Universiteit Utrecht

Department of Physics and Astronomy

Renormalization Group connected to Neural Networks

Bachelor Thesis

Jaco ter Hoeve

Supervisors:

Dr. Deb Panja
Department of Information and Computing Sciences

Prof. Dr. Ir. Henk Stoof
Institute of Theoretical Physics

June 12, 2018

Abstract

Even though deep learning has proved to be very powerful as the core method of machine learning, theoretical understanding behind its success is still unclear. It is been pointed out in recent years that the behaviour of deep neural networks is reminiscent of a fundamental framework in statistical physics: the renormalization group (RG). Motivated by this analogy, we develop an analytical method of directly obtaining the trained weights W resulting from a training set of 1D-Ising samples with coupling J and temperature T. The found relation $W(J)$ drives a flow that takes 1D-Ising configurations at non-zero temperature T to the critical point at $T_c = 0$. This behaviour is opposite to what a typical RG-flow dictates.

Contents

1 Introduction

Machine learning forms the very cornerstone of Artificial Intelligence and has proved to be very successful in various different disguises, ranging from voice recognition to computer vision. Amongst the different methods of machine learning that exist up to this day, the one that has been attracting much attention recently is deep learning. It has achieved record-breaking results in performing tasks such as natural language processing and image recognition. Despite its enormous success, relatively little is in fact understood as to why it works so incredibly well. Therefore, active research is currently being done in order to find the reason behind its success. One particular study, by Mehta and Schwab (2014), that has been devoted to this topic suggests that the answer might actually be found in physics. They argue that the behaviour of deep learning is quite reminiscent of the renormalization group (RG) framework within statistical physics and quantum field theory. RG is one of the most profound techniques of theoretical physics developed in the second half of the 20^{th} century. It was originally devised to explain phenomena in high energy physics, but thanks to the insight and contributions of K. Wilson [1] it was found that RG could also be applied to the scaling theory of critical phenomena. It it this latter application of RG that we shall focus on throughout the rest of this work.

Why is there reason to expect a connection between RG and deep learning in the first place? First and foremost, they both share the same goal, i.e. iteratively coarse-graining data to extract relevant features from large complicated data sets. At the same time, those features which are not as relevant get suppressed. For instance, very distinct physical systems can in fact exhibit exactly the same scaling behaviour close to a phase transition. This leads to the idea of universality and universality classes: the idea that all systems within a so-called universality class share the same scaling behaviour or, equivalently, critical exponents close to criticality. Universality forms the very basis of a RG-transformation. All those features of a physical system that are shared amongst the members within a universality class get emphasised by the RG-transformation while those that are not shared are considered irrelevant and thus get suppressed. Similarly, within the field of deep learning one follows this same pattern. For example, several images of a cat differing distinctly on a microscopic scale, i.e. the pixel values, can still all get classified as a cat. It is really the features that are shared amongst other cats, such as the shape of their ears or the fact that they have whiskers, that get emphasised so as to correctly label previously unseen cats. Other features, such as the background of the picture, do not get picked up on as strongly. These are considered irrelevant and get suppressed. Furthermore, both RG and deep learning seem to extract features by following a coarse-graining procedure. A deep neural network (DNN), for example, learns increasingly more macroscopic features as the data flows downstream. Similarly, RG also performs a coarse graining procedure by integrating out microscopic degrees of freedom and re-expressing the same physics in terms of a new set of degrees of freedom defined on a coarser scale.

However, even though the above account suggests the existence of a deep intimate relationship between deep learning and RG, Mehta and Schwab [2] did not manage to draw any connection between the more powerful ideas within RG and deep learning, such as the RG-flow and its associated fixed points or critical points. Another, more recent, account on this topic by S. Iso, S.Shiba and S.Yokoo [3] did in fact study the RG-flow in relation to Re-

stricted Boltzmann Machines (RBMs). They had to conclude, however, that the behaviour of RBMs is in fact opposite to what RG dictates.

In this thesis we develop a connection between RG and RBMs by analysing the behaviour of the RBM in relation to the 1D-Ising model. By studying the 1D-Ising model we were able to find an analytical expression relating the RG-flow to the RBM. This gives more mathematical insight into the very connection compared to earlier accounts on this topic [2, 3], which predominantly relied on numerical findings. Our main finding is that the RBM generates a flow towards the critical point of the 1D-Ising model. This is indeed opposite to the behaviour within the RG-framework, as was also found numerically by Ref. [3] in 2D.

The remainder of this thesis is organised as follows. In chapters two and three we first outline the technical prerequisites for understanding neural networks and the RG-group respectively. If the reader is already familiar with both neural networks and RG, we suggest skipping straight to chapter 4. There, we present our newly found results on the connection between RG and RBMs. We finish off by outlining possible future research questions that resulted from this very research.

2 Neural Networks

2.1 Introduction to Machine Learning

Even though the interest and advancements in the field of machine learning have increased very much recently, the concepts underlying the very idea do in fact date back to way earlier, to the 1960s [4]. It is only due to the increase in the power of computer processor units and the high availability of data, that the development of machine learning has taken a huge leap forward. The idea, nonetheless, is still fairly intuitive. Machine learning is about improving some performance measure P for a task T while increasing the experience E. Naturally, the machine's experience increases as it performs its task more often. The performance measure ensures that desired behaviour can get rewarded such that subsequent tasks will be performed more accurately. So to take a common example, the longer voice recognition software listens to a person his or her voice the more accurate it becomes. A suitable definition of what we mean by accuracy will however be needed in order to quantify the process of learning.

We generally consider two different schemes of machine learning, namely supervised and unsupervised learning. In both cases the machine is supplied with training data as an input in order to learn and thus to increase its performance on subsequent tasks. In supervised learning, however, a set of desired outcomes is passed along together with the input vector so as to be able to tell how well the machine performs its task, whereas in unsupervised learning this does not happen. Unsupervised learning is about learning an unknown probability distribution based on sample data drawn from that distribution, hence no labels telling how well the machine performs are involved. This can be useful nonetheless, for example, if one is interested in filling in parts of an unfinished image or to generate new samples from the learnt distribution that share the same statistics as the training samples. As for supervised learning, the goals is to ultimately correctly determine the class labels of unseen input vectors. So a learning algorithm must on the one hand adjust the model in such a way that the training examples are mapped to their respective desired outputs, on the other hand the machine must also allow for enough generality in order for new, unseen, inputs to be mapped to their correct class.

In both learning schemes, the machine can be interpreted as a parameterized function that maps an input to an output. The parameters give the machine a certain degree of flexibility such that the accuracy can be enhanced. The learning algorithm tells the machine how to change these parameters, which can be thought of as adjustable knobs, in order to achieve this higher accuracy.

The mathematical form of the parameterized function depends on the building blocks that are used for machine learning. A commonly used architecture in both supervised and unsupervised learning is a neural network. This is a collection of interconnected units, representing biological neurons, that each take a certain state and pass them on to neighbouring units. Because the very neuron itself is the building block of any neural network, we will start by discussing the workings of a single neuron. Then we will gradually expand this to Boltzmann Machines and Restricted Boltzmann Machines.

2.2 Deterministic Neural Networks

A single neuron

The fundamental building block constituting any neural network is a single neuron. This is a composite function that associates to I continuous inputs, labeled x_i, an output y. Its structure can be best understood by considering figure (1) [5].

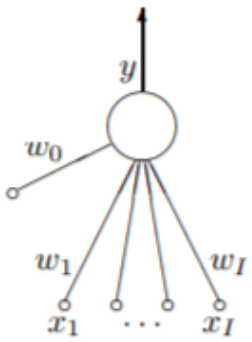

Figure 1: A single neuron with I inputs, weights w_i and output y.

The inputs are connected to the neuron via a set of weights w_i, expressing the relative importance of input x_i compared to the other inputs. Together they form a sum, called the activation a of the unit, where each input x_i gets weighted by w_i:

$$a = \sum_i w_i x_i. \tag{1}$$

The activity measures the degree to which a neuron gets triggered by external stimuli. The weights allow the neuron to be more sensitive to certain stimuli than others. If, however, the neuron has a preferred state of activity, we can take this into account by adding a bias term to the sum in equation (1). This causes the activity to be shifted to either a more active or inactive state regardless of what the neuron's external stimuli might be. Equivalently, this can be thought of as a weight associated to an extra input x_0 whose state is permanently set to one.

Given the activation a of the neuron, the activity y is computed by acting with the activation function σ on a in order to confine it to a certain range. The activation function that we will commonly use throughout this work is the sigmoid-function (or the logistic function):

$$\sigma(a) = \frac{1}{1 + e^{-a}}. \tag{2}$$

The higher the activation a flowing into a neuron the closer its activity will be to one. Conversely, a low activation will get converted to an activity close to minus one.

In the case of supervised learning the weight parameters w_i are chosen such that the neuron gives an output y sufficiently "close" to a target value t. Therefore, learning only a single neuron can be thought of as searching in the weight space for an optimum value.

What exactly is meant by "close" we will come to in the next subsection when discussing the objective function.

Learning the single neuron

Consider a training set $\{\mathbf{x}^{(n)}\}$ of size N with associated target values $\{t^{(n)}\}$. Here $t^{(n)} \in \{0, 1\}$, and n labels the training examples. The training set $\{\mathbf{x}^{(n)}\}$ gets mapped to $\{y^{(n)}\}$ by the function $y(\mathbf{x}, \mathbf{w})$, which represents a single neuron with inputs $\mathbf{x}$ and weights $\mathbf{w}$. Now, the objective function, or the error function, defines the discrepancy between these target values $t^{(n)}$ and the output $y^{(n)}$. It measures how well the neuron with a particular set of weights and biases fits the training data. We define the following objective function, denoted G:

$$G(\mathbf{x}) = -\sum_{n=1}^{N} \left[t^{(n)} \ln y(\mathbf{x}^{(n)}, \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}, \mathbf{w})) \right]. \tag{3}$$

where we sum over the entire training set. The more accurately $\{\mathbf{x}^{(n)}\}$ gets mapped to $\{t^{(n)}\}$ the smaller G must be. A natural way to understand how this particular form of the objective function comes about is to interpret the output $y(\mathbf{x}^{(n)}, \mathbf{w})$ as defining the probability that the input vector $\mathbf{x}^{(n)}$ belongs to the class $t = 1$, rather than $t = 0$:

$$\begin{aligned} P(t = 1|\mathbf{w}, \mathbf{x}) &= y, \\ P(t = 0|\mathbf{w}, \mathbf{x}) &= 1 - y. \end{aligned} \tag{4}$$

This interpretation allows for a Bayesian approach that will eventually result in the objective function stated in equation (3). In Bayesian statistics one defines a certain degree of believe in a hypothesis that is to account for the observed data. The key thing to statistical inference is to try and deduce what the underlying distribution dictating a statistical phenomena, such as coin flipping, might be. Different distributions that try to explain the observed data are characterized by a different set of parameters, such as the mean or standard deviation. One can formulate a hypothesis as corresponding to a certain choice of these parameters. Before any evidence, or data, is taken into account one's beliefs can be expressed in terms of the prior distribution over the parameter in the model. Then, after the data is observed, one adjusts this prior distribution by the likelihood: the probability of the observed data given the asserted hypothesis. After dividing by an overall normalizing constant, the adjusted prior, also called the posterior distribution, becomes:

$$P(\mathbf{w}|D) = \frac{P(D|\mathbf{w})P(\mathbf{w})}{P(D)}. \tag{5}$$

This is simply a version of Bayes' theorem applied to credibilities as opposed to the usually encountered probabilities. It also applies to supervised learning in which the aim is to learn a parameterised model by inferring from the target vectors $\{t^{(n)}\}$ a set of weights and biases that make the model fit the data as well as possible.

Recalling the probabilistic interpretation of the output $y^{(n)}$ in equation (4), we can rewrite (4) into one single equation:

$$P(t|\mathbf{x}, \mathbf{w}) = y^t(1 - y)^{1-t} = \exp\left[t \ln y + (1 - t) \ln(1 - y)\right]. \tag{6}$$

Assuming the target values are independent, the probability of the overall target set given the hypothesis factorises:

$$\mathcal{L} = P(\{t^{(n)}\}|\mathbf{w}) = \prod_n P(t^{(n)}|\mathbf{w}, \mathbf{x}) \tag{7}$$

$$= \exp\left[\sum_n t^{(n)} \ln y(\mathbf{x}^{(n)}, \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}, \mathbf{w}))\right]. \tag{8}$$

Substituting equation (3) into equation (8) gives for the likelihood $\mathcal{L}$:

$$\mathcal{L} = P(\{t^{(n)}\}|\mathbf{w}) = \exp\left[-G(\mathbf{x})\right]. \tag{9}$$

Increasing the neuron's performance by minimising the objective function thus corresponds to maximising the log-likelihood. This motivates the definition of equation (3).

