the neural network has successfully learned to perform its task. However, to improve on the performance of these unrestricted neural networks settling into local minima some changes can be made. These changes can either be of the network itself, for example by adding a convolution layer or by placing restrictions on the fitting parameters, or by changing the format of the inputted data. In this section we will follow the lead of Beach et al[24] and apply a convolutional neural network to the configurations, as well as performing some feature-engineering to feed the convolutional neural network a vortex configuration instead of an angle configuration.

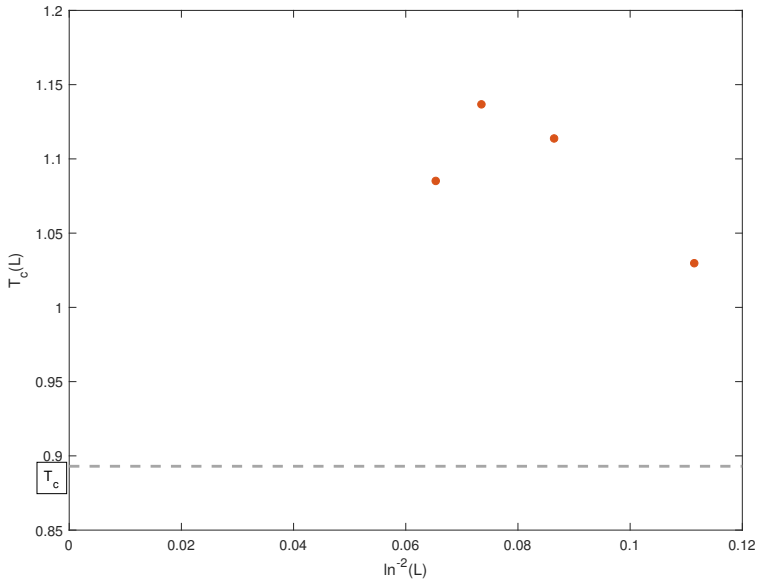


FIGURE 4.5: *Plot of the measured critical temperature $T_c(L)$ using a trained feed-forward neural network for different lattice sizes $L \times L$ of the xy-model. The grey dashed line is the expected value of T_c.*

4.2.2 Adding a convolution layer

One thing that might improve the performance of the neural network is to add a convolution layer in front of the hidden layer. By adding a set of filters which stride across the input configuration, the neural network might be able to pick up local translation invariant features of the system, such as the energy between neighbouring sites or the vorticity. The convolutional layer consists of four 2×2 filters which stride over each lattice site of the input configuration, applying periodic boundary conditions as padding. The number of hidden neurons is kept the same as for the feed-forward neural network (100 hidden neurons). The results of the critical temperatures as well as a least-squares fit of equation (3.95) is shown in figure 4.7. The fit gives $T_c = 1.029 \pm 0.164$, where again the uncertainty in $T_c(L)$ was not taken into account. While the result is near the expected value of $T_c \approx 0.893$, the uncertainty is still very high.

By looking at figure 4.7 one can see that the scaling of the finite-size critical temperatures is not convincing. Once again the problem appears to be the ambiguity in what the system picks up on. To get a better idea of what the neural network is learning we might look at the filters of the trained network. For the CNN trained on the 50×50 configurations the filters $\mathbf{f}$ have the values

$$f_1 = \begin{bmatrix} 0.283970 & -0.230723 \\ 0.267527 & -0.254183 \end{bmatrix}, \qquad f_2 = \begin{bmatrix} -0.224006 & -1.108324 \\ -0.673558 & 0.622721 \end{bmatrix},$$
$$f_3 = \begin{bmatrix} 0.822974 & 0.391843 \\ -0.054578 & 0.181431 \end{bmatrix} \quad \text{and} \quad f_4 = \begin{bmatrix} -0.577750 & -0.826530 \\ -0.620391 & 1.949122 \end{bmatrix}. \tag{4.4}$$

To make interpreting their forms more intuitive, the values have been transformed into a heatmap picture, as shown in figure 4.6.

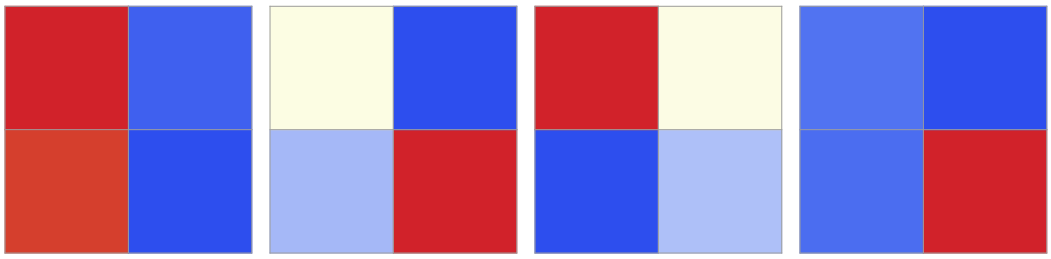


FIGURE 4.6: *Heatmap of the four filters* **f** *for a trained CNN.*

While it is possible to infer some information from these values, no clear conclusion can be drawn from it. One might say that f_1 appears to calculate the differences between horizontally neighbouring sites and f_4 calculates the difference between the right down site and the three sites left, up and diagonal from it. The functions of f_2 and f_3 are less clear however, and as such no clear function of the convolution layer appears to present itself. If we look at the filters **f** from CNNs trained on 40×40 and 30×30 configurations the filters have a very similar form, albeit with slightly different numerical values. The CNN trained on 20×20 configurations has a slightly different f_2 while the rest of the filters are similar to the ones of CNNs trained on larger configurations. The top left value is slightly higher (still negative) for this CNN. This might be the reason why the scaling is off, because the CNN trained on the smaller configurations might pick out slightly different quantities to use for its discrimination between phases. This results in a slightly different finite-size critical temperature prediction than those of the CNNs trained on larger configurations.

This highlights the importance of consistency of the quantity used to discriminate between phases. This is hard to remedy, because the outcome of gradient descent in loss functions with many local minima is very much dependent on the initial values chosen for the parameters, the kind of configurations the neural network is fed as well as the learning parameters of the neural network. While for computer science applications, such as discriminating between hand drawn numbers, any well-performing local minimum will be sufficient, for physics applications the local minimum the neural network chooses is crucial for finite-size scaling analysis. In order to force the system in a particular direction we transform the initial angle configurations to a vorticity configuration à la Beach et al[24].

4.2.3 Transforming the input configurations to vorticity configurations

To force the neural network to settle in a more clear minimum of the loss function so that finite-size scaling will give the expected results, we transform the input configurations to vorticity configurations. This transformation is performed by explicitly designing the convolution layer such that the hidden layer is essentially fed a transformation of the original angle configuration to a vorticity configuration. To do this the convolution layer consists of 4 filters **f** of the forms

$$f_1 = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}, \qquad f_2 = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix},$$
$$f_3 = \begin{bmatrix} 0 & 0 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad f_4 = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}. \tag{4.5}$$

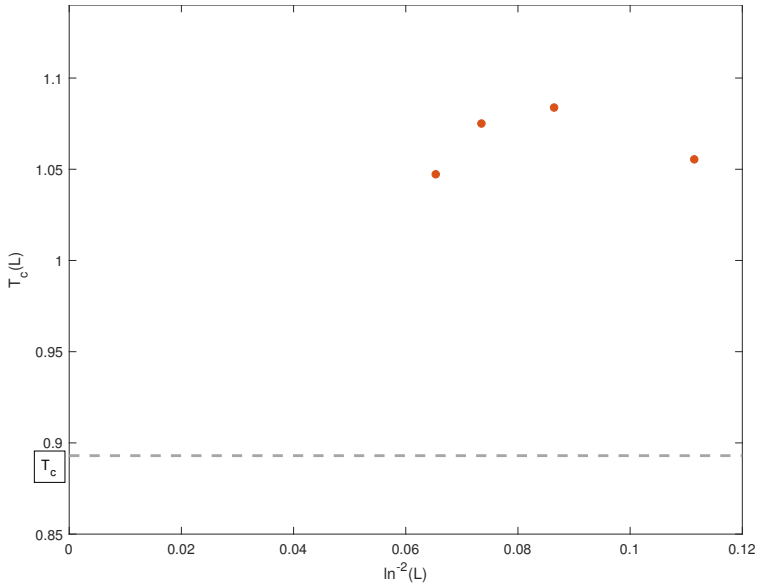


FIGURE 4.7: *Plot of the measured critical temperatures $T_c(L)$ as a function of the system size $L \times L$ (orange dots) obtained from a trained CNN. The grey dashed line is the expected value of T_c.*

These filters essentially take the angle differences of neighbouring sites which we shall call $\Delta\theta \in [-2\pi, 2\pi]$. These angle differences can then be mapped to the range $(-\pi, \pi]$ by applying the saw function:

$$\text{saw}(x) = \begin{cases} x + 2\pi, & x \leq -\pi \\ x & -\pi < x \leq \pi \\ x - 2\pi & x > \pi \end{cases}. \tag{4.6}$$

This function is applied to the angle differences corresponding to the four different filters, which are then added together to give a discrete approximation of the local vorticity as defined in equation (3.88). By performing a discrete sum over a 2×2 square of lattice sites' nearest neighbour angle differences, an approximation of the local vorticity is made whose value is mapped to a lattice site on a new lattice. This sum operation is then dragged over every single lattice site of the angle configuration, using periodic boundary conditions where necessary. The result is a new lattice where each lattice point corresponds to the value of one such sum of the angle configuration. Applying the filters to each lattice site thus results in a transformed configuration from angles on each lattice site to a local measure of the vorticity on each lattice site.

By explicitly transforming the angle configuration to a vortex configuration using a convolutional layer the hope is that the system will be able to distinguish between the two phases better and in a more consistent manner. By implicitly designing the convolutional layer as described above, the neural network can be trained on the same xy-model configurations as before. The predicted finite-size critical temperatures $T_c(L)$ for different system sizes $L \times L$ with a least squares fit of equation (3.95) is shown in figure 4.8. As is visible in this figure, the critical temperatures seem to follow the finite-size scaling much better than the FNN and the CNN trained on angle configurations. The least squares fit gives an estimation of the critical temperature of $T_c \approx 0.8545 \pm 0.0663$, where again no error estimation of the finite-size critical

temperature was included. Clearly this is a much better estimation of the critical temperature as for the FNN and CNN, which falls more closely to the expected critical temperature of $T_c \approx 0.893$. This result shows the power of feature-engineering the input data to steer the neural network to a better performance.

However, by explicitly performing these kind of transformations on the input data part of the allure of neural networks is lost. The main goal of neural network analysis of classical statistical physics systems' phase transitions is to study and compare the learned observables of the neural network to the typical theoretical and numerical tools used in these kinds of analyses. By explicitly transforming the input data you are essentially taking one of the steps which would make sense to do in the theoretical understanding of the model and doing it for the neural network, rather than seeing if it would figure it out on its own. While this makes it significantly easier to get better results, as is evident from the comparison between the three different neural networks used, part of the initial goal is lost. Ideally the neural network would indeed learn to transform the angle configuration to a vortex configuration, if this would lead to the best results, without any guidance by human hand. If that would have been the case, a much stronger argument for the application of neural networks to distinguish between phases of physical models would have been made. The power of a neural network to adapt and successfully find a minimum of the loss function which performs as we would like without any feature-engineering would have been a strong argument for naive application of neural networks to physical systems with unclear or hard to determine order parameters. Since this is not the case however, it seems like application of neural networks still need a firm guiding hand to get the network to do what we want. This lowers the neural networks appeal to more conventional, and as of now more accurate, numerical methods (which are often based on a theoretical understanding of the model) which are commonly used alongside Monte-Carlo simulations to get meaningful physical results.

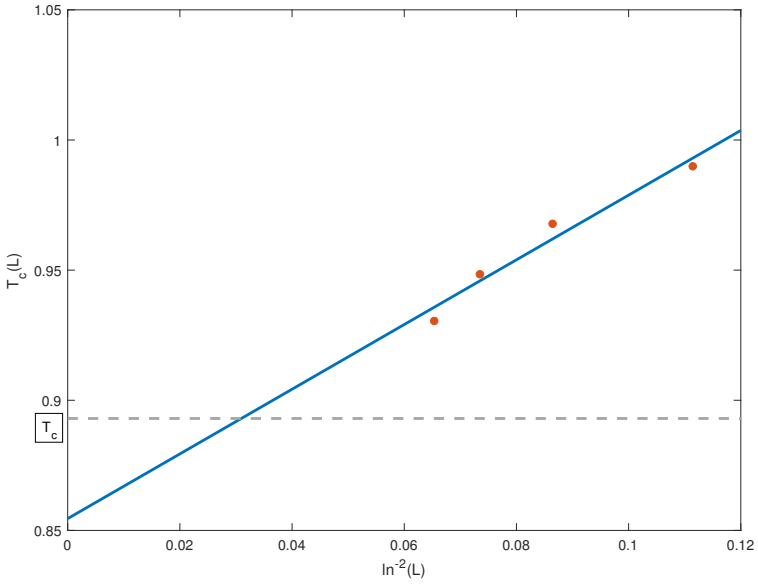


FIGURE 4.8: *Plot of the measured critical temperatures $T_c(L)$ as a function of the system size $L \times L$ (orange dots) obtained from a vortex CNN. The blue line is a least squares fit of equation* (3.95)*. The grey dashed line is the expected T_c.*

4.3 Anti-ferromagnetic Ising model on a triangular lattice

The anti-ferromagnetic Ising model on a triangular lattice is a bit of a different model compared to the others in this chapter. The model does not undergo a phase transition, but rather has a highly degenerate ground state due to geometric frustration. The goal of this neural network analysis is then not to distinguish between two different phases and find the critical temperature, but to see if the network is able to learn to discriminate between the frustrated groundstates and the high-temperature disordered phase by itself.

4.3.1 Evaluating the thermometer

To train the neural network for this task, it is designed a bit differently than the others. Rather than having two or more output neurons, a single output neuron is used. This output neuron is then compared to the temperatures associated with the configurations the network is fed, so that the neural network essentially acts as a thermometer. The hope is then that the network learns to recognise the typical frustrated configurations, compared to the purely disordered configurations. We assume that it is easier for the network to recognise the differences by employing a convolutional layer rather than just a single hidden layer. The convolutional layer consists of 6 filters of size 2×2. A hidden layer of 100 neurons is used. The system is then trained on configurations obtained via Monte Carlo simulations in the temperature range $T \in [0.1, 4.5]$ with temperature steps of $\Delta T = 0.2$. The network is fed a random set of configurations over the entire temperature range. The Monte Carlo simulations are repeated for the system sizes $L = \{21, 24, 30, 36, 42\}$.

The result for a temperature measurement for the CNN trained on a 42×42 triangular lattice is shown in figure 4.9. Clearly the performance is very good: the measured temperatures all fall very close to the Monte Carlo temperature. One thing to note is that the errorbars increase for higher temperatures. This is somewhat expected, since the higher temperatures include more noise and a wider spectrum of the possible configurations will be shown to the neural network. The performance of the CNN trained on a 42×42 lattice is in no way special. CNN's trained on the other system sizes showed similar behaviour. The 42×42 result is merely shown as an example of the learned behaviour.

To gain insight into what aspects of the physical model the neural network has learned, it may be helpful to look at the form of the convolution filters. Rather than giving the numeric values obtained after training, each individual filter can then be visualised as a 2×2 heatmap picture, shown in figure 4.10. It is difficult to obtain insight from the pictures, but by realising that the blue and red squares have the largest numeric difference, one can think of blue and red pairs as differences between the spins corresponding to the sites. Then the upper three filters can be seen as taking the difference between diagonal spins. The lower three filters seem to represent the difference between a single spin and the three others. This is a rather weak argument however, and we would like to find another way to really infer what the network has learned.

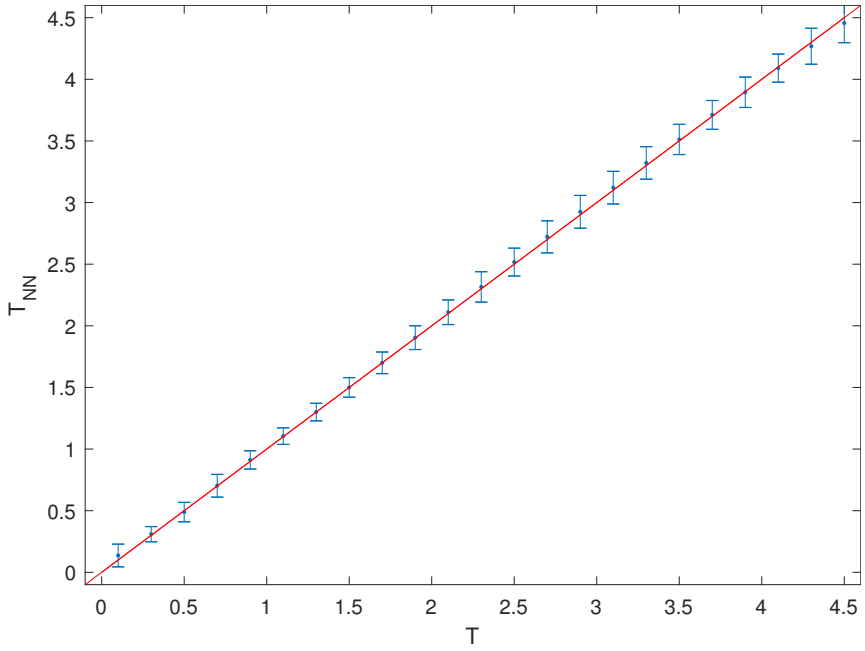


FIGURE 4.9: *Temperature measurements from a trained CNN, T_{NN}, as a function of the input configuration temperature, T, at which they were generated in Monte Carlo simulations. The CNN was trained on a 42×42 anti-ferromagnetic Ising model on a triangular lattice. The blue dots show the average temperature given by the CNN, with the errorbars indicating 1σ of error. The red line is a linear function $T_{\mathrm{NN}} = T$ for visual reference.*

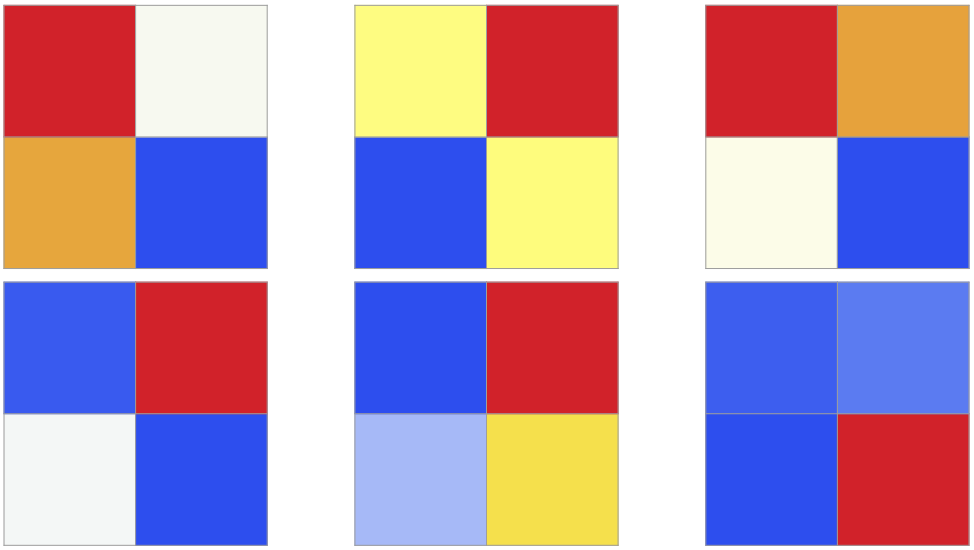


FIGURE 4.10: *Heatmap representations of the six convolution filters of the trained CNN. The colours run from blue (low) to red (high).*

Another way to see what the network has learned, is to track the output of the hidden layer for a set of configurations and to try and find a correlation between physical quantities of those configurations. For the two-dimensional Ising model this was a clear-cut procedure, the magnetisation is an obvious guess for a physical quantity to check the learned behaviour against. For the anti-ferromagnet on a triangular lattice this is less obvious. Since there is no phase transition, there is also no

order parameter. One can think of other physical quantities that increase with temperature, such as the average energy for example. In chapter 3 we also discussed that the distribution of frustrated and unfrustrated triangles should change with temperature, which is also something that a convolutional network might be able to pick up on. To check these guesses, we run configurations through the trained CNN while tracking the values of some of the hidden neurons. These outputs are then coupled to the energy and triangle distribution of the configurations.

Figure 4.11 shows the transformed inputs for the hidden layer $\mathbf{z}^{(2)}$ as a function of the energy E of the inputted configurations. The inputs are grouped together in groups of similar general form. Of these groups, two show near flat response to the energy, meaning that their value does not really change as a function of energy. This is quite peculiar, because this means that the inputs are also flat as a function of temperature. These inputs thus correspond to inactive neurons, which give of a near constant value for all inputs. Of the three groups left, all show some form of linear behaviour, albeit with a lot of noise. We see this as an indication that the network has indeed manage to learn the energy, or triangle distribution, of the physical system. However, performance is clearly not optimal, as there is a lot of noise in the response functions. A redesign of the CNN might be able to steer the network towards a better performance.

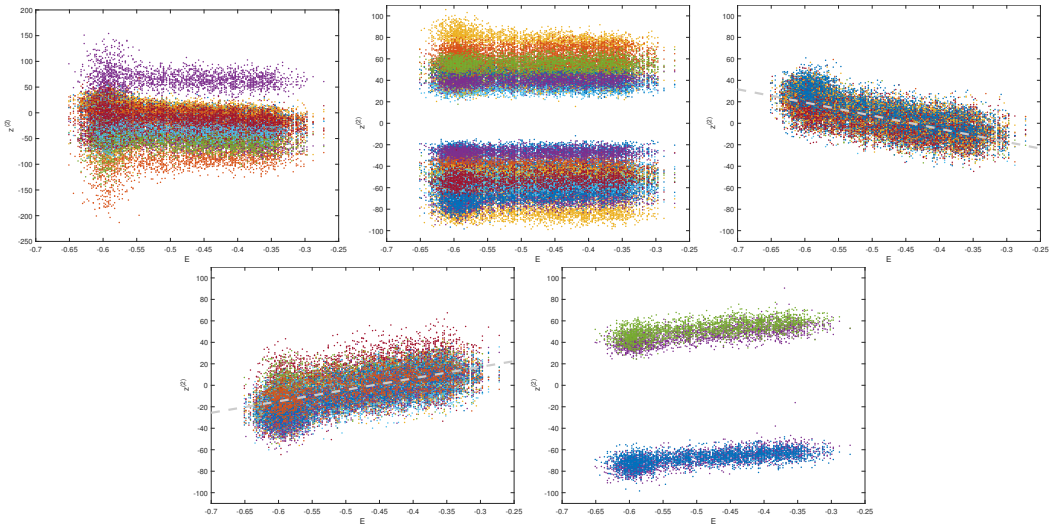


FIGURE 4.11: *Figures of the transformed inputs of the hidden layer* $\mathbf{z}^{(2)}$ *as a function of the energy E of the inputted configuration. The figures are grouped together in terms of similar responses to the input configurations. Different colours correspond to inputs for different hidden neuron*

4.3.2 A toy model learning the triangle sum histogram

To illustrate that the network is able to learn the triangle sum distribution we present a toy model CNN. The convolution layer consists of 4 different 3×2 filters with stride $S = [x, y] = [1, 2]$. That layer is then flattened and connected to two other 1×1 filters with different biases and ReLu activation functions. The output of the last convolution layer is then flattened and connected to four output neurons with ReLu activation functions, where each neuron corresponds to a possible triangle sum outcome. This way the network will effectively produce a histogram of the triangle sum distribution.

f	$\triangle$	f	$\triangledown$
1	$\begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$	3	$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$
2	$\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix}$	4	$\begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$

TABLE 4.1: *Convolution filters f corresponding to upwards pointing triangles $\triangle$ and downwards pointing triangles $\triangledown$.*

The four filters are designed such that they measure the triangle sums corresponding to either the upwards or downwards pointing triangles. Since the configurations are fed to the CNN as a square $L \times L$ black and white image, special care needs to be taken in the filter design with regards to the original triangular lattice. The resulting filters are shown explicitly in table 4.1. Precisely because of the triangular lattice and the design of the filters, taking the stride as $S = [1, 2]$ makes sure every triangle on the lattice is counted only once. The image is padded with periodic boundary conditions as well.

Applying these filters on the original configuration, or image, then creates a $L \times L \times 4$ three-dimensional matrix. Each filter dimension, or third dimension, now corresponds to each triangle defined by the filter. The convolution layer is then flattened so that it now consists of a vector of triangle sum outcomes, ordered by triangle type. The next convolution layer consists of two 1×1 filters of values $+1$ and -1 respectively going through a ReLu activation function. The result is a two-dimensional $L^2 \times 2$ matrix where the first filter corresponds to the positive triangle sum outcomes only, since the negative outcomes equate to 0. Likewise, the second filter corresponds to the negative triangle sum outcomes only. The matrix thus consists only of the values 3, 1 and 0.

Next another convolution layer is applied. This layer has two filters, $f_\pm = \pm 1$, with an associated bias $b_\pm = \mp 2$. The output are again going through a ReLu activation function. The result is a three-dimensional $L^2 \times 2 \times 2$ matrix, where the second dimension corresponds to the positive and negative triangle sums, and the third dimension corresponds to the 3 and 1 outcomes. By flattening this matrix to a one-dimensional vector we have a vector of four parts of length L^2, each of which corresponds to a different triangle sum outcome. The first part corresponds to triangle sums with outcome 3, the second to 1, third to -1 and the last to -3.

