UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Master Degree in Physics of Data

Final Dissertation

Explainable Machine Learning Applied to Proteins

Thesis supervisor

Prof. Marco Baiesi

Candidate

Lorenzo Rosset

Thesis advisors

Prof. Aurélien Decelle

Prof. Beatriz Seoane Bartolome

Academic Year 2021/2022

Contents

Introduction

The last decade witnessed a technological revolution in which Artificial Intelligence (AI), and in particular the branch of Machine Learning (ML), played a central role. The modern idea of ML, the branch of computer science that studies how to develop machines that are capable of learning information from data, dates back more than sixty years ago. However, it is only recently that the Big Data revolution and the modern hardware for parallel computing allowed to train powerful deep learning models. Nowadays, deep neural network models proved successful in many fields of knowledge, from science to engineering and economy. The problem with these architectures is that, in many cases, they are treated like black boxes over which we are not able to tell what they have learned from the data and why they make certain choices. Understanding what are the principles guiding the functioning of ML models is crucial in those fields in which even a small error might be critical, or where important decisions that might be aided by AI have to be taken. For this reason, in the last years, researchers have put a lot of effort in the direction of *explainable machine learning*, namely the development of methods and ML architectures that can be interpreted by humans.

The object of this thesis is the Restricted Boltzmann Machine (RBM), a neural network model which is inspired by the spin systems studied in the field of Statistical Physics. The main advantage of using this type of architecture is that we can apply the tools developed in the study of disordered systems for providing a macroscopic description of the model and for understanding how the machine learns from the data. Throughout this work, we will always put the accent on the explainability rather than on the performances. Among the practical applications of the RBM, a particular interest here is dedicated to biological datasets, and in particular to protein sequences.

This thesis is structured as follows. In chapter 1 we present the basic RBM and we discuss some possible extensions of the model that have been implemented in this thesis work, such as the possibility to deal with categorical (Potts) variables and to train the model in a semi-supervised fashion. Eventually, we present some theoretical results taken from the literature that help the understanding of the learning process. Chapter 2 is dedicated to the training of the RBM. Here we enter into the more technical aspects of the training and we present some experiments done with the model. In chapter 3 we discuss the mean-field theory and we extend it to the case of the Potts RBM. Finally, in chapter 4 we present a new method for generating relational trees of data by using the mean-field theory and the learning dynamics of the model, an application of the RBM that is particularly interesting for protein datasets.

The scientific contribution of this work consists in the algorithm presented in chapter 4 and in the derivation of the TAP equations for the Potts RBM that is discussed in chapter 3 which, to our knowledge, was not yet present in the literature. We also adapted the training program to work with categorical labels instead of one-hot vectors, and implemented the semi-supervised learning procedure.

Chapter 1

The Restricted Boltzmann Machine

In this chapter we present the Restricted Boltzmann Machine (RBM). Instead of starting with the most general case, we choose to begin our exposition with a short introduction to the learning of energy-based models and then to describe the most simple case of an RBM, in which both the visible and the hidden units are binary variables. We then progressively add complexity to this base model, first by generalizing the visible layer to the case of categorical variables and then allowing for a generic potential acting on the hidden units. Eventually, we will present the most general case coherent with the scopes of this work, in which we consider a semi-supervised setting where partially annotated data are used for the training. The last section of this chapter aims at showing how a physics-inspired approach can be profitable to study the model from a theoretical perspective, and in this context we will have the opportunity to introduce a useful language for describing the model that is borrowed from the field of disordered systems.

1.1 Definition of The Model

1.1.1 Unsupervised Learning with Energy-Based Models

In the framework of *unsupervised* Machine Learning (ML), we are given a dataset $\mathcal{D} = \{\boldsymbol{x}_d^{(k)}\}_{k=1,\dots,N_{\text{data}}}$, where $\boldsymbol{x}_d^{(k)}$ is a M-dimensional real vector, and we assume that the data are identically and independently distributed (i.i.d.) according to an unknown probability distribution $p(\boldsymbol{x})$. Our goal is to learn the empirical distribution of the data and to encode it in a computational model $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, parametrized by some parameters $\boldsymbol{\theta}$, by properly tuning the parameters of the model. The procedure of fitting the model on the data is called *training*. Once the model is properly trained, it is then possible to use it in a *generative* way, by sampling new data that have the same statistics as the training dataset.

We will restrict ourselves to the particular family of *energy-based models*, which is defined by a probability distribution written in the form

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}} e^{-E(\boldsymbol{x};\boldsymbol{\theta})}, \tag{1.1}$$

where E is called *energy function* or *Hamiltonian*. $Z_{\boldsymbol{\theta}}$ is called *partition function*, an it is the normalization factor of the probability distribution:

$$Z_{\boldsymbol{\theta}} = \sum_{\boldsymbol{x}} e^{-E(\boldsymbol{x},\boldsymbol{\theta})}. \tag{1.2}$$

In the expression above and throughout all the exposition of this work, we conventionally indicate as $\sum_{\boldsymbol{x}}$ (with a bold symbol as a subscript) the trace operator, which has to be intended either as a sum over all the configurations of the system if the variables are discrete, or as a multi-dimensional integral if the variables are continuous. It follows, then, that to define the probability distribution of an energy-based model it is sufficient to specify an energy function $E(\boldsymbol{x};\boldsymbol{\theta})$. In Statistical Physics, the probability distribution (1.1) is known under the name of *Boltzmann distribution*.

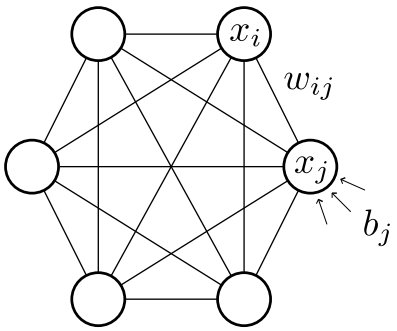

Figure 1.1: Probabilistic graph of a Boltzmann Machine: each node represent a random variable x_j and it is subject to a local bias (field) b_j. The graph is fully-connected, and the edges represent the interactions among variables. The entity of the interaction is described by the weights matrix $\boldsymbol{w}$.

At this point, it is yet not clear how to choose the functional form of the energy function. The inspiration comes from the Ising model in Statistical Physics, where a set of spin variables $s_i \in \{-1, 1\}$ interact on a lattice through the Hamiltonian

$$E(\boldsymbol{s}) = -\sum_{(i,j)} J_{ij} s_i s_j - \sum_i b_i s_i, \tag{1.3}$$

where $\boldsymbol{J}$ is the interaction matrix, $\boldsymbol{b}$ is an external magnetic field and the sum runs over all the connected pairs of spins ($J_{ij} \neq 0$). Physically, the interaction term rewards those configurations where two interacting spins s_i and s_j are aligned if $J_{ij} > 0$, and anti-aligned if $J_{ij} < 0$. On the other hand, the second term competes for aligning the spins along the direction of the field at each site, and we therefore say that the field acts as a *bias*. The probability distribution of the configurations of a spin system at equilibrium is exactly given by the Boltzmann distribution (1.1), where the parameters of the model are $\boldsymbol{\theta} = (\boldsymbol{J}, \boldsymbol{b})$.