The goal of training is to adjust the set of weight and biases so as to minimise the objective function. Therefore we take the derivative of $G(\mathbf{x})$ with respect to the weight w_i to find:

$$\frac{\partial G(\mathbf{x})}{\partial w_i} = \sum_{n=1}^{N} -(t^{(n)} - y^{(n)}) x_i^{(n)}. \tag{10}$$

Rather than equating this to zero and solving the simultaneous equations for the weights, one usually performs gradient descend on the objective function to find the set of weights that take the objective function sufficiently close to a stationary point.

The Hopfield network

In the last section, we studied the behaviour of a single neuron. Although this has already shed light on some of the principles underlying machine learning, the dynamical behaviour that could be described was still limited due to the small number of degrees of freedom involved. Let us therefore increase this by wiring several of these neurons together. The output of many single neurons will now collectively form the input flowing into another neuron, whose own output will in turn be passed on to other neurons too. This way we get a whole network of interconnected neurons: the neural network. Previously, the activation a of a neuron could be accurately expressed just in terms of a weight vector w_i and the continuous input vector x_i:

$$a = \sum_i w_i x_i.$$

However, now that we are wishing to describe a whole network of neurons, a weight can no longer uniquely be specified by a single index i. We also need a second index to specify the pair of neurons the weight belongs to. In addition to this, in our further discussion we only need discrete inputs, so we set $x_i \in \{-1, 1\}$. Our new definition of the activity of the i^{th} neuron, given its inputs x_j, becomes:

$$a_i = \sum_j w_{ij} x_j. \tag{11}$$

where the w_{ij}'s form a symmetric weight matrix, i.e. $w_{ij} = w_{ji}$. Also, by definition, no neuron can influence its own activity, that is $w_{ii} = 0$.

One particular important example of a neural network is the Hopfield network [6]. In the language of graph theory, this network is simply a complete graph, i.e. every unit in the network is connected to every other unit. The states of all the units in the network make up one configuration, to which an energy E gets associated expressed in terms of the activities and the weights connecting the units:

$$E = -\sum_i x_i b_i - \sum_{i<j} x_i x_j w_{ij}. \tag{12}$$

where we have explicitly written out term with bias b_i. The second term is a summation over all the pairs in the Hopfield network, taking into account the degree to which two units are connected.

Up to now we have only given a description of the architecture and its state, but a Hopfield network also has dynamics associated to it determining its passage through time. The units are binary threshold units, which means that they update their state (or activity) x_i according to whether its activation a_i passes a certain threshold or not above which the unit is set to the active state. If the a_i does not pass the threshold the unit x_i is switched off. More explicitly, the activity x_i, given its activation a_i, is obtained through[1]:

$$x_i(a_i) = \Theta(a_i) \equiv \begin{cases} 1 & \text{if } a_i \geq 0 \\ -1 & \text{if } a_i < 0 \end{cases} \tag{13}$$

with $\Theta(a_i)$ the step function. How does this relate to the energy function in equation (12)? Consider the change in energy ΔE_i of the network when the i^{th} unit updates its state according to the binary threshold decision rule:

$$\Delta E_i = E(x_i = -1) - E(x_i = 1) = 2\left(b_i + \sum_j x_j w_{ij}\right). \tag{14}$$

Notice that the change in energy ΔE_i is only affected by units x_j connecting to the flipped unit x_i. The bias term b_i can be absorbed into sum by introducing a permanently active unit associated to every single unit in the network. The weight connecting to this permanently active unit then effectively acts like a bias. Taking this adjustment into account, the energy difference ΔE_i can now be rewritten as:

$$\Delta E_i = 2\sum_j x_j w_{ij}. \tag{15}$$

Comparing this to the activation of a single neuron in equation (11) we conclude that the energy gap ΔE_i associated to unit i simply equals twice its activation a_i. Recalling the binary threshold decision rule in equation (13), we conclude that this causes the Hopfield network to go downhill in energy, since each unit gets updated to whichever of its two states gives the lowest global energy E. From this we can deduce the network's dynamics. For instance,

[1]The activity x_i was previously called y in figure (1), but since the activity itself gets passed on to other neurons in the Hopfield network it is also just an input x_i.

by letting the network start off in an arbitrary configuration and continuously picking a unit at random to update its state, it will eventually settle down to an energy minimum. This is however not guaranteed to be a global minimum, since the network can also get trapped at a local minimum.

So what is a Hopfield network good for? One application of Hopfield networks is storing memories. John Hopfield proposed that one could let energy minima of the network correspond to memories. The idea is that, if this can indeed be done, the binary threshold decision rule causes distorted memories to flow back to their stable points in the energy landscape. In this way, corrupted images can be restored to the way they were before they got distorted. Moreover, a stored item can be accessed by only knowing parts of its contents. Forcing some units to have fixed values corresponding to just parts a memory, the complete memory can get filled out if one sequentially applies the update rule to the remaining units that started off in a randomly initialized state.

The hard assumption underlying the storage ability of a Hopfield network is that the energy landscape can be deformed in such a way that certain memories can be made to correspond to local minima. This is done by changing the weights. Exactly how these should be altered to meet a particular purpose is generally given by a learning rule. In the case of the Hopfield network the learning rule goes under the name of the Hebbian learning rule. The idea is that given the memories the network is supposed to store, the Hebbian learning rule simply increments the weights between any units that are both switched on:

$$w_{ij} = \eta \sum_n x_i^{(n)} x_j^{(n)}, \tag{16}$$

where η is the learning rate. Note that the learning rule does not involve any target values that have been passed along, so the Hopfield network is a unsupervised neural network. Furthermore, the set $\{w_{ij}\}$ forms nothing more than the correlation matrix representing the degree to which different x_i's tend to be correlated. This is really where the associative nature of biological memory comes in. Neurons firing in isolation will cause other correlated neurons to become active as well. Applied to the storage of images, training the Hopfield network with the Hebbian learning causes the images to be remembered to correspond to stationary points of the energy function. Combining this with the updating dynamics of the Hopfield network, as described by the activity rule in equation (13), we see that a Hopfield network is indeed able to retrieve memories. Note however, that the stationary points are embedded in a very high dimensional space and the more and more memories we try to store, the rougher the energy landscape becomes. As a result, even memories distorted by only a tiny amount will sometimes not flow towards the desired stationary points, but will reside in some other local minima instead. A gross percentage of the distorted memories will therefore not be recovered correctly. So it is a matter of striking the right balance between the number of memories and their stability.

Numerous other activity rules other than equation (13) have been introduced to prevent the network from getting trapped, such as simulated annealing or a stochastic update rule [7]. This brings us now to the Boltzmann machine.

2.3 Boltzmann Machines

A stochastic network

In the last section we discussed Hopfield networks as a type of neural network that uses its local energy minima to store items. In other words, for a set of memories that we wish to store, the weights and biases are set according to the Hebbian learning rule, equation (16), so as to let these memories correspond to local energy minima. Suppose that the system is started near one of these local energy minima, the dynamical update rule, as stated by (13), then causes the system to eventually fall into that minimum. It is important to note that this happens in a completely deterministic way, i.e. given the weights and biases associated to the Hopfield network, the same initial configuration always gives rise to the same dynamical behaviour as the network approaches its local stable configuration.

As opposed to the Hopfield network, the so-called Boltzmann machine does not behave deterministically. Even though the Boltzmann machine shares the same architecture as the Hopfield network, i.e. every unit is connected to all other units with an appropriate weight, its dynamical behaviour, however, never settles down into a stable state. To put it differently, it simply keeps updating all of its units stochastically. By initialising all of the units in the Boltzmann machine arbitrarily and allowing it to continuously update its states freely, we can keep track of the fraction n_α/N the Boltzmann machine is in a particular state α as it passes through a total of N states. As N grows large, n_α/N becomes an increasingly more accurate approximation of the probability P_α to find the network in a state α. One might argue that P_α too simply keeps changing while the Boltzmann machine keeps updating its units stochastically, but it turns out that eventually the fraction n_α/N converges to a well defined value. This is the concept of thermal equilibrium [8]; even though the states of individual units may still fluctuate, the probability of a particular configuration has converged. So, once this equilibrium has been reached, the Boltzmann machine represents nothing but a probability distribution over its units. We will see later that this corresponds to the Boltzmann distribution.

Now, given a finite set of binary data vectors $\{\mathbf{x}^{(n)}\}$ drawn from some unknown probability distribution p, the Boltzmann machine should adjust its internal parameters so as to represent the distribution p as faithfully as possible. We assume that in principle the distribution p exists, but its analytic form is unknown to us and so we would like to infer it from the training data $\{\mathbf{x}^{(n)}\}$. Why would this be useful in any way? Well, for one thing, it can be used to generate additional samples $\{\mathbf{x}^{(n)}\}$ from the learnt distribution that share the same statistics as the already existing training examples.

We have in fact not yet defined the stochastic update rule that governs the dynamics of the Boltzmann machine. While the Hopfield network simply switches the i^{th} unit into whichever state gives rise to the lowest total energy, the Boltzmann machine, on the other hand, sets the i^{th} unit to the active state with a probability:

$$p_i = \frac{1}{1 + \exp(-\Delta E_i/T)} = \sigma(\Delta E_i/T) = \sigma(2a_i/T), \tag{17}$$

where ΔE_i is the energy difference associated to the network in going from a configuration with active unit x_i to one with an inactive unit x_i. The total energy E_i and its change ΔE_i are given by equation (12) and (14) respectively. Also, a_i is the activation of the i^{th}

unit and T is the "temperature" of the network controlling the size of fluctuations. Notice a few aspects of equation (17). Firstly, at zero temperature the stochastic update rule (17) simply reduces to the ordinary binary threshold rule, equation (13). This can be seen by the fact that upon taking the zero temperature limit, equation (17) becomes a step-function. Secondly, if the activation a_i of the i^{th} unit is large, the probability of this unit switching on is likewise large. So "neurons that fire together wire together", an often heard quote in psychology.

Equation (17) brings us to the realm of statistical physics. Although the network will usually fall into a state of lower energy, occasional visits to higher energy states are possible as well, though will occur less often. In this way the network can sometimes escape from a local minima, because occasionally the network will make a jump just high enough to overcome the energy barrier.

Suppose now that we initialise the network in an arbitrary configuration and pick a unit at random to update its state according to the stochastic update rule (17). We then pick another one to update that one too. Because equation (17) satisfies the detailed balance equation, by successively following this procedure the network will eventually reach thermal equilibrium. Even though at non-zero temperature the states of individual units may still fluctuate, the probability $P(\{x_i\})$ of a configuration $\{x_i\}$ no longer changes and is given by the Boltzmann distribution:

$$P(\{x_i\}) = \frac{e^{-E(\{x_i\})/T}}{\sum_{\{x_i\}} e^{-E(\{x_i\})/T}}, \tag{18}$$

where the energy E associated to the configuration $\{x_i\}$ is given by equation (12). Two important aspects to note here are, firstly, upon having reached thermal equilibrium the system has "forgotten" about its initial configuration. This is because the probability distribution (18) will be the same regardless of which state the network started off in. Secondly, the probability of a state is completely determined by its associated energy.

Visible and hidden units

So far we have made the restrictive assumption (for didactic reasons) that the number of units in the Boltzmann machine should correspond to the number of components of the training vector. This is an unnecessary restriction, as we can also introduce an extra group of so called hidden units that do not directly take on values from the environment. We say that the number of units is split into visible (or observed) units corresponding to the components of the observations, and hidden units given by the remaining units. The hidden units function to capture higher order correlations in the environmental data set that could otherwise not have been modelled by the visible units alone. We shall denote the overall configuration of the network by $(\{v_i\}, \{h_a\})$, where $\{v_i\}$ and $\{h_a\}$ represent the configurations of the visible units and hidden units respectively.

The network is said to run freely when all the units, both visible and hidden, are allowed to update their states according the stochastic update rule (17). So we may start by setting both the visible and hidden units to a random state, and by successively applying the stochastic update rule the network will eventually reach thermal equilibrium.

As opposed to a network running freely, we can also decide to clamp the visible units to their corresponding environmental components. Now, when applying the stochastic update rule, it is only the hidden units that change their states as we gradually approach thermal equilibrium.