The final step to obtain the triangle sum histogram is to connect the final vector to the four output neurons. Since each part of the final vector corresponds to different outcomes of the triangle sums, it suffices to define the weight-matrix connecting the final vector to the output neurons as

$$\mathbf{W} = \frac{1}{L^2} \begin{pmatrix} \overbrace{1 \quad 1 \cdots 1}^{L^2} & \overbrace{0 \quad 0 \cdots 0}^{L^2} & \overbrace{0 \quad 0 \cdots 0}^{L^2} & \overbrace{0 \quad 0 \cdots 0}^{L^2} \\ 0 \quad 0 \cdots 0 & 1 \quad 1 \cdots 1 & 0 \quad 0 \cdots 0 & 0 \quad 0 \cdots 0 \\ 0 \quad 0 \cdots 0 & 0 \quad 0 \cdots 0 & 1 \quad 1 \cdots 1 & 0 \quad 0 \cdots 0 \\ 0 \quad 0 \cdots 0 & 0 \quad 0 \cdots 0 & 0 \quad 0 \cdots 0 & 1 \quad 1 \cdots 1 \end{pmatrix}. \tag{4.7}$$

Essentially this matrix separates and normalises the triangle sums as computed by the foregoing convolution layers. Since the layers already take the triangle sums and reduce them to values of 1 and 0, the normalisation just consists of summing over all the triangle sums with the same values and dividing by the amount of triangles. Figure 3.7 shows a schematic picture of the CNN design.

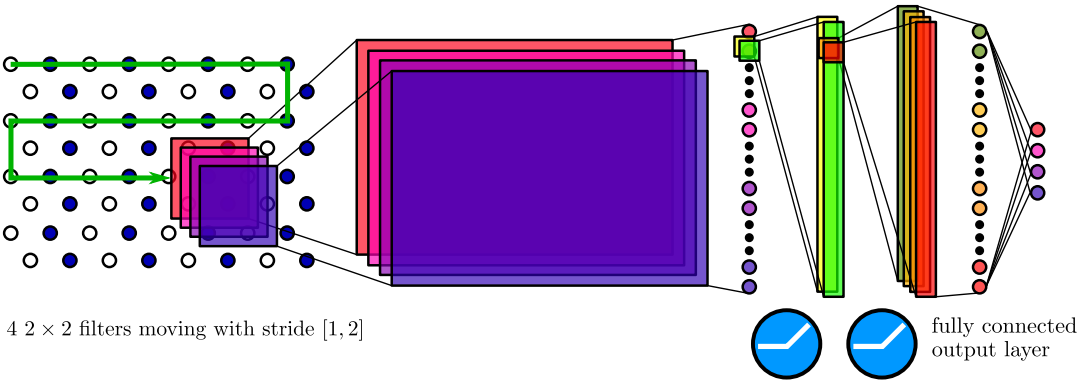


FIGURE 4.12: *Schematic picture of the CNN design for counting triangles. The blue circles with white lines indicate that the layer uses a ReLu activation function. The output layer is a fully connected layer with a weight matrix given by equation* (4.7).

4.4 Two-dimensional Coulomb Gas

Now we consider a model which contains a less well-defined phase transition, the two-dimensional Coulomb Gas. As discussed in chapter 3 there are arguments for two different kinds of phase transitions (PT) in this model, a KT-like and an Ising-like PT. There exists analytic argumentation based on approximate RG equations[66] that predicts the existence of a KT-like transition followed by a second-order Ising-like transition, which is further supported by numerical simulations[65]. The detection of the different PTs is usually done by tracking two different order parameters, ϵ^{-1} and M_S for the KT-like and Ising-like transition respectively, and extrapolating to the thermodynamic limit using the finite-size scaling equations appropriate for the type of transition. In this section we are interested in letting the neural network detect the phase transitions, and extrapolating to the thermodynamic limit using finite-size scaling.

To this end we obtain a set of configurations for a $L \times L$ lattice using Monte Carlo simulations. Two sets are generated, one ranging over a large range of temperatures: $0.01 \leq T \leq 1.0$ with $\Delta T = 0.01$, and the other near the predicted critical temperatures $0.4900 \leq T \leq 0.5495$ with $\Delta T = 0.0005$. This model is time-consuming to simulate, which is why the range of system sizes is limited to $L = \{10, 16, 20, 30\}$. These configurations can then be used to train the neural network. A couple of different training procedures and neural network designs will be used in this section. First we consider a regular CNN trained on the low- and high-temperature phases from just the first dataset. Next we consider a CNN trained on all three phases, using the second dataset to obtain configurations for the phase in between the KT-like and Ising-like transitions. Finally we consider a confusion scheme FNN on the first dataset, followed by a confusion scheme FNN on the second dataset.

4.4.1 CNN on the low- and high-temperature phases

We first consider a CNN with 4 filters of size 2×2 moving with unitary stride on the configurations padded with periodic boundary conditions. The convolution layer has neurons with ReLu activation functions. The convolution layer is flattened and connected to a hidden layer of 100 nodes with sigmoid activation functions. The output layer consists of two nodes representing the two phases, with a softmax activation function. Only the first dataset over the large range of temperatures is considered. This makes it difficult to assign configurations for the in-between phase, which is why the CNN is trained on the low- and high-temperature phases only. The interesting results of this simulation will then be in what transition the neural network will detect. Instead of linking the input of the hidden layer to physical quantities, the learned quantity is only analysed in finite-size analysis. How the quantity scales will then tell us whether the quantity is related to the Ising-like or KT-like transition.

The CNN reaches 100% accuracy within 20 epochs for all system sizes, where each epoch consists of 1000 training steps based on mini-batches of size 100. The system is only trained on low- and high-temperature configurations, where little ambiguity about which phase the system is in exists. The values of the output layer Y of the trained CNN is measured for all Monte Carlo configurations of the system size the network was trained on. The resulting values are plotted in figure 4.13. The location of the phase transition is then identified with the location where the network is most uncertain, when $Y_1 = Y_2 = 0.5$. As can be seen in the figure, the critical temperature

gets higher for larger system sizes.

Plotting the measured critical temperature per system size $T_c(L)$ shows us that the measured critical temperature scales according to the second-order Ising-like scaling equation (3.36). The neural network has thus learned to identify the Ising-like transition. The measured temperatures fitted with equation (3.36) using a least-squares method is shown in figure 4.14.

The least-squares fit gives an extrapolation of the critical temperature to the thermodynamic limit. It finds $T_c = 0.53 \pm 0.02$, where the uncertainty corresponds to a 2σ deviation. This is an underestimation of the real uncertainty, since no uncertainty in the network performance or extrapolation of the Y outputs to the 0.5-crossings was included. The found critical temperature agrees with earlier simulations, which found $T_I = 0.532 \pm 0.004$ [64]. However, the uncertainty range also includes the critical transition for the KT-like transition $T_{\mathrm{KT}} = 0.516 \pm 0.008$. We would like the neural network to be able to differentiate between the two transitions, so that an estimate of the critical temperatures for both transitions can be made. The result of this neural network analysis is then that the network naturally tends to pick up the Ising-like transition, as is evidenced by the finite-size scaling analysis.

This could also be expected by noting that the staggered magnetisation M_S is a more natural quantity to pick up from real-space configurations of the model. The inverse dielectric constant on the other hand is defined in for k-space, and might be something the neural network is less keen to pick up on. It should also be noted that the high-temperature phase obviously corresponds to a pure disordered phase, while the low-temperature phase corresponds to an ordered state. Purely in terms of the types of configurations the network gets to see, there is very little difference between the Coulomb Gas configurations and the anti-ferromagnetic Ising model, which also has the staggered magnetisation as order parameter. To circumvent this problem we need configurations representing the in-between phase, so that the network is able to learn two different order parameters.

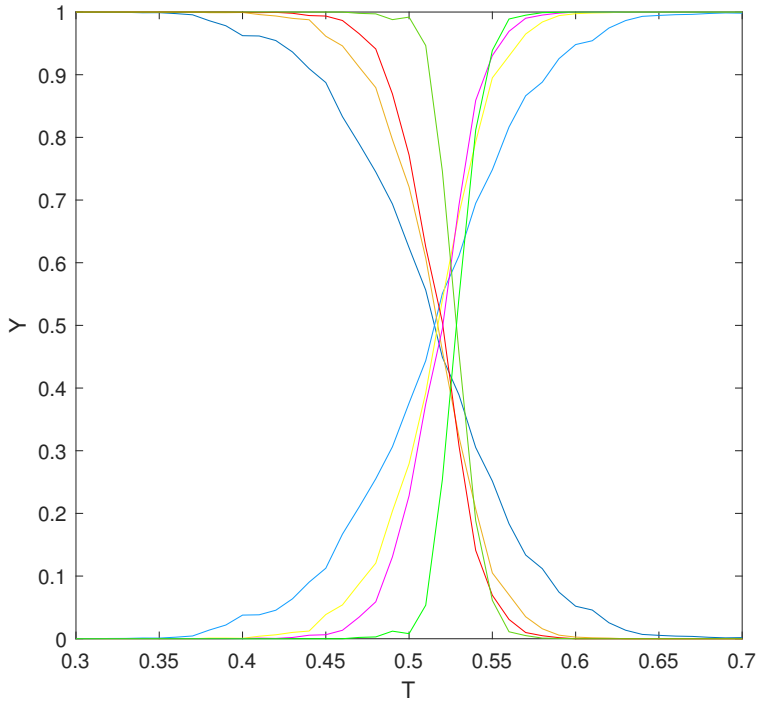


FIGURE 4.13: *Output Y as a function of T of the CNN trained on the low- and high-temperature phases. The output of Y_1 and Y_2 for the same system size are related by colour, with Y_2 corresponding to the brighter colour. Different colour-pairs correspond to different system sizes.*

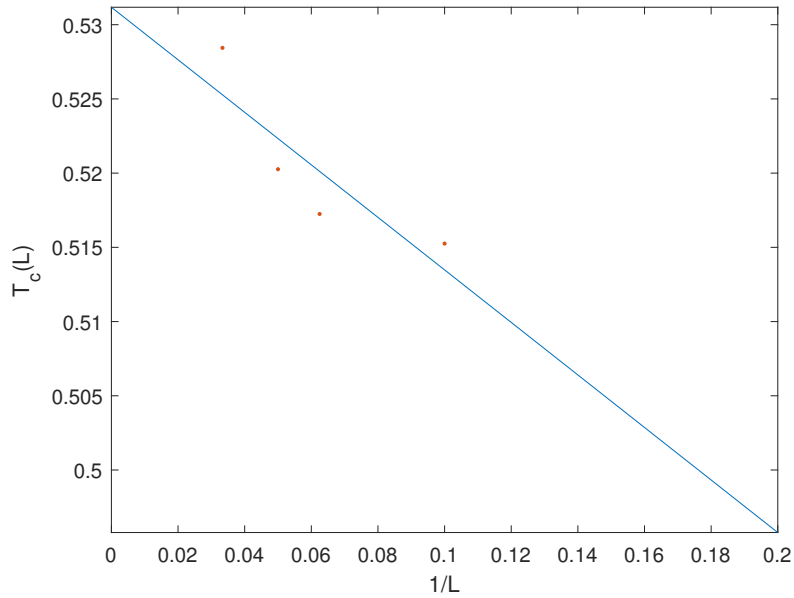


FIGURE 4.14: *Finite-size scaling of the critical temperature at system size $L \times L$ as measured by a CNN trained on low- and high-temperature phases. The blue line is a fit of equation* (3.36) *to the datapoints, the orange dots, by a least-squares fit.*

4.4.2 CNN on all three phases

To be able to train a CNN to differentiate between three phases, the network needs to be fed configurations representing all three phases. The first dataset over a large range of temperatures does not have enough configurations which can unambiguously be qualified as being in the second (in-between) phase. Adding the second dataset, ranging over a range of temperatures near the critical temperatures might be able to give us configurations being in the second phase. However, since the two phase transitions are so close to another, according to earlier simulations, even with temperature steps as small as $\Delta T = 0.0005$ it may be difficult to label configurations which are unambiguously in the second phase.

Nevertheless, we will attempt to train a CNN with 8 filters of size 2×2, moving with unitary stride. Again the filters are connected to neurons with ReLu activation functions. The convolution layer is connected to a hidden layer of 50 hidden neurons with sigmoid activation functions. The output layer now consists of three neurons with a softmax activation function. To train the network the training data is divided in three parts. The first part consists of low-temperature configurations, taken from the first dataset with a cut-off temperature of $T_1 = 0.4$. The second part is the high-temperature configurations, also taken from the first dataset, which run from $T_2 = 0.6$ to $T_f = 1.0$. The third part then consists of configurations in the in-between phase taken from the second dataset, taking a temperature interval of $0.5255 < T < 0.5320$.

The outputs Y of the trained CNN for $L = \{16, 20\}$ is shown in figure 4.15. The first row shows the output over the first dataset, while the second row shows the output over the second dataset. While the behaviour of the output over the first dataset seems fine at first glance, the locations of the phase transitions are a ways off. The scaling is also not consistent between different system sizes. The output over the second dataset also shows quite different behaviour for the two different system sizes. Different divisions of the temperature for the in-between phase were tried, as well as different distributions of the training configurations and different learning parameters, but to no avail. It seems like the CNN is unable to learn to differentiate between the three phases. Since this might be a problem to do with unclear labelling of the data, we turn to the confusion scheme neural network next.

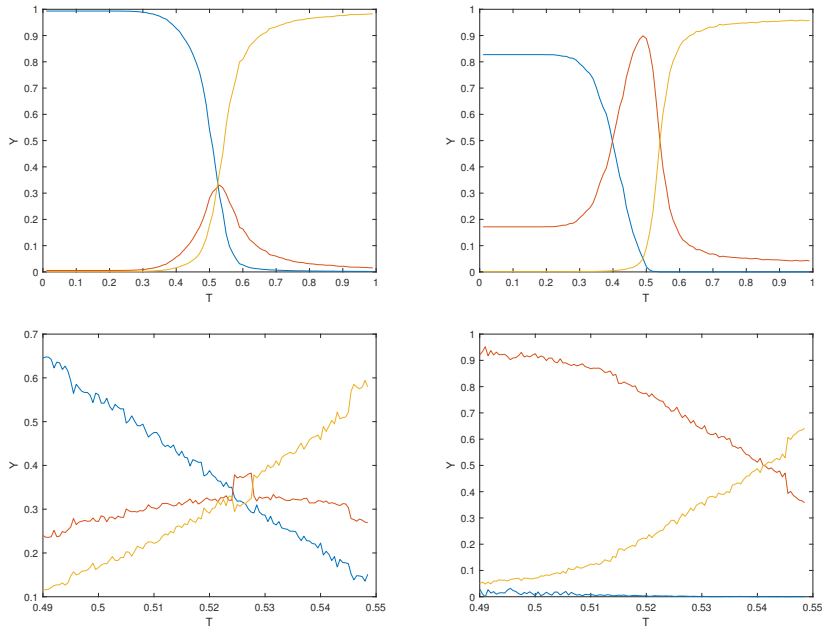


FIGURE 4.15: *CNN output Y for $L = 16$ ($L = 20$) on the left (right) for dataset 1 on the first row and dataset 2 on the second row.*

4.4.3 Finding the phase transition with confusion

Since the in-between region is hard to define, it may be useful to apply the confusion scheme to try to find the locations of the phase transitions. We attempt to do this in two different settings. The confusion scheme is first applied to the first dataset only, to see if the confusion scheme picks up two or three phases and which phase that is. After that we apply the confusion scheme to the second dataset as well, testing the accuracy $a(T')$ on the second dataset only.

The confusion scheme is applied by cycling through the possible values for T', training the neural network for 100 epochs per T'. Each epoch a save of the network state is made, and an accuracy test over a separate test dataset not shown during training is made. The iteration with the highest test dataset accuracy is then loaded after the training is over to prevent working with an overfitted network. The accuracy is then tested through all configurations for each temperature, taking the average over the results of the entire temperature range.

The first dataset

The results for the accuracy per T' is shown in figure 4.16 for $L = \{16, 20, 30\}$. $L = 10$ was omitted because no clear W-shape could be obtained. The expected W-shape is clearly visible for all other system sizes. An extrapolation to the thermodynamic limit can then be made by taking the middle-highest point of the W-shape as the critical temperature for that system size: $T_c(L)$. By doing a finite-size scaling analysis one can then find the critical temperature in the thermodynamic limit T_c. A comparison between the Ising-like and KT-like finite-size scaling shows that the Ising-like fit is the closest match. The datapoint with the fit are shown in figure 4.17. The found critical temperature is $T_c \approx 0.51 \pm 0.14$. The uncertainty is quite high since there are only three datapoints, and since the confusion scheme took temperature-steps of $\Delta T' = 0.01$ that is the highest precision possible. Unlike for the regular neural

network analysis, there is no interpolation to infer $T_c(L)$, so the results are stuck to the used values for T'.

We take the results as an indication that the network is unable to differentiate between the three phases of the physical system with just the first dataset. The network does find a single phase transition, which is the Ising-like transition. To try to find both phase transitions present in the system we need to include the second dataset.

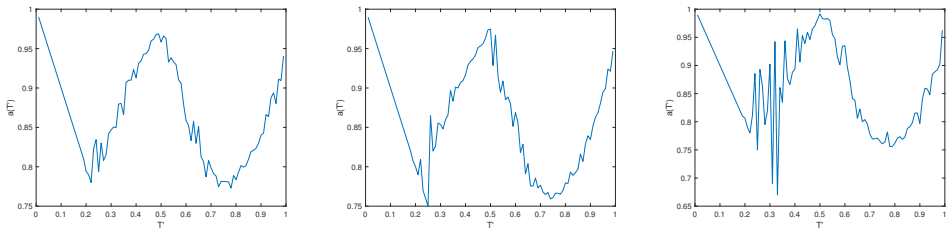


FIGURE 4.16: *Accuracy over the first dataset for* $0.01 \leq T' < 1.00$ *for* $L = \{16, 20, 30\}$ *from left to right.*

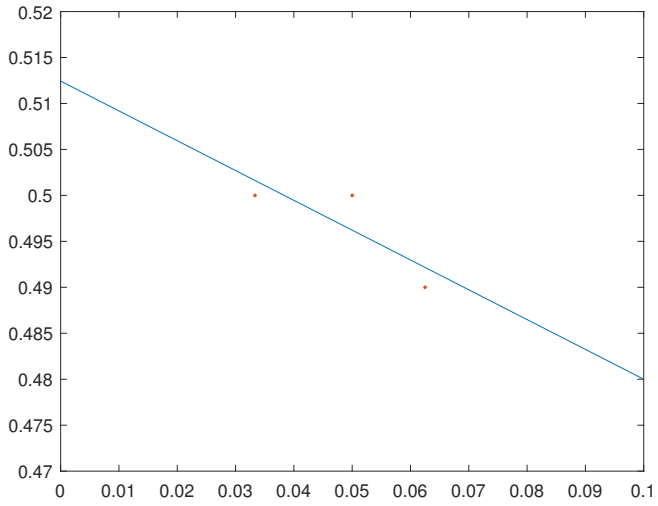


FIGURE 4.17: *Ising-like Finite-size scaling for* $T_c(L)$ *found from the confusion scheme for the first dataset.*

The second dataset

Adding the second dataset to the set of configurations fed to the neural network during the training phase does not seem to allow the network to learn two different phase transitions. The training procedure of the confusion scheme network is the same as for the first dataset, except now the configurations near the critical temperatures of the second dataset are also included in the training process. The accuracy of the trained neural network is then tested on all the configurations of the second dataset to hopefully find a VVV-shape depicting three different phase transitions. Unfortunately this is not the case. Figure 4.18 shows $a(T')$ for $L = 16$ and $L = 20$. Instead of a VVV-shape of even a W-shape, just a V-shape is found. The neural network is unable to find a classifier to distinguish between the different phases within dataset 2. Different learning parameters and configuration distributions were used,

but to no avail.

This leaves us no choice but to conclude that the neural network is unable to differentiate between the two phase transitions of the two-dimensional Coulomb gas. The numerical technique can not be used to find an unbiased basis for finding out if the model really has two separate phase transitions and three distinct phases. In principle the neural network should be able to differentiate between Ising-like and KT-like phase transitions, as indicated by the earlier neural network analysis done on the Ising model and xy-model. In this case however, the network is unable to differentiate. This is again an indication of the limited power of neural networks as a naive tool to implement for physical systems.

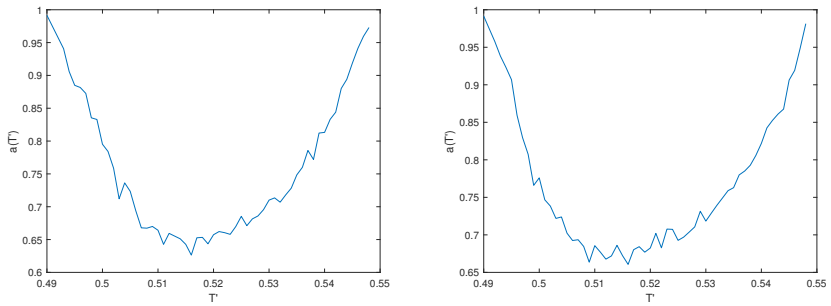


FIGURE 4.18: *Accuracy over the second dataset for* $0.49 < T' < 0.5495$ *for* $L = \{16, 20\}$ *from left to right.*

Chapter 5

Restricted Boltzmann Machines and spin models

This chapter breaks away from the focus of classifying different phase transitions using neural networks, and rather focuses on another class of neural networks: restricted Boltzmann machines (RBM). As described in chapter 2, RBM are a type of generative unsupervised neural network. Their primary function is to model the underlying probability configuration of the configurations it is fed as best as it can. The RBM does this by minimising the difference between its own underlying probability distribution $p(\mathbf{v})$ and the probability distribution we wish to model $q(\mathbf{v})$, where $\mathbf{v}$ is a configuration of the visible neurons. The difference between the probability distributions are measured via the Kullback-Leibler divergence (KL-divergence) given in equation (2.25).

The main goal of this chapter is to figure out what kind of properties the RBM is able to pick up from physical probability distributions, and whether there is any physical relevance to these properties. Specifically we are interested in studying RBMs where the hidden layer consists of less nodes than the hidden layer. The information entropy in the hidden layer will then be less than the information entropy of the visible layer. The RBM can then never retain the same amount of information when transitioning from the visible to hidden layer, as done in block Gibbs sampling. The RBM will thus have to make choices in what information to keep, which is conceptually analogous to real-space RG's relevant and irrelevant parameters. We aim to study this by a combination of analytic and numerical arguments related to training a RBM on both the one-dimensional and two-dimensional square-lattice Ising models. This work is a continuation of preliminary work by Ter Hoeve[69].

This chapter will start with a discussion of the general structure of an RBM, followed by a short recap of the results by Ter Hoeve, which will be followed by an exact mapping of the one-dimensional Ising model to a RBM. We will then continue to an analysis of a RBM with a restricted number of free weight-matrix components based on translation invariance arguments. The RBM flow is then computed and analysed for the different RBM designs. Finally a continuation of the one-dimensional results to the two-dimensional Ising model is discussed, where the one free W-value case is discussed in detail and a comparison to mean-field theory is made.

5.1 An investigation into the general RBM structure

In this section the structure of an unrestricted RBM, meaning no restrictions are placed on the weight-matrix $\mathbf{W}$, is investigated. Some results of this investigation may come in handy when discussing possible restrictions of the RBM. The only restriction set on the RBM is that the biases are 0. In analogy to the spins of the target distribution, the possible values of the visible and hidden neurons are $\{+1, -1\}$ as opposed to the usual binary values $\{1, 0\}$. The energy-function for the RBM is then

$$E = \sum_{i,a} v_i W_{ia} h_a. \tag{5.1}$$

From this we can derive the probability distribution of the visible neurons by summing over all possible hidden neuron values:

$$\begin{aligned} p(\mathbf{v}) &= \sum_{\{\mathbf{h}\}} p(\mathbf{v}, \mathbf{h}) \\ &= \frac{2^N}{Z} \prod_{a=1}^{N} \cosh\left(\sum_{i=1}^{L} v_i W_{ia}\right), \end{aligned} \tag{5.2}$$

where N and L are the number of hidden and visible nodes respectively, and Z is the partition function of the whole probability distribution of the RBM $p(\mathbf{v}, \mathbf{h})$.