Following the Ising model analogy, we can think of describing our data by using an *undirected graph* made of M connected nodes, where the nodes of the graph represent random variables associated with the M components of the data vectors. The edges of the graph represent statistical interactions among the variables, and the strength of the interactions is described by the *weight matrix* $\boldsymbol{w} = \{w_{ij}\}$ (we change notation to be coherent with the computer science community), such that $w_{ii} = 0 \ \forall i = 1, \dots, M$. Finally, we consider an "external field", $\boldsymbol{b}$, that acts as a local bias for the variables. In this way, we can introduce an effective energy function,

$$E(\boldsymbol{x}; \boldsymbol{\theta}) = -\sum_{i,j} w_{ij} x_i x_j - \sum_i b_i x_i, \tag{1.4}$$

and define a probability function over the graph given by Equation (1.1). Instead of considering spin variables, i.e. that can assume values in $\{-1, 1\}$, we move to the computer science representation of binary variables $x_i \in \{0, 1\}$. The Hamiltonian (1.4) does not carry any physical meaning about the data, but it has to be intended as an effective model that we use as a tool for extracting information from the data. A computational model that implements the probability distribution associated with an energy function of the kind (1.4) is generally called *Boltzmann Machine* (BM) [13].

1.1.2 Boltzmann Machine Learning

Given a set of examples $\mathcal{D} = \{\boldsymbol{x}_d^{(k)}\}_{k=1,\dots,N_{\text{data}}}$, we are interested in adjusting the parameters $\boldsymbol{\theta}$ of the generative model so that it approximates the empirical distribution of the data

$$p_{\text{data}}(\boldsymbol{x}) = \frac{1}{N_{\text{data}}} \sum_{k=1}^{N_{\text{data}}} \delta\left(\boldsymbol{x} - \boldsymbol{x}_d^{(k)}\right), \tag{1.5}$$

where $\delta(x - y) = 1$ if $x = y$ and 0 otherwise. A standard way of doing this is to translate the inference into an optimization problem, where the parameters are tuned in such a way to minimize

the Kullback-Leibler (KL) divergence between the empirical distribution and the inferred one with respect to the parameters $\boldsymbol{\theta}$:

$$\begin{aligned} D_{\mathrm{KL}}(p_{\mathrm{data}}||p_{\boldsymbol{\theta}}) &= \sum_{\boldsymbol{x}} p_{\mathrm{data}}(\boldsymbol{x}) \log\left(\frac{p_{\mathrm{data}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}\right) = \\ &= \sum_{\boldsymbol{x}} p_{\mathrm{data}}(\boldsymbol{x}) \log p_{\mathrm{data}}(\boldsymbol{x}) - \sum_{\boldsymbol{x}} p_{\mathrm{data}}(\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}). \end{aligned} \tag{1.6}$$

The first term is (minus) the Shannon entropy of the empirical distribution, and it does not depend on the parameters of the model. This means that minimizing the KL divergence is equivalent to maximizing the term

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) &= \sum_{\boldsymbol{x}} p_{\mathrm{data}}(\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \\ &\propto \sum_{k=1}^{N_{\mathrm{data}}} \sum_{\boldsymbol{x}} \delta\left(\boldsymbol{x} - \boldsymbol{x}_d^{(k)}\right) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \\ &= \sum_{k=1}^{N_{\mathrm{data}}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_d^{(k)}) = \log\left(\prod_{k=1}^{N_{\mathrm{data}}} p_{\boldsymbol{\theta}}(\boldsymbol{x}_d^{(k)})\right), \end{aligned} \tag{1.7}$$

which is nothing but the Log-Likelihood (LL) of the model with respect to the dataset $\mathcal{D}$. The presence of the partition function prevents us from directly evaluating the LL (or, equivalently, the KL divergence), and therefore standard optimization routines do not help. However, we can still maximize the LL using the *Gradient Ascent* (GA) method, which consists of iteratively updating the parameters of the model by moving in the direction of the gradient of the LL evaluated on the data:

$$\theta_i(t+1) \leftarrow \theta_i(t) + \eta \Delta\theta_i \quad \text{with} \quad \Delta\theta_i \equiv \frac{\partial \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})}{\partial \theta_i}, \tag{1.8}$$

where η is a (small) hyper-parameter called learning rate and $t = 1, \ldots, N_{\mathrm{upd}}$. This optimization method can be made *stochastic* – and in this case we talk about *Stochastic Gradient Ascent* (SGA) – by subdividing the data into branches, called *minibatches*, and evaluating the gradient on these subsets rather than on the whole dataset. In this way, we introduce a sort of "effective temperature" in the system that helps the algorithm escape local maxima and improves the speed of the learning.

Now, for each parameter θ_i, we can associate a *conjugate variable* defined as

$$\mathcal{C}_{\theta_i}(\boldsymbol{x}) \equiv -\frac{\partial E(\boldsymbol{x};\boldsymbol{\theta})}{\partial \theta_i}. \tag{1.9}$$

Then, for a Boltzmann-like probability distribution, the gradient of the LL with respect to its parameters is particularly simple to write down:

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i}\left[\frac{1}{N_{\mathrm{data}}} \sum_{k=1}^{N_{\mathrm{data}}} \left(-E(\boldsymbol{x}_d^{(k)};\boldsymbol{\theta}) - \log Z_{\boldsymbol{\theta}}\right)\right] = \\ &= \frac{1}{N_{\mathrm{data}}} \sum_{k=1}^{N_{\mathrm{data}}} \mathcal{C}_{\theta_i}(\boldsymbol{x}_d^{(k)}) - \sum_{\boldsymbol{x}} \mathcal{C}_{\theta_i}(\boldsymbol{x}) \frac{e^{-E(\boldsymbol{x};\boldsymbol{\theta})}}{Z_{\boldsymbol{\theta}}} = \\ &= \langle \mathcal{C}_{\theta_i} \rangle_{\mathcal{D}} - \langle \mathcal{C}_{\theta_i} \rangle_{E}, \end{aligned} \tag{1.10}$$

where we denoted as $\langle\cdot\rangle_{\mathcal{D}}$ the average over the empirical distribution (1.5) and as $\langle\cdot\rangle_E$ the average according to the probability distribution of the generative model. In particular, the gradient of the LL associated with the model (1.4) is

$$\begin{aligned} \Delta w_{ij} &= \frac{\partial \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})}{\partial w_{ij}} = \langle x_i x_j \rangle_{\mathcal{D}} - \langle x_i x_j \rangle_E, \\ \Delta b_i &= \frac{\partial \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})}{\partial b_i} = \langle x_i \rangle_{\mathcal{D}} - \langle x_i \rangle_E. \end{aligned} \tag{1.11}$$

From these equations, we can see that the parameters of the model are tuned in such a way as to make the one-point and two-point correlation functions predicted by the model coincide with the empirical ones obtained from the data. This is not a coincidence, since it can be directly shown that the Boltzmann distribution associated with the Hamiltonian (1.4) is the maximum entropy model that matches the firsts two moments of the data.