The goal of learning is to adjust the internal parameters of the network so that when it is running freely the probability distribution

$$P'(\{v_i\}) = \sum_{\{h_a\}} P'(\{v_i\}, \{h_a\}), \tag{19}$$

obtained by marginalizing over the hidden units $\{h_a\}$, approximates the environmental distribution $P(\{v_i\})$ underlying the training set as faithfully as possible. In order to understand how the Boltzmann machine should be trained to increase its performance, we first need to discuss how we might in general carry out an unsupervised learning problem.

Training an Unsupervised Network

Suppose that we have a training set $S = \{\mathbf{x}^{(n)}\}$ drawn from some unknown probability distribution $p(\mathbf{x})$. The aim is to model this distribution as well as possible using another distribution $q(\mathbf{x}|\boldsymbol{\theta})$ with adjustable parameters $\boldsymbol{\theta}$. In order to be able to tell how well $q(\mathbf{x}|\boldsymbol{\theta})$ reflects $p(\mathbf{x})$ we need an objective function to measure their discrepancy, the so called Kullback-Leiber divergence (KL-divergence):

$$KL(p||q) = \sum_{\boldsymbol{x}} p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x}|\boldsymbol{\theta})}, \tag{20}$$

where the sum is over the finite state space of the distribution.

The better the parameterized distribution $q(\mathbf{x}|\boldsymbol{\theta})$ reflects $p(\mathbf{x})$ the smaller the KL-divergence becomes. Therefore, unsupervised learning is performed by adjusting the parameters $\boldsymbol{\theta}$ such that the KL-divergence is minimised. The problem arising, however, is that the probability distribution $p(\mathbf{x})$ is unknown to us (it has to be modelled after all). The way out of this is to use a finite set of N training examples $\{\boldsymbol{x}^{(n)}\}$, drawn from $p(\mathbf{x})$, to approximate equation (20) by:

$$KL(p||q) \approx \frac{1}{N} \sum_{n=1}^{N} \{-\ln q(\mathbf{x}^{(n)}|\boldsymbol{\theta}) + \ln p(\mathbf{x}^{(n)})\}, \tag{21}$$

The second term on the right side in equation (21) is independent of $\boldsymbol{\theta}$ and will therefore not play a role in minimising the KL-divergence with respect to the parameters. The first term is the negative log-likelihood function for the parameters $\boldsymbol{\theta}$ under the distribution $q(\boldsymbol{x}|\boldsymbol{\theta})$ evaluated using the training set. This let us conclude that maximising the log-likelihood is simply equivalent to minimising the KL-divergence. Therefore, learning an unsupervised network amounts to nothing more than the standard way of estimating the parameters in a statistical inference problem.

Training the Boltzmann Machine

Recall from the end of the second to last section that the Boltzmann machine is trained by adjusting the weights in such a way that when the network is running freely the probability distribution $P'(\{v_i\})$ over the visible units $\{v_i\}$ fits the unknown environmental (or clamped) distribution $P(\{v_i\})$ as well as possible. We express their discrepancy in terms of the KL-divergence

$$KL(P(\{v_i\}||P'(\{v_i\}))) = \sum_{\{v_i\}} P(\{v_i\}) \ln \frac{P(\{v_i\})}{P'(\{v_i\})}, \tag{22}$$

where the sum is over all possible configurations of the visible units. The goals is to minimise this divergence with respect to the weights. This brings us to the learning algorithm of the Boltzmann machine.

Finding stationary points of (22) analytically is a daunting task computationally wise due to the large number of degrees of freedom w_{ij} involved. Therefore, the learning algorithm is based on gradient descend on the KL-divergence instead. Given the initial values of the weights w_{ij}, the updated values are found by subtracting the gradient of (22) with respect to w_{ij} evaluated at the original point. By iterating this updating procedure, the hope is that the parameters will eventually make (22) stationary. In the Appendix we will derive this gradient of the KL-divergence with respect to the weights.

The result derived in the Appendix gives the following learning rule for the Boltzmann machine:

$$\frac{\partial KL}{\partial w_{ij}} = \langle x_i x_j \rangle_{model} - \langle x_i x_j \rangle_{data} \,. \tag{23}$$

It takes a surprisingly simple form, i.e. finding the gradient with respect to the weight w_{ij} only requires local information about the units it connects. Having found the gradient, the updated set of weights w_{ij} is now given by:

$$\Delta w_{ij} = \eta \left[\langle x_i x_j \rangle_{data} - \langle x_i x_j \rangle_{model} \right] . \tag{24}$$

where η is the learning rate that determines the speed and accuracy at which we go downhill in KL-divergence space.

We can attach a meaning to both terms in (24) as referring to two distinct phases in the learning process. The first term refers to the learning phase in which the visible units are clamped to a particular training example. The hidden units are not clamped, and are allowed to update their states freely. Starting off in an initial configuration, we first let the the hidden units update their states until the network has reached thermal equilibrium. Only then do we collect statistics by sampling several times the product $x_i x_j$. The average of these samples gives us the first term in equation (24). This tells us that we should increment any weight w_{ij} connecting two units x_i and x_j that are both active on average. Note that this is reminiscent of the Hebbian learning rule in the case of the Hopfield network. The second term in equation (24) refers to the so-called dreaming phase in which the network withdraws within itself and is allowed to update all of its states freely. Starting off in a complete random initial state, the network will eventually reach thermal equilibrium. Once reached, we sample

the products $x_i x_j$ in order to determine an accurate approximation of $\langle x_i x_j \rangle_{model}$. Finally we decrement any weight w_{ij} connecting two units x_i and x_j that are both switched off on average while the network is running freely.

It is important to note that once the weights are updated according to our rule (24), we first need to wait until the network has settled down again to thermal equilibrium before collecting any further statistics. Needles to say, having to wait until thermal equilibrium after every update only to collect many statistics that determine the increment of a single weight, is a computationally high-demanding task to put into practise. Therefore, we shall restrict the network's topology in order to speed up the learning algorithm. This shall be studied in the next section.

2.4 Restricted Boltzmann Machines

We have just discussed the Boltzmann machine, which allowed for all possible connections between the units it contained; any unit was connected to any other unit. We also distinguished between two different types of units, that is, we considered hidden and visible units. The set of visible units stood in direct contact with the environment that had to be modelled, which means that the components of a training vector were passed directly onto the visible units. The set of hidden units, however, did not stand in direct contact with the environment and thus remained hidden, or unseen. They served to increase the modelling capacity of the distribution over the visible units.

As its name already suggests, the restricted Boltzmann machine (RBM) still shares all of the general features that the Boltzmann machine already had, but imposes some further restrictions on the network's architecture in order to speed up the learning process. In the RBM no mutual connections within the set of either visible or hidden units are allowed. This partitions the network into two layers with no intra-layer connections: the visible layer with variables $\{v_i = \pm 1\}$ and the hidden layer with variables $\{h_a = \pm 1\}$. The configuration of the entire network is collectively given by $\{v_i, h_a\}$.

Associated to each such configuration is an energy E given by the following Hamiltonian[2]:

$$H(\{v_i\}, \{h_i\}) = -\left(\sum_{i,a} W_{ia} v_i h_a + \sum_i b_i^{(v)} v_i + \sum_a b_a^{(h)} h_a \right). \tag{25}$$

For a given set of weights W_{ia} and biases b an initial configuration of the network will again change its state stochastically according to equation (17). The network is said to be thermalised. As a result, the network will never settle down into a stable state because it simply keeps updating the state of each of its units. However, the fraction of distinct configurations it passes through as time goes on, will eventually converge. When this has happened the system has reached thermal equilibrium and the probability of a configuration $\{v_i, h_a\}$ does only depend on its energy and is given by

$$p(\{v_i\}, \{h_a\}) = \frac{1}{\mathcal{Z}} e^{-\mathcal{H}(\{v_i\}, \{h_a\})}, \tag{26}$$

[2]For those unfamiliar with the concept of the Hamiltonian, it suffices to think of it as simply being the energy that we encountered before in our discussion on the Hopfield network and Boltzmann machines.

where $\mathcal{Z}$ is the partition function defined by $\mathcal{Z} = \sum_{\{v_i,h_a\}} e^{-\mathcal{H}(\{v_i\},\{h_a\})}$. Configurations $\{v_i, h_a\}$ with a low associated energy are preferred by the network over those with a high energy, so a positive weight W_{ia} stimulates a positive correlation between the units v_i and h_a.

From $p(\{v_i\},\{h_a\})$ we can also easily derive the marginalized distribution over $\{v_i\}$:

$$p(\{v_i\}) = \frac{1}{\mathcal{Z}} \sum_{\{h_a\}} e^{-\mathcal{H}(\{v_i\},\{h_a\})}. \tag{27}$$

One other important aspect to notice straightaway is that due to the absence of intra-layer connections, given the visible units the hidden units are statistically independent and vice versa, so we can write:

$$p(\{h_a\}|\{v_i\}) = \prod_a p(h_a|\{v_i\}), \tag{28}$$

$$p(\{v_i\}|\{h_a\}) = \prod_i p(v_i|\{h_a\}). \tag{29}$$

These can be made more specific when applied to the RBM, since it holds that

$$\begin{aligned} p(h_a|\{v_i\}) &= \frac{1}{1+\exp\left[-2h_a(\sum_i w_{ia}v_i + b_a^{(h)})\right]}, \\ p(v_i|\{h_a\}) &= \frac{1}{1+\exp\left[-2v_i(\sum_a w_{ia}h_a + b_i^{(v)})\right]}. \end{aligned} \tag{30}$$

Using equations (30) the expectation value of a hidden unit being turned on given the visible units can also be obtained:

$$\begin{aligned} \langle h_a \rangle_{\{v_i\}} &= \sum_a h_a p(h_a|\{v_i\}) = p(h_a = 1|\{v_i\}) - p(h_a = -1|\{v_i\}) \\ &= \frac{1}{1+\exp\left[-2(\sum_i w_{ia}v_i + b_a^{(h)})\right]} - \frac{1}{1+\exp\left[2(\sum_i w_{ia}v_i + b_a^{(h)})\right]} \\ &= \tanh\left(\sum_i w_{ia}v_i + b_a^{(h)}\right). \end{aligned} \tag{31}$$

Hence,

$$\langle h_a \rangle_{\{v_i\}} = \tanh\left(\sum_i w_{ia}v_i + b_a^{(h)}\right), \tag{32}$$

$$\langle v_i \rangle_{\{h_a\}} = \tanh\left(\sum_a w_{ia}h_a + b_i^{(v)}\right). \tag{33}$$

where we have likewise carried out the same computation for $\langle v_i \rangle_{\{h_a\}}$.

Suppose that we have a set of samples $\{\sigma_i^A\}$ drawn from an unknown environmental distribution, with A denoting the various samples ($A = 1, ..., N$). It is not important how these samples might be handed over to us, let us simply assume that they are. The higher the number of samples N we have, the more faithfully the environmental distribution will be characterised by

$$q(\{v_i\}) = \frac{1}{N} \sum_A \delta(v_i - \sigma_i^A), \tag{34}$$

where δ is the Kronecker-delta symbol contributing to the sum whenever $\{v_i\}$ matches one of training samples $\{\sigma_i^A\}$.

Just like when training the Boltzmann machine, the goal now in training the RBM is to let $p(\{v_i\})$ over the visible units represent the sampling distribution $q(\{v_i\})$ as faithfully as possible. In order to quantify their discrepancy, the KL-divergence is again taken as our objective function:

$$KL(q||p) = \sum_{\{v_i\}} q(\{v_i\}) \ln \frac{q(\{v_i\})}{p(\{v_i\})}. \tag{35}$$

The learning algorithm derived in the Appendix applied to the Boltzmann machine, also holds for the RBM, since the latter is simply a special case of the former. Hence, the learning rule of the Boltzmann machine can readily be carried over to the RBM with only a few adjustments; only derivatives with respect to intra-layer weights can be considered, because of the restrictive architecture of the RBM.

$$\frac{\partial KL(q||p)}{\partial W_{ia}} = \langle v_i h_a \rangle_{model} - \langle v_i h_a \rangle_{data}. \tag{36}$$

This leads, again, to the following simple training rule:

$$\Delta W_{ia} = \eta(\langle v_i h_a \rangle_{data} - \langle v_i h_a \rangle_{model}), \tag{37}$$

where $\langle ... \rangle_{data}$ denotes an expectation value taken with respect to the conditional distribution of the hidden units given the visible units, and $\langle ... \rangle_{model}$ denotes the expectation value with respect to the joint distribution of the visible and the hidden units. The learning rate η again affects the step size while doing gradient descent.