The RBM tries to minimise the KL-divergence (or rather maximise the likelihood, which is equivalent to minimising the KL-divergence) to best represent the target probability distribution $q(\mathbf{v})$. The partial derivative of the KL-divergence (denoted by KL) with respect to a weight-matrix component W_{jb} is

$$\begin{aligned} \frac{\partial \mathrm{KL}}{\partial W_{jb}} &= -\sum_{\{\mathbf{v}\}} \frac{q(\mathbf{v})}{p(\mathbf{v})} \frac{\partial p(\mathbf{v})}{\partial W_{jb}} \\ &= 2^N \sum_{\{\mathbf{v}\}} \frac{q(\mathbf{v})}{p(\mathbf{v})} \Big[\frac{\partial Z}{\partial W_{jb}} \frac{1}{Z^2} \prod_{a=1}^{N} \cosh\left(\sum_{i=1}^{L} v_i W_{ia}\right) \\ &\quad - \frac{1}{Z} \prod_{a \neq b}^{N} \cosh\left(\sum_i v_i W_{ia}\right) \sinh\left(\sum_i v_i W_{ib}\right) v_j \Big] \\ &= \frac{2^N}{Z} \frac{\partial Z}{\partial W_{jb}} - 2^N \sum_{\{\mathbf{v}\}} q(\mathbf{v}) v_j \tanh\left(\sum_i v_i W_{ib}\right). \end{aligned} \tag{5.3}$$

Here we have used that $q(\mathbf{v})$ does not depend on W, Z does not depend on $\mathbf{v}$ and $\sum_{\{\mathbf{v}\}} q(\mathbf{v}) = 1$. We want to update the weight-matrix components such that the KL-divergence is at its global minimum. That corresponds to setting the partial derivatives of the KL-divergence w.r.t. the weight-matrix components to 0. We can further expand on the Z-term:

$$\frac{2^N}{Z} \frac{\partial Z}{\partial W_{jb}} = 2^N \frac{\sum_{\{\mathbf{v},\mathbf{h}\}} v_j h_b \exp\left(\sum_{i,a} v_i h_a W_{ia}\right)}{\sum_{\{\mathbf{v},\mathbf{h}\}} \exp\left(\sum_{i,a} v_i h_a W_{ia}\right)}, \tag{5.4}$$

so that the equation we need to solve for every W_{jb} weight-matrix component is

$$\frac{\sum_{\{\mathbf{v},\mathbf{h}\}} v_j h_b \exp\left(\sum_{i,a} v_i h_a W_{ia}\right)}{\sum_{\{\mathbf{v},\mathbf{h}\}} \exp\left(\sum_{i,a} v_i h_a W_{ia}\right)} = \sum_{\{\mathbf{v}\}} q(\mathbf{v}) v_j \tanh\left(\sum_i v_i W_{ib}\right). \tag{5.5}$$

We can further expand on the left-hand-side of the equation. The denominator, Z, can be evaluated further as

$$\begin{aligned} Z &= \sum_{\{\mathbf{v},\mathbf{h}\}} \prod_a \exp\left(\sum_i v_i h_a W_{ia}\right) \\ &= \sum_{\{\mathbf{v}\}} \prod_a \left[\exp\left(\sum_i v_i W_{ia}\right) + \exp\left(-\sum_i v_i W_{ia}\right)\right] \\ &= 2^N \sum_{\{\mathbf{v}\}} \prod_a \cosh\left(\sum_i v_i W_{ia}\right). \end{aligned} \tag{5.6}$$

The numerator can be evaluated in the same vein:

$$\begin{aligned} \frac{\partial Z}{\partial W_{jb}} &= \sum_{\{\mathbf{v},\mathbf{h}\}} v_j h_b \prod_a \exp\left(\sum_i v_i h_a W_{ia}\right) \\ &= \sum_{\{\mathbf{v}\}} v_j \prod_{a\neq b} \left[\exp\left(\sum_i v_i W_{ia}\right) + \exp\left(-\sum_i v_i W_{ia}\right)\right] \\ &\times \left[\exp\left(\sum_i v_i W_{ib}\right) - \exp\left(-\sum_i v_i W_{ib}\right)\right] \\ &= 2^N \sum_{\{\mathbf{v}\}} v_j \prod_{a\neq b} \cosh\left(\sum_i v_i W_{ia}\right) \sinh\left(\sum_i v_i W_{ib}\right). \end{aligned} \tag{5.7}$$

The equation that minimises the KL-divergence with regards to W_{jb} is now

$$\frac{2^N}{Z} \sum_{\{\mathbf{v}\}} v_j \prod_{a\neq b} \cosh\left(\sum_i v_i W_{ia}\right) \sinh\left(\sum_i v_i W_{ib}\right) = \sum_{\{\mathbf{v}\}} q(\mathbf{v}) v_j \tanh\left(\sum_i v_i W_{ib}\right). \tag{5.8}$$

Since $\tanh -x = -\tanh x$, $\cosh -x = \cosh x$ and $\sinh -x = -\sinh x$ we can see that this equation is invariant under global spin flips $\mathbf{v} \to -\mathbf{v}$ just like the Ising spin models we are considering. It is also invariant under a change of sign of all W_{ib}. It is not invariant under a translation $v_i \to v_{i+1}$, unlike Ising models with periodic boundary conditions.

We can compare the equations that minimise W_{jb} and W_{j+1b} to try and find other symmetries, since $q(\mathbf{v})$ is invariant under translations. Since both equations should vanish, we can equate the two. By explicitly summing over $v_j = \{-1, +1\}$ and $v_{j+1} = \{-1, +1\}$ we get an equation which might give more insight in possible symmetries between W_{jb}- and W_{j+1b}-solutions that minimise the KL-divergence. The $(v_j, v_{j+1}) = (1, 1)$ and $(-1, -1)$ contributions in the configuration sums cancel each

other, so that we end up with the final equation

$$
\begin{aligned}
2 \sum_{\{\mathbf{v}\}}^{(v_j,v_{j+1})=(1,-1)} & q(\mathbf{v}) \tanh\Big(\sum_{i\neq j,j+1} v_i W_{ib} + W_{jb} - W_{j+1b}\Big) \\
-2 \sum_{\{\mathbf{v}\}}^{(v_j,v_{j+1})=(-1,1)} & q(\mathbf{v}) \tanh\Big(\sum_{i\neq j,j+1} v_i W_{ib} - W_{jb} + W_{j+1b}\Big) = \\
\frac{2^N}{Z}\Big[2 \sum_{\{\mathbf{v}\}}^{(v_j,v_{j+1})=(1,-1)} & \prod_{a\neq b} \cosh\Big(\sum_{i\neq j,j+1} v_i W_{ia} + W_{ja} - W_{j+1a}\Big) \\
& \times \sinh\Big(\sum_{i\neq j,j+1} v_i W_{ib} + W_{jb} - W_{j+1b}\Big) \\
-2 \sum_{\{\mathbf{v}\}}^{(v_j,v_{j+1})=(-1,1)} & \prod_{a\neq b} \cosh\Big(\sum_{i\neq j,j+1} v_i W_{ia} - W_{ja} + W_{j+1a}\Big) \\
& \times \sinh\Big(\sum_{i\neq j,j+1} v_i W_{ib} - W_{jb} + W_{j+1b}\Big)\Big].
\end{aligned} \tag{5.9}
$$

While this equation gives no obvious hard conditions, one can see that having $W_{ja} = W_{j+1a}\ \forall a$ fulfils the equation since the sums in the hyperbolic functions will no longer contain the j- and $j+1$-terms and $q(\mathbf{v})$ has translation invariance. The terms will thus cancel out and vanish. This is by no means the only solution to the equation however. If we take the condition to be true, that means that the weights are independent of the visible layer, in other words independent of j. Since there is nothing special about the $j+1$-th site we took, it should hold for any other visible site.

With the weights independent of j equation (5.8) becomes

$$
\frac{2^N}{Z} \sum_{\{\mathbf{v}\}} v_j \prod_{a\neq b} \cosh\left(VW_a\right) \sinh\left(VW_b\right) = \sum_{\{\mathbf{v}\}} q(\mathbf{v}) v_j \tanh\left(VW_b\right). \tag{5.10}
$$

Again, since the equation only differs in the product of hyperbolic cosines for weights of different hidden nodes there probably is a solution for which the weights are independent of the hidden sites. There seems to be no a priori indication that this is the solution which minimises the KL-divergence the most.

5.2 The Ising chain and the mean-field RBM

This section is an analytic and numerical analysis on the training of a RBM with a single weight-matrix W value as per Ter Hoeve[69]. The choice to reduce $L \times N$ weight-matrix $\mathbf{W}$, where L and N are the number of visible and hidden neurons respectively, to a matrix where every component is equal to W can be understood as follows. In general the weight-matrix component W_{ia} encodes the interaction strength between the visible node i and hidden node a. Since we known that the one-dimensional Ising chain is translation invariant, we can demand that the RBM probability distribution is also translation invariant in the visible neurons. There are multiple ways to enforce this. As was shown by equation (5.9) one way of enforcing translation invariance is by removing the visible i-dependency of the weight-matrix on the hidden nodes: $W_{ia} \to W_a$. The probability distribution now factorises over a:

$$\begin{aligned} p(\mathbf{v}) &= \frac{1}{Z} \sum_{\{\mathbf{h}\}} \exp\left(\sum_{i,a} v_i W_{ia} h_a\right) \\ &= \frac{1}{Z} \sum_{\{\mathbf{h}\}} \prod_{a}^{N} \exp\left(\sum_{i}^{L} v_i W_a h_a\right) \\ &= \frac{1}{Z} \prod_{a}^{N} 2\cosh\left(\sum_{i}^{L} v_i W_a\right). \end{aligned} \tag{5.11}$$

Since the optimisation of the KL-divergence with respect to each W_a will give a set of N equal equations, a solution to this equation is sure to be a solution to the others as well. This means that setting $W_a = W$ is able to give us a minimum of the KL-divergence as long as such a minimum also existed for W_a. Note that setting $W_{ia} = W$ might not give the solution with the lowest KL-divergence, as we will see later in this chapter. The greatest advantage of reducing the degrees of freedom of $\mathbf{W}$ to one is that this allows us to take several analytic derivations where this would have been impossible for a general $\mathbf{W}$. $p(\mathbf{v})$ can now be written as

$$p(\mathbf{v}) = \frac{2^N}{Z} (\cosh WV)^N, \tag{5.12}$$

where $V = \sum_i^L v_i$. The partition sum is

$$Z = \sum_{\{\mathbf{v}\}} 2^N (\cosh WV)^N. \tag{5.13}$$

We refer to this reduction of the $\mathbf{W}$-matrix to a single W-value as the mean-field RBM. For a comparison between the two-dimensional mean-field Ising model and the mean-field RBM see section 5.7. Since the $\mathbf{W}$-matrix is reduced to a single free W parameter, the learning process of the RBM consists solely of minimising the KL-divergence with respect to W. The KL-divergence depends on the probability distribution of the Ising chain $q(\mathbf{v})$:

$$q(\mathbf{v}) = \frac{1}{Z_{\text{Ising}}} \sum_{\{\mathbf{v}\}} \exp\left(K \sum_{i}^{L} v_i v_{i+1}\right), \tag{5.14}$$

where $K = \beta J > 0$ is the temperature-dependent interaction strength. The partition sum is given in equation (3.10), which, for zero magnetic field ($h = 0$), can be written

as

$$Z_{\text{Ising}} = (2\cosh K)^L + (2\sinh K)^L. \tag{5.15}$$

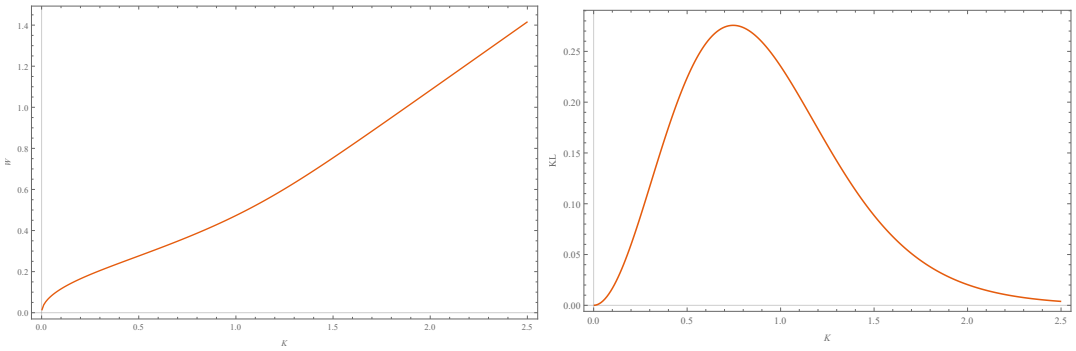
The minimum of the KL-divergence can be found by taking the derivative of the KL-divergence with respect to W:

$$\begin{aligned}\frac{\partial \text{KL}}{\partial W} &= -\sum_{\{\mathbf{v}\}} \frac{q(\mathbf{v})}{p(\mathbf{v})} \frac{\partial p(\mathbf{v})}{\partial W} \\ &= -\sum_{\{\mathbf{v}\}} \left[q(\mathbf{v}) NV \tanh(WV) + q(\mathbf{v}) \frac{1}{Z} \frac{\partial Z}{\partial W} \right].\end{aligned} \tag{5.16}$$

The second term on the right hand side can be significantly simplified by noting that Z is independent of the configuration of $\mathbf{v}$ and that $q(\mathbf{v})$ is normalised to 1. Minimising the KL-divergence then comes down to solving the equation

$$\frac{\partial \ln Z}{\partial W} = \sum_{\{\mathbf{v}\}} q(\mathbf{v}) NV \tanh(WV). \tag{5.17}$$

Ter Hoeve does not solve this equation analytically, but rather finds numerical solutions for finite-sized configurations using *Mathematica*. We repeated the numerical calculations for a 6×3 RBM trained on a Ising chain of 6 sites at a interaction strength K, and repeated the calculation for a multitude of K-values in the range $(0, 2.5]$. This results in a curve equating K to W such that the KL-divergence is minimised. The resulting curve is shown in figure 5.1a. The final KL-divergence associated with the solution of equation (5.17) is shown in figure 5.1b.



(A) $W(K)$ (B) *KL-divergence*

FIGURE 5.1: *Graphs showing the relation between W and K obtained by solving equation* (5.17) *numerically for a 6×3 RBM, as well as the KL-divergence of the trained RBM.*

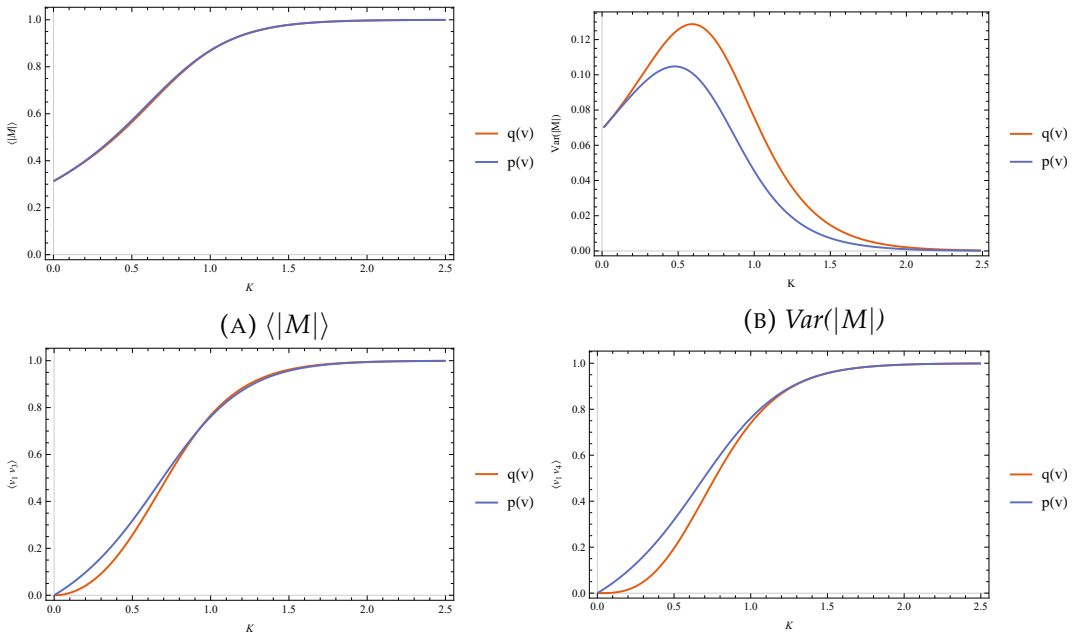
As is visible in figure 5.1, the single W-valued RBM is most inaccurate around $K \approx 0.6$, when the Ising chain is transitioning from its preference for disordered states in the high temperature (low K) region to more ordered states in the low temperature (high K) region. Note that while the Ising chain in the thermodynamic limit exhibits no phase transition and is always disordered for any finite temperature, a finite size system will show different behaviour. A finite sized system can technically never undergo a phase transition, since a phase transition is only defined in the thermodynamic limit. That the RBM is unable to capture essential features from the Ising

probability distribution is made more clear in figure 5.2, which shows a comparison between the average absolute magnetisation $\langle|M|\rangle$, the magnetic susceptibility χ and two different two-spin correlations: $\langle v_1 v_3\rangle$ and $\langle v_1 v_4\rangle$.

While the RBM is able to model the average absolute magnetisation $\langle|M|\rangle$ extremely well, it fails to capture the fluctuations in the magnetisation present in the magnetic susceptibility of the absolute magnetisation:

$$\chi_{|M|} = LK\left(\langle|M|^2\rangle - \langle|M|\rangle^2\right) = LK \cdot \mathrm{Var}(|M|). \tag{5.18}$$

This can be understood conceptually from the fact that by restricting the RBM to a single W-value, $p(\mathbf{v})$ essentially becomes a distribution of the total magnetisation $LM = \sum_i v_i = V$ only. Hence it should make sense that the RBM manages to model the mean of M as well as it does since it is one of the central characteristics of a probability distribution in M. The variance of $|M|$, which is equal to the magnetic susceptibility $\chi_{|M|}$ up to a linear K-dependent scaling, is evidently not captured as well by the RBM. Precisely because the RBM is a function of V only, it is unable to fully capture correlations between individual spins (the v_i's), which are included in the $\langle M^2\rangle$ term. This point is supported by the two spin correlations shown in figure 5.2, which show significant deviation between the RBM and the Ising chain in the small K region. For large K's, the correlations between spins grow in the Ising chain, leading to similar behaviour between all two-spin correlations. In the RBM, all two-spin correlations are the same, so for large K one can expect the difference in two-spin correlations to matter less. This explains why the two-spin correlations for the Ising chain and RBM are again similar in the large K region, which results in similar Var($|M|$) behaviour as well.



(A) $\langle|M|\rangle$ (B) *Var*($|M|$)

(C) $\langle v_1 v_3\rangle$ (D) $\langle v_1 v_4\rangle$

FIGURE 5.2: *Graphs comparing* $\langle|M|\rangle$, *Var*($|M|$), $\langle v_1 v_4\rangle$ *and* $\langle v_1 v_4\rangle$ *of a trained* 6×3 *RBM to a 6-sited Ising chain of interactions strength* K.

While the mean-field RBM is unable to fully capture the Ising chain, it is able to capture the average absolute magnetisation of the Ising chain. In the large K and $K \approx 0$

regions the RBM is able to model the Ising chain really well. This can be understood by the arguments mentioned above. Additional insight can be gained by investigating certain limits of the RBM, where a connection to the RBM and renormalisation group (RG) can be made.

5.2.1 Limit behaviour of $W(K)$ and connection to RG

The limit behaviour of $\lim_{K\to\infty} W(K)$ can be derived by evaluating the right- and left hand sides of equation (5.17). It is relatively straightforward to show that in the limit of $K \to \infty$ only the configurations with all spins pointing up or down will contribute to the probability distribution:

$$\begin{aligned}\lim_{K\to\infty} q(\mathbf{v}) &= \lim_{K\to\infty} \frac{\exp\left(K\sum_i v_i v_{i+1}\right)}{(2\cosh K)^L + (2\sinh K)^L} \\ &\approx \lim_{K\to\infty} \frac{1}{2}\exp\left(K\sum_i v_i v_{i+1} - KL\right) \\ &= \begin{cases} 0 & \text{if} \quad \sum_i^L v_i v_{i+1} < L \\ \frac{1}{2} & \text{if} \quad \sum_i^L v_i v_{i+1} = L \end{cases}. \end{aligned} \tag{5.19}$$

So only when $\sum_i^L v_i v_{i+1} = L$ will $q(\mathbf{v})$ have a non-vanishing contribution. This is only the case when all spins are pointing either up or down. All other configurations add up to less than L, since they must include at least one anti-aligned spin pair which reduces the sum by 2 compared to the spin pair being aligned. A more detailed derivation of this limit is included in the appendix of Ter Hoeve[69]. So in the limit of $K \to \infty$ the right hand side of equation (5.17) reduces to

$$\lim_{K\to\infty} \sum_{\{\mathbf{v}\}} q(\mathbf{v}) NV \tanh(WV) = NL \tanh(WL), \tag{5.20}$$

where we used that $\tanh -x = -\tanh x$. For the left hand side of equation (5.17) we get:

$$\begin{aligned}\frac{1}{Z}\frac{\partial Z}{\partial W} &= \frac{1}{\sum_{\{\mathbf{v}\}}(2\cosh WV)^N} \sum_{\{\mathbf{v}\}} NV(2\cosh WV)^{N-1} 2\sinh WV \\ &= \frac{\sum_{\{\mathbf{v}\}} NV(\cosh WV)^N \tanh WV}{\sum_{\{\mathbf{v}\}}(\cosh WV)^N}. \end{aligned} \tag{5.21}$$

This equation will be equal to equation (5.20) in the limit $W \to \infty$:

$$\lim_{W\to\infty} \frac{1}{Z}\frac{\partial Z}{\partial W} \approx \lim_{W\to\infty} \frac{\sum_{\{\mathbf{v}\}} NV \exp(WVN) \tanh WV}{\sum_{\{\mathbf{v}\}} \exp(WVN)} \tag{5.22}$$

$$= \lim_{W\to\infty} \frac{\sum_{\{\mathbf{v}\}} NV \exp(WVN)\exp(-WLN) \tanh WV}{\sum_{\{\mathbf{v}\}} \exp(WVN)\exp(-WLN)} \tag{5.23}$$

$$= NL. \tag{5.24}$$

In deriving the last step we used $\lim_{x\to\infty} \tanh x = 1$. Evidently this is equal to equation (5.20) in the limit $W \to \infty$. Thus if K diverges, W does as well. By similar arguments one can show that if $K = 0$, equation (5.17) will hold for $W = 0$.

To illustrate the connection between the RG and the RBM flow the expected value

for the hidden neurons given a configuration $\{\mathbf{v}\}$ is given by the average value of h_a, where the average is taken over the probabilities given by equation (2.49). The resulting average is

$$\begin{aligned}\langle h_a \rangle_{\{\mathbf{v}\}} &= \sum_{\{h_a\}} h_a p(h_a|\mathbf{v}) \\ &= \tanh\left(\sum_i W v_i\right). \end{aligned} \tag{5.25}$$

The expected value for v_i can be calculated in a similar fashion using equation (2.48):

$$\langle v_i \rangle_{\{\mathbf{h}\}} = \tanh\left(\sum_a W h_a\right). \tag{5.26}$$

A flow of configurations is now created by iterating new configurations of the hidden and visible layer given the configurations of the other layer. If the RBM is trained on the Ising chain at $K \to \infty$, the weight will also diverge. If the trained RBM is then fed an arbitrary configuration $\{\mathbf{v}\}$, the hidden layer will iterate to

$$\langle h_a^{(0)} \rangle = \lim_{W\to\infty} \tanh\left(W \sum_i v_i^{(0)}\right) = \begin{cases} 1 & \text{if} \quad \sum_i v_i^{(0)} > 0 \\ -1 & \text{if} \quad \sum_i v_i^{(0)} < 0 \\ 0 & \text{if} \quad \sum_i v_i^{(0)} = 0 \end{cases}, \tag{5.27}$$

where the superscript on the hidden and visible neurons indicates the configuration iteration, where (0) is the initial configuration. Clearly the hidden layer configuration will consist of all spins up (down) if V is larger (smaller) than 0. When $V = 0$, the probability of getting an up or down spin is the same, and the hidden neuron can go either way. With equation (5.26) the next iteration of the visible configuration can be determined:

$$\langle v_i^{(1)} \rangle = \lim_{W\to\infty} \tanh\left(W \sum_a h_a^{(0)}\right) = \begin{cases} 1 & \text{if} \quad \sum_a h_a^{(0)} > 0 \\ -1 & \text{if} \quad \sum_a h_a^{(0)} < 0 \\ 0 & \text{if} \quad \sum_a h_a^{(0)} = 0 \end{cases}. \tag{5.28}$$

It is clear that the visible configuration will flow to a configuration of either all up or all down spins eventually, if not already after the first iteration. This iterative procedure can be seen as a flow of configurations at some finite T (if we assume the initial configuration has at least some disorder in it) to a configuration at $T_c = 0$. It is precisely this flow of configurations to the critical point which is analogous to the flow seen in RG, where degrees of freedom are iteratively integrated out until the system reaches a critical point, effectively changing the temperature of the classical system along the way.

The derivation above effectively shows the RBM is able to model the Ising chain at $K \to \infty$ perfectly well, and mimics the iterative procedure of flow towards the critical point. However at finite temperatures the system will not flow towards the critical point, as W will be finite and so the probability of getting a single type of spin will always be smaller than 1. Likewise, we would like to obtain a physical interpretation in what the RBM learns from the model, specifically when the number of hidden neurons is smaller than the number of visible neurons. To obtain some physical insight to the RBM we first devise an exact mapping between the RBM and the Ising chain.

5.3 A spin barrier based mapping between the Ising chain and $L \times L$ RBM

Since the Ising chain's probability distribution only depends on the amount of spin-barriers in each spin configuration, one might expect that a RBM based on spin-barriers will be able to model the Ising chain really well. To explore this idea, we begin with the RBM probability distribution over the visible units for a given spin configuration $\{\mathbf{v}\}$:

$$p(\{\mathbf{v}\}) = \frac{1}{Z}\prod_{a}^{N} 2\cosh\left(\sum_{i}^{L} v_i W_{ia}\right). \tag{5.29}$$

A natural way to force the distribution to model to barriers is to only consider the sum of spin with its neighbour for each hyperbolic cosine argument, so $v_i + v_{i+1}$. This term will be labelled by B_i, which represents the barrier between sites i and $i+1$:

$$B_i = v_i + v_{i+1}. \tag{5.30}$$

We have a barrier if $B_i = 0$, so when the spins at i and $i+1$ are anti-aligned. If the spins have the same sign we have either $B_i = -1$ or $B_i = +1$, which leads to the same factor in the probability distribution since cosh is invariant under a sign change in the argument. In this scheme each hyperbolic cosine argument carries only one B_i term, so we need as many hyperbolic cosines as there are sites. We thus have L hyperbolic cosines. This means that there are as many hidden as visible neurons in the RBM. With this set-up only a single W value is needed, as the term in the hyperbolic cosine should be the same for $B_i = +1$ or $B_i = -1$, as well as taking into account the translation invariance of the system. This gives us

$$p(\{\mathbf{v}\}) = \frac{1}{Z}\prod_{i}^{L} 2\cosh\left(WB_i\right). \tag{5.31}$$

The relatively simple form of this equation allows us to find an explicit expression for the partition function Z using the transfer-matrix method:

$$Z = \sum_{\{\mathbf{v}\}}\prod_{i}^{L} 2\cosh\left(W(v_i + v_{i+1})\right) \tag{5.32}$$

$$= 2^L \,\mathrm{Tr}[\mathcal{T}^L], \tag{5.33}$$

where the transfer-matrix $\mathcal{T}$ is given by

$$\mathcal{T} = \begin{bmatrix} \cosh(2W) & 1 \\ 1 & \cosh(2W) \end{bmatrix}, \tag{5.34}$$

which has the eigenvalues

$$\lambda_{\pm} = \cosh(2W) \pm 1. \tag{5.35}$$

We can now easily calculate Z, since the trace of a matrix is equal to the sum of its eigenvalues, and the eigenvalues of a product of equal matrices are the product of the eigenvalues of the individual matrices corresponding to the same eigenvector.