The first term of the gradient (1.10), the so-called *positive term*, is readily evaluated from the data. The difficult part comes when we want to evaluate the second part – the *negative term* – because the presence of the intractable partition function prevents us from computing the average. This problem can be overcome if we renounce computing the negative term exactly and we rather try to approximate it using dynamic Monte Carlo methods. Indeed, if we are able to sample N i.i.d. random variables from a probability distribution $p(\boldsymbol{x})$, the average of a generic function f under p can be estimated by the empirical mean, i.e.,

$$\langle f \rangle = \sum_{\boldsymbol{x}} f(\boldsymbol{x}) p(\boldsymbol{x}) \overset{N\to\infty}{\approx} \frac{1}{N} \sum_{k=1}^{N} f\left(\boldsymbol{x}^{(k)}\right) = \overline{f} \quad \text{with} \quad \boldsymbol{x}^{(k)} \sim p(\boldsymbol{x}). \tag{1.12}$$

In this way, the problem of evaluating an intractable probability distribution translates into the problem of sampling i.i.d. samples from the model.

We can sample configurations from $p_{\boldsymbol{\theta}}$ by running a *Markov Chain Monte Carlo* (MCMC) process with (Boltzmann-)Gibbs sampling. The dynamics of the chain consists of picking up a node randomly, say x_k, and considering all the other nodes, that we indicate as $\boldsymbol{x}_{-k}$, as fixed. Then, we update the state of the node using *heat-bath* dynamics, i.e. according to the conditioned probability

$$\begin{aligned} p_{\boldsymbol{\theta}}(x_k = 1 | \boldsymbol{x}_{-k}) &= \left. \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{-k})} \right|_{x_k=1} = \left. \frac{\exp\left[\sum_i x_i \left(b_i + \sum_j w_{ij} x_j\right)\right]}{\sum_{x_k=0,1} \exp\left[\sum_i x_i \left(b_i + \sum_j w_{ij} x_j\right)\right]} \right|_{x_k=1} = \\ &= \left. \frac{\exp\left[x_k \left(b_k + \sum_j w_{kj} x_j\right)\right]}{\sum_{x=0,1} \exp\left[x \left(b_k + \sum_j w_{kj} x_j\right)\right]} \right|_{x_k=1} = \\ &= \frac{1}{1 + e^{-(b_k + I_k(\boldsymbol{x}_{-k}))}} = \text{sigmoid}(b_k + I_k(\boldsymbol{x}_{-k})), \end{aligned} \tag{1.13}$$

where we introduced the *sigmoid* function $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$ and we defined the total "current" coming from the other nodes as

$$I_k(\boldsymbol{x}_{-k}) \equiv \sum_j w_{kj} x_j \quad \text{with} \quad w_{kk} = 0. \tag{1.14}$$

We will refer to the total current arriving at a node plus the contribution of the local field as the *activation* of the node, and we will call *activation function* the mapping from the activation to the probability of the node to be active (the sigmoid function, in this specific case).

The algorithm for sampling from $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is described in Algorithm 1. It can be proven that the algorithm satisfies the detailed balance condition, which in turn guarantees the convergence of the chain toward the target equilibrium distribution.

Under mild conditions, the previous algorithm is guaranteed to converge to the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ in the limit $t \to \infty$. We therefore have to make sure that N_{Gibbs} is large enough to permit the Markov Chain to reach the equilibrium, that is, we want the final state of the chain to be de-correlated from the initial condition $\boldsymbol{x}_0$. Once the BM is successfully trained, we can use the Algorithm 1 for generating new data starting from a random initial state.

The Boltzmann Machine has the advantage of being a simple model whose learning dynamics is easily interpretable: just match the firsts two moments of the distribution with the empirical ones. An

Algorithm 1 Gibbs sampling MCMC for BMs

Input: Initial state: $\boldsymbol{x}_0$, Number of steps: N_{Gibbs}
Output: Final state: $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$
for $t = 1, \dots, N_{\text{Gibbs}}$ **do**
  pick k randomly, $k \in \{1, \dots, M\}$
  set $x_k = +1$ with probability $p_{\boldsymbol{\theta}}(x_k = 1 | \boldsymbol{x}_{-k}) = \text{sigmoid}(b_k + I_k(\boldsymbol{x}_{-k}))$
  otherwise set $x_k = 0$
end for

example of the application of BMs to biological data is the determination of contact maps in protein-protein interactions [10].
On the other hand, this kind of model presents some serious drawbacks. First, the fact that we are reproducing only the second-order statistics of the environment leaves out all the higher-other of correlations. Although it has been shown that in some cases the BM is able to reproduce also the third moment of the dataset [10], still the problem remains. The expressivity of the model can be improved by introducing *hidden variables* that perform "feature extraction", as we will see in the next section. Second, training and sampling data from a BM is highly time-consuming. Indeed, Algorithm 1 is not parallelisable (each variable x_i must be updated sequentially since it interacts with all its neighbors), and equilibrating requires a number of iterations N_{Gibbs} sufficiently high to run over all the nodes of the graph multiple times. This means that the sampling time scales linearly with the input dimension, M, thus making the training prohibitive for high-dimensional data.

1.1.3 The Restricted Boltzmann Machine

The *Restricted Boltzmann Machine* (RBM) [13] is a variant of the classical BM that aims at improving the expressivity of the model while drastically reducing the training time. The first modification consists of introducing a set of so-called *hidden variables*. Those variables do not have any physical meaning (and for this reason we refer to them as *hidden*, or *latent*, units), but rather represent high-order features of the "physical variables". Roughly speaking, we can think of the latent variables as representing hypotheses that the model produces over the data. As we will show in subsection 1.5.1, this modification allows us to approximate the empirical distribution of the data up to any order of correlations.
The second major change consists on defining the model on a *bipartite graph*, where a layer of N_v *visible units*, $\boldsymbol{v} = \{v_i\}_{i=1,\dots,N_v}$, – the "physical variables" – is connected with a layer of N_h hidden units, $\boldsymbol{h} = \{h_\mu\}_{\mu=1,\dots,N_h}$, through the weight matrix $\boldsymbol{w} = \{w_{i\mu}\}$, but no direct connection is present within each layer. A schematic representation of the RBM is shown in Figure 1.2. Throughout the present work, we will conventionally use a *latin* index when we refer to visible units, and a *greek* index when dealing with hidden units.
For ease of exposition, let us focus for the moment on the particular case in which both the visible and the hidden variables can take a binary value in $\{0, 1\}$; we will generalize this setting in the later sections.
We call $\boldsymbol{a}$ the field interacting with the visible variables and $\boldsymbol{b}$ the field acting on the hidden layer. The parameters of the model are therefore $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}\}$. Overall, the Hamiltonian of the RBM can be written as