The first term with the visible units clamped is easy to compute due to the absence of mutual connections between hidden units in the RBM. It requires taking samples from the conditional distribution $p(\{h_a\}|\{v_i\})$, computing the product $h_a v_i$ for the respective samples and averaging the result. More concretely, clamp the states of the visible units to a traning vector and compute the probability $p(h_a = 1|\{v_i\})$ using equation (30). Introduce a uniform distribution $U[0, 1]$ between 0 and 1 from which we draw a sample, x_i. If $x_i < p(h_a = 1|\{v_i\})$, set $h_a = 1$ and $h_a = -1$ otherwise. Compute the product $h_a v_i$ and repeat the same steps again for another x_i drawn from $U[0, 1]$. Finally average all of the products $h_a v_i$. Note that the entire hidden layer can be sampled jointly given the visible layer due to the absence of mutual connections within a layer. It is because of this that the learning algorithm for the RBM is significantly faster.

However, the second term where the model is allowed to update all of its states freely is more difficult to compute directly. This would require taking a sample from the joint

distribution which depends on the full partition function $\mathcal{Z}$ of the network. The summation over both the visible and the hidden units makes this intractable for regular sized RBMs. So how can this second term be determined?

The solution is to employ Gibbs sampling which allows one to sample from a joint distribution when one can easily sample from the conditional distributions. Here, the conditional probability distributions $p(\{h_a\}|\{v_i\})$ and $p(\{v_i\}|\{h_a\})$ are easy to sample from as was described in detail above. In fact, all variables within a layer could be sampled jointly due to their mutual independence given the other layer. By virtue of this observation, a whole sample $\{h_a\}$ drawn from $p(\{h_a\}|\{v_i\})$ can be obtained directly. Given this new sample $\{h_a\}$ the same procedure can be applied to sample $\{v_i\}$ based on $p(\{v_i\}|\{h_a\})$. This last sample is then again used to sample $\{h_a\}$ once more, etc. This is referred to as block Gibbs sampling. By continuing this sampling process one forms what is called a Markov chain [9]. Eventually this chain will converge to a stationary point. These samples at the end of the chain correspond to the same samples that one could also have obtained in principle by sampling directly from the joint distribution. In this way, one circumvents the problem of having to carry out an intractable double sum.

Being able to sample jointly greatly speeds up the learning process, and this is one of the main reasons of preferring a RBM for implementation over a regular Boltzmann machine. Because if the units within a layer had not been independent, samples could not have been drawn jointly, but would have to sampled subsequently instead.

In practise, however, one adopts even additional approximations since letting the Markov chain run all the way until stationarity is too inefficient to be implemented. This can be understood by realising that while performing gradient descent the Markov chain has to run all the way until equilibrium at each subsequent point in weight space. One additional approximation frequently employed in practise is contrastive divergence (CD) devised by G.Hinton (2002).

Contrastive Divergence (CD)

Instead of letting the Gibbs (or Markov) chain run all the way until it equilibrium, we can also wait for only k steps to measure the statistics. It turns out that despite this short-cut the learning still works. This is referred to as CD^k. The learning rule is still the same as before, except for only a slight change in the last term in (36):

$$\Delta w_{ij} = \epsilon(\langle v_i h_a \rangle^0 - \langle v_i h_a \rangle^k), \tag{38}$$

where the superscript indicates the number of times the Gibbs-chain should run. Note that the term that we are using to build up negative statistics is actually only an approximation, but it still causes gradient descent on average.

3 The Renormalization Group

Here we shall give a qualitative description of the ideas behind the renormalization group that will hopefully be a useful guide later on when discussing its connection to RBMs. Especially, we will look at what motivates the introduction of these ideas and how these can be applied to the 1D-Ising model.

Scale-invariance

The basic idea underpinning all of RG is the fact that at the critical temperature a statistical system becomes scale invariant. To see this, we follow the reasoning in Ref. [1]. Consider a macroscopic piece of material and measure some of its properties, such as the density. Next imagine it being cut up into two equally sized pieces and measure the same properties again. Assuming that the external parameters, such as temperature and pressure, have been kept fixed, the properties of the individual parts will be the same as those of the original part which they constituted. By repeatedly carrying out this procedure, i.e. cutting up a piece into two parts and measuring its properties, one eventually reaches a scale at which the outcomes of these measurements will in fact no longer stay the same. The length scale at which this so happens is called the correlation length. It is the typical length scale over which statistical degrees of freedom are correlated with respect to one another. This is precisely why the individual parts kept having the same properties until the correlation length was reached; degrees of freedom outside this range could not influence each other anyway, so effectively these were already disconnected before the actual separation took place.

Applied to the Ising model, the correlation length has the intuitive interpretation of the degree to which individual spins separated a distance d apart tend to align or not. At high temperature thermal fluctuations dominate and any structure the model previously might have had is completely lost. The correlation length in this case is close to zero. However, at low temperatures the model is more likely to reside in one of its energetically low states so that the spins tend to be aligned. Consequently, the correlation length will be much larger than the lattice spacing a. Clearly, the correlation length depends on the temperature, and at one very specific temperature, the so-called critical temperature T_c, the correlation length in fact diverges off to infinity. This signifies a second-order phase transition and the fluctuations of statistical degrees of freedom are now correlated over all length scales. In the case of Ising model this means that the fluctuations of every spin are correlated to every other spin at the critical temperature. In other words, no single spin can possibly be unaffected by changes of the other spins, no matter how far apart these might be. Because of this, clusters with aligned spins of all sizes will start to appear as the system approaches T_c. Once reached, the system has become scale invariant, since there is no longer a length scale that characterizes the typical length at which the dynamics of the system takes place. Even though there are phenomena that are dictated by local fluctuations as well, the physics describing a phase transition is really only determined by the long range correlations. This phenomena is exactly what the renormalization group captures, which shall be further discussed next and it is in this framework that phase transitions are best described.

Coarse-graining

The idea of describing nature at different length scales and how these descriptions relate to one another is very important. Whenever one wants to describe a natural phenomena one has to choose the right variables that are most appropriate to the associated scale. It would absolutely not make sense, for example, to construct a model of the atmosphere starting from the elementary particles. In other words, one always describes nature at some scale and depending on the scale at which one is interested the description will change. Consider again the 1D-Ising model of spins a distance a apart with the following Hamiltonian:

$$-\beta\mathcal{H}_a(\{\sigma_i\}) = J\sum_{<ij>}\sigma_i\sigma_j + h\sum_i \sigma_i,$$

where the subscript a attached to the Hamiltonian serves as a reminder of the resolution or scale at which the system is described. Now imagine tuning the external parameters of the environment in which the Ising model is embedded so that the system approaches the critical point. As was argued in the previous section, at criticality the correlation length of the Ising model diverges. So clearly having a microscopic description at scale of order a is not the ideal setting to study a phase transition. Therefore, one defines a different set of degrees of freedom on a coarser scale without changing the large distance physics. This is the scale that really matters after all when describing a phase transition. This idea is nicely summarised by J.Cardy [1]: "The underlying idea of the renormalization group is to re-express the parameters which define a problem in terms of some other, perhaps simpler, set while keeping unchanged those physical aspects of the problem which are of interest." More concretely, in the case of a statistical system this amounts to re-expressing the same partition function in terms of a coarser set of degree of freedoms:

$$Z = Tr_\sigma e^{-H(\sigma)} = Tr_{\sigma'} e^{-H'(\sigma')}. \tag{39}$$

Given a coarse-grained set of degrees of freedom $\{\sigma_i'\}$, the next step is now to find a renormalized hamiltonian H' such that this equation is satisfied. Starting from this requirement we can argue that in general upon performing a renormalization group transformation all possible couplings between interconnected spins might be generated. This is because whenever a summation over certain spins on a lattice is performed all of the remaining spins that were once connected to that very spin must now be statistically coupled to each other so as to represent their original statistical dependence. Therefore we must allow for all possible generated couplings between arbitrary domains of spins. Consequently a flow of coupling parameters or a mapping $\{K\} \to \{K'\} \to ...$ will be obtained when successively coarse graining the system. Besides these newly generated couplings we also allow the renormalized Hamiltonian to contain a spin independent term as this merely boils down to a shift of the energy levels. This spin independent term can thus be taken outside the configurational sum of equation (39) and so translates into a constant contribution to the free energy. As our most general expression for the renormalized Hamiltonian, assuming our original system contains N degrees of freedom, we therefore take [10]:

$$\mathcal{H}'(\{\sigma_i'\}) = Ng(K) + \sum_\alpha K'_\alpha S_\alpha, \tag{40}$$

where $g(K)$ represents the spin independent term. It can be thought of as a trivial field added to the bond action between the two interacting spins before coarse-graining. This can always be done. As was pointed out earlier, we allow for the most general type of interactions between domain of spins and this is what K'_α, a coupling constant, and S_α account for:

$$S_\alpha = \prod_{i \epsilon I_\alpha} \sigma_i,$$

where I_α represents a domain of spins labelled by α and σ_i are the spins within this domain. Although the general idea has now been outlined, a renormalization group transformations always has to be adjusted to the problem at hand. Therefore, we shall next look at such a problem.

Example: 1D-Ising model

A simple model in which these fairly abstract ideas can be made more apparent is the interacting 1D-Ising model [11]. It is already known from other accounts that this does not exhibit a true phase transition. Nevertheless, it is still instructive to see this same result come out explicitly from the perspective of the renormalization group as well. The 1D-Ising model shall also be of considerable importance in the next chapter, in which we discuss the connection between RG and neural networks. Therefore, it is necessary to study its behaviour from a mere RG perspective first.

Let's start off by stating the usual Hamiltonian for the interacting 1D-Ising model:

$$\mathcal{H}_a = -J_a \sum_i \sigma_i \sigma_{i+1}, \tag{41}$$

where the subscript a denotes again the resolution at which the alignment of spins is viewed. As argued above, performing a RG transformation is about adopting a coarse-grained set of degrees of freedom without changing the large distance physics. Therefore, we shall integrate out every other spin while leaving the partition function $\mathcal{Z}$ untouched:

$$\mathcal{Z} = \sum_{\{\sigma_i\}} e^{-\beta \mathcal{H}_a \{\sigma_i\}} = \sum_{\{\sigma_i\}} e^{K \sum_i \sigma_i \sigma_{i+1}} = \sum_{\{\sigma_i\}} e^{\sum_i w(\sigma_i, \sigma_{i+1})},$$

where we have defined $K \equiv \beta J$ and denoted the local bond action by $w(\sigma_i, \sigma_{i+1}) \equiv K\sigma_i\sigma_{i+1}$ in order to make the connection to the renormalized bond action, which will be derived shortly, more prominent.

$$\begin{aligned}
\mathcal{Z} &= \sum_{\{\sigma_i\}} \left[e^{w(\sigma_1,\sigma_2)} e^{w(\sigma_2,\sigma_3)}\right] \times \left[e^{w(\sigma_3\sigma_4)} e^{w(\sigma_4,\sigma_5)}\right] ... \\
&= \sum_{\{\sigma'_i\}} \sum_{\sigma_2=\pm1,\sigma_4=\pm1,...} \left[e^{w(\sigma_1,\sigma_2)} e^{w(\sigma_2,\sigma_3)}\right] \times \left[e^{w(\sigma_3\sigma_4)} e^{w(\sigma_4,\sigma_5)}\right] ... \\
&= \sum_{\{\sigma'_i\}} \sum_{\sigma_2=\pm1} \left[e^{w(\sigma_1,\sigma_2)} e^{w(\sigma_2,\sigma_3)}\right] \sum_{\sigma_4=\pm1} \left[e^{w(\sigma_3\sigma_4)} e^{w(\sigma_4,\sigma_5)}\right] ... \\
&\equiv \sum_{\{\sigma'_i\}} e^{w'(\sigma_1,\sigma_3)} e^{w'(\sigma_3,\sigma_5)} ...
\end{aligned} \tag{42}$$

By symmetry we only need to study how the local bond action $w(\sigma_i, \sigma_{i+1})$ gets renormalised:

$$e^{w'(\sigma_i,\sigma_{i+2})} = \sum_{\sigma_{i+1}=\pm1} \left[e^{w(\sigma_i,\sigma_{i+1})} e^{w(\sigma_{i+1},\sigma_{i+2})}\right]. \tag{43}$$