So the partition function becomes

$$\begin{aligned} Z &= 2^L \left[(\lambda_+)^L + (\lambda_-)^L \right] \\ &= 2^L \left[(\cosh(2W) + 1)^L + (\cosh(2W) - 1)^L \right] \\ &= 2^L \left[(2\cosh^2(W))^L + (2\sinh^2(W))^L \right]. \end{aligned} \tag{5.36}$$

Note that this looks very similar to the partition function for the Ising chain given in equation (5.15). To see if this distribution is indeed capable of modelling the Ising chain we follow the usual procedure of minimising the KL-divergence. We start by calculating the left hand side term:

$$\begin{aligned} \frac{1}{Z}\frac{\partial Z}{\partial W} &= \frac{2^L}{Z} L[4(2\cosh^2(W))^{L-1}\cosh(W)\sinh(W) \\ &\quad + 4(2\sinh^2(W))^{L-1}\sinh(W)\cosh(W)] \\ &= \frac{2L\sinh(W)\cosh(W)\left(\cosh^{2L-2}(W) + \sinh^{2L-2}(W)\right)}{\cosh^{2L}(W) + \sinh^{2L}(W)}. \end{aligned} \tag{5.37}$$

The right hand side (rhs) is slightly more complicated, but can also be brought into a nice expression. After some rewriting we end up with

$$\begin{aligned} \text{rhs} &= \sum_{\{\mathbf{v}\}} q(\{\mathbf{v}\}) \frac{1}{p(\{\mathbf{v}\})} \frac{1}{Z} \frac{\partial}{\partial W} \left(\prod_i^L 2\cosh(WB_i)\right) \\ &= \sum_{\{\mathbf{v}\}} q(\{\mathbf{v}\}) \frac{1}{\prod_i \cosh(WB_i)} \sum_j^L B_j \sinh(WB_j) \prod_{i \neq j} \cosh(WB_i) \\ &= \sum_{\{\mathbf{v}\}} q(\{\mathbf{v}\}) \sum_j^L B_j \tanh(WB_j) \\ &= \frac{1}{Z_{\text{Ising}}} \sum_{\{\mathbf{v}\}} \prod_i^L \exp(K(v_i v_{i+1})) \sum_j^L (v_j + v_{j+1}) \tanh(W(v_j + v_{j+1})). \end{aligned} \tag{5.38}$$

This expression can also be calculated using the transfer-matrix method, albeit with some extra steps. We note that it is possible to write the configurational sum as the trace of the product of the transfer matrices we would have for the Ising chain as well as one different matrix representing the j to $j+1$ transfer terms. We first focus on just one of the $v_j + v_{j+1}$ terms, as all the terms will just contribute the same thing to the rhs. We can write one such term as

$$\begin{aligned} &\sum_{\{\mathbf{v}\}} \prod_i^L \exp(K(v_i v_{i+1}))(v_j + v_{j+1}) \tanh(W(v_j + v_{j+1})) \\ &= \text{Tr}[\mathcal{T}_1 \mathcal{T}_2 \cdots \mathcal{K}_j \cdots \mathcal{T}_L], \end{aligned} \tag{5.39}$$

where the transfer-matrices denoted by $\mathcal{T}_i$ are the usual Ising chain transfer matrices, while $\mathcal{K}_j$ is the transfer matrix corresponding to the j and $j+1$ terms:

$$\mathcal{T}_i = \begin{bmatrix} \exp(K) & \exp(-K) \\ \exp(-K) & \exp(K) \end{bmatrix} \tag{5.40}$$

$$\mathcal{K}_j = \begin{bmatrix} \exp(K) 2\tanh(2W) & 0 \\ 0 & \exp(K) 2\tanh(2W) \end{bmatrix}, \tag{5.41}$$

here we used that $\tanh -x = -\tanh x$. Since $\mathcal{K}_j$ only has diagonal terms which are equal, we can write it as a 2×2 identity matrix, $\mathbb{1}$, multiplied with a prefactor. This allows us to rewrite the trace as

$$\begin{aligned} \mathrm{Tr}[\mathcal{T}_1 \dots \mathcal{K}_j \dots \mathcal{T}_L] &= 2\mathrm{e}^K \tanh(2W) \mathrm{Tr}[\mathcal{T}_1 \cdots \mathbb{1} \cdots \mathcal{T}_L] \\ &= 2\mathrm{e}^K \tanh(2W) \mathrm{Tr}[\prod_{i \neq j} \mathcal{T}_i] \\ &= 2\mathrm{e}^K \tanh(2W) \left((2\cosh(K))^{L-1} + (2\sinh(K))^{L-1} \right). \end{aligned} \tag{5.42}$$

Note that the calculation of this term did not depend on the j and $j+1$ sites, which means that the sum over j in the rhs equation will just give an extra factor L. So we end up with

$$\mathrm{rhs} = \frac{\cosh^{L-1}(K) + \sinh^{L-1}(K)}{\cosh^L(K) + \sinh^L(K)} \cdot L\mathrm{e}^K \tanh(2W). \tag{5.43}$$

So minimising the KL-divergence equates to solving the equation

$$\frac{1}{Z}\frac{\partial Z}{\partial W} = \mathrm{rhs}. \tag{5.44}$$

A solution to equation (5.44) can be found by first rewriting the equations in terms of

$$\begin{aligned} x &= \mathrm{e}^{2W}, \qquad \text{and} \\ y &= \mathrm{e}^K. \end{aligned} \tag{5.45}$$

The left hand side equation (5.37) then takes the following form:

$$\begin{aligned} \frac{\partial \ln Z}{\partial W} &= \frac{2L(x^{1/2} - x^{-1/2})(x^{1/2} + x^{-1/2}) \left[(x^{1/2} + x^{-1/2})^{2L-2} + (x^{1/2} - x^{-1/2})^{2L-2} \right]}{(x^{1/2} + x^{-1/2})^{2L} + (x^{1/2} - x^{-1/2})^{2L}} \\ &= \frac{2L(x - x^{-1}) \sum_{k=\text{even}}^{2L-2} \binom{2L-2}{k} x^{L-1-k}}{\sum_{k=\text{even}}^{2L} \binom{2L}{k} x^{L-k}}. \end{aligned} \tag{5.46}$$

The rhs takes the form

$$\begin{aligned} \mathrm{rhs} &= \frac{2L(y + y^{-1})^{L-1} + (y - y^{-1})^{L-1}}{(y + y^{-1})^L + (y - y^{-1})^L} y \frac{x - x^{-1}}{x + x^{-1}} \\ &= \frac{2L \sum_{k=\text{even}}^{L-1} \binom{L-1}{k} y^{L-1-k}}{\sum_{k=\text{even}}^{L} \binom{L}{k} y^{L-k}} y \frac{x - x^{-1}}{x + x^{-1}}. \end{aligned} \tag{5.47}$$

A solution can then be found by considering the $L = 2$ case:

$$\frac{-1 + x^4}{1 + 6x^2 + x^4} = \frac{(-1 + x^2)y^4}{(1 + x^2)(1 + y^4)}. \tag{5.48}$$

This equation can be solved for $y > 1$ and gives

$$x_\pm = y^2 \pm \sqrt{y^4 - 1}, \tag{5.49}$$

which can be transformed back in terms of W and K to give

$$W_\pm = \frac{1}{2} \ln \left(\exp(2K) \pm \sqrt{\exp(4K) - 1} \right). \tag{5.50}$$

Here the $\pm$ comes from the W-sign symmetry present in $p(\mathbf{v})$. While proving that equation (5.50) is a general solution that minimises the KL-divergence, for example by induction, is not as straightforward as it sounds, it can be confirmed to hold numerically up to small system sizes. This is taken as a sign that this equation holds for any system size L. The KL-divergence can be calculated numerically as well, and gives 0 up to numerical precision. We take this as a strong indication that equation (5.50) gives the value of W expressed in K which exactly maps the RBM as described in equation (5.31) to the classical Ising chain. Figure 5.3 shows a plot of equation (5.50). It can easily be seen that in the limit $K \to \infty$, $\lim_{K\to\infty} W_\pm = \pm\infty$. So we expect similar flow behaviour of the RBM in the $K \to \infty$ region as was seen for the mean-field RBM of the previous section.

While the exact mapping between the Ising chain and the RBM might give insight in solutions found by numerical training of the RBM on an Ising chain, it is ultimately based on having an equal number of visible and hidden nodes. While having an equal number of visible and hidden nodes is not uncommon, we are primarily interested in what the RBM will pick up when the degrees of freedom are effectively reduced by making the step from the visible to hidden layer. This is done by making sure the hidden layer has less nodes than the visible layer, in analogy with real-space RG, where the number of spins in the system is reduced in a step-wise manner.

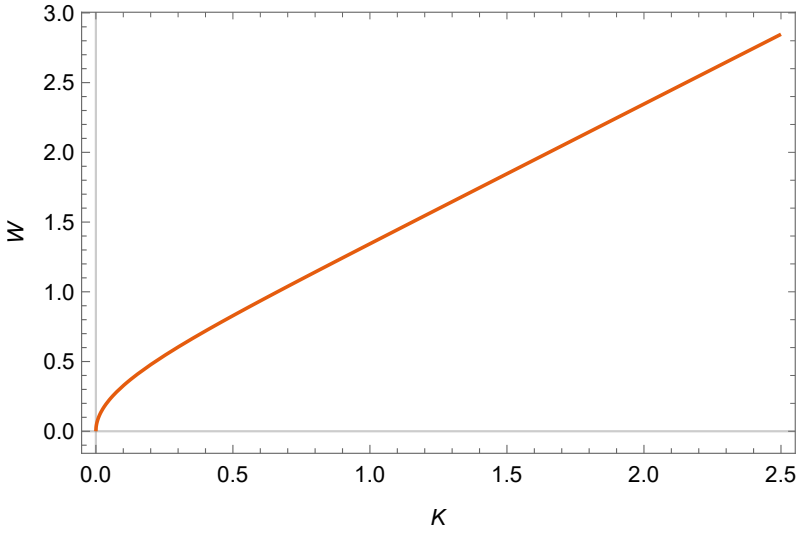


FIGURE 5.3: *Plot of $W(K)$ given by equation* (5.50).

5.4 Finding W-restrictions based on translation invariance: the weave-method

It should be clear that reducing $\mathbf{W}$ to a single W-value is not sufficient to model the Ising chain, at least not for a RBM with less hidden neurons than visible neurons. In this section we aim to reduce the degrees of freedom in $\mathbf{W}$ by symmetry arguments. The spin-flip symmetry $\mathbf{v} \to -\mathbf{v}$ of the Ising chain is already made present in $p(\mathbf{v})$ by setting the biases to 0. Restrictions on $\mathbf{W}$ can be made by assuming that the translation invariance present in the Ising chain should also be present in the RBM. We investigate this assumption by studying a 6×3 RBM. For this RBM, the probability distribution is given by

$$p(\mathbf{v}) = \frac{1}{Z} 2^3 \cosh\left(\sum_{i=1}^{6} v_i W_{i1}\right) \cosh\left(\sum_{i=1}^{6} v_i W_{i2}\right) \cosh\left(\sum_{i=1}^{6} v_i W_{i3}\right). \tag{5.51}$$

Note that the spin-flip symmetry is already present in the probability distribution, since $\cosh(-x) = \cosh(x)$. There are multiple ways to implement the translation invariance. Equation (5.51) should be equal for all configurations connected through translations. Since the probability distribution consists of a product of three hyperbolic cosines this equating has multiple solutions, which can be visualised as three weaves representing the equating arguments of each individual hyperbolic cosine. This is visualised in figure 5.4, with a more thorough interpretation of the figure given in its description. Changing the weaves and configurations and connecting the different solutions will give different constraints on the weight-matrix components.

TABLE 5.1: *Spin configurations ordered by probability, N_B is the number of barriers (anti-aligned spin pairs).*

N_B	Spin configurations	$\lvert M\rvert$	N° configurations	$Z_{\text{Ising}} p(\mathbf{v})$
0	$\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow$, $\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$	6	2	$\exp(6K)$
2	$\downarrow\uparrow\uparrow\uparrow\uparrow\uparrow$ + translations & spin-flip	5	12	$\exp(2K)$
	$\downarrow\downarrow\uparrow\uparrow\uparrow\uparrow$ + translations & spin-flip	4	12	
	$\downarrow\downarrow\downarrow\uparrow\uparrow\uparrow$ + translations & spin-flip	3	6	
4	$\downarrow\uparrow\downarrow\uparrow\uparrow\uparrow$ + translations & spin-flip	4	12	$\exp(-2K)$
	$\downarrow\uparrow\uparrow\downarrow\uparrow\uparrow$ + translations & spin-flip	4	6	
	$\downarrow\downarrow\uparrow\downarrow\uparrow\uparrow$ + translations & spin-flip	3	12	
6	$\downarrow\uparrow\downarrow\uparrow\downarrow\uparrow$, $\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow$	3	2	$\exp(-6K)$

Table 5.1 shows the spin configurations sharing the same probability in the Ising chain and the absolute magnetisation, barriers in the configuration and the total number of configurations which share all these qualities. We start with the braiding of the $|M| = 5$ terms, so the terms with one spin pointing in the opposite direction to all the other spins. We braid the arguments such that $\sum_i W_{ia}$ relates to $\sum_i W_{ia+1}$ for the translation of the spin configuration by one site. This can be visualised as displayed in figure 5.4, where the three different lines symbolising equating the arguments all run in parallel to one another with periodic boundary conditions. Solving this set of

equations gives the three arguments:

$$\sum_i W_{ia} v_i = W_{11}v_1 + W_{31}v_2 + W_{21}v_3 + W_{11}v_4 + W_{31}v_5 + W_{21}v_6 \tag{5.52}$$

$$\sum_i W_{i2} v_i = W_{21}v_1 + W_{11}v_2 + W_{31}v_3 + W_{21}v_4 + W_{11}v_5 + W_{31}v_6 \tag{5.53}$$

$$\sum_i W_{i3} v_i = W_{31}v_1 + W_{21}v_2 + W_{11}v_3 + W_{31}v_4 + W_{21}v_5 + W_{11}v_6. \tag{5.54}$$

So there are three dependent weight-matrix components: W_{11}, W_{12} and W_{13}. With this choice of braiding the translational invariance of each spin configuration explicitly shown in table 5.1 is guaranteed, and they all keep the same braiding equivalence. One observation has to be noted: Spin configurations with $|M| = 3$ and 2 or 4 barriers all have hyperbolic cosine arguments equal to 0 with this choice of braiding. This means that the RBM will be unable to differentiate between $|M| = 3$ states with 2 or 6 barriers, even though the Ising chain clearly favours the 2 barriers.

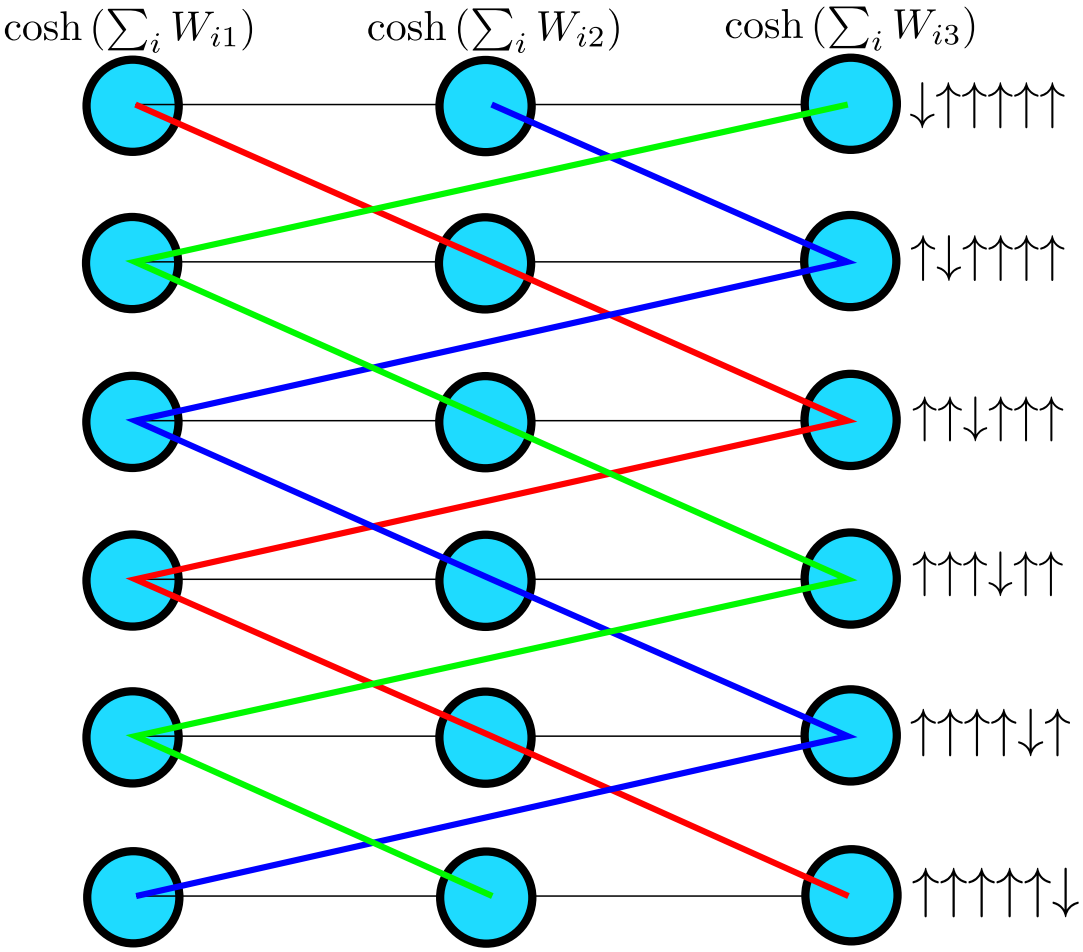


FIGURE 5.4: *Graphical representation of possible solution to a translation invariant $p(\mathbf{v})$ given the translation invariance of the spin configuration ↓↑↑↑↑↑ and its translations in $q(\mathbf{v})$. Each black dot represents the argument of the hyperbolic cosine terms relating to one of the hidden nodes as indicated above and a corresponding configuration as indicated to the right. A black line between the dots indicates a product between the hyperbolic cosines. Each coloured line (red, blue, green) represents equating the connected hyperbolic cosine arguments so that the product of hyperbolic cosines is equivalent for each spin configuration.*

In general the **W**-matrix with $L \times N$ degrees of freedom (d.o.f.) can be reduced by $N_{\text{tr}} \times N$ d.o.f. at most, where N_{tr} is the number of translations of a particular spin configurations which leaves $q(\mathbf{v})$ unchanged and N is the number of hidden neurons. For our 6×3 RBM reduced by the $|M| = 5$ configurations the weight-matrix is reduced by $5 \times 3 = 15$ d.o.f., so that the reduced **W** has $18 - 15 = 3$ d.o.f. left. The way the weaves are connected determines the form of the matrix, but the d.o.f. should be the same, as long as the weaves do not contain doubly counted equations. Thus there is a certain ambiguity in choosing the weave to use in reducing the d.o.f., especially since there is no clear cut way to make sure the chosen weave gives the lowest KL-divergence.

To counteract this problem and provide an argument that the weave-formalism is a sensible one, a comparison to the unrestricted RBM is made. The comparison consists of numerically finding a minimum in the KL-divergence using conventional numerical techniques, and repeating this several times from different (randomly chosen) initial positions, choosing the solution with the minimal KL-divergence overall. We call a solution generated with this procedure a swarm-solution. The swarm-solution is then compared to the solution which minimises the KL-divergence of the reduced **W**-matrix. If the performance of the reduced solution is lower or comparable to the performance of the unrestricted RBM, the **W**-reduction is deemed valid.

The comparison between the mean-field RBM and the translation invariant RBM can be made in a similar manner. Figure 5.5 shows the KL-divergence for the 6×3 RBM restricted with the weave, as well as the mean-field 6×3 RBM. The results for the 4×2 RBMs are shown as well. The figure shows a clear improvement for the 4×2 model, while the 6×3 model shows only a very small improvement. This is as expected, as both models are unable to fully model the Ising chain. Nevertheless we expect correlations to be picked up better by the translation invariant RBM, as will be seen in comparisons of the variance of $|M|$.

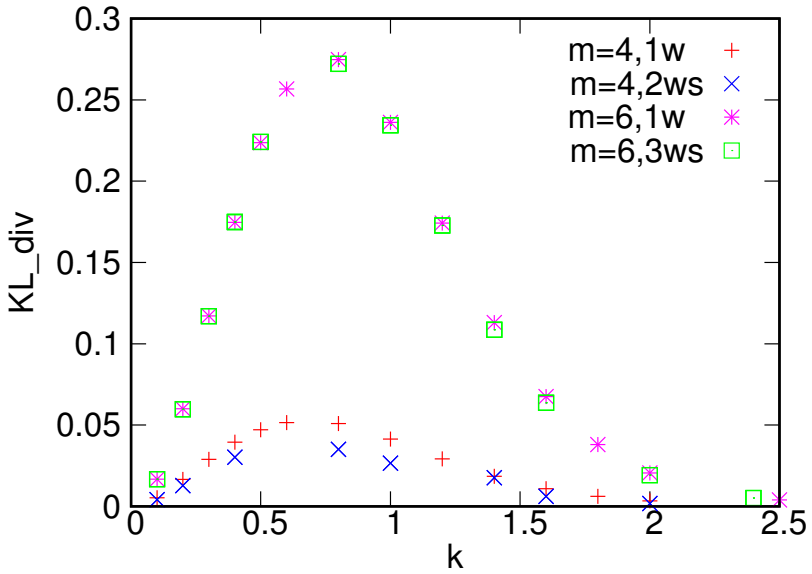


FIGURE 5.5: *KL-divergence comparison between the mean-field RBM and translation invariant RBM for* 4×2 *and* 6×3 *RBMs. Figure courtesy of W. Zhong[70].*

5.5 A study of the translation invariant RBM

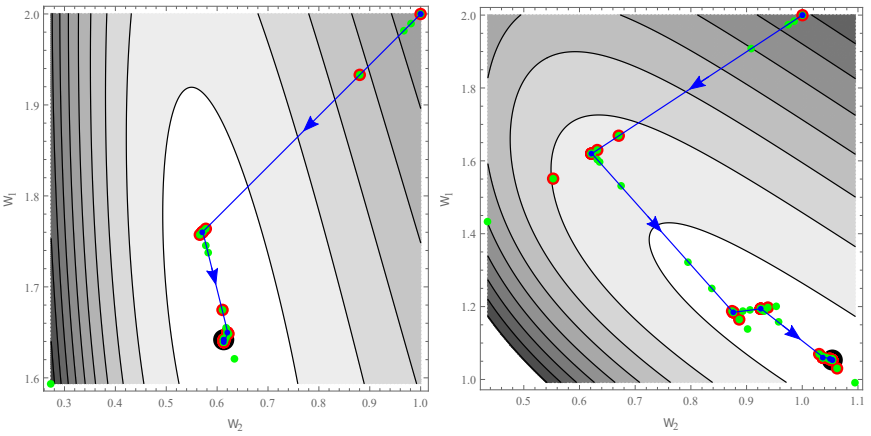
The weave-method described in the section above provides a systematic method for restricting the **W**-matrix in such a manner that the RBM is translation invariant. By restricting the **W**-matrix, the parameter-space gets significantly restricted. This allows for a faster finding of solutions that minimise the KL-divergence. However, there is no guarantee that the found minimum in the restricted parameter-space corresponds to a local minimum in the unrestricted parameter-space. The solution found in the restricted parameter-space could for example be a saddle-point in the unrestricted parameter-space. To get a better understanding of how the restricted RBM compares to the unrestricted case, a numerical analysis for a small system size $L = 4$ was done.

5.5.1 The stability of a 4×2 translation invariant RBM

We start with a very small 4×2 RBM. Using the weave-formalism, we can derive that there are two candidates that restrict **W** such that the RBM is translation invariant:

$$\mathbf{W}_1 = \begin{pmatrix} W_1 & W_2 & W_2 & W_1 \\ W_2 & W_1 & W_1 & W_2 \end{pmatrix}, \quad \text{and} \quad \mathbf{W}_2 = \begin{pmatrix} W_1 & W_2 & W_1 & W_2 \\ W_2 & W_1 & W_2 & W_1 \end{pmatrix}. \tag{5.55}$$

A solution that minimises the KL-divergence can then by found by the conjugate gradient method, starting from some initial set of W's. An example of such a minimisation is shown in figure 5.6. This process is repeated for many randomly chosen initial values, where the solution with the lowest final KL-divergence is chosen as the ultimate solution. This procedure does not guarantee that the RBM ends up in the absolute minimum of the KL-divergence, but the chances of ending in a 'sufficiently good' minimum are high. The final solutions for $\mathbf{W}_1$ and $\mathbf{W}_2$ for $K = 1.2$ are $(W_1, W_2) \approx (1.64, 0.61)$ and $(W_1, W_2) \approx (1.05, 1.05)$ respectively. Since the solution for $\mathbf{W}_2$ corresponds to the single-W valued RBM, this choice of restriction is discarded.



(A) $\mathbf{W}_1$ (B) $\mathbf{W}_2$

FIGURE 5.6: *Plots visualising the conjugate gradient minimisation method of the TI RBMs from the initial value $(W_1, W_2) = (1, 2)$ at $K = 1.2$. Green and red dots correspond to function and gradient evaluations respectively. The blue dots and lines show the gradient steps, where the arrow indicates the direction of the step.*

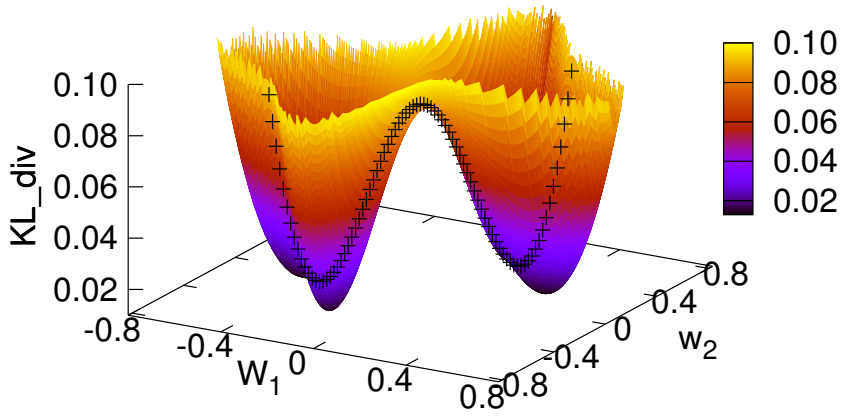


FIGURE 5.7: *Visualisation of the two-parameter* **W***-space KL-divergence. The black plusses show the* $W_1 = W_2$ *restriction to a single W-parameter. Figure courtesy of W. Zhong[70].*

The stability of the $\mathbf{W}_1$ solution with respect to the unrestricted RBM at $K = 1.2$ can be tested by initialising a gradient descend in the unrestricted parameter-space starting from the $\mathbf{W}_1$ solution. If the gradient descent is initialised from exactly the $\mathbf{W}_1$ solution, the solution remains unchanged. However, if a small deviation δ is made in any W_{ij}-direction, gradient descent flow to another solution. This is an indication that, at least at this particular value of K, the $\mathbf{W}_1$ solution is either a very shallow local minimum in the unrestricted parameter-space or a relatively flat saddle-point. As an example of this, consider the KL-divergence for a model with two free W-parameters: W_1 and W_2. The single-W restriction would then be $W_1 = W_2$. As shown in figure 5.7, the minimum of the KL-divergence for the single-W restriction clearly falls on the saddle-points of the KL-divergence in the unrestricted **W**-space.