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\sum_i a_i v_i - \sum_\mu b_\mu h_\mu - \sum_{i\mu} v_i w_{i\mu} h_\mu. \tag{1.15}$$

All the results we derived for the BM remain valid once we properly consider the role of the visible and hidden units in the equations. Once we redefine the average over the data of an observable $\mathcal{O}$ as

$$\langle \mathcal{O} \rangle_{\mathcal{D}} = \sum_{\boldsymbol{v}, \boldsymbol{h}} \mathcal{O}(\boldsymbol{v}, \boldsymbol{h})\ p(\boldsymbol{h}|\boldsymbol{v}) p_{\text{data}}(\boldsymbol{v}), \tag{1.16}$$

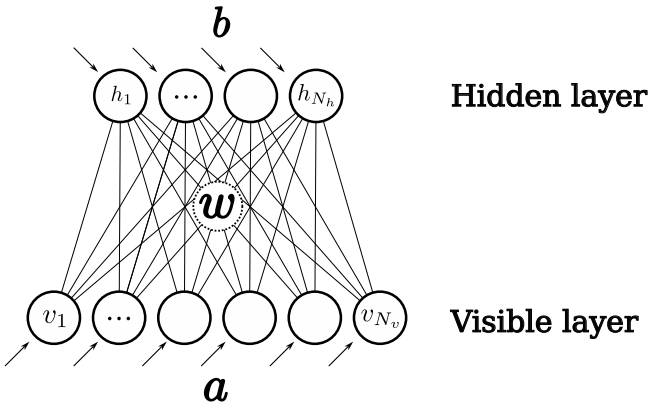

Figure 1.2: Schematic representation of an RBM. Each visible node v_i interacts through the matrix $\boldsymbol{w}$ with all the hidden units, but it is independent of all the other visible nodes.

the gradient of the LL can be written as

$$\begin{aligned} \Delta a_i &= \langle v_i \rangle_{\mathcal{D}} - \langle v_i \rangle_E \,, \\ \Delta b_\mu &= \langle h_\mu \rangle_{\mathcal{D}} - \langle h_\mu \rangle_E \,, \\ \Delta w_{i\mu} &= \langle v_i h_\mu \rangle_{\mathcal{D}} - \langle v_i h_\mu \rangle_E \,. \end{aligned} \tag{1.17}$$

In the particular case of binary variables $v_i \in \{0,1\}$, $h_\mu \in \{0,1\}$, the conditioned probability distribution for a single node can be easily derived along the line of what we have done for the BM. Let us define $I_\nu(\boldsymbol{v}) = \sum_i w_{i\nu} v_i$ as the input current coming from the visible layer to the hidden node h_ν. Then:

$$\begin{aligned} p(h_\nu = 1|\boldsymbol{v}) &= \sum_{\boldsymbol{h}_{-\nu}} \frac{p(\boldsymbol{h},\boldsymbol{v})}{p(\boldsymbol{v})} = \sum_{\boldsymbol{h}_{-\nu}} \frac{e^{-E(\boldsymbol{h},\boldsymbol{v};\boldsymbol{\theta})}}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{h},\boldsymbol{v};\boldsymbol{\theta})}} = \\ &= \frac{\exp\left(b_\nu + \sum_i w_{i\nu} v_i\right)}{\sum_{h_\nu=0,1} \exp\left(h_\nu b_\nu + \sum_i v_i w_{i\nu} h_\nu\right)} = \text{sigmoid}\left(b_\nu + I_\nu(\boldsymbol{v})\right). \end{aligned} \tag{1.18}$$

Symmetrically, one can write the probability distribution of a visible node conditioned to the knowledge of the hidden state as

$$p(v_i = 1|\boldsymbol{h}) = \text{sigmoid}\left(a_i + I_i(\boldsymbol{h})\right), \tag{1.19}$$

where $I_i(\boldsymbol{h}) = \sum_\mu w_{i\mu} h_\mu$.
The advantage of using a bipartite structure for the underlying probabilistic graph is that the probability distribution of one layer conditioned to the knowledge of the state in the other layer factorizes, and this allows for parallelization. We therefore have

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_{i=1}^{N_v} p(v_i|\boldsymbol{h}) \quad \text{and} \quad p(\boldsymbol{h}|\boldsymbol{v}) = \prod_{\mu=1}^{N_h} p(h_\mu|\boldsymbol{v}). \tag{1.20}$$

This difference with the BM is crucial when it comes to estimating the negative term of the gradient of the LL. In fact, Equations (1.20) implies that the nodes of a layer are independent of each other when conditioned on the state of the opposite layer. This means that we can run a MCMC in which, iteratively, we clump the variables of one layer to the previous values of the chain and we sample the variables of the opposite layer *all at once*:

$$\begin{aligned} \boldsymbol{h}_{t+1} &\sim p(\boldsymbol{h}|\boldsymbol{v}_t), \\ \boldsymbol{v}_{t+1} &\sim p(\boldsymbol{v}|\boldsymbol{h}_{t+1}). \end{aligned} \tag{1.21}$$

for $t = 1, \ldots, N_{\text{Gibbs}}$. This method is called *alternating* or *block Gibbs sampling*, and the computational time is – up to the parallelization capabilities of the hardware – independent of both the size of the input and the number of hidden units. In Figure 1.3 we present a sketch of the sampling procedure.

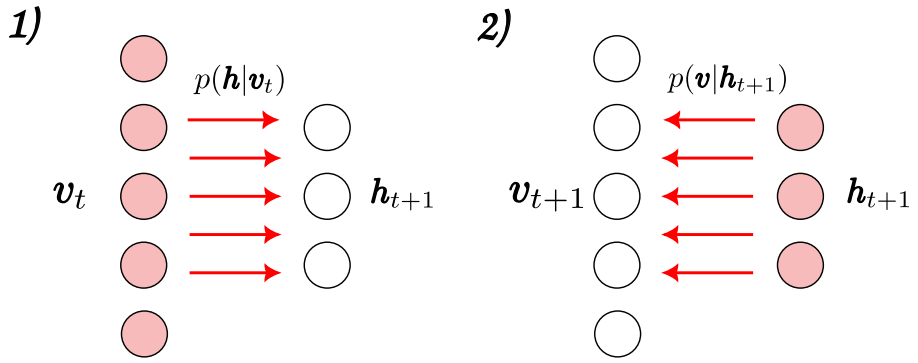

Figure 1.3: Sketch of the alternating Gibbs sampling: One generates hidden variables at time $t+1$ all at once by just knowing the state of the visible vector at time t. Subsequently, one can use this latent vector in order to generate all visible configurations in parallel at time $t+1$.