In order to make progress we notice that the following holds for $\sigma_i \in \{-1, 1\}$:

$$\begin{aligned}
e^{w(\sigma_i,\sigma_{i+1})} = e^{K\sigma_i\sigma_{i+1}} &= \cosh(K) + \sigma_i\sigma_{i+1}\sinh(K) \\
&= \cosh(K)[1 + \sigma_i\sigma_{i+1}\tanh(K)].
\end{aligned} \tag{44}$$

As argued above, we allow for a trivial field K_0 so as to be able to write w' in the same form as w:

$$e^{w(\sigma_i,\sigma_{i+1})} = e^{K_0} e^{K\sigma_i\sigma_{i+1}}. \tag{45}$$

By plugging equation (44) into (43) we find:

$$\begin{aligned}
e^{w'(\sigma_i,\sigma_{i+2})} &= \sum_{\sigma_{i+1}=\pm1} e^{2K_0}\cosh^2(K)[1 + \sigma_i\sigma_{i+1}\tanh(K)][1 + \sigma_{i+1}\sigma_{i+2}\tanh(K)] \\
&= 2e^{2K_0}\cosh^2(K)[1 + \sigma_i\sigma_{i+2}\tanh^2(K)].
\end{aligned} \tag{46}$$

On the other hand w' must be written in the same form as w, so we also have:

$$e^{w'(\sigma_i,\sigma_{i+2})} = e^{K'_0}\cosh(K')[1 + \sigma_i\sigma_{i+2}\tan(K')]. \tag{47}$$

By equating equation (46) to equation (47) we are forced to conclude:

$$\tanh(K') = \tanh^2(K), \tag{48}$$

for the ferromagnetic coupling, and

$$K'_0 = \ln\left[\frac{2e^{2K_0}\cosh^2(K)}{\cosh(K')}\right], \tag{49}$$

for the trivial field. This tells us how the couplings K get renormalised into new couplings K' while the system gets coarse-grained into larger length scales. Note that the only two fixed points of this recursive equation correspond to $K = 0$ and $K = \infty$, for only then gets K mapped onto itself. All other values of K flow towards the stable fixed point, $K = 0$, which is equivalent to the high temperature phase (figure (2)), or the paramagnetic phase. So we see from this little analysis that the 1D-Ising model is in a paramagnetic phase at all temperatures, except for $T = 0$, which corresponds to the critical point of the 1D-Ising model.

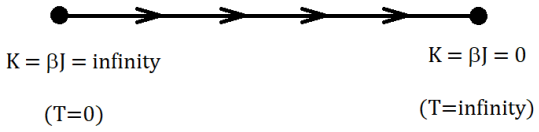

Figure 2: RG flow for the 1D-Ising model.

4 Connection between RG and Neural Networks

The potential existence of a connection between RG and deep learning got first hinted at by Mehta and Schwab [2]. Since then, this subject has been studied in greater detail [3, 12]. Especially, S.Iso, S.Shiba, and S.Yokoo [3] used a RBM applied to a two-dimensional Ising model so as to generate a flow of temperatures which could subsequently be compared to the flow that one would obtain within the RG-framework. Surprisingly though, their findings pointed at an exact opposite behaviour; the RBM constructed a flow towards the critical temperature T_c, whereas in RG the flow would be directed away from the critical point. Their conclusions were largely based on numerical findings. Although such an approach can initially be quite insightful and helpful to obtain concrete results, it does not give one a real insight into the analytical structure of the very connection between RG and neural networks. For this reason, in this section we shall develop an analytical approach that will help us understand the connection between RG and neural networks from a more mathematical perspective. But first, let us pause to elaborate on the methods used by the researchers in Ref [3].

The general pattern followed in their paper [3] goes as follows.

1. They first generate, using the Metropolis algorithm, a large set of configurations of the 2D-Ising model at temperatures ranging within a discrete set $T = \{0, 0.25, 0.5, ..., 6\}$. At each distinct temperature T in this set they generate 1000 spin configurations, adding up to a total of 25000 configurations.

2. These configurations then serve as input data to train a RBM. This is done through the usual methods as described in section (2.4).

3. After the RBM has been successfully trained, the set of weights W_{ia} do no longer get updated and are thus fixed. Now, they generate a completely new configuration $\{v_i^{(0)}\}$

over the visible units at some specific temperature T^*. It is important to note that $\{v_i^{(0)}\}$ has not been part of the training set that was used previously to train the RBM.

4. Next, they assign $\{v_i^{(0)}\}$ to the visible layer in the RBM, each spin spin serving as an input to one visible unit. In the background of this visible layer they compute the average hidden layer, using equation (33), in which the a^{th} hidden unit is given by

$$h_a^{(0)} = \tanh\left(\sum_i W_{ia} v_i^{(0)}\right), \tag{50}$$

where W_{ia} is the trained weight found in step (2), connecting the i^{th} visible unit to the a^{th} hidden unit. By doing this for all a's in the hidden layer the entire set $\{h_a^{(0)}\}$ gets constructed.

5. In the background of $\{h_a^{(0)}\}$ they essentially repeat the same computation as in step (4) to find $\{v_i^{(1)}\}$, where an individual $v_i^{(1)}$ is given by

$$v_i^{(1)} = \tanh\left(\sum_i W_{ia} h_a^{(0)}\right). \tag{51}$$

By plugging $v_i^{(1)}$ back into equation (50) they repeat steps (4) and (5) to obtain $v_i^{(2)}$. In this way, i.e. repeatedly going over steps (4) and (5), they obtain a flow of configurations. This flow is called the RBM-flow.

6. Having computed the flow $\{v_i^{(0)}\} \to \{h_a^{(1)}\} \to \{v_i^{(1)}\} \to \{h_a^{(2)}\} \to \cdots$ starting from one particular sample $\{v_i^{(0)}\}$ over the visible units at temperature T^*, the next step is to convert this flow of configurations to a flow of model parameters, i.e. temperature.

7. This is done through a multilayer-perceptron (MLP). This is an additionally trained supervised neural network on top of the already trained RBM that takes the k^{th} configuration $\{v_i^{(k)}\}$ within the flow to measure its associated temperature T. The MLP has been trained by the same training set that was used in step 2 to train the RBM. However, in the case of the MLP the training set also has associated target values (i.e. temperatures). Hence, the flow of configurations $\{v_i^{(0)}\} \to \{v_i^{(1)}\} \to \{v_i^{(2)}\} \to \cdots$ gets converted to $T^* \to T' \to T'' \to ...$ by the MLP.

Coming back to our comment expressing our wish to develop an analytical approach as opposed to a numerical one, we will apply the same basic steps outlined above to the 1D-Ising model instead. This has the big advantage that by so doing we don't have to train a RBM numerically, since the 1D-Ising model is an exactly solvable system[3]. As we shall derive in the upcoming section, the set of weights W_{ia} that minimises the $KL(q||p)$, with q the environmental distribution and p the model distribution, can directly be obtained as

[3]One could argue that the same holds for the 2D-Ising model, since this was exactly solved by L.Onsager. Nonetheless, the computations resulting from this are still analytically hard to analyse. Therefore, we shall stick to the 1D-Ising model for now.

a function of the couplings J in the one-dimensional case. These correspond to the set of weights W_{ia} that one otherwise would have obtained when training the RBM through the usual methods as described in section (2.4). However, by virtue of the relation $W(J)$ we circumvent the problem of having to resort to numerical computations.

4.1 Relating the Weights W to the Couplings J

Motivation

In order to compute the the transition from a given configuration over the visible units $\{v_i^{(k)}\}$ to the next configuration $\{v_i^{(k+1)}\}$ in the flow, we need an expression for W_{ia}. Otherwise the flow as dictated by equations (50) and (51) cannot be computed.

First, note that the distribution $q(\{v_i\})$ underlying the training set treats every unit v_i the same. All the training examples are samples from a 1D-Ising model with periodic boundary conditions after all. Any unit is as likely to be active as any other unit. Hence, all visible units are equivalent to one another and thus all the weights W_{ia} equal the same constant W by symmetry. The fact that $W_{ia} = W$ will tremendously simplify our computations that follow and allows us to derive an analytical expression that could not have been derived otherwise. Also, the distribution $p(\{v_i\})$ in equation (27) simplifies now that we can set $W_{ia} = W$:

$$p(\{v_i\}) = \frac{1}{Z} \sum_{\{h_a\}} \exp\left[W \sum_{i,a} v_i h_a \right], \tag{52}$$

where the external magnetic field, or the bias, has been set to zero for it would otherwise unnecessarily complicate our computations. The partition function Z is obtained by summing the Boltzmann weight over all possible configurations of the network:

$$Z = \sum_{\{v_i, h_a\}} \exp\left[W \sum_{i,a} v_i h_a \right]. \tag{53}$$

Next, the general idea that one should have mind is that the environmental distribution $q(\{v_i\})$ is linked to the model distribution $p(\{v_i\})$ via the KL-divergence $KL(q||p)$. As always with training, the aim is to tune W such that $KL(q||p)$ is minimised. Since our training set involves samples of the 1D-Ising model drawn from $q(\{v_i\})$ that depend on the couplings J, and the model distribution $p(\{v_i\})$ depends on the weights W through equation (52), the weights W can be expressed in terms of the couplings J by enforcing the minimisation requirement of $KL(q||p)$. Therefore, from now on until the end of this section, we shall be concerned with minimising $KL(q||p)$ with respect to W.

The derivation of W(J)

Recall,

$$KL(q||p) = \sum_{\{v_i\}} q(\{v_i\}) \ln \frac{q(\{v_i\})}{p(\{v_i\})}.$$

Since $q(\{v_i\})$ is independent of W,

$$\begin{aligned}\frac{\partial KL(q||p)}{\partial W} &= -\sum_{\{v_i\}} \frac{\partial}{\partial W}\left[q(\{v_i\})\ln p(\{v_i\})\right] \\ &= -\sum_{\{v_i\}} \frac{q(\{v_i\})}{p(\{v_i\})}\frac{\partial p(\{v_i\})}{\partial W}.\end{aligned} \tag{54}$$

In order to make progress, we next consider:

$$\begin{aligned}\frac{\partial p(\{v_i\})}{\partial W} &= \frac{\partial}{\partial W}\left[\frac{1}{Z}\sum_{\{h_a\}}\exp\left(W\sum_{i,a}v_i h_a\right)\right] \\ &= -\frac{\sum_{\{h_a\}}\exp\left(W\sum_{i,a}v_i h_a\right)}{Z^2}\frac{dZ}{dW} + \frac{1}{Z}\sum_{\{h_a\}}\left(\sum_{i,a}v_i h_a\right)\exp\left(W\sum_{i,a}v_i h_a\right).\end{aligned} \tag{55}$$

The last term on the far right hand side contributes to equation (54) via a term

$$-\sum_{\{v_i\}}\frac{q(\{v_i\})}{\sum_{\{h_a\}}\exp\left(W\sum_{i,a}v_i h_a\right)}\cdot\left(\sum_i v_i\right)\sum_{\{h_a\}}\left(\sum_a h_a\right)\exp\left(W\sum_{i,a}v_i h_a\right),$$

where Z has conveniently cancelled and the magnetization of the visible units has been pulled outside the configurational sum over the hidden units. To proceed, note that the following sum factorises over the N hidden units:

$$\begin{aligned}&\sum_{\{h_a\}}\exp\left(W\sum_{i,a}v_i h_a\right) = \sum_{\{h_a\}}\exp\left[W\left(\sum_i v_i\right)(h_1+\cdots+h_N)\right] \\ &= \left(\sum_{h_1=\pm1}\exp\left[W\left(\sum_i v_i\right)h_1\right]\right)\cdots\left(\sum_{h_N=\pm1}\exp\left[W\left(\sum_i v_i\right)h_N\right]\right) \\ &= \prod_{i=1}^{N}\left(\sum_{h_i=\pm1}\exp\left[W\left(\sum_i v_i\right)h_i\right]\right) \\ &= \prod_{i=1}^{N}2\cosh\left(W\sum_i v_i\right) \\ &= \left[2\cosh\left(W\sum_i v_i\right)\right]^N.\end{aligned} \tag{56}$$

Rewriting this result with $V \equiv \sum_i v_i$, gives a nice intermediate result:

$$\sum_{\{h_a\}} \exp\left(W \sum_{i,a} v_i h_a\right) = \left[2\cosh\left(WV\right)\right]^N . \tag{57}$$