Naturally this can be taken as an argument to disregard the weave-method and instead look at the unrestricted case. However, we would like to argue that the weave-method is still valid. The first argument is based on the difference in KL-divergence between the shallow minimum of the $\mathbf{W}_1$ solution and the minimum in the unrestricted parameter space. For $K = 1.2$ this difference is of the order $\mathcal{O}(10^{-2})$, which we consider relatively small. It is almost guaranteed that the unrestricted RBM will perform better than the restricted RBM, since it is likely overdetermined. However, we are interested in which aspects of the physical system the RBM picks up the most. In an overdetermined system all aspects of the physical system are in principle included, which makes making statements of relevant and irrelevant variables as decided by the RBM impossible.

The second argument is based on the importance of translation invariance. In physical systems this symmetry is a fundamental aspect of their behaviour. The unrestricted RBM solution does not necessarily have this symmetry, while the weave-formalism guarantees translation invariance. The small difference in KL-divergence is then a small price to pay for retaining this symmetry.
Perhaps the most important argument of all is the generalisation to larger system

sizes. By restricting the free parameters by a significant amount, the computation time decrease tremendously. This allows for a study of larger system sizes.

If we take the weave-method to be able to give us valuable answers, we can take $\mathbf{W}_1$ as the $\mathbf{W}$-matrix and minimise the KL-divergence for a range of K. This minimisation can be done numerically by the swarm-method described earlier. The found values for W_1 and W_2 are shown in figure 5.8 together with the average values of $|M|$, M^2 and $\mathrm{Var}(|M|)$. While this is only a very small system size, the variance of $|M|$ is already marginally better than variance measured for the single-W case. This is most likely a direct result from the inclusion of more W-values, which allow the RBM to model the spin-spin correlations more accurately. These correlations are precisely the quantities that are important in capturing the variance of $|M|$.

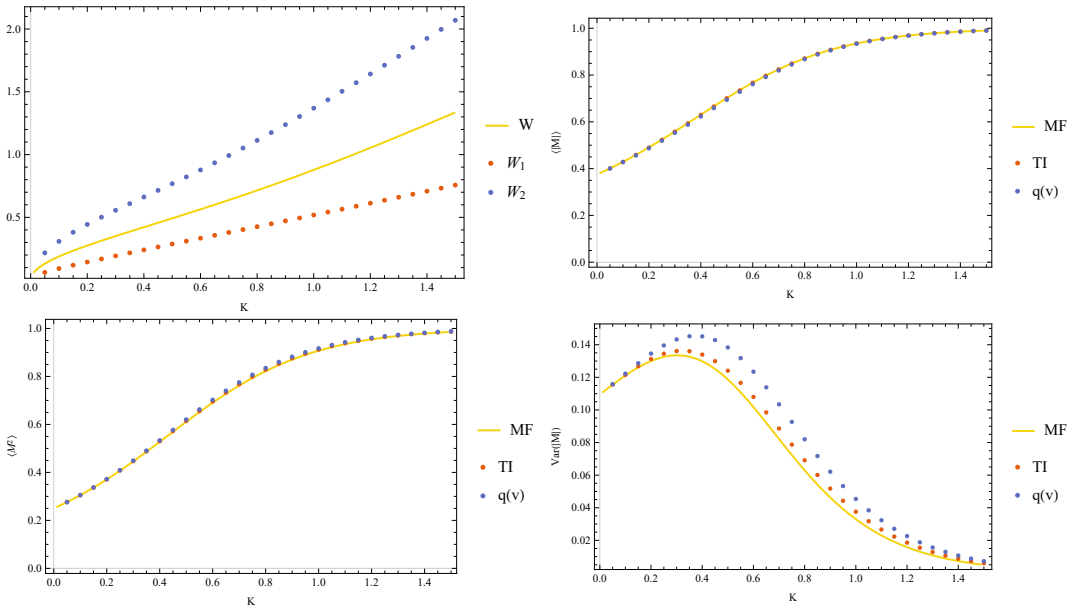


FIGURE 5.8: *The upper left figure depicts the* $\mathbf{W}_1$ *values that minimise the KL-divergence for a* 4×2 *translation invariant RBM (TI) for different values of K. The other figures show different physical quantities related to M comparing the results from the TI RBM to the Ising chain. The results for a* 4×2 *mean field RBM (MF) are shown in yellow for comparison.*

5.6 Analysing the RBM flow

In the previous sections we have discussed several different iterations of the RBM. We now turn our attention to the flow of these particular instances of the RBMs. To this end we first need to calculate the conditional probability:

$$\begin{aligned} p(\mathbf{h}|\mathbf{v}) &= \frac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{v})} \\ &= \frac{\mathrm{e}^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\{\mathbf{h}\}} \mathrm{e}^{-E(\mathbf{v},\mathbf{h})}}. \end{aligned} \tag{5.56}$$

Note that this conditional probability is normalised over all distributions of $\{\mathbf{h}\}$ given any $\mathbf{v}$. To iterate back to the visible layer we will also need the other conditional probability:

$$\begin{aligned} p(\mathbf{v}|\mathbf{h}) &= \frac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{h})} \\ &= \frac{\mathrm{e}^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\{\mathbf{v}\}} \mathrm{e}^{-E(\mathbf{v},\mathbf{h})}}. \end{aligned} \tag{5.57}$$

The RBM flow of probabilities will then be given by the mapping

$$\pi^{(n)}(\mathbf{v}) = \frac{1}{Z_n} \sum_{\{\mathbf{h}\}} p(\mathbf{v}|\mathbf{h}) \sum_{\{\mathbf{v}\}} p(\mathbf{h}|\mathbf{v}) \pi^{(n-1)}(\mathbf{v}), \tag{5.58}$$

where the superscript (n) indicates the iteration number starting from some initial $n = 0$ distribution $\pi^{(0)}$. Z_n is the partition function making sure $\pi^{(n)}(\mathbf{v})$ is properly normalised, it is defined as

$$Z_n = \sum_{\{\mathbf{v}\}} \sum_{\{\mathbf{h}\}} p(\mathbf{v}|\mathbf{h}) \sum_{\{\mathbf{v}\}} p(\mathbf{h}|\mathbf{v}) \pi^{(n-1)}(\mathbf{v}). \tag{5.59}$$

The Markov chain defined by equation (5.58) then provides insight into the block Gibbs sampling performed by the trained RBM, given some initial $\{\mathbf{v}\}$ distribution. Since RBM's are most commonly used as generative models, a trained RBM is used to come up with new configurations that are similar to the set of configurations it trained itself on. This generating of new configurations can be done through Gibbs sampling, where the first input configuration could for example be chosen at random and a new configuration can be generated after a few iterations of Gibbs sampling. Here, rather that inputting a single configuration, we consider the results of putting in configurations from some initial distribution $\pi^{(0)}(\mathbf{v})$, and seeing what the distribution of the generated configurations at a step k, $\pi^{(k)}(\mathbf{v})$, looks like.

It can easily be checked that $p(\mathbf{v})$ is a stationary solution to this mapping by just

filling it in for the n- and $n-1$-th iterations:

$$\begin{aligned} p(\mathbf{v}) &= \frac{1}{Z_n} \sum_{\{\mathbf{h}\}} p(\mathbf{v}|\mathbf{h}) \sum_{\{\mathbf{v}\}} p(\mathbf{h}|\mathbf{v}) p(\mathbf{v}) \\ &= \frac{1}{Z_n} \sum_{\{\mathbf{h}\}} p(\mathbf{v}|\mathbf{h}) \sum_{\{\mathbf{v}\}} p(\mathbf{v}, \mathbf{h}) \\ &= \frac{1}{Z_n} \sum_{\{\mathbf{h}\}} \frac{p(\mathbf{v}, \mathbf{h})}{p(\mathbf{h})} p(\mathbf{h}) \\ &= \frac{1}{Z_n} p(\mathbf{v}). \end{aligned} \tag{5.60}$$

Since $Z_n = \sum_{\{\mathbf{v}\}} p(\mathbf{v}) = 1$, the equation clearly holds.

5.6.1 4-sited Ising chain and the RBM flow

We first consider a small system of size 4×2. We consider the translation invariant RBM with two free W-values, the mean-field RBM with a single W value, and the unrestricted RBM with all W's free. A comparison of these RBMs to the 4×4 exact mapping of the RBM to the Ising chain is made as well.

We first consider the RBM flow starting from the same distribution the RBM was trained on, so that

$$\pi^{(0)} = q(\mathbf{v}, K_0 = K), \tag{5.61}$$

where the dependence of q on K_0 is shown explicitly for clarity. This flow corresponds to training an RBM on Ising configurations at some fixed K_0, and generating new configurations from the RBM by Gibbs sampling starting from the same Ising configurations. Figure 5.9 shows the average absolute magnetisation and variance of the absolute magnetisation for the different RBMs after 20 Gibbs sampling iterations ($k = 20$).

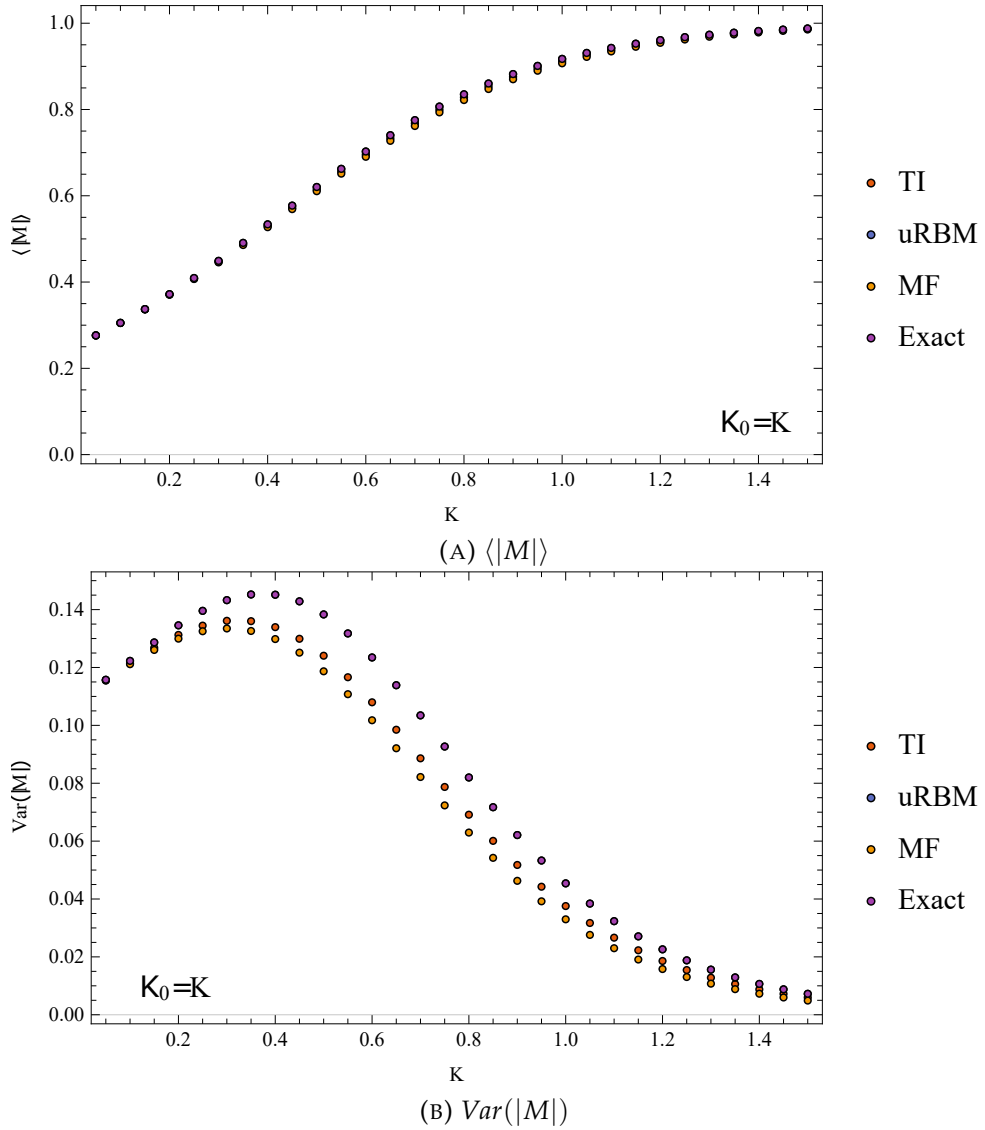


FIGURE 5.9: *Average absolute magnetisation and variance of the absolute magnetisation for the RBM flow configuration $\pi^{(k)}$ after 20 iterations ($k = 20$) starting from $\pi^{(0)} = q(\mathbf{v}, K)$. The different RBMS are labelled by different colours as depicted in the legend.*

From the figures we can see that the flow behaviour is as expected, after 20 iterations all RBMs have reached their stationary distributions $p(\mathbf{v})$, at least in terms of the observable behaviour of the distribution. Most RBMs reach the equilibrium distribution in as little as 1 or 2 iterations, which makes sense starting from a distribution which should already be very close to the stationary distribution. The unrestricted RBM and the exact RBM are the closest to the behaviour of $q(\mathbf{v}, K)$, as they should be. The exact RBM is an exact mapping between the Ising chain and the RBM, so that the stationary distribution $p(\mathbf{v}, W)$ is equivalent to $q(\mathbf{v}, K)$. The unrestricted RBM is perhaps overdetermined, it is able to minimise the KL-divergence to machine precision, thus reaching behaviour close to that of the Ising chain.

A common way to obtain configurations from the RBM is by initialising the Gibbs sampling from a random configuration of $\mathbf{v}$. This corresponds to starting the RBM flow from a discrete uniform distribution $U(\{\mathbf{v}\})$, where every distribution has

equal probability of occurring. The discrete uniform distribution is equivalent to the Ising chain at infinite temperature, at $K_0 = 0$. Thus we start the RBM flow from

$$\pi^{(0)} = U(\{\mathbf{v}\}) = q(\mathbf{v}, K_0 = 0). \tag{5.62}$$

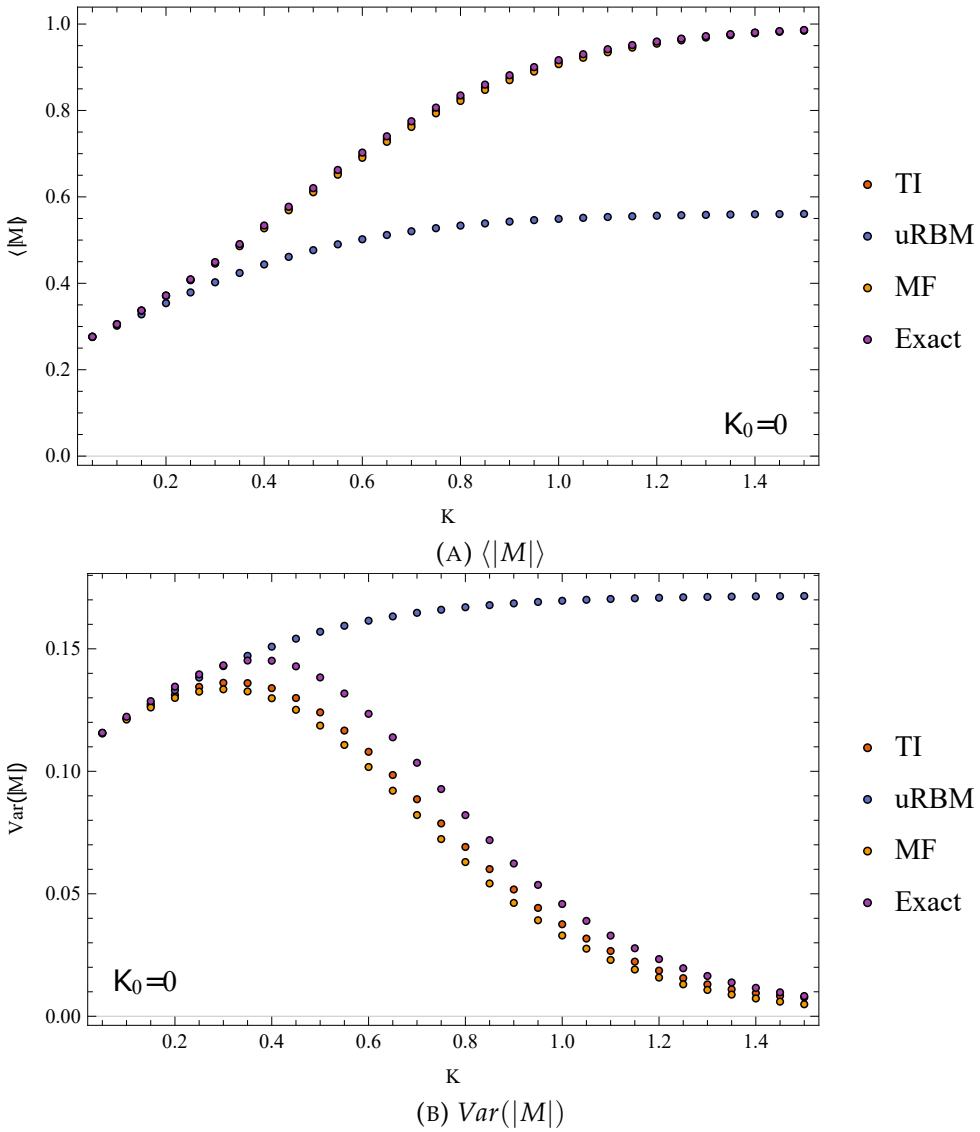


FIGURE 5.10: *Average absolute magnetisation and variance of the absolute magnetisation for the RBM flow configuration $\pi^{(k)}$ after 20 iterations ($k = 20$) starting from $\pi^{(0)} = q(\mathbf{v}, K = 0)$. The different RBMS are labelled by different colours as depicted in the legend.*

Figure 5.10 shows the average absolute magnetisation and variance of the absolute magnetisation for $k = 20$ of the RBM flow. The mean-field RBM, translation invariant RBM and exact RBM all flow towards the stationary distribution $p(\mathbf{v})$, as expected. The unrestricted RBM shows quite unexpected behaviour however, it flows to a stationary distribution, but this distribution does not behave like $p(\mathbf{v})$ at all. This is quite unexpect and will be explored in more detail after a discussion of the RBM flows starting from $\lim_{K_0 \to \infty} q(\mathbf{v}, K_0)$.

The unexpected behaviour of the unresticted RBM shows a deviation from the RBM

flow to $p(\mathbf{v})$, that the other RBMs do appear to adhere to. To explore the RBM flow in more detail we initialise the flow from the other extreme: $q(\mathbf{v}, K \to \infty)$. This corresponds to the Ising model at zero temperature, which means that only the configurations with all spins up or down have a finite probability: $1/2$. We start the RBM flow from

$$\pi^{(0)} = q(\mathbf{v}, K \to \infty). \tag{5.63}$$

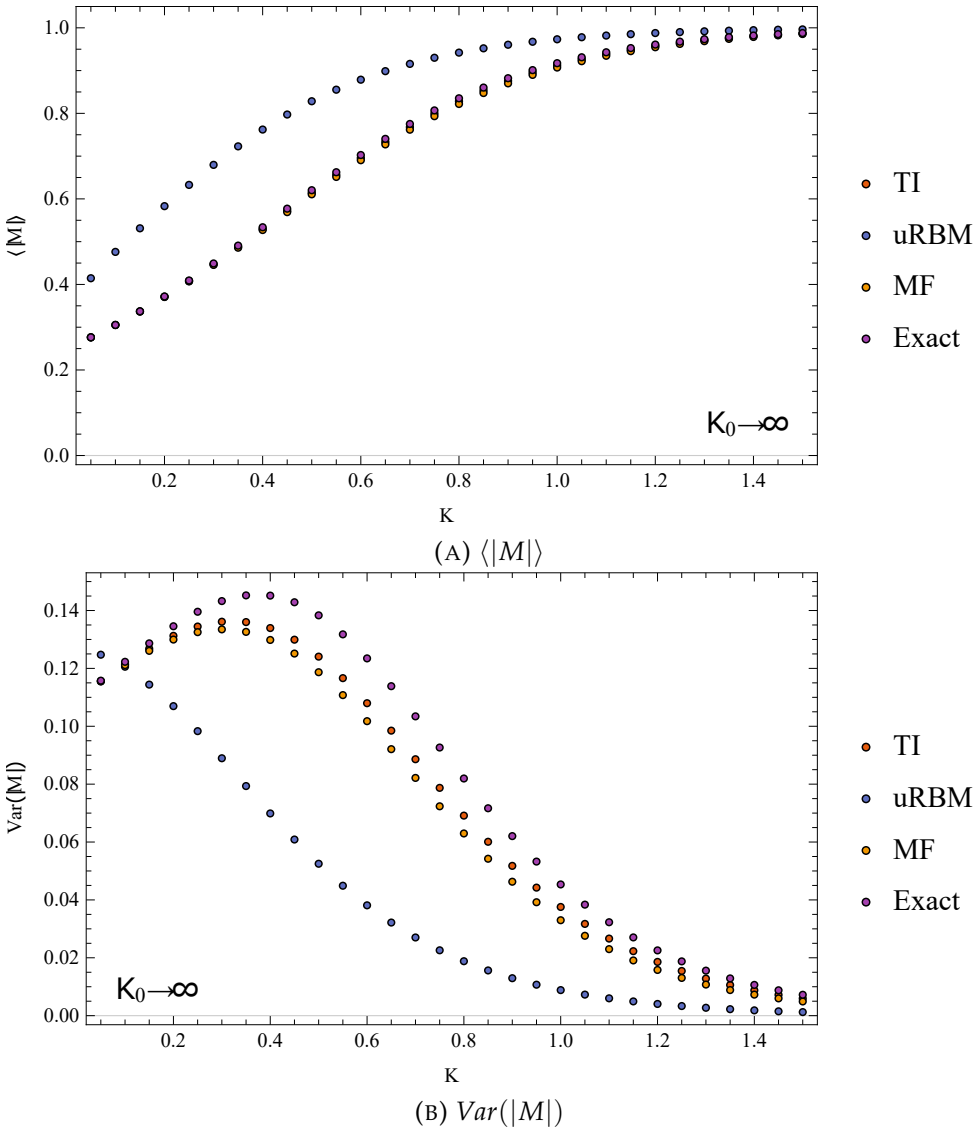


FIGURE 5.11: *Average absolute magnetisation and variance of the absolute magnetisation for the RBM flow configuration $\pi^{(k)}$ after 20 iterations ($k = 20$) starting from $\pi^{(0)} = q(\mathbf{v}, K_0 \to \infty)$. The different RBMS are labelled by different colours as depicted in the legend.*

Figure 5.11 show the average absolute magnetisation and variance of the absolute magnetisation of the RBM flow at $k = 20$ starting from $q(\mathbf{v}, K_0 \to \infty)$. Again all RBMs but the unrestricted RBM flow towards their respective stationary distribution $p(\mathbf{v})$. This is a strong indication that these RBM have a flow that always flows towards the K the RBM was trained on. This was also checked by initialising the RBM flow from distributions at the K-values $K_0 = \{0.4, 0.9, 1.5\}$. The RBM flow always flowed towards a distribution showing similar behaviour as the $p(\mathbf{v})$ distribution.

These RBM behave as one would naively expect a trained RBM to behave. Once the RBM is trained, Gibbs sampling should take any configuration to the stationary distribution $p(\mathbf{v})$ which the RBM was trained for to make as close to the target distribution $q(\mathbf{v})$ as possible. That precisely the unrestricted RBM, which is the most general starting point for RBM simulations, does not show this behaviour is quite peculiar and warrants further inspection.

5.6.2 6-sited Ising chain and the RBM flow

Before continuing onto a further inspection of the RBM flow for the unrestricted RBM, we first consider the 6×3 RBM flow for a six sited Ising chain. Three different RBMs will be considered: a mean-field RBM with a single W-value, a translation-invariant RBM with 3 free Ws and a RBM with all Ws left free (the unrestricted RBM). The translation-invariant RBM is obtained from choosing a weave that gives the a restriction on the $\mathbf{W}$ matrix so that there are only three free Ws left. The resulting $\mathbf{W}$ matrix takes the form

$$\mathbf{W} = \begin{pmatrix} W_1 & W_3 & W_2 & W_2 & W_3 & W_1 \\ W_2 & W_1 & W_3 & W_3 & W_1 & W_2 \\ W_3 & W_2 & W_1 & W_1 & W_2 & W_3 \end{pmatrix}. \tag{5.64}$$

The RBM flow is computed starting from $\pi^{(0)} = q(\mathbf{v}, K_0)$, with $K_0 = \{0, K, \infty\}$ as in the previous subsection. The RBMs take longer to settle to their stationary distributions, so we take 100 RBM flow iterations ($k = 100$) to find an accurate estimate for the stationary distributions. The results are shown in figure 5.12.

As for the 4×2 RBM, the RBM flows of the mean-field and translation-invariant RBMs seem to flow to their corresponding stationary distribution $p(\mathbf{v})$. The unrestricted RBM deviates from this behaviour again, only if the flow is initialised from $K_0 = K$ will the unrestricted RBM settle into their stationary distribution $p(\mathbf{v})$. This deviation from the expected flow will be further investigated in the next section.