Algorithm 2 Alternating Gibbs Sampling for RBMs

Input: Initial state: $(\boldsymbol{v}_0, \boldsymbol{h}_0)$, Number of steps: N_{Gibbs}
Output: Final state: $(\boldsymbol{v}, \boldsymbol{h}) \sim p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})$ (if converging to equilibrium)
for $t = 1, \ldots, N_{\text{Gibbs}}$ **do**
  sample $\boldsymbol{h}_t \sim \text{sigmoid}(\boldsymbol{b} + \boldsymbol{I}(\boldsymbol{v}_{t-1}))$
  sample $\boldsymbol{v}_t \sim \text{sigmoid}(\boldsymbol{a} + \boldsymbol{I}(\boldsymbol{h}_t))$
end for

To sum up, we can sample visible and hidden configurations from $p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})$ using the Algorithm 2. There are different ways of choosing the initial conditions of the chain throughout the training, and each method differs considerably from the others both in terms of the training's dynamics and the properties of the trained machine. We will address this problem in chapter 2.

As a final remark, let us point out that Algorithm 2 can be employed, once the RBM has been properly trained, to generate new samples distributed according to the data distribution. This is typically done by initializing a Markov Chain with a random state and by running t_G Gibbs-steps. Random initialization is crucial to ensure an independent generation and to evaluate the correctness of the learnt model.

1.2 Generalization to Categorical Variables

In the previous section, we introduced the RBM model in the most basic setup in which both the visible and the hidden layers are binary random vectors. There are many cases, though, in which the data we want to train our model on are not sequences of Ising variables, but rather chains of categorical variables that can take N_q possible values (in this sense, the Ising variables are a special case in which $N_q = 2$). This is often the case, for instance, when one wants to study sequences of proteins (which are chains made of 20 possible amino acids) or nucleotide sequences in the case of DNA (4 possible nucleotides). It is always possible to employ the binary RBM model for studying these kinds of datasets by first applying a *one-hot encoding* of the data. This consists of encoding each possible value of the sequence in a N_q-dimensional vector with just one entry set to 1 and all the others set to 0:

$$\{0, \ldots, N_q\} \ni a \mapsto v_j = \begin{cases} 1 & \text{if } j = a, \\ 0 & \text{otherwise} \end{cases} \quad \text{for} \quad j = 1, \ldots, N_q.$$

By itself, this transformation does not prevent us to create states with more than one possible label, which adds up mainly to unphysical states (in the sense that they do not exist in the database). One standard solution is to impose the one-category constraint in the model by hand. However, there is a much simpler and elegant approach that we discuss below.

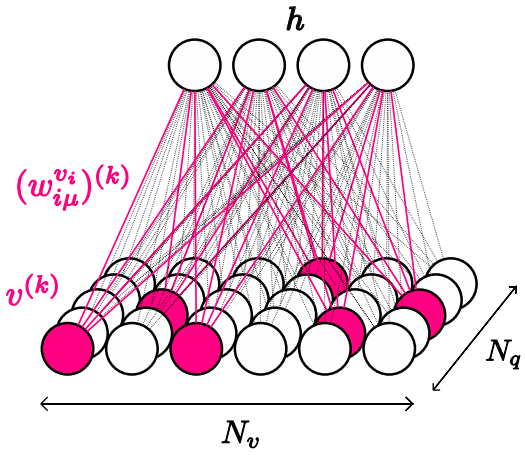

Figure 1.4: Scheme of the Potts RBM. Now the weights matrix is a 3-dimensional tensor $\boldsymbol{w} = \{w_{i\mu}^q\}$. When an input data $\boldsymbol{v}$ enters the machine, it selects at each site i the connection with the hidden component μ that has the proper color v_i.

1.2.1 Potts RBM

A better option is to redefine the model such that it can handle directly categorical variables. In statistical mechanics, the extension of the Ising model to categorical variables is called the Potts model. Here, spins are replaced with nodes that can assume one of the N_q possible values (called *colors* in the jargon), $s_i \in \{1, \dots, N_q\}$, and that interact through the Hamiltonian

$$E_p(\boldsymbol{s}) = -\sum_{(i,j)} w_{ij}\delta_{s_i,s_j} - \sum_i h_{s_i}, \tag{1.22}$$

where the first sum runs over all the connected nodes, δ is the Kronecker delta and $\boldsymbol{h}$ is an external magnetic field. In other words, in the Potts model, two nodes interact through the matrix $\boldsymbol{w}$ only when they are connected and when they assume the same color. We can generalize the RBM by using Potts variables in the visible layer, and by introducing a local field, $\boldsymbol{a} = \{a_i^q\}$, that can assume different values for different colors. The Hamiltonian of the Potts RBM with binary hidden units is therefore defined as

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\sum_{i=1}^{N_v}\sum_{\mu=1}^{N_h}\sum_{q=1}^{N_q} h_\mu w_{i\mu}^q \delta_{v_i,q} - \sum_{i=1}^{N_v}\sum_{q=1}^{N_q} a_i^q \delta_{v_i,q} - \sum_{\mu=1}^{N_h} b_\mu h_\mu, \tag{1.23}$$

where i is an index that runs over the input sequence and v_i is the color of the state $\boldsymbol{v}$ at position i. A scheme of the structure of the Potts RBM is shown in Figure 1.4.

1.2.2 Marginal distributions and gradient of the Log-likelihood

Analogously to what we did for the binary RBM, we can compute the conditional distributions of hidden and visible variables. For the latent variables (which remain binary) the computation remains the same, and we end up with a sigmoid activation function:

$$p(h_\nu = 1|\boldsymbol{v}) = \text{sigmoid}\left(b_\nu + I_\nu(\boldsymbol{v})\right) \quad \text{with} \quad I_\nu(\boldsymbol{v}) = \sum_{iq} \delta_{v_i,q} w_{i\mu}^q. \tag{1.24}$$

For the visible layer, instead, the activation function becomes the generalization of the sigmoid to

categorical variables, also known as the *softmax* function. Let $r \in \{1, \dots, N_q\}$, then:

$$\begin{aligned} p(v_j = r|\boldsymbol{h}) &= \sum_{\boldsymbol{v}_{-j}} \frac{\exp\left(\sum_{iq} a_i^q \delta_{v_i,q} + \sum_{iq\mu} h_\mu w_{i\mu}^q \delta_{v_i,q}\right)}{\sum_{\boldsymbol{v}} \exp\left(\sum_{iq} a_i^q \delta_{v_i,q} + \sum_{iq\mu} h_\mu w_{i\mu}^q \delta_{v_i,q}\right)} \\ &= \frac{\exp\left(a_j^r + \sum_\mu h_\mu w_{j\mu}^r\right)}{\sum_{q=1}^{N_q} \exp\left(a_j^q + \sum_\mu h_\mu w_{j\mu}^q\right)} \\ &= \mathrm{softmax}_r\left(a_j^r + I_j^r(\boldsymbol{h})\right) \quad \text{with} \quad I_j^r(\boldsymbol{h}) = \sum_\mu h_\mu w_{j\mu}^r, \end{aligned} \tag{1.25}$$

where we put a subscript in the softmax definition in order to track the index over which the argument has to be normalized:

$$\mathrm{softmax}_i(x_{ij}) \equiv \frac{e^{x_{ij}}}{\sum_k e^{x_{kj}}}. \tag{1.26}$$