From equation (57) we can easily derive the closely related quantity

$$\begin{aligned}
&\sum_{\{h_a\}} \left(\sum_a h_a\right) \exp\left[W \sum_{i,a} v_i h_a\right] = \\
&\sum_{\{h_a\}} \left(\sum_a h_a\right) \exp\left[WV \sum_a h_a\right] = \\
&\frac{\partial}{\partial(VW)} \sum_{\{h_a\}} \exp\left[VW \sum_a h_a\right] = \\
&\frac{\partial}{\partial(VW)} \left[2\cosh\left(WV\right)\right]^N = \\
&N\left[2\cosh(VW)\right]^N \tanh(VW).
\end{aligned} \tag{58}$$

Plugging equations (57) and (58) into (56) gives:

$$-N \sum_{\{v_i\}} q(\{v_i\}) V \tanh(VW).$$

Lastly, we still need to evaluate the first term on the right hand side of equation (55):

$$\begin{aligned}
&-\frac{\sum_{\{h_a\}} \exp\left[W \sum_{i,a} v_i h_a\right]}{Z} \cdot \frac{1}{Z}\frac{dZ}{dW} = \\
&-p(\{v_i\}) \frac{d\ln Z}{dW}.
\end{aligned} \tag{59}$$

Collecting all derived quantities in equation (55) and substituting these into (54) gives:

$$\frac{\partial KL(q||p)}{\partial W} = -\sum_{\{v_i\}} \frac{q(\{v_i\})}{p(\{v_i\})} \left(-p(\{v_i\}) \frac{d\ln Z}{dW}\right) - N \sum_{\{v_i\}} q(\{v_i\}) V \tanh(VW). \tag{60}$$

Cancelling common terms and pulling the derivative of $\ln Z$ outside the summation, since it doesn't depend on the visible units, we arrive at:

$$\frac{\partial KL(q||p)}{\partial W} = \frac{d\ln Z}{dW} \sum_{\{v_i\}} q(\{v_i\}) - N \sum_{v_i} q(\{v_i\}) V \tanh(VW). \tag{61}$$

Because the environmental distribution $q(\{v_i\})$ is normalised to unity, we can write the final result as:

$$\frac{\partial KL(q||p)}{\partial W} = \frac{d\ln Z}{dW} - N \sum_{v_i} q(\{v_i\}) V \tanh(VW). \tag{62}$$

The goal of training is to minimise $KL(q||p)$ with respect to W. Therefore, we must set the last equation to zero and solve for W:

$$\frac{d\ln Z}{dW} = N\sum_{\{v_i\}} q(\{v_i\})V\tanh(VW). \tag{63}$$

The information about the couplings J is contained in $q(\{v_i\})$. In order to see how it can be extracted, we look at a simple example in which our training set solely contains Ising configurations generated at one single temperature T^*. This makes $q(\{v_i\})$ to simply correspond to the Boltzmann-distribution, as we shall now show. In general we allow for a training set that contains samples at various different temperatures, starting at $T = 0$ up to T_0:

$$\begin{aligned} q(\{v_i\}) &= \int_0^{T_0} P(\{v_i\}, T)dT \\ &= \int_0^{T_0} P(\{v_i\}|T)P(T)dT. \end{aligned} \tag{64}$$

In our simple examples, however, we now take $P(T) = \delta(T - T^*)$, since all training samples have been generated at the same single temperature T^*. Therefore the environmental distribution simplifies to

$$q(\{v_i\}) = P(\{v_i\}|T^*), \tag{65}$$

by the sifting property of the dirac-delta function. But this is nothing more than the ordinary Boltzmann distribution, so:

$$q(\{v_i\}) = \frac{\exp\left(-\beta\mathcal{H}(\{v_i\})\right)}{\mathcal{Z}_{Ising}}, \tag{66}$$

with $\mathcal{H}$ the Hamiltonian of the interacting 1D-Ising model with periodic boundary conditions, and $\mathcal{Z}_{Ising}$ its associated partition function:

$$\mathcal{H}(\{v_i\}) = -J\sum_{i=1}^{M} v_i v_{i+1}, \tag{67}$$

$$\mathcal{Z}_{Ising} = (2\cosh(K))^M + (2\sinh(K))^M, \tag{68}$$

where M corresponds to the number of visible units, and $K = \beta J$.

Plugging equation (66) into equation (63) gives an expression relating W to J at temperature T^*, for this specific example:

$$\frac{d\ln Z}{dW} = \frac{N\sum_{\{v_i\}}\exp\left(\beta J\sum_i^M v_i v_{i+1}\right)V\tanh(VW)}{(2\cosh(\beta J))^M + (2\sinh(\beta J)))^M}. \tag{69}$$

Note that this is an expression relating W to J, since all the other variables are summed over. The partition function Z must, however, still be evaluated. To this end, we manipulate equation (53) as follows:

$$Z = \sum_{\{v_i,h_a\}} \exp\left[W \sum_{i,a} v_i h_a\right]$$
$$= \sum_{\{v_i,h_a\}} \exp\left[W \sum_{i} v_i \sum_{a} h_a\right].$$

Defining for the sake of notation H$\equiv \sum_a h_a$, we get

$$Z = \sum_{\{v_i,h_a\}} \exp\left[WH(v_1 + \cdots v_N)\right].$$

Factorising the exponential gives

$$Z = \sum_{\{h_a\}} \left(\sum_{v_1=\pm 1} \exp\left[v_1 WH\right]\right) \cdots \left(\sum_{v_M=\pm 1} \exp\left[v_M WH\right]\right)$$
$$= \sum_{\{h_a\}} \prod_{i=1}^{M} \left(\sum_{v_i=\pm 1} \exp\left[v_i WH\right]\right)$$
$$= \sum_{\{h_a\}} \left(2\cosh(WH)\right)^M = 2^M \sum_{\{h_a\}} \cosh(WH)^M. \tag{70}$$

Because the hidden units come in only through the magnetization H of the hidden layer, terms in the summation giving rise to the same magnetization H can be collected and multiplied by the degeneracy of H. Furthermore, because the summand is even in H, terms contributing by $-H$ can be grouped under terms contributing by $+H$.

Let $d(H)$ denote the degeneracy or, equivalently, the number of ways N_+ active units can be arranged amongst a total of N hidden units. Then equation (70) can be expanded out as:

$$\frac{Z}{2^M} = d(0) + 2d(2)\cosh^M(2W) + \cdots$$
$$+ 2d(N-2)\cosh^M((N-2)W) + 2d(N)\cosh^M(NW). \tag{71}$$

Note that the magnetization H can only change in steps of two, since flipping a unit's state from active to inactive, or vice versa, alters H by a value of two. Here, we have assumed an even number of hidden units. For the odd-case, which we shall not further consider here, the term $d(0)$ drops out and we only get odd factors of W.

The degeneracy $d(H)$ is given by the binomial coefficient $\binom{N}{N_+}$, where N_+ is the number of active hidden units giving rise to a magnetization H. Equivalently, $d(H)$ can also be expressed only in terms of N and H via

$$d(H) = \frac{N!}{(\frac{N+H}{2})!(\frac{N-H}{2})!}. \tag{72}$$

With all terms in equation (71) well defined, we can now condense it into a shorter notation:

$$Z = 2^M d(0) + 2^{M+1} \sum_{n=1}^{N/2} d(2n) \cosh^M(2nW). \tag{73}$$

Going back to equation (69), which we have copied again down here for convenience, we notice that only the configurational sum over the visible units still has to be carried out:

$$\frac{d \ln Z}{dW} = \frac{N \sum_{\{v_i\}} \exp\left(\beta J \sum_i^M v_i v_{i+1}\right) V \tanh(VW)}{(2\cosh(\beta J))^M + (2\sinh(\beta J)))^M}.$$

However, this time we cannot evaluate this sum without having to resort to numerical methods. In want of a closed form expression, we have used *Mathematica* to explicitly evaluate this term for certain specific cases.

Equation (69) is of the form

$$f(W) = g(W, K), \tag{74}$$

where $K = \beta J$. At a fixed temperature T, changes in K only amount to associated changes in J. Therefore, from now on, we shall use K instead of J. The inverse temperature β and J always occur in pairs after all. By letting K run within a certain range $[K_{min}, K_{max}]$, and numerically solving (74) for W, we build up information about $W(K)$.

For illustrative purposes, we shall show how we find W for a particular $K \in [K_{min}, K_{max}]$. Plotting both $f(W)$ and $g(W, K)$ together in one plot against W, allows us to graphically find the value of W that solves equation (69) for this particular K. In figure (3) you can see the result where we have taken $K = 1$. The architecture that we used contained $M = 8$ visible units, and $M/2$ hidden units. Reading off the point of intersection in figure (3) gives

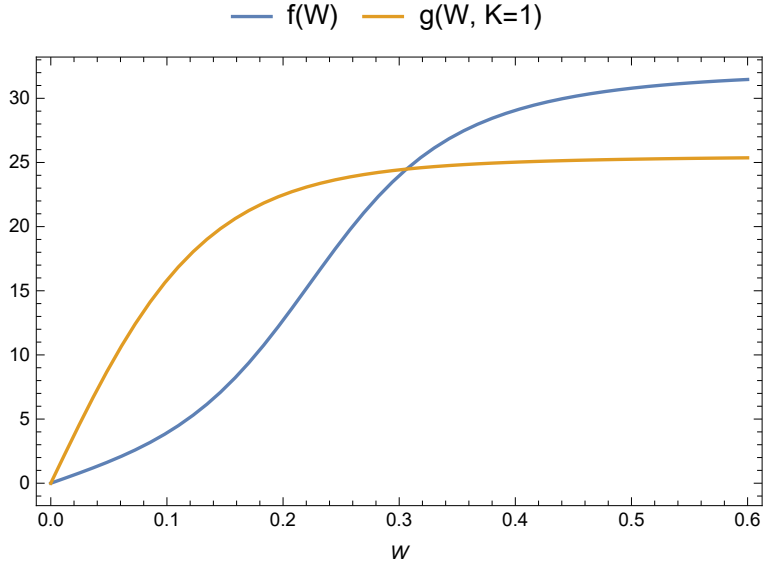

Figure 3: Graphical solution to equation (69), for $K = 1$.

$W = 0.30$. To put it differently, if a training set consists of samples of the 1D-Ising model with $K = 1$, the RBM is learned when we set $W = 0.30$.

Repeating the same procedure for all other values of K within $[K_{min}, K_{max}]$, we get the result shown in figure (4). Note that the previously found solution $W = 0.30$ indeed corresponds to $K = \beta J = 1.0$.

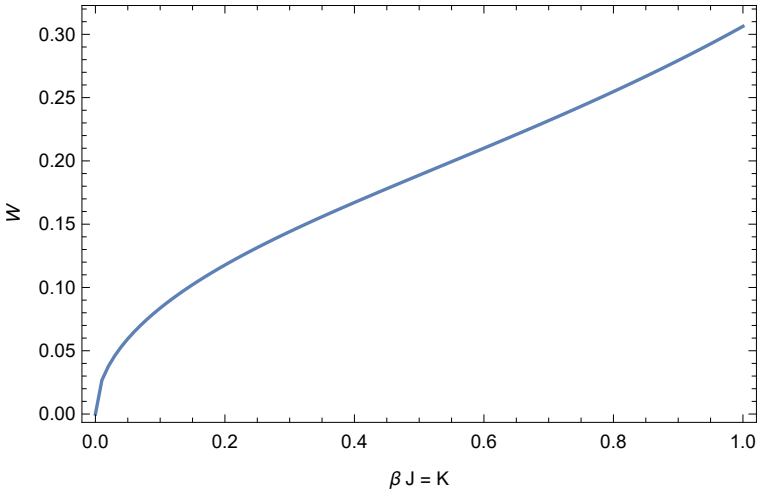

Figure 4: Weights W as a function of βJ, with $M = 8$ and $N = 4$.

A few interesting limits of W(K)

Now that we have found a method of determining $W(K)$ for the specific architecture $M = 8$ and $N = M/2$, we can study a few interesting limits. Firstly, note that at $\beta J = 0$ the corresponding weight W is zero. This is also what we would expect intuitively, because the Ising model is completely disordered at zero coupling (or equivalently, in the high temperature limit). Therefore, the RBM should reproduce the micro-canonical ensemble; every unit in the RBM must have an equal probability of $1/2$ to update its state. This can only be realised when the weights W are all set to zero, since this gives rise to zero activity and thus $P(\pm 1) = \sigma(a = 0) = 1/2$, with a the activity of a unit.