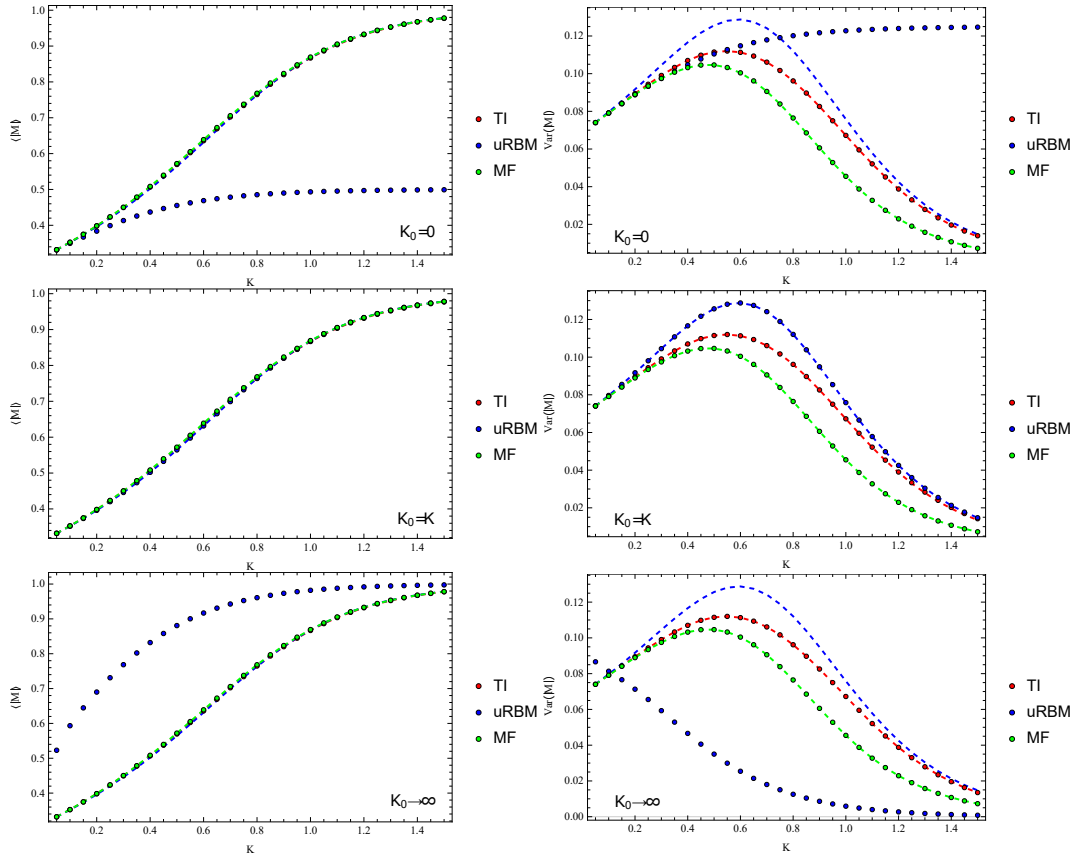


FIGURE 5.12: *The average and variance of the absolute magnetisation for the stationary distributions of the RBM flows initialised from* $K_0 = \{0, K, \infty\}$ *for the translation-invariant RBM (TI), the unrestricted RBM (uRBM), and the mean-field RBM (MF). The dashed lines are the measured quantities for* $p(\mathbf{v})$ *of the corresponding RBM, the dots are the results for the stationary distribution of the RBM flow (*$k = 100$*).*

5.6.3 The unrestricted RBM flow

The behaviour of the unrestricted RBM shown in figures 5.10, 5.11 and 5.12 is quite strange and different than one might expect. In this subsection we attempt to explain the unrestricted RBM behaviour by looking at the form of the **W**-matrix for the trained unrestricted 4×2 RBM. To investigate the flow of the unrestricted 4×2 RBM further, we calculated the RBM flow from a range of initial distributions $\pi^{(0)} = q(\mathbf{v}, K_0)$, where $K_0 = \{0, 0.4, 0.9, 1.5, \infty, K\}$. Here K is the temperature the RBM was trained on. The average and variance of the absolute magnetisation for the 20th RBM flow iteration distribution $\pi^{(20)}$ are shown in figure 5.13. The RBM flow shows very little deviation from the distribution at $k = 20$, which we therefore take to be good estimates of the equilibrium distribution of the RBM flow.

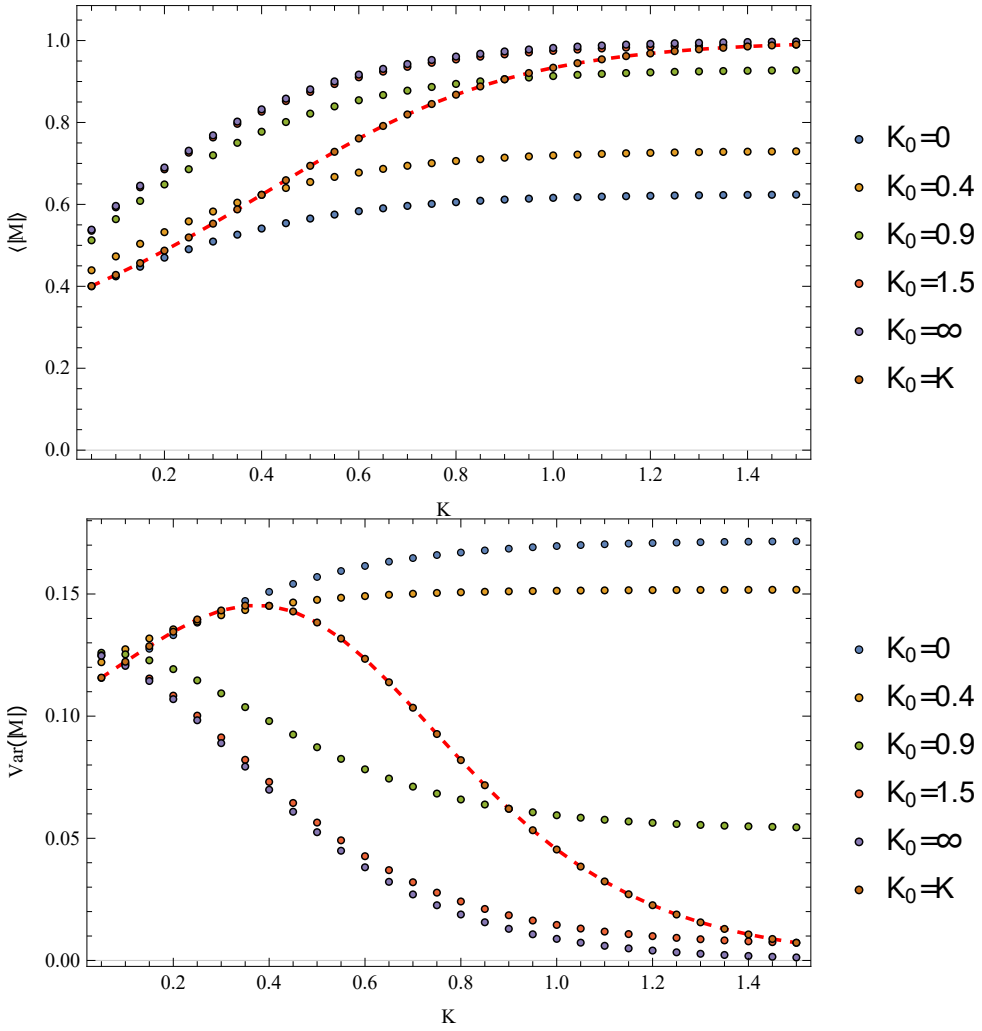
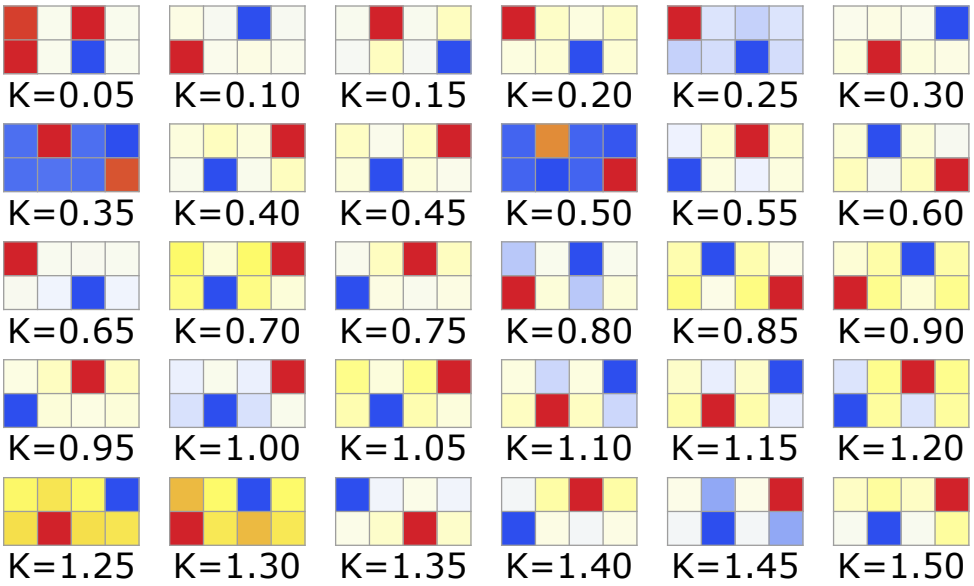
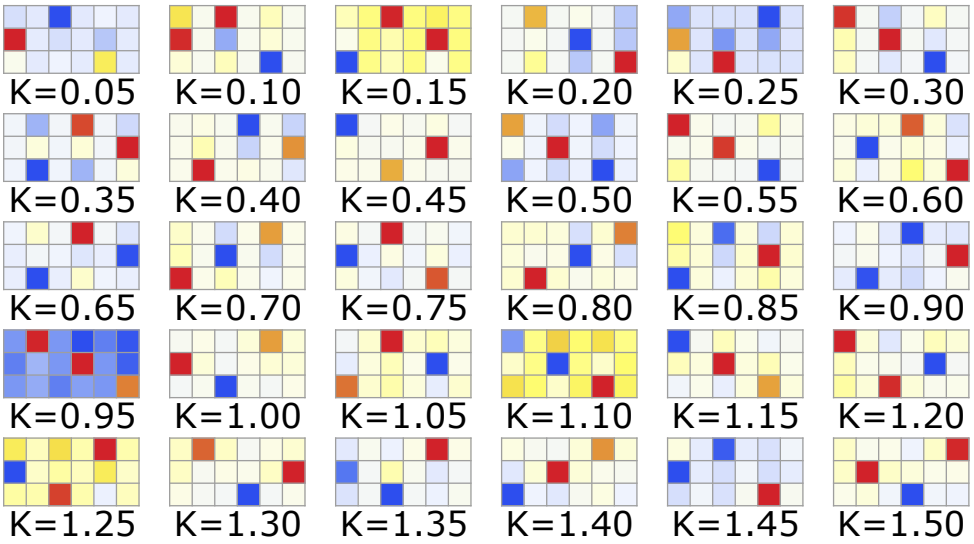


FIGURE 5.13: *Average and variance of the absolute magnetisation of $\pi^{(20)}$ for a trained unrestricted RBM starting from $\pi^{(0)} = q(\mathbf{v}, K_0)$ for different values of K_0. The dashed red line corresponds to $p(\mathbf{v})$. Different colours correspond to different initial values of K_0, as indicated in the legend.*

From the figure we see that the behaviour for $K_0 = 0$ and $K_0 \to \infty$ holds for starting from other values of $K_0 \neq K$. Rather than flowing to a distribution similar to $p(\mathbf{v})$, it flows to an entirely different distribution. This flow is not universal however, meaning that the flows starting from any $K_0 \neq K$ does not flow to the same stationary distribution. Rather, the stationary distributions the flow settles in seems related to the initial distribution. To try to understand this behaviour it may be useful to inspect the **W**-matrices for the trained RBMs at different Ks in more detail. Figure 5.14 shows a heatmap image of the **W**-matrix for the two unrestricted RBMs trained at different Ks.



(A) 4×2 *RBM.*



(B) 6×3 *RBM.*

FIGURE 5.14: *heatmap for the* **W***-matrix for the trained unrestricted* 4×3 *and* 6×3 *RBMs at different values of K.*

From the heatmaps a couple of interesting things can be noted. The first is that in a lot of the **W**-matrices there only tends to be a single large absolute (bright red or deep blue) value per hidden layer. The spins corresponding to these large values tend to be separated by a single other spin, as well the values being of opposite sign in the 4×2 RBM. This tells us the RBM prefers to model its hidden layer dependence via single spins per layer, having the multiple hidden nodes take care of the spin-spin interactions. This is different than our exact mapping, which explicitly models the difference between two neighbouring spins per hidden node. To see how this form plays a role in the RBM flow we first look at the conditional probability of a single hidden neuron:

$$p(h_a|\{\mathbf{v}\}) = \frac{e^{\sum_i v_i W_{ia} h_a}}{2\cosh\left(\sum_i W_{ia} v_i\right)}. \tag{5.65}$$

The average value the neuron will take is then

$$\langle h_a \rangle = \tanh\left(\sum_i W_{ia} v_i\right). \tag{5.66}$$

The average value the hidden neuron will take thus depends solely on the input configuration $\mathbf{v}^{(0)}$ and the weight-matrix row corresponding to that hidden neuron. We can then attempt to understand the RBM flow by analysing $\langle \mathbf{h}^{(0)} \rangle$ given some initial configuration $\mathbf{v}^{(0)}$. The average hidden neuron values can in turn be used to produce an estimate for the next iteration via

$$p(v_i^{(1)}|\mathbf{h}) = \frac{e^{\sum_i v_i^{(0)} W_{ia} h_a^{(0)}}}{2\cosh\left(\sum_i W_{ia} v_i^{(0)}\right)}, \tag{5.67}$$

$$\langle v_i^{(1)} \rangle = \tanh\left(\sum_a W_{ia} h_a^{(0)}\right) \tag{5.68}$$

Understanding the $K_0 \to \infty$ flow

The $K \to \infty$ flow can be understood by noting that only configurations with all spins pointing either up or down will be part of the initial configurations. The average values for the hidden neurons need then only be calculated for these two configurations. Since the weight-matrix has a single large absolute value for each hidden layer, the average value for h_a depends mostly on the value of the spin with the strongest matrix-value. Lets assume for each hidden layer a there is a single matrix value $|W_{j_a a}| >> W_{ia}$ with $i \neq j_a$. The average value for the hidden neuron will then be

$$\langle h_a^{(0)} \rangle \approx \text{sign}(W_{j_a a}) v_{j_a}^{(0)}, \tag{5.69}$$

so that the average value of h_a is always $\{+1, -1\}$ depending on the sign of the matrix-element and the value of v_{j_a}. A more even distribution of W-values could have $\mathbf{v}$-configurations where $\langle h_a \rangle = 0$ for example, where the actual value of the hidden neuron can be $+1$ or -1 with a 50% probability for either. When the unrestricted RBM is fed configurations of only spin up or down, the values of the hidden neurons depends on the relative sign between the largest absolute matrix values per matrix-row. For the 4×2 RBM for example, for most of the $\mathbf{W}$, the largest absolute values per matrix-row have opposite sign. That means that the expected values for the hidden neurons is going to be

$$\langle (h_1, h_2) \rangle_{\pm} \approx (\pm 1, \mp 1), \tag{5.70}$$

for example, where the $\pm$ sign corresponds to the two input configurations. The next visible layer can then be configured by considering equation (5.68). The expected values for the hidden nodes need then be multiplied by the $\mathbf{W}$ matrix-columns. Note that $j_a \neq j_b$ for $a \neq b$, so that the large absolute matrix-values are never in the same matrix-column. Here a difference between the low- and high-temperature trained RBM comes in. For the visible nodes with the large absolute matrix-values the expected value is still dominated by the large matrix-value, so that

$$\langle v_{j_a} \rangle \approx \text{sign}(W_{j_a a}) \langle h_a \rangle \approx v_{j_a}. \tag{5.71}$$

The expectation value of the other visible nodes differs from the low- to high-temperature trained RBMs. For the high-temperature trained RBMs the expectation value of

$\langle v_i^{(1)} \rangle$ is small, while it grows with increasing K. The expectation value of these visible nodes as a function of K is shown in figure 5.15 from an initial configuration of all spins up. The expectation is inverted for the initial configuration with all spins down.

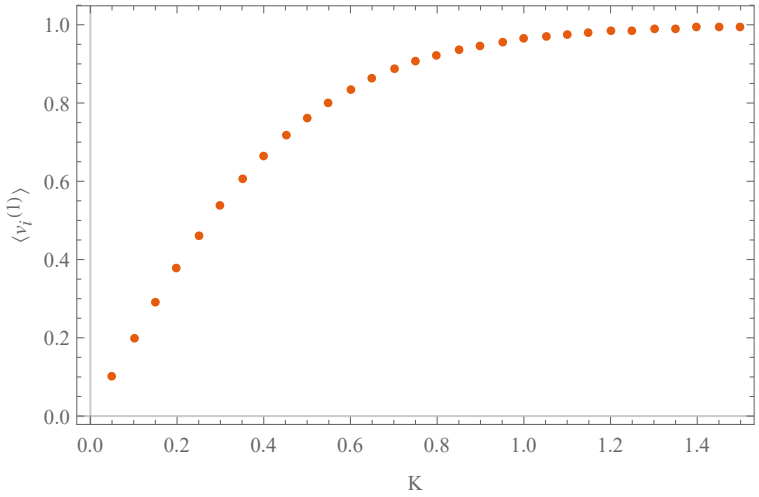


FIGURE 5.15: *Expectation value for the first iteration of the small visible nodes for the unrestricted* 4×2 *RBM, obtained from iterative application of equations* (5.66) *and* (5.68) *for* v_i *corresponding to* **W***-matrix columns with near equal small values. The initial configuration had all spins up.*

Since the average hidden node values depend solely on v_{j_a}, and v_{j_a} remains the same from iteration to iteration we can now understand the $K \to \infty$ behaviour. The iterative generated configurations have v_{j_a} the same as the input configurations, which means that these spins are pointing in the same direction for each new iteration. The input configurations consists solely of a configuration with all spins up and a configuration with all spins down, so the spins corresponding to the large absolute matrix-values in iterative configurations remain pointing in the same direction as for the initial configuration. The other spins are either going in random directions, as for the high-temperature trained RBMs, or stay in the same direction as the initial configurations as for the low-temperature trained RBMs. This allows us to understand the $\langle |M| \rangle$ behaviour.

In the high-temperature RBM a set of N (the number of hidden nodes) visible nodes are pointing either up or down together. The other $L - N$ nodes are pointing either up or down at random. So there are $2 + (L - N)^2$ different configurations with equal probability. The average absolute magnetisation then becomes

$$\langle |M| \rangle = \frac{1}{L} \sum_{\{\mathbf{v}\}} \frac{|\sum_i^L v_i|}{2 + (L - N)^2} = \frac{N}{L} \tag{5.72}$$

So for the high-temperature RBM we expect $\langle |M| \rangle = 1/2$, which is also what we saw in the RBM flow. For the low-temperature RBM $\langle v_i \rangle \approx v_i$ for all i, so that the iterative configurations are the same as the input configuration. The average absolute magnetisation then becomes $\langle |M| \rangle = 1$, which is also seen in the RBM flow. The variance $\mathrm{Var}(|M|)$ is then expected to be the same as for $\lim_{K_0 \to \infty} q(\mathbf{v}, K_0)$. The behaviour of the unrestricted RBM $K \to \infty$ flow can thus be understood as a direct consequence of the **W**-matrix form.

Understanding the $K_0 = 0$ flow

The unrestricted RBM flow behaviour for an initial distribution $\pi^{(0)} = q(\mathbf{v}, K_0)$ with $K_0 = 0$ can be understood by a similar analysis as for the $K_0 \to \infty$ case. The main difference between the two cases is that the set of initial configurations $\{\mathbf{v}^{(0)}\}$ now consists of all possible $\mathbf{v}$-configurations with equal probability for all. The observation that $\langle v_{j_a}^{(1)} \rangle \approx v_{j_a}^{(0)}$ still holds, the $\langle v_i^{(1)} \rangle$ for $i \neq j_a$ requires a bit more inspection. The expectation value still follows figure 5.15 if v_{j_a} for all a are pointing in the same direction. If one of the v_{j_a} has an opposite sign to the others, for some v_i two of the matrix-column terms will cancel each other since the matrix-terms are of similar size but opposite sign. One thing which is not visible in figure 5.14 is that the matrix-columns without a large absolute value tend to have two matrix-values of equal absolute value but opposite sign, and the other vanishing (for the 6×3 RBM). This holds for both the low- and high-temperature trained RBMs. This means that if one of the v_{j_a} has an opposite sign to the others, a few of the visible nodes will have equal probability to end up in either the spin up or down state.

The behaviour of the unrestricted RBM flow trained on the high-temperature distribution can now be understood. The visible nodes corresponding to large absolute matrix values v_{j_a} keep the same distribution as the initial configuration. The other visible nodes v_i, $i \neq j_a$, have a small preference for the direction they were already pointing in in the previous configuration if all v_{j_a} are pointing in the same direction. If a single or more v_{j_a} have a different value than the other v_{j_a}, a couple of visible spins v_i will have an expectation value of 0, meaning they can be pointing either way with equal probability. So for an initial configuration with all spin configurations at equal probability the iteration configurations will follow the same distribution as the initial distribution. This explains why the behaviour of the RBM flow iteration is close to the initial distribution behaviour for the high-temperature unrestricted RBMs.

For low-temperature unrestricted RBMs the initial configurations with v_{j_a} pointing in the same direction will go through the iterations relatively unchanged. The configurations with a single or more v_{j_a} different from the other v_{j_a} will have a number of v_i equal to the number of pairs of different v_{j_a} which will have expectation values of 0. These configurations will thus have equal probability to be transformed into a configuration where the v_i spin is flipped. As a consequence certain configurations will see their relative weight change compared to the initial distribution where every configuration was equally likely. Carefully computing the iterative distributions with the new weights and averaging over them will give the $\langle |M| \rangle$ values that were calculated for the stationary unrestricted RBM flow distribution.

Understanding the entire flow

Now that we know why the results are as they are for the $K_0 = 0$ and $K_0 \to \infty$ results, we can understand the general results on a qualitative level. The low- and high-temperature RBMs both a little different behaviour in terms of which spins they leave unchanged and which they do not. The low-temperature RBM leaves the initial distribution relatively unchanged, with only a little bias inserted due to the small but non-zero $\langle v_i \rangle$. The high-temperature RBM tends to randomise the initial distribution, with the strong spins v_{j_a} kept the same. When the initial configuration distribution for the unrestricted RBM trained for inverse temperature K is the same

$q(\mathbf{v}, K)$, the result is close to $p(\mathbf{v})$, which is a stationary distribution of the RBM flow. Clearly the RBM flow does not pull the configurations to this $p(\mathbf{v})$, but rather seems to land in a mixture of the initial configuration distribution and randomness added in by variation of the v_i spins, but keeping the v_{j_a} spins the same.

Overall this is quite odd behaviour, but it can be explained by the form of the **W**-matrix. Apparently the form of this matrix minimises the KL-divergence more than other possible configurations, as the values for the **W**-matrix was found by the swarm-method. This results in a very close match between $p(\mathbf{v})$ and $q(\mathbf{v})$, but in a RBM flow that is unable to pull other initial distributions to $p(\mathbf{v})$. This makes the found solution for the unrestricted RBM not applicable for generating new configurations from its distribution, which tends to be the goal of RBM learning. Nevertheless, this particular analysis was done for the results that where found by numeric minimisation of the KL-divergence, with $q(\mathbf{v})$ explicitly present. Usually this minimisation is done through contrastive divergence or adjacent methods, which use the Gibbs sampling procedure already within the learning process. Since it is clear that Gibbs sampling does not converge to the $p(\mathbf{v})$, it might be that learning through contrastive divergence will not find this particular solution to the unrestricted RBM. On the other hand, the learning is done through configurations obtained from $q(\mathbf{v})$, to which the flow does converge to $p(\mathbf{v})$. It would be interesting to see if a conventionally trained RBM will converge to the same general **W**-matrix form as was found by minimising the KL-divergence directly. If so, the restrictions of the RBM that were suggested in this thesis, the mean-field RBM and the translation invariant RBM, are able to provide much better convergence of the Gibbs chain to $p(\mathbf{v})$ while also being less computationally demanding.

5.7 Two-dimensional Ising model and a single-W valued RBM

While the Ising chain is an excellent model to study application to a RBM on, since finite-sized results can be calculated exactly, the interesting behaviour of the model is rather limited. There is no phase transition present in the model for finite temperatures, and the RG flow is not very exciting. The two-dimensional Ising model on a square lattice does have a phase transition, which is relatively well understood. The RG equations are known and can be derived using the block spin method. This makes the two-dimensional Ising model an excellent candidate for studying the learning behaviour of a RBM on.