Applying Equation (1.10), the gradient of the LL can be written as

$$\begin{aligned} \Delta a_i^q &= \langle \delta_{v_i,q} \rangle_{\mathcal{D}} - \langle \delta_{v_i,q} \rangle_E \,, \\ \Delta b_\mu &= \langle h_\mu \rangle_{\mathcal{D}} - \langle h_\mu \rangle_E \,, \\ \Delta w_{i\mu}^q &= \langle \delta_{v_i,q} h_\mu \rangle_{\mathcal{D}} - \langle \delta_{v_i,q} h_\mu \rangle_E \,. \end{aligned} \tag{1.27}$$

1.2.3 Gauge freedom

It is a known fact in the literature, that the Potts model is not completely defined, but brings some degree of arbitrariness [29]. To see this in the case of the Potts RBM, consider the conditional probability distribution (1.25). The arbitrariness emerges by noticing that the probability distribution remains unchanged under the gauge transformations

$$\begin{aligned} w_{i\mu}^q &\leftarrow w_{i\mu}^q + A_{i\mu} \quad \forall i = 1, \dots, N_v, \quad \forall \mu = 1, \dots, N_h \\ a_i^q &\leftarrow a_i^q + B_i \quad \forall i = 1, \dots, N_v \end{aligned} \tag{1.28}$$

with $\{A_{i\mu}\}$ and $\{B_i\}$ constants, so that there are $N_v \cdot N_h + N_v$ degrees of freedom. To fix them, two gauge-fixing choices are typically used in the literature:

- Zero-sum gauge:

$$\sum_q a_i^q = 0, \quad \sum_q w_{i\mu}^q = 0 \quad \forall i = 1, \dots, N_v, \quad \forall \mu = 1, \dots, N_h, \tag{1.29}$$

- Lattice-gas gauge:

$$a_i^{N_q} = 0, \quad w_{i\mu}^{N_q} = 0 \quad \forall i = 1, \dots, N_v, \quad \forall \mu = 1, \dots, N_h, \tag{1.30}$$

  which is equivalent to measuring the energy with respect to the configuration $\mathbb{1}N_q$.

Although the two gauge-fixing choices are equivalent from a theoretical perspective, the zero-sum gauge is convenient in the algorithmic implementation, because centring the parameters of the model at zero is beneficial for numerical stability. The lattice-gas gauge, on the other hand, is convenient in analytical computations, and we will make use of it in Chapter 3.

Fixing the gauge is not just a mathematical requirement to have a well-defined model, as we empirically verified that implementing zero-sum gauge leads to a significant improvement in terms of performance.

Once we have chosen a gauge, we still have to make sure that it is preserved by the dynamics of the learning. The parameters update rule preserves the zero-sum gauge for the local visible field. Indeed,

if we assume $a_i^q(t)$ to fulfill (1.29), then $a_i^q(t+1)$ will do it as well only if $\sum_q \Delta a_i^q = 0$. It is easy to show that this last condition is indeed true in the update (1.27):

$$\begin{aligned}\sum_q \Delta a_i^q &= \sum_q \langle \delta_{v_i,q} \rangle_{\mathcal{D}} - \sum_q \langle \delta_{v_i,q} \rangle_E = \\ &= \sum_q \left(\frac{1}{N_{\text{data}}} \sum_{k=1}^{N_{\text{data}}} \delta_{v_{d,i}^{(k)},q} \right) - \sum_q \frac{\sum_{v_i} \delta_{v_i,q}\, e^{a_i^{v_i} + \sum_\mu w_{i\mu}^{v_i} h_\mu}}{\sum_p e^{a_i^p + \sum_\mu w_{i\mu}^p h_\mu}} = \\ &= \frac{1}{N_{\text{data}}} \sum_{k=1}^{N_{\text{data}}} \underbrace{\left(\sum_q \delta_{v_{d,i}^{(k)},q} \right)}_{=1} - \underbrace{\frac{\sum_q e^{a_i^q + \sum_\mu w_{i\mu}^q h_\mu}}{\sum_p e^{a_i^p + \sum_\mu w_{i\mu}^p h_\mu}}}_{=1} = 0.\end{aligned}$$

The same reasoning shows that the gradient ascent update does not preserve the zero-sum gauge for the weight tensor. That has to be fixed by hand, by re-centring the weight tensor along the color dimension after each parameter update:

$$w_{i\mu}^q \leftarrow w_{i\mu}^q - \frac{1}{N_q} \sum_{p=1}^{N_q} w_{i\mu}^p. \tag{1.31}$$

1.2.4 Weights and biases Initialization

In order to achieve better performances and faster convergence of the training, the weights of the RBM must be properly initialized. A typical way of initializing the interaction tensor is to independently sample its entries from a normal distribution centred at zero and with small variance ($\sim 10^{-4}$). It is important to start with small weights in order to initialize the system in its paramagnetic phase, because if the weights are too large the system may find itself in a spin glass phase and remain stuck there. The criterion is that the eigenvalues of the weights matrix should be much smaller that the learning-threshold (which is 4 for $\{0,1\}$ variables and 1 for $\{-1,1\}$ variables). We will discuss more in detail these points in chapter 2.

If the weights of the interaction tensor are small, we can assume that, at the very beginning of the training, the RBM can be approximated by the non-interacting model defined by the Hamiltonian

$$E_0(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\sum_{iq} \delta_{v_i,q} a_i^q - \sum_\mu h_\mu b_\mu. \tag{1.32}$$

Hence, a clever choice is to initialize the local fields in such a way as to minimize the distance between the empirical frequencies and the averages predicted by the non-interacting model. For the visible field, we have

$$\langle \delta_{q,v_i} \rangle_{\mathcal{D}} = \frac{e^{a_i^q}}{\sum_p e^{a_i^p}} = \frac{e^{a_i^q}}{Z_i} \Rightarrow a_i^q = \log \langle \delta_{v_i,q} \rangle_{\mathcal{D}} + c_i, \tag{1.33}$$

where $c_i = \log Z_i$. Imposing the zero-sum gauge $\sum_q a_i^q = 0$, we get

$$\begin{aligned}\sum_{q=1}^{N_q} a_i^q = \sum_q \log \langle \delta_{v_i,q} \rangle_{\mathcal{D}} + N_q\ c_i \Rightarrow c_i = -\frac{1}{N_q} \sum_q \log \langle \delta_{v_i,q} \rangle_{\mathcal{D}}\,, \\ a_i^q = \log \langle \delta_{v_i,q} \rangle_{\mathcal{D}} - \frac{1}{N_q} \sum_q \log \langle \delta_{v_i,q} \rangle_{\mathcal{D}}\,.\end{aligned} \tag{1.34}$$

The hidden local field, on the other hand, can be initialized by setting all entries at zero, because at the beginning of the learning no latent representation of the data is meaningful.