Secondly, from figure (4) we deduce that $W(K)$ looks like a monotonically increasing function of K. We would like to make this observation more explicit by actually proving that as K tends to infinity, the weight W gets arbitrarily large as well. To be more specific, consider the three consecutive pictures in figure (5) corresponding to an increasing value of K from left to right. The figure suggests that at infinitely large K, the trained W tends to infinity as well. In other words, as the temperature T approaches the critical point $T_c = 0$, the weight W diverges. We will come back to this important observation later on when discussing the flow generated by the trained RBM. It will in fact turn out to carry very rich consequences. But first, we refer to the Appendix for an explicit proof of the diverging behaviour of W.

The fact that W diverges as K tends to infinity can also be understood from a physical perspective. To see this, note that the high coupling phase is equivalent to the low-temperature phase, in which the 1D-Ising model is completely ordered. Therefore, the probability of a

unit in the RBM adopting a particular state must have converged to one as the couplings approached infinity. This can only be true when the weights W and thus the activity a grow infinitely large. Recall $\lim_{a\to\infty} P(\pm 1) = \lim_{a\to\infty} \sigma(a) = 1$, as required in the fully ordered phase at zero temperature.

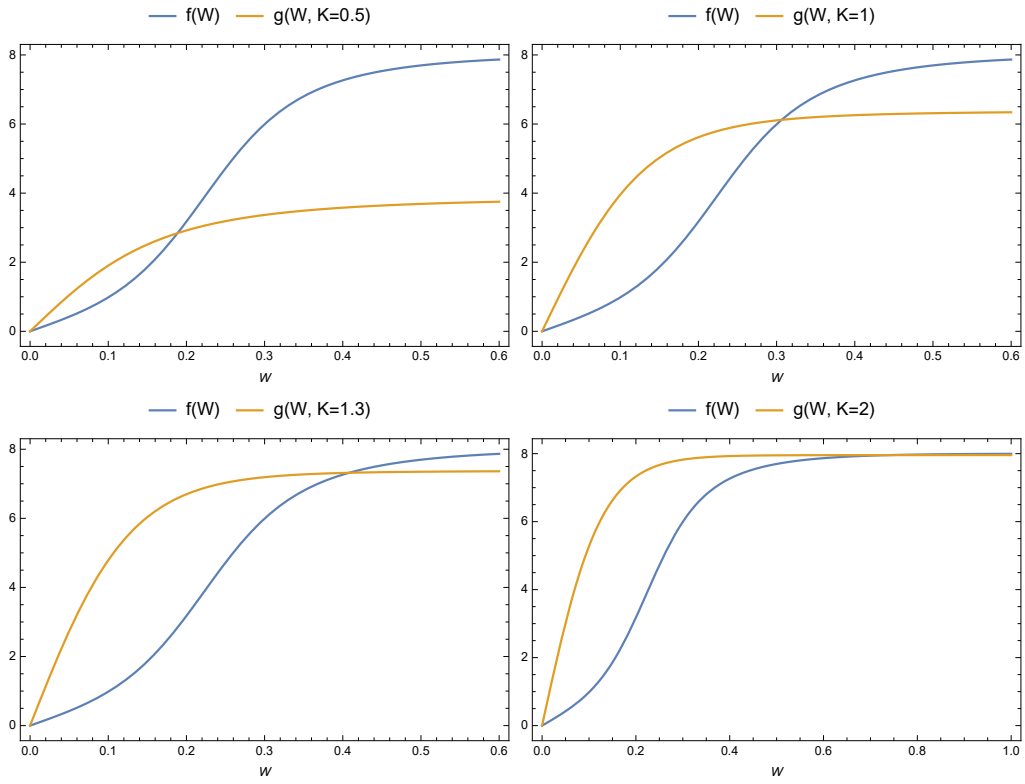

Figure 5: The solution W to equation (69) diverges to infinity as $K = \beta J$ is increased. The architecture that was used is $M = 8$ and $N = 4$.

4.2 RBM Flow towards the Critical Point

Before moving on we first briefly recapitulate the results we have obtained so far. We generated a training set characterised by the environmental distribution $q(\{v_i\})$. It consisted of samples of the 1D-Ising model generated at one single temperature T and coupling J. The relation $W(K)$ allowed us to directly obtain the trained weight W given our training set. Especially, we saw then when the training set was solely made up of samples at the critical temperature $T_c = 0$, the trained weight W diverged.

Having outlined our results so far, we now consider their consequences for the RBM-flow in step (6) at the beginning of this chapter. As we are dealing with equal weights, i.e. $W_{ia} = W$, the flow as stated in equations (50) and (50) simplifies to, again neglecting any external magnetic field or bias term:

$$h_a^{(k+1)} = \tanh\left(W \sum_i v_i^{(k)}\right), \tag{75}$$

$$v_i^{(k+1)} = \tanh\left(W \sum_a h_a^{(k+1)}\right). \tag{76}$$

Now, letting our training set consist of samples of the 1D-Ising model at $T = T_c = 0$, we know that in this case $W \to \infty$. As a result the flow as stated in equation (75) simplifies even further when applied to this particular training set:

$$h_a^{(k+1)} = \lim_{W\to\infty} \tanh\left(W \sum_i v_i^{(k)}\right) = \begin{cases} 1 & \text{if } \sum_i v_i^{(k)} > 0 \\ -1 & \text{if } \sum_i v_i^{(k)} < 0 \\ 0 & \text{otherwise} \end{cases}$$

Equation (76) therefore results in:

$$v_i^{(k+1)} = \lim_{W\to\infty} \tanh\left(W \sum_a h_a^{(k+1)}\right) = \begin{cases} 1 & \text{if } \sum_i v_i^{(k)} > 0 \\ -1 & \text{if } \sum_i v_i^{(k)} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{77}$$

In other words, the flow (originating from a RBM that was trained with samples at $T = 0$) takes any arbitrary configuration $\{v_i^{(k)}\}$ that's not been used for training to a completely aligned configuration, depending on the magnetization of the initial configuration in the flow. If we imagine starting off with an ensemble of configurations all generated at a non-zero temperature $T \neq 0$, then according to equation (77) this whole set of configurations will get mapped onto a set of perfectly aligned spins. Since the whole ensemble at $T \neq 0$ flows towards perfect ordering, no matter how large the ensemble, we conclude that the RBM-flow goes from $T \neq 0$ to the critical temperature $T_c = 0$. Note that this corresponds to the numerical findings of the researchers in Ref. [3] in the 2D-case.

Consider next a training set that only contains samples at some non-zero temperature T. In that case the flow will not take a configuration $\{v_i\}$ to the critical point $T_c = 0$. This can be seen as follows. Recall that the trained weights W corresponding to a training set at non-zero temperature remain finite. As a result, equations (50) and (51) do not take an arbitrary spin $v_i^{(k)}$ to either an up-state or a down-state with 100% certainty. Since a slightly disordered configuration does not correspond to $T_c = 0$, we conclude that the flow does not flow towards the critical point.

4.3 Training Sets with a Continuous Range of Temperatures

Note that we have restricted ourselves throughout this section to the rather special case in which $P(T) = \delta(T - T^*)$. In other words, all training examples were certain to have been generated at one single temperature T^*. In the paper by S.Iso, S. Shiba, and S. Yokoo [3], however, the training set contained spin configurations corresponding to different temperatures as well. Therefore, we should in fact generalise our results to a training distribution

containing samples drawn uniformly from a whole range of temperatures $[0, T_0]$, where we only consider training samples up to some finite temperature T_0. Going back to the point just before we made the restrictive assumption that the temperature distribution is sharply peaked around T^* in equation (64), we now get for the environmental distribution, assuming a uniform distribution for $P(T)$ between $[0, T_0]$:

$$\begin{aligned} q(\{v_i\}) &= \frac{1}{T_0}\int_0^{T_0} P(\{v_i\}|T)dT \\ &= \frac{1}{T_0}\int_0^{T_0} \frac{\exp(-\beta\mathcal{H}(\{v_i\}))}{\mathcal{Z}_{Ising}}dT \\ &= \frac{1}{T_0}\int_0^{T_0} \frac{\exp(K\sum_i v_i v_{i+1})}{(2\cosh(K))^M + (2\sinh(K))^M}dT. \end{aligned} \tag{78}$$

Plugging this equation back into equation (63) gives:

$$\begin{aligned} \frac{1}{N}\frac{d\ln Z}{dW} &= \frac{1}{T_0}\int_0^{T_0} \frac{\sum_{\{v_i\}}\exp\left(K\sum_i^M v_i v_{i+1}\right)V\tanh(VW)}{(2\cosh(K))^M + (2\sinh(K))^M}dT \\ &= \frac{1}{T_0}\sum_{\{v_i\}} V\tanh(VW)\int_0^{T_0} \frac{\exp\left(K\sum_i^M v_i v_{i+1}\right)}{(2\cosh(K))^M + (2\sinh(K))^M}dT. \end{aligned} \tag{79}$$

Since we are interested in the behaviour of the flow generated by a RBM which has been trained with samples close to and including the critical temperature $T_c = 0$, one should in future research perform a low temperature expansion of the integrand in equation (79) about $T = 0$.

5 Conclusions, Discussion, and Outlook

In this thesis we developed a connection between the renormalization group and neural networks. We specifically looked at the flow generated by a RBM which got trained by a training set consisting of 1D-Ising model snapshots at a single temperature T^* and periodic boundary conditions. By making use of the translational symmetry within this training set, we were able to derive an implicit analytical expression for the trained weights W as a function of the effective coupling $K = \beta J$ in the 1D-Ising model. The results obtained in chapter 4 pointed at the existence of a monotonically increasing relationship $W(K)$. Consequently, a training set at zero coupling resulted in trained weights W that were all equal to zero. This corresponded to the disordered phase. At infinite coupling, however, the trained weights of the RBM were found to diverge. We also constructed a flow generated by the trained RBM. In the case of a RBM trained at the critical temperature, the flow was found to go towards the critical temperature, i. e. an arbitrary 1D-Ising sample at finite temperature T flowed towards the critical point. This is consistent with the numerically obtained findings in Ref. [3] for the 2D-Ising model. However, this behaviour is exactly opposite to what was dictated by the RG-group in chapter 3. Nevertheless, the RBM assigns special meaning to the critical point. This begs the question of which special features the samples of the 1D-Ising at the critical temperature posses compared to samples at non-zero temperature.

From a more practical point of view, it would in fact be more useful if a RBM generated a flow that takes samples away from the critical point, since the degree of noise in an image should be suppressed rather than increased in order to enhance the image's quality. In our case, however, the correlation length of noise, or fluctuations, increases along the RBM-flow.

In obtaining our results, we restricted ourselves to training sets that contained samples of the 1D-Ising at one single temperature. In future research it would, however, be interesting to consider a training set which represents a continuous range of temperatures instead. It is important to develop results for two such different temperature ranges. One in which the critical temperature is included within the range, and another one in which the critical temperature is excluded. Because the critical point has zero measure within the continuous temperature range, we do in fact expect both training sets to generate similar RBM-flows. Whether these flow towards or away from the critical point still needs be found out though.

References

[1] J. Cardy, P. Goddard, and J. Yeomans, *Scaling and Renormalization in Statistical Physics*, Cambridge Lecture Notes in Physics (Cambridge University Press, 1996), ISBN 9780521499590, URL `https://books.google.nl/books?id=Wt804S9FjyAC`.

[2] P. Mehta and D. J. Schwab, ArXiv e-prints (2014), `1410.3831`.

[3] S. Iso, S. Shiba, and S. Yokoo, **97**, 053304 (2018), `1801.07172`.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, 2006), ISBN 978-0387-31073-2, URL `http://research.microsoft.com/en-us/um/people/cmbishop/prml/`.

[5] D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms* (Cambridge university press, 2003).

[6] G. Hinton, *Hopfield nets with hidden units*, URL `https://www.youtube.com/watch?v=vVEju0zMCaA&t=60s&list=PLoRl3Ht4JOcdU872GhiYWf6jwrk_SNhz9&index=53`.

[7] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, in *Readings in Computer Vision* (Elsevier, 1987), pp. 522–533.

[8] S. Roweis, Lecture notes (1995).

[9] A. Fischer and C. Igel, in *Iberoamerican Congress on Pattern Recognition* (Springer, 2012), pp. 14–36.

[10] K. Huang, *Statistical mechanics* (Wiley, 1987), ISBN 9780471815181, URL `https://books.google.nl/books?id=M8PvAAAAMAAJ`.