To investigate what the RBM learns from the two-dimensional Ising model, we again start with a single W-value and no biases. Since we take the same assumptions as for the Ising chain, we can start from

$$\begin{aligned} Z &= 2^N \sum_{\{\mathbf{v}\}} \cosh^N(WV) \\ &= 2^N \sum_{V=-L}^{L} \cosh^N(WV) \\ &= 2^N \sum_{l=0}^{L} \binom{L}{l} \cosh^N((2l-L)W), \end{aligned} \tag{5.73}$$

where l is the number of active visible neurons (with $v = +1$). The possibility of writing the partition sum in this form is a direct result of limiting the degrees of freedom of $\mathbf{W}$ to 1. The RBM now only depends on $V = \sum_i v_i$, or equivalently the unnormalised magnetisation of the Ising model of size L. The Ising model distribution is

$$q(\mathbf{v}) = \frac{1}{Z_{\text{Ising}}} \exp\left(K \sum_{(i,j)}^{L} v_i v_j\right), \tag{5.74}$$

where Z_{Ising} is now the partition function corresponding to the finite-sized two-dimensional Ising model and (i,j) denotes that the sum is taken over nearest-neighbour pairs i, j. The equation that minimises the KL-divergence is then

$$\frac{\partial \ln(Z)}{\partial W} = L \sum_{\{\mathbf{v}\}} q(\mathbf{v}) V \tanh(VW), \tag{5.75}$$

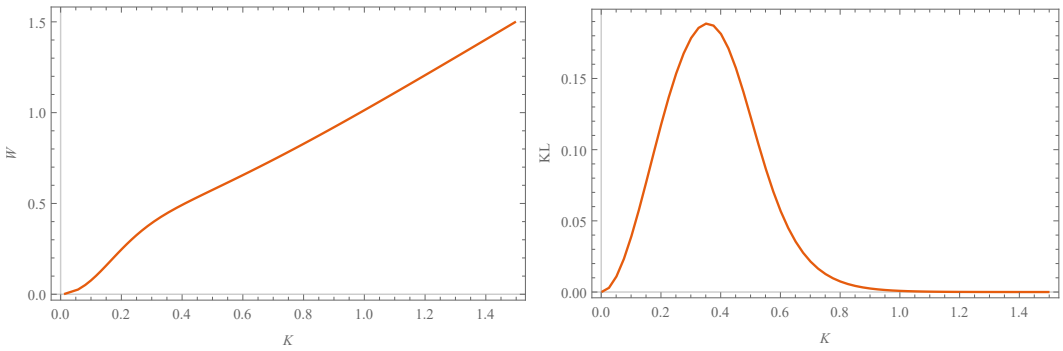
which we refer to in terms of the left-hand-side and right-hand-side equations:

$$f(W) = g(W, K). \tag{5.76}$$

Again an analytic expression is not possible, but a numerical evaluation is. The point where these two functions intersect minimises the KL-divergence. The W-value of the intersection between the two functions increases for higher K, as in the one-dimensional case. As was shown for the one-dimensional case, if W goes to infinity the system will flow to ordered states (see equation (5.28)). It seems likely that if $K \to \infty$, W will follow suit. Figure 5.16 shows a plot of $W(K)$ for which W minimises the KL-divergence for a given K.

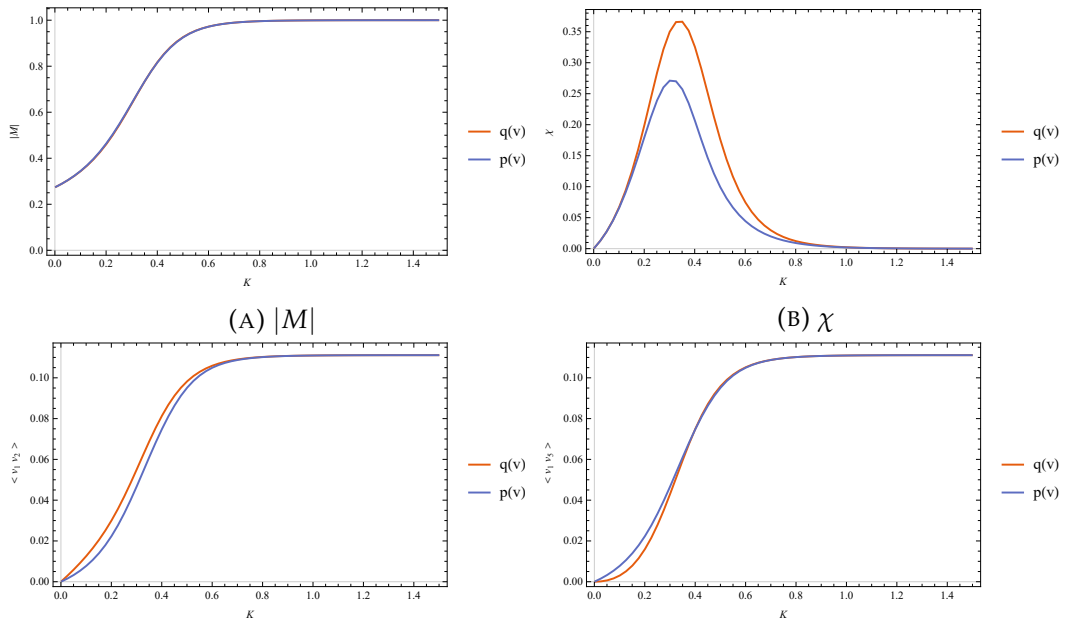
Inspecting figure 5.16 we see a similar form as for the Ising chain, which immediately tells us that the RBM learned something which does not diverge or disappear

at the phase transition. We can compare some observables the systems produced by calculating the expectation value (or average) of these observables. Figure 5.17 shows the average absolute magnetisation $\langle |M| \rangle$, the magnetic susceptibility χ and two two-spin correlations $\langle v_1 v_2 \rangle$ and $\langle v_1 v_5 \rangle$ for a 3×3 two-dimensional Ising model on a square lattice and the corresponding trained 9×4 RBM. The RBM appears to replicate the magnetisation very well, which is to be expected since $p(\mathbf{v})$ is a function of $V = L \cdot M$ only. The RBM performs less well on the magnetic susceptibility, which is directly linked to the variance of M. This is also a direct consequence of the single W-value restriction, since the two-spin correlations in the RBM are all equal. Any spin-pair is equally connected in the single W-value RBM. The plots of the two-spin correlations in figure 5.17 shows that the RBM is able to follow the basic features of the spin correlations, but fails on the details. The two correlations are equivalent for $p(\mathbf{v})$. It is exactly these details which are important for χ, which is why the single W-value RBM was unable to fully capture the details of χ.



(A) $W(K)$ (B) *KL-divergence*

FIGURE 5.16: *Figure (A) shows W as a function of K for a 9×4 RBM trained on a 3×3 Ising model on a square lattice, obtained via numerical evaluations of equation* (5.76)*. Figure (B) shows the KL-divergence corresponding to the solution given by $W(K)$.*



(A) $|M|$ (B) χ

(C) $\langle v_1 v_2 \rangle$ (D) $\langle v_1 v_5 \rangle$

FIGURE 5.17: *Absolute magnetisation $|M|$, magnetic susceptibility χ and two-spin correlations comparisons of a 9×4 RBM ($p(\mathbf{v})$) trained on a 3×3 Ising model on a square lattice ($q(\mathbf{v})$).*

5.7.1 Connection to mean-field theory

That the RBM fails to fully capture the details near the critical point is also shown in the KL-divergence, which has its maximum around $K \approx 0.4$ when the difference in χ is the largest. This is around the critical value of the Ising model: $K_c \approx 0.44$. The RBM is thus unable to capture the details of the model around the critical point. The single W-valued RBM is in many ways reminiscent of a mean-field theory. The spins in $p(\mathbf{v})$ all interact with the same value W:

$$p(\mathbf{v}) = \frac{1}{Z} \prod_a^N 2 \cosh \left(W \sum_i^L v_i \right). \tag{5.77}$$

This is very similar to how a mean-field theory of the two-dimensional Ising model is usually derived. In the low-temperature limit the individual spins are rewritten as

$$v_i = m + (v_i - m) \equiv m + \delta v_i, \tag{5.78}$$

where m is the average magnetisation of the spin if it was taken to be free and δv_i is taken to be small so that second-order contributions can be ignored. Essentially this equates to ignoring correlations between spins, which is obviously not valid as long as $|r_i - r_j| < \xi$, where r_i is the distance from the origin to the spin at site i and ξ is the correlation length. This means that their product can be written as

$$\begin{aligned} v_i v_j &= (m + \delta v_i)(m + \delta v_j) \\ &\approx m^2 + m\delta v_j + m\delta v_i \\ &= -m^2 + m(v_i + v_j). \end{aligned} \tag{5.79}$$

The Hamiltonian for the mean-field Ising model then becomes

$$-H_{\text{MFT}} = -JLm^2 + 2Jm\sum_i^L v_i, \tag{5.80}$$

so that the probability distribution looks like

$$q_{\text{MFT}}(\mathbf{v}) = \frac{1}{Z_{\text{MFT}}} \mathrm{e}^{-KLm^2} \exp\left(2Km\sum_i^L v_i\right). \tag{5.81}$$

The value for m is then determined by minimising the mean-field free energy f_{MFT} with respect to m. The free-energy is defined as

$$\begin{aligned} f_{\text{MFT}} &\equiv -\frac{1}{\beta L}\ln Z_{\text{MFT}} \\ &= -Jm^2 + \frac{1}{\beta}\ln\left(2\cosh 2Km\right). \end{aligned} \tag{5.82}$$

This equation is minimal when the self-consistent mean-field equation holds:

$$m = \tanh 2Km. \tag{5.83}$$

In a sense equation (5.77) is similar to the mean-field Ising model (5.81) in that they both consist of uncorrelated spins interacting via a constant interaction strength, K in the Ising model and W in the RBM. They are not equivalent however, as the form of the probability distribution, a product of cosh-terms versus a single exponential term, are sufficiently different. No simple mapping equating the two probability distributions appears to exist.

Chapter 6

Discussion and conclusion

In this chapter a short recap of the results obtained in chapters 4 and 5 will be given. The results of the chapters will be discussed separately, followed by a conclusion of the results, suggestions for further research and some final remarks.

6.1 A brief recap of the neural network analysis results

In chapter 4 we first discussed the application of a feed-forward neural network to find the finite-size critical temperature from configurations of a two-dimensional Ising model on a square lattice. The neural network was able to find the critical temperatures with relative ease, and finite-size scaling was used to infer the critical temperature in the thermodynamic limit $T_c = 2.25 \pm 0.03$. The learned behaviour of the network could be explained using a simple toy model, showing that the neural network has learned the magnetisation. Next a convolution layer was added to the neural network, which did not improve on the accuracy of the found critical temperature. An analysis of the input values for the hidden layer showed that the neural network is now able to learn both the magnetisation and energy of the physical system.

A similar analysis was applied to the xy-model. A simple feed-forward neural network was able to differentiate between the different phases of the system. However, when comparing the neural network results for multiple system sizes and performing finite-size scaling analysis, the results did not fit the expected form. A convolution layer was added to circumvent this problem, but to no avail. The correct finite-size scaling was finally achieved by explicitly setting the values for four filters so as to transform the angle-configuration to a vortex-configuration. With this transformation a simple feed-forward neural network was able to successfully learn the KT-transition with the expected finite-size scaling.

Next a different application of the neural network was then shown for the antiferromagnetic Ising model on a triangular lattice (TIAF). Rather than trying to learn a phase transition, which the system does not undergo, the neural network has learned to infer the temperature of a given configuration. The performance for a convolutional neural network was shown to be very accurate, decreasing in accuracy slightly for larger temperatures. An investigation of the filters and inputs for the hidden layer was unable to conclusively show what the system had learned to differentiate between the configurations. A simple toy model of a neural network was introduced to show that a neural network is able to count the number of frustrated triangles.

Finally we considered the two-dimensional Coulomb gas. This model is said to

have both an Ising-like and KT-like phase transition in relative close proximity to one-another. Different neural networks were deployed with the task of differentiating between the three phases of the system and finding the right finite-size scaling relations. First a convolution neural network was applied on just the low- and high-temperature phase. It found the Ising-like transition with the correct finite-size scaling. A convolutional neural network trained on configurations of the in between phase was unable to successfully differentiate between the different phases with the correct finite-size scaling. A confusion scheme was applied to circumvent the problem of false labelling. The confusion scheme was able to find the Ising-like transition for the dataset over a large temperature range. Including a new dataset over a smaller temperature range near the critical temperatures did not enable the confusion scheme neural network to find the three phases.

6.2 Discussion and conclusion of the neural network analysis

The goal of the neural network analysis was to study the application of neural networks as a novel numerical tool to find phase transitions in classical systems. While the success of the initial application of the neural network on the Ising model was a great start, the rest of the analysis was not as smooth. When the neural networks became more complex, for example by including a convolution layer, then so became the difficulty of training the network. While for the Ising model and xy-model the neural networks, when applied to configurations at a single system size, showed good results in terms of accuracy, this success did not carry on when regarding the neural network results in a bigger picture. Getting the correct finite-size scaling from the neural networks turned out to be a great difficulty, resulting in a great deal of energy needing to be spent tweaking the learning parameters and network topology. The more complex the neural network, the harder to get consistent results over all system sizes. This greatly hinders neural network application to physical systems, as most numerical applications of physics aim to get results valid in the thermodynamic limit to be compared to theory. The network tends to settle in a minimum of the cost function which simply does not give the desired physical result. This was most evident in the analysis of the xy-model. Only with sufficient feature-engineering were we able to obtain useful results from the neural network.

Another difficulty with neural networks is the interpretability of the learned behaviour. In a sense this ties in with the difficulty of getting the correct finite-size scaling. Precisely because these networks are so complex, it is hard to infer what it has learned. As such, it is also difficult to design the system in such a way that it picks up the correct behaviour. This defeats the purpose of neural networks as a tool to apply when theory is lacking in how to discern between different phases in a physical model. Because the learned behaviour of the neural network is so hard to discern, it seems unlikely that any theoretical insight can be obtained from it.

The accumulation of these observations is perhaps most clear in the analysis of the two-dimensional Coulomb Gas model. The neural network analysis was unable to give any meaningful contribution to the discussion regarding the two phases in this model. It seems that the most obvious conclusion to draw from these results is that the neural network is a rather limited tool in numerical phase transition analysis in classical systems. Apart from a few models with clearly defined phase transitions, it

could be useful as a more specialised tool with the proper feature-engineering. An example would be to give an estimate for the temperature of a given configuration, as was done for the TIAF. The hope that this tool could be used as powerful all-round phase classifier seems unlikely.

With that being said, there are plenty of directions to go into with neural networks in physics. In context of this thesis specifically, a more in depth neural network analysis of the two-dimensional Coulomb gas could still shed some light on the different phases of the model. More configurations in the in between phase for larger system sizes might help, as could trying different, more advanced neural networks. There are still many classical models for which neural networks could most likely be used to differentiate between the different phases. There are also other applications in quantum mechanical models represent the ground state with neural networks[22, 71]. They can also be used to help with error-correction in quantum computers[36, 72], or to model the nuclear distribution function to help with particle physics calculations[37].

6.3 A brief recap of the RBM and spin models

Chapter 5 was concerned with training and analysing the behaviour of different iterations of RBM on the Ising model. First a discussion of the general structure of the RBM was made, showing that restricting the weight-matrix $\mathbf{W}$ to a single W-value is an extremum of the KL-divergence for a translation invariant target distribution $q(\mathbf{v})$. This allowed us to devise the mean-field RBM in the next section. The mean-field RBM was then trained on a one-dimensional Ising model, comparing the learned behaviour of the trained RBM to the Ising chain. The mean-field RBM learned the magnetisation very well, but lacked in the spin correlations. The RBM flow behaviour for $K \to \infty$ was investigated, showing that $W \to \infty$, so that the RBM flow follows the RG flow.

Next an exact mapping from a RBM with equal number of visible and hidden nodes to the Ising chain was derived using spin barriers and a single free W-parameter. Another RBM iteration was derived by discussing a way to force translation invariance upon the RBM. The weave-method was introduced as a method to obtain such a RBM. The stability and performance of a 4×2 translation invariant RBM was discussed in detail as well.

All the different RBM iterations for the Ising chain were brought together to analyse the Gibss sampling, or RBM, flow of each different iteration. The trained RBM were initialized from different initial distributions, and all but the unrestricted RBM were shown to converge to the stationary distribution $p(\mathbf{v})$ starting from any initial distribution. The unrestricted RBM only converged to $p(\mathbf{v})$ if it was initialised from the Ising chain distribution it was trained on. The peculiar behaviour of the unrestricted RBM was discussed in detail and an explanation based on the $\mathbf{W}$-elements was given.

Finally the application of the mean-field RBM to the two-dimensional Ising model on a square lattice was discussed, including a derivation of the weight-matrix component W as a function for the Ising coupling constant K. The KL-divergence was

found to be largest near the critical region. A comparison of the mean-field RBM to mean-field theory of the Ising model was given as well.

6.4 Discussion and conclusion of RBM and spin models

Chapter 5 is very much the initial, exploratory part of possible further research. As such, from the things discussed in chapter 5, only some preliminary conclusions can be drawn. The work done on the one-dimensional Ising model primarily goes to show the validity of certain restrictions on the RBM, and the interpretation of that. The analytic arguments combined with numerical calculations has shown a restriction of the $\mathbf{W}$-matrix forces the system to pick up on a limited number of aspects of the Ising chain, mainly focusing on the magnetisation before encoding higher order moments. At the same time restricting $\mathbf{W}$ in a certain way enforces translation invariance in the RBM. In fact, the RBM flow behaviour of the RBM is more well behaved for the restricted RBMs, meaning that the RBM flow always goes towards the stationary distribution $p(\mathbf{v})$. This is an indication that restricting the RBM based on physical symmetries can help in getting better behaving RBMs as well as being computationally cheaper to train. The weave-method was shown to be a great general method to apply to restrict the RBM such that $p(\mathbf{v})$ has translation invariance, while still performing better than the mean-field RBM and having a better behaved RBM flow than the unrestricted RBM.

The next step would be to go to the two-dimensional Ising model. This model clearly shows more interesting behaviour than the Ising chain, and as such the RBM behaviour is expected to be more interesting as well. Apart from the mean-field RBM, the translation invariant RBM can also be applied to the two-dimensional case. To this end, an extrapolation of the weave-method to two dimensions needs to be formulated. The RBM flow behaviour of the mean-field RBM, translation invariant RBM, and unrestricted RBM can then be compared, providing a stronger argument for the advantages of restricting the RBM. It would also be interesting to see what the RBM learns when exposed to a range of temperatures during training, instead of a single temperature, and seeing if the results are similar to the results of Iso et al.[31]

Appendix A

Monte Carlo simulations

A.1 Introduction to Monte Carlo simulations

Monte Carlo simulations are an often used numerical tool to simulate finite physical systems and obtain estimates for physical observables. The Monte Carlo method approximates the average of an observable O by measuring the observable for each configuration, averaging over all encountered configurations:

$$\langle O \rangle = \frac{1}{N} \sum_{i=1}^{N} O_i, \tag{A.1}$$

where N is the number of configurations, and O_i is the measured observable for configuration i. To obtain physical results, the configurations must be sampled from a physical distribution, defined by the physical system the Monte Carlo simulation is supposed to be modelling. This so-called importance sampling of the configurations then follows the Boltzmann distribution

$$p(\{\mathbf{s}\}) = \frac{\mathrm{e}^{-\beta H(\{\mathbf{s}\})}}{Z}, \tag{A.2}$$

where $\{\mathbf{s}\}$ is a configuration of spins, H is the Hamiltonian and Z is the partition sum. The configurations for the importance sampling are usually obtained from a Markov chain, where from some initial configuration μ a new one ν is obtained according to some transition probability $P(\mu \to \nu)$, which does not vary over time and only depends on the properties of the states μ and ν. Naturally the transition probability needs to be normalised. This process is then repeated so that a chain of configurations is created, where the transition probability is chosen such that after a while the distribution of created states follows the Boltzmann distribution. For this to hold we need the conditions of ergodicity and detailed balance.

Ergodicity simply means that it must be possible to reach any state in the statespace of the Boltzmann distribution from any other state in a finite number of steps. Any state with a finite probability in the Boltzmann distribution must be able to be reached by the Markov chain, otherwise the Markov chain can never truly model the Boltzmann distribution. This is something that needs to be checked for every new algorithm which defines a new set of transition probabilities.

The detailed balance condition is a way to ensure that the Boltzmann distribution is a stationary distribution of the Markov chain, and that there are no limit cycle solutions to the chain. For a detailed derivation we point the interested reader to Newmann et al.[73]. For our purposes it is sufficient to simply state that the detailed

balance equation

$$p(\mu)P(\mu \to \nu) = p(\nu)P(\nu \to \mu) \tag{A.3}$$

must hold. To make the equilibrium distribution equal to the Boltzmann distribution of equation (A.2) we can see from the detailed balance equation that

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{p(\nu)}{p(\mu)} = \mathrm{e}^{-\beta(H(\nu)-H(\mu))}. \tag{A.4}$$

So the fraction of transition probabilities from μ to ν and back has to be equal to the exponential of the energy difference between the two states. Different algorithms have different transition probabilities that are computationally faster or have higher acceptance rates, depending on the physical system being model. A common choice is the Metropolis algorithm, which is based of the splitting of the transition probability into two parts:

$$P(\mu \to \nu) = g(\mu \to \nu)A(\mu \to \nu). \tag{A.5}$$

The probability $g(\mu \to \nu)$ is the selection probability, giving the probability of generating the state ν given a state μ. $A(\mu \to \nu)$ is the acceptance ratio which gives the probability of accepting the newly generated state ν. The Metropolis algorithm is then based on choosing $g(\mu \to \nu)$ to be symmetric, so that

$$\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \mathrm{e}^{-\beta(H(\nu)-H(\mu))}. \tag{A.6}$$

There are many choices for A that satisfy this condition, but the Metropolis algorithm chooses

$$A(\mu \to \nu) = \begin{cases} \exp\left(-\beta[H(\nu) - H(\mu)]\right) & \text{if } H(\nu) < H(\mu) \\ 1 & \text{if } H(\nu) \geq h(\mu) \end{cases}. \tag{A.7}$$

This essentially means that a trial move, generated from $g(\mu \to \nu)$, will be accepted always if the energy for the new configuration ν is lower than the old configuration μ. If this is not the case, the new configuration will be accepted with the probability equal to the exponential of the energy difference between the two states, weighted with the inverse temperature.

Monte Carlo simulations then consists of generating many configurations according to this Markov chain method, making care to sample from the stationary distribution of the chain by letting the chain reach equilibrium. Depending on the algorithm used it might be necessary to take multiple steps between sequential observable measurements to prevent unphysical correlations between the configurations.

A.2 Model specific details

In this section the specific algorithms used to generate configurations for the different physical models will be discussed.

A.2.1 Ising and xy-model

For the Ising and xy-model a mixture of the spin-flip Metropolis algorithm and the Wolff algorithm[67] was used. The spin-flip algorithm consists of picking a random spin σ_i in the current spin configuration μ, and then flipping that spin $\sigma_i \to -\sigma_i$ (or

rotating it with some randomly chosen angle for the xy-model) to generate a new possible configuration ν. This new configuration is then accepted with the probability given by equation (A.7). This algorithm suffers in the low-temperature region, where many possible moves will be rejected and not all of statespace will be explored within a reasonable time. To counteract this the algorithm is augmented with the Wolff algorithm.

The Wolff algorithm consists of growing a cluster of spins, and flipping (or rotating) the cluster all in one go. It is based on the reflection operation with respect so the hyperplane orthogonal to $\mathbf{r} \in S_{n-1}$ for $O(n)$ spin-models

$$R(r)\sigma_i = \sigma_i - 2(\sigma_i \cdot \mathbf{r})\mathbf{r}, \tag{A.8}$$

where for the Ising model ($n = 1$) this reflection simply consist of flipping the spin. Note that the Hamiltonian is invariant under application of this operation to all spins. The Wolff algorithm then consists of first choosing a random reflection $\mathbf{r} \in S_{n-1}$ and a random initial lattice site a marking the start of the cluster. The spin on this site is then 'flipped': $\sigma_a \to R(r)\sigma_a$ and marked. All nearest neighbours of a are then considered to be added to the cluster, a nearest neighbour b is added with probability

$$P(\sigma_a, \sigma_b) = 1 - \exp\left(\min(0, J\beta\sigma_a \cdot [1 - R(\mathbf{r})]\sigma_b)\right) \tag{A.9}$$

$$= 1 - \exp\left(\min[0, 2J\beta(\mathbf{r} \cdot \sigma_a)(\mathbf{r} \cdot \sigma_b)]\right). \tag{A.10}$$

If b is added, it is flipped and marked. This process continues for each newly added spin and its nearest neighbours. It can be checked that equation (A.4) holds for newly generated configurations using this procedure, as does the ergodicity condition.

This cluster algorithm greatly increases the speed of obtaining new uncorrelated configurations in the low-temperature limit and near the critical point.

A.2.2 Anti-ferromagnetic Ising model on a triangular lattice

The anti-ferromagnetic Ising model on a triangular lattice has a highly degenerate groundstate, which means that if you want to generate configurations that are representative of the model at low temperatures, most of these groundstates, or small excitations thereof, will need to be reached. With just the Metropolis algorithm this is nearly impossible. When starting from some initial configuration the simulation will tend to settle in one of these groundstates, and apart from the occasional small excitation of a single spin flip, it will stay there. The consequent Metropolis steps to take to reach the other groundstates are simply so unlikely to happen that the system is stuck in that particular groundstate.

This problem can be circumenvented by applying a cluster algorithm by Zhang et al.[74]. The algorithm first divides the lattice in a chequerboard pattern of triangles, randomly choosing one of the two possible sets of triangles (the black or white tiles). Each individual triangle in the set is then considered. The energy for each triangle is calculated, and based on the two options the bonds in the triangle are either frozen or deleted.

If the energy of the triangle is $E = -3J$, so that all spins are either pointing up or down, the three unsatisfied bonds are deleted with probability $p = 1$. If the energy of the triangle is $E = J$, then with probability $p = \exp(4\beta J)$ all three bonds of the triangle are deleted. Otherwise, one of the two satisfied bonds is frozen and the other two deleted. The bond to be frozen is chosen at random. This is done for every triangle in the set, so that a cluster is created of spins connected by frozen bonds. Two spins belong in the same cluster if there is a line of frozen bonds connecting them. Then each cluster is flipped with a probability $1/2$. This algorithm, combined with Metropolis steps in between, is able to reach all of statespace of the model in the low temperatures.

A.2.3 Coulomb gas

The Coulomb gas model is different than the previous spin models in that instead of having just nearest-neighbour interactions, every charge is related to every other charge. This greatly slows down the simulation when applying the Metropolis algorithm, since for every step one would need to sum over all charges to calculate the new energy of the suggested state. To circumvent this problem a algorithm by Grest[64] is used. This algorithm is based on the simple Metropolis step, but the energy difference is rewritten in a clever way.