1.2.5 Centring Trick

The first idea of removing the mean from the nodes was originally proposed for feed-forward neural networks [19]. The idea is that the stability of the learning should be enhanced if the gradient of the LL is evaluated with respect to a new set of variables whose mean is centred at zero, so that it is less dependent on the dataset parameterization. To do this, we can rewrite the Hamiltonian of the RBM in terms of a new set of parameters $\tilde{\boldsymbol{\theta}} = (\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}})$ that interact only with the centered variables, and then derive the change of variables between the old "un-centred" parameters and the new ones [20]. The Hamiltonian in the new coordinates is

$$\begin{aligned} -\mathcal{E}(\boldsymbol{v}, \boldsymbol{h}; \tilde{\boldsymbol{\theta}}) &= \sum_{iq} \tilde{a}_i^q \left(\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}}\right) + \sum_{\mu} \tilde{b}_\mu \left(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}\right) + \sum_{i\mu q} \tilde{w}_{i\mu}^q \left(\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}}\right)\left(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}\right) \\ &= \sum_{iq} \delta_{v_i,q} \left(\tilde{a}_i^q - \sum_{\mu} \tilde{w}_{i\mu}^q \langle h_\mu \rangle_{\mathcal{D}}\right) + \sum_{\mu} h_\mu \left(\tilde{b}_\mu - \sum_{iq} \tilde{w}_{i\mu}^q \langle \delta_{v_i,q} \rangle_{\mathcal{D}}\right) + \sum_{i\mu q} \tilde{w}_{i\mu}^q h_\mu \delta_{v_i,q} + f(\tilde{\boldsymbol{\theta}}), \end{aligned} \tag{1.35}$$

where the last term does not depend on $\boldsymbol{v}$ and $\boldsymbol{h}$. That is equivalent to the un-centred Hamiltonian if we impose

$$\begin{cases} a_i^q = \tilde{a}_i^q - \sum_\mu \tilde{w}_{i\mu}^q \langle h_\mu \rangle_{\mathcal{D}}\,, \\ b_\mu = \tilde{b}_\mu - \sum_{iq} \tilde{w}_{i\mu}^q \langle \delta_{v_i,q} \rangle_{\mathcal{D}}\,, \\ w_{i\mu}^q = \tilde{w}_{i\mu}^q. \end{cases} \tag{1.36}$$

Now we want to write the gradient of the LL of the original uncentered machine in terms of the gradient computed on the centred system. To do this, we can write

$$\begin{aligned} a_i^q(t+1) &= \tilde{a}_i^q(t+1) - \sum_\mu \tilde{w}_{i\mu}^q(t+1) \langle h_\mu \rangle_{\mathcal{D}} \\ &= \tilde{a}_i^q(t) + \eta \Delta \tilde{a}_i^q - \sum_\mu \left(\tilde{w}_{i\mu}^q(t) + \eta \Delta \tilde{w}_{i\mu}^q\right) \langle h_\mu \rangle_{\mathcal{D}} \\ &= a_i^q(t) + \cancel{\sum_\mu \tilde{w}_{i\mu}^q(t) \langle h_\mu \rangle_{\mathcal{D}}} + \eta \Delta \tilde{a}_i^q - \cancel{\sum_\mu \tilde{w}_{i\mu}^q(t) \langle h_\mu \rangle_{\mathcal{D}}} - \eta \sum_\mu \Delta \tilde{w}_{i\mu}^q \langle h_\mu \rangle_{\mathcal{D}}\,. \end{aligned} \tag{1.37}$$

Therefore, we obtain

$$\Delta a_i^q = \Delta \tilde{a}_i^q - \sum_\mu \Delta \tilde{w}_{i\mu}^q \langle h_\mu \rangle_{\mathcal{D}}\,,$$

where

$$\Delta \tilde{a}_i^q = \left\langle (\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}}) \right\rangle_{\mathcal{D}} - \left\langle (\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}}) \right\rangle_{\mathcal{E}} = \langle \delta_{v_i,q} \rangle_{\mathcal{D}} - \langle \delta_{v_i,q} \rangle_{E}\,. \tag{1.38}$$

The same kind of computation leads to the following update of the parameters in terms of the centred gradient:

$$\begin{cases} \Delta w_{i\mu}^q = \left\langle (\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}})(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}) \right\rangle_{\mathcal{D}} - \left\langle (\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}})(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}) \right\rangle_{\mathcal{E}}\,, \\ \Delta a_i^q = \langle \delta_{v_i,q} \rangle_{\mathcal{D}} - \langle \delta_{v_i,q} \rangle_{E} - \sum_\mu \Delta w_{i\mu}^q \langle h_\mu \rangle_{\mathcal{D}}\,, \\ \Delta b_\mu = \langle h_\mu \rangle_{\mathcal{D}} - \langle h_\mu \rangle_{E} - \sum_{iq} \Delta w_{i\mu}^q \langle \delta_{v_i,q} \rangle_{\mathcal{D}}\,. \end{cases} \tag{1.39}$$

1.3 Generalization to a Generic Hidden Potential

So far, for simplicity, we dealt with a Hamiltonian of the form (1.23), where we assumed the hidden nodes to be binary variables $h_\mu \in \{0, 1\}$, at variance with the visible variables that could be categorical.

Now we want to extend the discussion to a more general setting in which the latent variables can be continuous and interact with a generic potential $\mathcal{U}_\mu(h_\mu;\boldsymbol{\vartheta}_\mu)$:

$$E(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) = -\sum_{iq} a_i^q \delta_{v_i,q} + \sum_\mu \mathcal{U}_\mu(h_\mu;\boldsymbol{\vartheta}_\mu) - \sum_{iq\mu} \delta_{v_i,q} w_{i\mu}^q h_\mu. \tag{1.40}$$

The new set of parameters that define the model is $\boldsymbol{\theta} = (\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{\vartheta})$, with $\boldsymbol{\vartheta} = \{\boldsymbol{\vartheta}_\mu\}_{\mu=1,\dots,N_h}$ being the parameters of the hidden potential. The analytical form of $\mathcal{U}$ can be obtained by first assuming a specific distribution P over the hidden variables, and then writing the potential as

$$\mathcal{U}(h;\boldsymbol{\vartheta}) = -\log P(h;\boldsymbol{\vartheta}). \tag{1.41}$$

1.3.1 Bernoulli Potential

For instance, the Hamiltonian (1.23) can be obtained by assuming $\boldsymbol{h}$ to be binary variables following a Bernoulli distribution,

$$P(h;\rho) = \rho^h\ (1-\rho)^{1-h} \quad \text{with} \quad \rho \in [0,1], \tag{1.42}$$

where we just have to reparametrize $\rho = \text{sigmoid}(b)$ with $b \in \mathbb{R}$. For this reason, from now on we will refer to the RBM with binary hidden variables as "Bernoulli RBM".