[11] L. D. Debbio, *The renormalisation group: general idea and application to the 1d ising model*, Blackboard Learn, University of Edinburgh (2017).

[12] H. W. Lin, M. Tegmark, and D. Rolnick, Journal of Statistical Physics **168**, 1223 (2017), `1608.08225`.

A The BM Learning Algorithm

In this short appendix we shall derive the learning algorithm for the Boltzmann machine, which was first referred to in section (2.3). The derivation follows closely that of G.Hinton's in his paper on learning algorithms for Boltzmann machines [7].

The setting is as follows. We have a Boltzmann machine with both visible and hidden units - their configurations denoted by $\{v_i\}$ and $\{h_a\}$ respectively. Since the Boltzmann machine forms a complete graph (so including mutual connections within the visible and hidden layer), we shall denote the weight connecting two arbitrary units x_i and x_j by w_{ij}. Given a training set, we would now like to find the set of weights that minimises the KL-divergence between $P'(\{v_i\})$ and $P(\{v_i\})$, with P' denoting the model distribution and P the environmental distribution:

$$KL(P(\{v_i\}||P'(\{v_i\}))) = \sum_{\{v_i\}} P(\{v_i\}) \ln \frac{P(\{v_i\})}{P'(\{v_i\})}. \tag{80}$$

Because $P(\{v_i\})$ is the environmental distribution, it does not depend on any learning parameter w_{ij}. Therefore, terms in equation (80) that only contain $P(\{v_i\})$ can be regarded constant and will vanish upon taking derivatives with respect to the weights w_{ij}. Consequently,

$$\frac{\partial KL}{\partial w_{ij}} = -\sum_{\{v_i\}} \frac{P(\{v_i\})}{P'(\{v_i\})} \frac{\partial P'(\{v_i\})}{\partial w_{ij}}. \tag{81}$$

In order to make progress we need to take a closer look at the model distribution P':

$$P'(\{v_i\}) = \sum_{\{h_a\}} P'(\{v_i\}, \{h_a\}) = \frac{\sum_{\{h_a\}} \exp\left[-E(\{v_i, h_a\})\right]}{\sum_{\{v_i, h_a\}} \exp\left[-E(\{v_i, h_a\})\right]}, \tag{82}$$

where we have used the Boltzmann distribution in which the energy $E(\{v_i, h_a\})$ of the network in the configuration $\{v_i, h_a\}$ is given by:

$$E(\{v_i, h_a\}) = -\sum_{i<j} w_{ij} x_i x_j, \tag{83}$$

where w_{ij} denotes the weight between the i^{th} and j^{th} unit, and the sum is over all connected pairs. Note that this includes connected pairs of either both visible or hidden units. In order to differentiate P' in equation (82) with respect to the weight w_{ij}, we first need:

$$\frac{\partial \exp\left[-E(\{v_i, h_a\})\right]}{\partial w_{ij}} = x_i x_j e^{-E(\{v_i, h_a\})}. \tag{84}$$

Hence, differentiating equation (82) with respect to w_{ij} gives, using (84):

$$\frac{\partial P'(\{v_i\})}{\partial w_{ij}} = \frac{\sum_{\{h_a\}} \exp[-E(\{v_i, h_a\})] x_i x_j}{\sum_{\{v_i,h_a\}} \exp[-E(\{v_i, h_a\})]} - \frac{\sum_{\{h_a\}} \exp[-E(\{v_i, h_a\})] \sum_{\{v_i,h_a\}} \exp[-E(\{v_i, h_a\})] x_i x_j}{(\sum_{\{v_i,h_a\}} \exp[-E(\{v_i, h_a\})])^2}.$$

Or, writing the last line in terms of probabilities:

$$\frac{\partial P'(\{v_i\})}{\partial w_{ij}} = \sum_{\{h_a\}} P'(\{v_i, h_a\}) x_i x_j - P'(\{v_i\}) \sum_{\{v_i,h_a\}} P'(\{v_i, h_a\}) x_i x_j.$$

Substituting this into equation (81), we get:

$$\frac{\partial KL}{\partial w_{ij}} = -\sum_{\{v_i\}} \frac{P(\{v_i\})}{P'(\{v_i\})} \left[\sum_{\{h_a\}} P'(\{v_i, h_a\}) x_i x_j - P'(\{v_i\}) \sum_{\{v_i,h_a\}} P'(\{v_i, h_a\}) x_i x_j \right]. \tag{85}$$

Next, we use the product rule for probability distributions:

$$\begin{aligned} P(\{v_i, h_a\}) &= P(\{h_a\}|\{v_i\}) P(\{v_i\}), \\ P'(\{v_i, h_a\}) &= P'(\{h_a\}|\{v_i\}) P'(\{v_i\}). \end{aligned} \tag{86}$$

Furthermore, we realise:

$$P'(\{h_a\}|\{v_i\}) = P(\{h_a\}|\{v_i\}), \tag{87}$$

since the probability of a configuration over the hidden units, given some visible configuration, is the same in thermal equilibrium regardless of whether the visible units were clamped or arrived at $\{v_i\}$ by free-running. Combining this with the product rule (86), we obtain:

$$P'(\{v_i, h_a\}) \frac{P(\{v_i\}}{P'(\{v_i\})} = P(\{v_i, h_a\}). \tag{88}$$

Combining equation (88) with (85), results in:

$$\frac{\partial KL}{\partial w_{ij}} = -\sum_{\{v_i,h_a\}} P(\{v_i, h_a\}) x_i x_j + \sum_{\{v_i\}} P(\{v_i\}) \sum_{\{v_i,h_a\}} P'(\{v_i, h_a\}) x_i x_j. \tag{89}$$

Naturally, the distribution over the clamped visible units is normalized to unity:

$$\sum_{\{v_i\}} P(\{v_i\}) = 1.$$

Therefore, equation (89) becomes:

$$\frac{\partial KL}{\partial w_{ij}} = -\sum_{\{v_i,h_a\}} P(\{v_i, h_a\}) x_i x_j + \sum_{\{v_i,h_a\}} P'(\{v_i, h_a\}) x_i x_j. \tag{90}$$

Finally define the following two quantities:

$$\langle x_i x_j \rangle_{data} \equiv \sum_{\{v_i, h_a\}} P(\{v_i, h_a\}) x_i x_j, \tag{91}$$

and

$$\langle x_i x_j \rangle_{model} \equiv \sum_{\{v_i, h_a\}} P'(\{v_i, h_a\}) x_i x_j. \tag{92}$$

Substituting these two definitions into equation (90), we finally arrive at the gradient of the KL-divergence in simple form:

$$\frac{\partial KL}{\partial w_{ij}} = \langle x_i x_j \rangle_{model} - \langle x_i x_j \rangle_{data}. \tag{93}$$

This completes the derivation of the learning algorithm for the Boltzmann machine.

B Behaviour of the Intersection Point

Here we consider the limiting behaviour of the intersection point in figure (5) of the main-text. We shall prove that the intersection point diverges to infinity as the temperature approaches the critical point $T_c = 0$.

Let $f(W)$ and $g(W, K)$ be defined as in the main-text:

$$\begin{aligned} f(W) &= \frac{1}{N} \frac{d \ln Z}{dW}, \\ g(W, K) &= \frac{\sum_{\{v_i\}} \exp(K\alpha)\, V \tanh(VW)}{(2\cosh(K))^M + (2\sinh(K)))^M}, \end{aligned} \tag{94}$$

where the partition function Z of the RBM is given by

$$Z = 2^M d(0) + 2^{M+1} \sum_{n=1}^{N/2} d(2n) \cosh^M(2nW),$$

with $d(n)$ the degeneracy of the hidden layer, as defined by equation (72). Also, we have defined for notational convenience $\alpha \equiv \sum_i^M v_i v_{i+1}$ and $V \equiv \sum_i^M v_i$. Note that α is restricted to lie within the discrete set $\{-M, -(M+4), ..., (M+4), M\}$. Likewise, V takes on values within $\{-M, -(M+2), ..., M+2, M\}$.

We are interested in the limiting behaviour of $f(W)$ and $g(W, K)$ in the zero-temperature limit, which corresponds to the critical point or, equivalently, infinitely high K. Therefore, we first determine the limit of $g(WK)$ as K tends to infinity:

$$\lim_{K\to\infty} g(W, K) = \sum_{\{v_i\}} V \tanh(VW) \lim_{K\to\infty} \left(\frac{\exp(K\alpha)}{(2\cosh(K))^M + (2\sinh(K))^M} \right). \tag{95}$$

As the sum runs over all configurations over the visible units, α will change accordingly. We claim that only those configurations $\{v_i\}$ that give rise to $\alpha = M$ give a non-zero contribution to equation (95). To see this, consider the last factor in equation (95):

$$\lim_{K\to\infty} \frac{\exp(K\alpha)}{(2\cosh(K))^M + (2\sinh(K))^M} = \lim_{K\to\infty} \frac{e^{K\alpha}}{(e^K + e^{-K})^M + (e^K - e^{-K})^M} =$$
$$\lim_{K\to\infty} \left(\frac{1}{\exp(K(1-\alpha/M)) + \exp(-K(1+\alpha/M))} + \frac{1}{\exp(K(1-\alpha/M)) - \exp(-K(1+\alpha/M))} \right)^M.$$

Now let $-M \leq \alpha < M$, then the exponents are bounded by

$$0 < 1 - \alpha/M \leq 2$$
$$0 < 1 + \alpha/M \leq 2,$$

and therefore we get:

$$\lim_{K\to\infty} \frac{\exp(K\alpha)}{(2\cosh(K))^M + (2\sinh(K))^M} = 0. \tag{96}$$

However, when $\alpha = M$ we get a non-zero contribution to the sum in equation (95):

$$\lim_{K\to\infty} \frac{\exp(K\alpha)}{(2\cosh(K))^M + (2\sinh(K))^M} = 1/2. \tag{97}$$

The only two configurations $\{v_i\}$ that give rise to $\alpha = M$ are the ones with all their spins either pointing up or down. Because of this equation (95) simplifies to:

$$\lim_{K\to\infty} g(W,K) = 2M\tanh(MW)\cdot\frac{1}{2} = M\tanh(MW). \tag{98}$$

Now that we have taken the low temperature limit, we can take the limit of W going to infinity:

$$\lim_{\substack{K\to\infty \\ W\to\infty}} g(W,K) = \lim_{\substack{K\to\infty \\ W\to\infty}} M\tanh(MW) = M. \tag{99}$$

Next, we determine the asymptotic behaviour of $f(W)$ as W tends to infinity. We first write $f(W)$ as:

$$\begin{aligned} Nf(W) = \frac{d\ln Z}{dW} &= \frac{2^{M+1}\sum_{n=1}^{N/2} d(2n) M\cosh^{M-1}(2nW)\sinh(2nW)2n}{2^M d(0) + 2^{M+1}\sum_{n=1}^{N/2} d(2n)\cosh^M(2nW)}, \\ &= \frac{4M\sum_{n=1}^{N/2} d(2n)\cosh^M(2nW)\tanh(2nW)n}{d(0) + 2\sum_{n=1}^{N/2} d(2n)\cosh^M(2nW)}. \end{aligned} \tag{100}$$

Using $\lim_{W\to\infty}\tanh(2nW) = 1$, we get:

$$\lim_{W\to\infty} Nf(W) = \lim_{W\to\infty} \frac{4M\sum_{n=1}^{N/2} d(2n)\cosh^M(2nW)n}{d(0) + 2\sum_{n=1}^{N/2} d(2n)\cosh^M(2nW)}.$$

Dividing by $\cosh^M(NW)$, gives:

$$\lim_{W\to\infty} Nf(W) = \lim_{W\to\infty} \frac{4M\sum_{n=1}^{N/2} d(2n)\cosh^M(2nW)n\cosh^{-M}(NW)}{d(0)\cosh^{-M}(NW) + 2\sum_{n=1}^{N/2} d(2n)\cosh^M(2nW)\cosh^{-M}(NW)}.$$

Since $\cosh^M(2nW)/\cosh^M(NW)$ goes to zero for all $n \in \{1, 2, ..., N/2\}$ as W tends to infinity, except for when $n = N/2$, the sum simplifies to:

$$\lim_{W\to\infty} Nf(W) = \frac{4MNd(N)/2}{2d(N)} = MN. \tag{101}$$

Therefore, we conclude:

$$\lim_{W\to\infty} f(W) = M = \lim_{\substack{K\to\infty \\ W\to\infty}} g(W, K). \tag{102}$$

So both curves in figure (5) converge to the same asymptote.