Since charge must be conserved, a pair of nearest-neighbour sites (i_0, i_1) is chosen at random. If the pair has opposite charge, the charge is flipped. Otherwise the move is rejected. The flipped charge is accepted with the Metropolis probability $\exp(-\beta\Delta E)$, where ΔE is the energy difference between the two states. Evaluating this ΔE is time-consuming, but Grest noticed that it can be written as

$$\Delta E = \sum_{i=i_0,i_1,j} [\Delta q_i V'(\mathbf{r}_i - \mathbf{r}_j) q_j] + \Delta q_{i_0} V'(\mathbf{r}_{i_0} - \mathbf{r}_{i_1}) \Delta q_{i_1}. \tag{A.11}$$

In this form each evaluation of ΔE is a computation of order $L \times L$, for sweeping the entire lattice. This can be sped up by rewriting it in terms of the total potentials per site. The total potential at site i is defined as

$$F_i \equiv \sum_j V'(\mathbf{r}_i - \mathbf{r}_j) q_j. \tag{A.12}$$

Now the energy difference can be rewritten as

$$\Delta E = \Delta q_{i_0} F_{i_0} + \Delta q_{i_1} F_{i_1} + \Delta q_{i_0} V'(\mathbf{r}_{i_0} - \mathbf{r}_{i_1}) \Delta q_{i_1}. \tag{A.13}$$

This is now a computation of order 1. However, now it is necessary to update the total potentials for each site every time the charge configuration changes:

$$F_{i,\text{new}} = F_{i,\text{old}} + V'(\mathbf{r}_i - \mathbf{r}_j) \Delta q_{i_0} + V'(\mathbf{r}_i - \mathbf{r}_{i_1}) \Delta q_{i_1}, \tag{A.14}$$

which is again a computation of order $L \times L$. However, only when an excitation is accepted do the total potentials need to be updated. This algorithm thus provides a substantial increase in speed compared to the direct computation of ΔE.

Appendix B

One-dimensional *xy*-model and RBM

Following the analysis as for the Ising chain, we set all the weights W_{ia} equal to a single value W. The main difference is that we now consider a RBM with real valued visible nodes $v_i = [0, 1]$ which are modelled by the probability of the visible node being equal to 1 in the binary RBM. So

$$v_i^{\text{real}} = p(v_i^{\text{binary}} = 1|\mathbf{h}). \tag{B.1}$$

This leaves the overall energy-function intact apart from having real valued visible nodes and requires only minor changes in the sampling procedure. The learning process remains the same[45]. We start with the probability distribution of the visible layer:

$$p(\{v_i\}) = \frac{1}{Z} \sum_{\{h_a\}} \exp\left[W \sum_{i,a} v_i h_a\right] \tag{B.2}$$

where we have again put the bias to zero. The partition function then takes the regular form

$$Z = \sum_{\{\mathbf{v},\mathbf{h}\}} \exp\left[W \sum_{i,a} v_i h_a\right]. \tag{B.3}$$

Since the RBM's aim is to minimise the KL-divergence $KL(q||p)$ with respect to W, the rest of the derivation will be concerned with minimising the KL-divergence.

The KL-divergence is given by

$$\text{KL} = \sum_{\{\mathbf{v}\}} q(\mathbf{v}) \ln \frac{q(\mathbf{v})}{p(\mathbf{v})}. \tag{B.4}$$

We can then rewrite the derivative as

$$\begin{aligned} \frac{\partial \text{KL}}{\partial W} &= -\sum_{\{\mathbf{v}\}} \frac{\partial}{\partial W} [q(\mathbf{v}) \ln p(\mathbf{v})] \\ &= -\sum_{\{\mathbf{v}\}} \frac{q(\mathbf{v})}{p(\mathbf{v})} \frac{\partial p(\mathbf{v})}{\partial W}, \end{aligned} \tag{B.5}$$

because $q(\mathbf{v})$ is independent of W. We can rewrite the partial derivative of p in more explicit terms

$$\begin{aligned}\frac{\partial p(\mathbf{v})}{\partial W} &= \frac{\partial}{\partial W}[\frac{1}{Z}\sum_{\{\mathbf{h}\}}\exp\left(W\sum_{i,a}v_i h_a\right)] \\ &= -\frac{\sum_{\{\mathbf{h}\}}\exp\left(W\sum_{i,a}v_i h_a\right)}{Z^2}\frac{\mathrm{d}Z}{\mathrm{d}W} + \frac{1}{Z}\sum_{\{\mathbf{h}\}}(\sum_{i,a}v_i h_a)\exp\left(W\sum_{i,a}v_i h_a\right). \end{aligned} \tag{B.6}$$

If we plug the second term on the right hand side of equation (B.6) back into equation (B.5) we get the term

$$-\sum_{\{\mathbf{v}\}}\frac{q(\mathbf{v})}{\sum_{\{\mathbf{h}\}}\exp\left(W\sum_{i,a}v_i h_a\right)}\cdot(\sum_i v_i)\sum_{\{\mathbf{h}\}}(\sum_a h_a)\exp\left(W\sum_{i,a}v_i h_a\right). \tag{B.7}$$

We can simplify this expression by explicitly performing some of the sums present in the term. Note that

$$\sum_{\{\mathbf{h}\}}\exp\left(W\sum_{i,a}v_i h_a\right) = \sum_{\{\mathbf{h}\}}\exp\left(W(\sum_i v_i)(h_1+h_2+...+h_N)\right) \tag{B.8}$$

can be factorised as

$$\begin{aligned} &= \left(\sum_{h_1=\{0,1\}}\exp\left[W(\sum_i v_i)h_1\right]\right)...\left(\sum_{h_N=\{0,1\}}\exp\left[W(\sum_i v_i)h_N\right]\right) \\ &= \prod_{a=1}^{N}\left(\sum_{h_a=\{0,1\}}\exp\left[WVh_a\right]\right) \\ &= \prod_{a=1}^{N}\left(1+\exp\left[WV\right]\right) \\ &= \left(1+\exp\left[WV\right]\right)^N. \end{aligned}$$

Here we take the sum over all possible values of $\{\mathbf{h}\}$ since h is still a discrete variable with $h_a = \{0,1\}$ and $V = \sum_i v_i$. So we have that

$$\sum_{\{\mathbf{h}\}}\exp\left(W\sum_{i,a}v_i h_a\right) = \left(1+\exp\left[WV\right]\right)^N. \tag{B.9}$$

We can use this expression to derive

$$\begin{aligned}\sum_{\{\mathbf{h}\}}(\sum_a h_a)\exp\left(W\sum_{i,a}v_i h_a\right) &= \sum_{\{\mathbf{h}\}}(\sum_a h_a)\exp\left(WV\sum_a h_a\right) \\ &= \sum_{\{\mathbf{h}\}}\frac{\partial}{\partial WV}\exp\left(WV\sum_a h_a\right) \\ &= \frac{\partial}{\partial WV}\left(1+\exp\left[WV\right]\right)^N. \end{aligned} \tag{B.10}$$

Performing the derivative gives us

$$\frac{\partial}{\partial WV}\left(1+\exp\left[WV\right]\right)^N = N\left(1+\exp\left[WV\right]\right)^{N-1}\exp\left[WV\right],$$

which we can plug back in equation (B.7) together with equation (B.9) to get

$$\begin{aligned}&\sum_{\{\mathbf{v}\}} -q(\mathbf{v})\ \left(1+\exp\left[WV\right]\right)^{-N} VN\left(1+\exp\left[WV\right]\right)^{N-1}\exp\left[WV\right]\\&= -N\sum_{\{\mathbf{v}\}}\frac{q(\mathbf{v})V}{1+\exp\left[-WV\right]}.\end{aligned} \tag{B.11}$$

We still need to evaluate the first term on the right hand side of equation (B.6):

$$-\frac{\sum_{\{\mathbf{h}\}}\exp\left(W\sum_{i,a}v_i h_a\right)}{Z^2}\frac{\mathrm{d}Z}{\mathrm{d}W} = -p(\mathbf{v})\frac{\mathrm{d}\ln\left(Z\right)}{\mathrm{d}W} \tag{B.12}$$

and plug it in equation (B.5). We find

$$-\sum_{\{\mathbf{v}\}}\frac{q(\mathbf{v})}{p(\mathbf{v})}\cdot\left(-p(\mathbf{v})\frac{\mathrm{d}\ln\left(Z\right)}{\mathrm{d}W}\right) = \frac{\mathrm{d}\ln\left(Z\right)}{\mathrm{d}W}, \tag{B.13}$$

where we have used that $q(\mathbf{v})$ is normalised to unity. We then find for the derivative of the KL-divergence:

$$\frac{\partial \mathrm{KL}}{\partial W} = \frac{\mathrm{d}\ln\left(Z\right)}{\mathrm{d}W} - N\sum_{\{\mathbf{v}\}}\frac{q(\mathbf{v})V}{1+\exp\left[-WV\right]}. \tag{B.14}$$

The RBM wants to minimise the KL-divergence, so to find an expression for W in terms of the coupling parameter J we need to solve

$$\frac{\mathrm{d}\ln Z}{\mathrm{d}W} = N\sum_{\{\mathbf{v}\}}\frac{q(\mathbf{v})V}{1+\exp\left[-WV\right]}. \tag{B.15}$$

All the information on the coupling constant is contained in $q(\mathbf{v})$. If we only look at the distribution at a fixed temperature T^* the distribution will correspond to the Boltzmann distribution

$$q(\mathbf{v}) = \frac{\exp\left(-\beta\mathcal{H}(\mathbf{v})\right)}{Z_{\mathrm{xy}}}, \tag{B.16}$$

where $\mathcal{H}$ corresponds to the Hamiltonian of the 1 dimensional XY-model with periodic boundary conditions and L sites. Z_{xy} is its associated partition function: [75]

$$\mathcal{H}(\mathbf{v}) = -J\sum_{i=1}^{L}\cos\left(2\pi(v_i - v_{i+1})\right) \tag{B.17}$$

$$Z_{\mathrm{xy}} = (2\pi)^L\sum_{p=-\infty}^{+\infty}\left(I_p(K)\right)^L. \tag{B.18}$$

L corresponds to the number of visible units, $K = \beta J$ and $I_p(K)$ is the modified Bessel function of the first kind of order p. All together we now have

$$\frac{\mathrm{d}\ln Z}{\mathrm{d}W} = N\sum_{\{\mathbf{v}\}}\frac{\exp\left(K\sum_{i=1}^{L}\cos\left(2\pi(v_i - v_{i+1})\right)\right)}{(2\pi)^L\sum_{p=-\infty}^{+\infty}\left(I_p(K)\right)^L}\cdot\frac{V}{1+\exp\left[-WV\right]}. \tag{B.19}$$

Now we only need to evaluate Z, we can write

$$\begin{aligned} Z &= \sum_{\{\mathbf{h}\}} \prod_{i=1}^{L} \left(\int_0^1 \mathrm{d}v_i \, \exp\left[WHv_i\right] \right) \\ &= \sum_{\{\mathbf{h}\}} \left(\frac{1}{WH} (\exp\left[WH\right] - 1) \right)^L . \end{aligned} \tag{B.20}$$

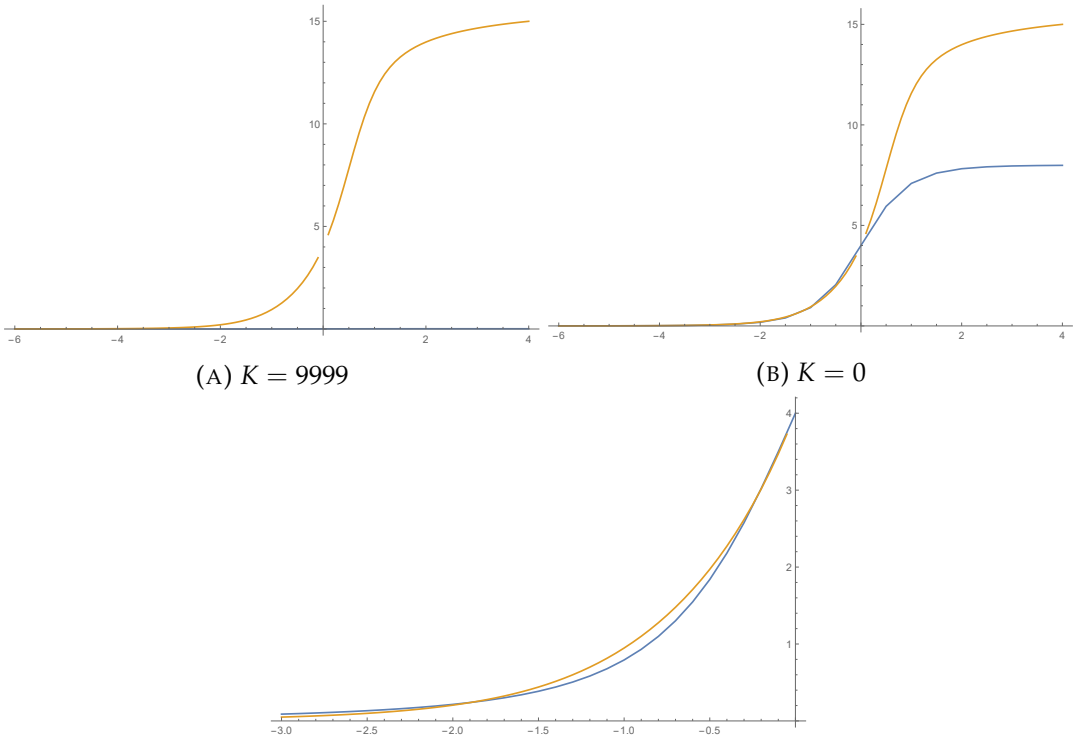
We would again like to explicitly calculate this quantity which depends only on $H = \sum_{a=1}^{N} h_a$. We can take nearly the same approach as for the Ising case, only now $h_a = \{0, 1\}$. The partition function Z then becomes

$$Z = d(0) + \sum_{n=1}^{N} d(n) \left(\frac{1}{Wn} (\exp\left[Wn\right] - 1) \right)^L . \tag{B.21}$$

Here $d(H)$ describes the degeneracy and is given by the binomial coefficient:

$$d(H) = \binom{N}{H}. \tag{B.22}$$

Again we need to resort to *Mathematica* to find solutions to equation (B.19). Figure B.1 shows the left- and right hand side of equation (B.19) for different values of the coupling strength K for the simplest case, a two-sited xy-model and a 2×1 RBM. *Mathematica* is unable to find solutions to the equations, even numerically. From the figures one can see that there never is a solution with $W(K) > 0$, and that for large K, W tends to be small. This corresponds to a strong preference for $V = 0$ in $p(\mathbf{v})$. This immediately highlights a problem with the single W-valued RBM for the xy-model: $p(\mathbf{v})$ is a function of V only, but $q(\mathbf{v})$ does not depend explicitly on V. Meaning that since $q(\mathbf{v})$ depends on the differences between neighbouring spins only, changes in V of a configuration will not necessarily alter the probability associated with that configuration, as long as the differences between the spins are conserved. When K goes to 0 one would expect $W \to 0$, since all configurations would then be equally likely. However, as is seen in the figure, it appears as if any $W \leq 0$ would be an extremum of the KL-divergence. For any finite K it appears as if $K = 0$ would always form a solution as well as some other negative K value.
From this brief numerical study we conclude that a single W-valued continuous RBM is unable to learn the one-dimensional xy-model.



(A) $K = 9999$

(B) $K = 0$

(C) $K = 5$

FIGURE B.1: *Equation* (B.19) *left hand side (yellow) and right hand side (blue) plotted for different values of K for a* 2×1 *RBM and a two-sited xy-model.*

Bibliography

1. Minsky, M. *Neural Nets and the Brain Model Problem* PhD thesis (Princeton University, 1954).
2. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* **65,** 386 (1958).
3. Michie, D. Experiments on the mechanization of game-learning Part I. Characterization of the model and its parameters. *The Computer Journal* **6,** 232–236 (1963).
4. Minsky, M. & Papert, S. An introduction to computational geometry. *Cambridge tiass., HIT* (1969).
5. Linnainmaa, S. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics* **16,** 146–160 (1976).
6. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics* **36,** 193–202 (1980).
7. Rumelhart, D. E., Hinton, G. E., Williams, R. J., *et al.* Learning representations by back-propagating errors. *Cognitive modeling* **5,** 1 (1988).
8. Markoff, J. What's the best answer? It's survival of the fittest. *New York Times* (1990).
9. Campbell, M., Hoane Jr, A. J. & Hsu, F.-h. Deep blue. *Artificial intelligence* **134,** 57–83 (2002).
10. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *nature* **529,** 484 (2016).
11. Amatriain, X. *Big & personal: data and models behind netflix recommendations* in *Proceedings of the 2nd international workshop on big data, streams and heterogeneous source Mining: Algorithms, systems, programming models and applications* (2013), 1–6.
12. Rosenberg, C. Improving photo search: A step across the semantic gap. *Google Research Blog* **12** (2013).
13. Binder, K. *et al. Monte Carlo methods in statistical physics* (Springer Science & Business Media, 2012).
14. Araki, H., Mizoguchi, T. & Hatsugai, Y. Phase Diagram of Disordered Higher Order Topological Insulator: a Machine Learning Study. <https://arxiv.org/abs/1809.09865> (2018).
15. Hsu, Y.-T., Li, X., Deng, D.-L. & Sarma, S. D. *Machine learning many-body localization: Search for the elusive nonergodic metal* May 2018.
16. Ch'ng, K., Carrasquilla, J., Melko, R. G. & Khatami, E. Machine Learning Phases of Strongly Correlated Fermions. *Physical Review X* **7,** 031038. ISSN: 2160-3308 (Aug. 2017).

17. Dong, X.-Y., Pollmann, F. & Zhang, X.-F. Machine learning of quantum phase transitions. <https://arxiv.org/abs/1806.00829> (2018).

18. Deng, D.-L., Li, X. & Sarma, S. D. Machine learning topological states. *Physical Review B* **96,** 195145 (2017).

19. Levine, Y., Sharir, O., Cohen, N. & Shashua, A. *Bridging Many-Body Quantum Physics and Deep Learning via Tensor Networks* Mar. 2018.

20. Liang, X. *et al.* Solving frustrated quantum many-particle models with convolutional neural networks. *Physical Review B* **98,** 104426 (Sept. 2018).

21. Nelson, J., Tiwari, R. & Sanvito, S. Machine learning density functional theory for the Hubbard model. <https://arxiv.org/abs/1810.12700> (Oct. 2018).

22. Gao, X. & Duan, L.-M. Efficient representation of quantum many-body states with deep neural networks. *Nature communications* **8,** 662 (2017).

23. Kashiwa, K., Kikuchi, Y. & Tomiya, A. Phase transition encoded in neural network. *arXiv preprint arXiv:1812.01522* (2018).

24. Beach, M. J. S., Golubeva, A. & Melko, R. G. Machine learning vortices at the Kosterlitz-Thouless transition. *Physical Review B* **97,** 045207 (Jan. 2018).

25. Carrasquilla, J. & Melko, R. G. Machine learning phases of matter. *Nature Physics* **13,** 431–434. ISSN: 1745-2473 (Feb. 2017).

26. Iakovlev, I. A., Sotnikov, O. M. & Mazurenko, V. V. Supervised learning approach for recognizing magnetic skyrmion phases. *Physical Review B* **98,** 174411 (Nov. 2018).

27. Van Nieuwenburg, E. P., Liu, Y.-H. & Huber, S. D. Learning phase transitions by confusion. *Nature Physics* **13,** 435 (2017).

28. Tanaka, A. & Tomiya, A. Detection of Phase Transition via Convolutional Neural Networks. *Journal of the Physical Society of Japan* **86,** 063001. ISSN: 0031-9015 (Apr. 2017).

29. Lee, S. S. & Kim, B. J. Confusion scheme in machine learning detects double phase transitions and quasi-long-range order. *Physical Review E* **99,** 043308 (2019).

30. Funai, S. S. & Giataganas, D. Thermodynamics and Feature Extraction by Machine Learning. <https://arxiv.org/abs/1810.08179> (2018).

31. Iso, S., Shiba, S. & Yokoo, S. Scale-invariant feature extraction of neural network and renormalization group flow. *Physical Review E* **97,** 053304 (2018).

32. Torlai, G. & Melko, R. G. Learning thermodynamics with Boltzmann machines. *Physical Review B* **94,** 165134 (2016).

33. Efthymiou, S., Beach, M. J. & Melko, R. G. Super-resolving the Ising model with convolutional neural networks. *Physical Review B* **99,** 075113 (2019).

34. Casert, C., Vieijra, T., Nys, J. & Ryckebusch, J. Interpretable Machine Learning for Inferring the Phase Boundaries in a Non-equilibrium System. <https://arxiv.org/abs/1807.02468> (2018).

35. Biamonte, J. *et al.* Quantum machine learning. *Nature* **549,** 195–202. ISSN: 0028-0836 (Sept. 2017).

36. Liu, Y.-H. & Poulin, D. Neural belief-propagation decoders for quantum error-correcting codes. *Physical Review Letters* **122,** 200501 (2019).

37. Khalek, R. A., Ethier, J. J. & Rojo, J. Nuclear Parton Distributions from Neural Networks. *arXiv preprint arXiv:1811.05858* (2018).
38. Lin, H., Tegmark, M. & Rolnick, D. Why Does Deep and Cheap Learning Work So Well? *Journal of Statistical Physics* **168,** 1223–1247. ISSN: 0022-4715 (Sept. 2017).
39. Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation* **14,** 1771–1800 (2002).
40. Mehta, P. *et al. A high-bias, low-variance introduction to Machine Learning for physicists* Mar. 2018.
41. Zeiler, M. D. & Fergus, R. *Visualizing and understanding convolutional networks* in *European conference on computer vision* (2014), 818–833.
42. Sobel, I. An Isotropic 3x3 Image Gradient Operator. *Presentation at Stanford A.I. Project 1968* (Feb. 1968).
43. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* **15,** 1929–1958 (2014).
44. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. <https://www.tensorflow.org/>.
45. Fischer, A. & Igel, C. Training restricted Boltzmann machines: An introduction. *Pattern Recognition* **47,** 25–39 (2014).
46. Tieleman, T. & Hinton, G. *Using fast weights to improve persistent contrastive divergence* in *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), 1033–1040.
47. Jaeger, G. The Ehrenfest Classification of Phase Transitions: Introduction and Evolution. *Archive for History of Exact Sciences* **53,** 51–81. ISSN: 1432-0657 (May 1998).
48. Ising, E. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik* **31,** 253–258. ISSN: 0044-3328 (Feb. 1925).
49. Onsager, L. Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition. *Physical Review* **65,** 117–149 (Feb. 1944).
50. Yang, C. N. The spontaneous magnetization of a two-dimensional Ising model. *Physical Review* **85,** 808 (1952).
51. Landau, D. P. & Binder, K. *A guide to Monte Carlo simulations in statistical physics* (Cambridge university press, 2014).
52. Ferdinand, A. E. & Fisher, M. E. Bounded and inhomogeneous Ising models. I. Specific-heat anomaly of a finite lattice. *Physical Review* **185,** 832 (1969).
53. Vvedensky, D. *Renormalization Group for the Two-Dimensional Ising Model* http://www.lorentzcenter.nl/lc/web/2010/404/presentations/VvedenskyI_2.pdf. Lecture Notes.
54. Wannier, G. H. Antiferromagnetism. The Triangular Ising Net. *Physical Review* **79,** 357–364 (July 1950).
55. Stephenson, J. Ising-Model Spin Correlations on the Triangular Lattice. III. Isotropic Antiferromagnetic Lattice. *Journal of Mathematical Physics* **11,** 413–419 (1970).
56. Chaikin, P. M., Lubensky, T. C. & Witten, T. A. *Principles of condensed matter physics* (Cambridge university press Cambridge, 1995).

57. Kosterlitz, J. M. & Thouless, D. J. Ordering, metastability and phase transitions in two-dimensional systems. *Journal of Physics C: Solid State Physics* **6,** 1181 (1973).

58. Nelson, D. R. & Kosterlitz, J. Universal jump in the superfluid density of two-dimensional superfluids. *Physical Review Letters* **39,** 1201 (1977).

59. Chung, S. G. Essential finite-size effect in the two-dimensional XY model. *Phys. Rev. B* **60,** 11761–11764 (16 Oct. 1999).

60. Olsson, P. Monte Carlo analysis of the two-dimensional XY model. II. Comparison with the Kosterlitz renormalization-group equations. *Phys. Rev. B* **52,** 4526–4535 (6 Aug. 1995).

61. Komura, Y. & Okabe, Y. Large-scale Monte Carlo simulation of two-dimensional classical XY model using multiple GPUs. *Journal of the Physical Society of Japan* **81,** 113001 (2012).

62. Villain, J. Two-level systems in a spin-glass model. I. General formalism and two-dimensional model. *Journal of Physics C: Solid State Physics* **10,** 4793 (1977).

63. Minnhagen, P. & Wallin, M. New phase diagram for the two-dimensional Coulomb gas. *Physical Review B* **36,** 5620 (1987).

64. Grest, G. S. Critical behavior of the two-dimensional uniformly frustrated charged Coulomb gas. *Physical Review B* **39,** 9267 (1989).

65. Lee, J.-R. & Teitel, S. Phase transitions in classical two-dimensional lattice Coulomb gases. *Physical Review B* **46,** 3247 (1992).

66. Minnhagen, P. & Weber, H. Non-universal Kosterlitz-Thouless jumps and two-dimensional XY-type models. *Physica B: Condensed Matter* **152,** 50–55 (1988).

67. Wolff, U. Collective Monte Carlo updating for spin systems. *Physical Review Letters* **62,** 361 (1989).

68. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

69. Ter Hoeve, J. *Renormalization Group connected to Neural Networks.* Bachelor Thesis (Utrecht University, June 2018).

70. Zhong, W. private communication. May 29, 2019.

71. Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **355,** 602–606 (2017).

72. Maskara, N., Kubica, A. & Jochym-O'Connor, T. Advantages of versatile neural-network decoding for topological codes. *arXiv preprint arXiv:1802.08680* (2018).

73. Newman, M. & Barkema, G. *Monte carlo methods in statistical physics chapter 1-4* (Oxford University Press: New York, USA, 1999).

74. Zhang, G. M. & Yang, C. Z. Cluster Monte Carlo dynamics for the antiferromagnetic Ising model on a triangular lattice. *Physical Review B* **50,** 12546–12549 (Nov. 1994).

75. Mattis, D. C. Transfer matrix in plane-rotator model. *Physics Letters A* **104,** 357–360 (1984).