1.3.2 Truncated Gaussian Potential

The use of a binary hidden layer brings several advantages. For instance, its simplicity allows for a relatively easy analytical treatment of the model, the algorithmic implementation is simple and the training of the RBM is generally quite stable. Nevertheless, one can argue that having units that can only be active or idle, without giving any additional information such as the intensity of the activation, may result in a quite poor expressive power of the model. A step toward enhancing the expressivity of the model is to consider for the hidden layer a set of continuous random variables definite in the positive real line, $h \in [0,+\infty)$, which are distributed according to the *Truncated Gaussian* (TG) probability density function [22, 28, 29]. This is typically defined in the literature in the form

$$P(h;\mu,\sigma,a,b) = \frac{1}{\sigma}\sqrt{\frac{2}{\pi}} \frac{e^{-\frac{1}{2}\left(\frac{h-\mu}{\sigma}\right)^2}}{\text{erf}\left(\frac{b-\mu}{\sqrt{2}\sigma}\right) - \text{erf}\left(\frac{a-\mu}{\sqrt{2}\sigma}\right)} \mathbb{1}_{[a,b]}, \tag{1.43}$$

where μ and σ are the parameters of a Gaussian distribution, $\mathbb{1}$ is the indicator function and a and b are respectively the lower and the upper bounds of the domain of the distribution. erf indicates the *error function*, defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \mathrm{d}z\ e^{-z^2}. \tag{1.44}$$

Now, in order to have a handier notation for our purposes, let us define the transformation of parameters

$$\sigma \mapsto \frac{1}{\sqrt{\gamma}}; \qquad \mu \mapsto -\frac{b}{\gamma}. \tag{1.45}$$

If we make this change and we set $a = 0$, $b = +\infty$, we can rewrite the expression (1.43) as:

$$P(h;b,\gamma) = \sqrt{\frac{2\gamma}{\pi}} \frac{e^{-\frac{1}{2}\gamma h^2 - bh}}{1 - \text{erf}\left(\frac{b}{\sqrt{2\gamma}}\right)} \mathbb{1}_{[0,+\infty]}. \tag{1.46}$$

Once we neglect inessential normalization factors, this allows us to identify the hidden potential

$$\mathcal{U}_\mu(h_\mu;\gamma_\mu,b_\mu) = \frac{\gamma_\mu h_\mu^2}{2} + b_\mu h_\mu. \tag{1.47}$$

Marginal Distributions

For practical purposes, let us define the function

$$\Phi(x) = \sqrt{\frac{\pi}{2}} e^{\frac{x^2}{2}} \left(1 - \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right), \tag{1.48}$$

such that the normalization of the Truncated Gaussian distribution can be written as

$$Z_\mu = \int_0^\infty \mathrm{d}h \ \exp\left(-\mathcal{U}_\mu(h;\gamma_\mu,b_\mu)\right) = \int_0^\infty \mathrm{d}h \ \exp\left(-\frac{\gamma_\mu h^2}{2} - b_\mu h\right) = \frac{1}{\sqrt{\gamma_\mu}}\Phi\left(\frac{b_\mu}{\sqrt{\gamma_\mu}}\right). \tag{1.49}$$

The probability distribution of the visible nodes conditioned on the latent variables is not affected by the particular distribution of $\boldsymbol{h}$, and it is therefore given by Equation (1.25). Concerning the conditioned probability of the hidden nodes, it is again a Truncated Gaussian but with a *shift parameter* b_μ that gets shifted by the input current coming from the visible layer:

$$p(h_\mu|\boldsymbol{v}) = \frac{e^{-\frac{\gamma_\mu h_\mu^2}{2} - h_\mu(b_\mu - I_\mu(\boldsymbol{v}))}}{\int \mathrm{d}h \exp\left(-\frac{\gamma_\mu h^2}{2} - (b_\mu - I_\mu(\boldsymbol{v}))h\right)} = \frac{e^{-\frac{\gamma_\mu h_\mu^2}{2} - h_\mu(b_\mu - I_\mu(\boldsymbol{v}))}}{\frac{1}{\sqrt{\gamma_\mu}}\Phi\left(\frac{b_\mu - I_\mu(\boldsymbol{v})}{\sqrt{\gamma_\mu}}\right)}. \tag{1.50}$$

To sample the hidden state from (1.50) we can use the method of the inverse cumulative density function, which consists on sampling uniform variables $z \sim \mathcal{U}(0,1)$ and then transforming them as

$$h_\mu|\boldsymbol{v} = \sqrt{\frac{2}{\gamma_\mu}} \ \text{erf}^{-1}\left[z + \text{erf}\left(\frac{b_\mu - I_\mu(\boldsymbol{v})}{\sqrt{2\gamma_\mu}}\right)(1-z)\right] - \frac{b_\mu - I_\mu(\boldsymbol{v})}{\gamma_\mu}. \tag{1.51}$$

Unfortunately, using this method for sampling the latent variables leads to a series of numerical issues, mainly devoted to the instability of the *inverse* erf *function* when its argument is close to 1. This is why sampling from the Truncated Gaussian is typically done through other methods, such as rejection sampling. The problem with those methods is that they rely on *for* cycles, which heavily slow down the computation and are not suitable for training an RBM. For this reason, we still implemented the formula (1.51) by using some careful tricks in order to avoid numerical issues. More about the technicality of the sampling is described in the Appendix B.

An approximation that is often found in the literature consists on avoiding to sample directly from (1.50) by considering the mode of the distribution, $h_{\max} = \frac{I(\boldsymbol{v})-b}{\gamma}$, and extracting

$$h = \max\left(0, h_{\max} + n\right) = \text{ReLU}\left(h_{\max} + n\right) \quad \text{with} \quad n \sim \mathcal{N}(0,\sigma), \tag{1.52}$$

where $\mathcal{N}(0,\sigma)$ is a Normal distribution with mean 0 and standard deviation σ. That is the reason why people often refer to the RMB with Truncated Gaussian potential as "ReLU" RBM. Nevertheless, we chose to stick with the "exact" sampling in order to keep the training as controlled as possible.

To conclude this part, we point out the existence of many other versions of RBM, depending on the type of data one has to deal with (discrete/continuous) and the type of potential one chooses in the visible and hidden layers. Here, we treated only those cases that are interesting for our scopes, and we remand to [7] for a review of all the other models.

1.4 Semi-supervised Learning with RBMs

RBMs are typically trained in a purely unsupervised way, namely, they are fed with the data without providing any additional information about them (e.g. labels). By doing this, we let the model discover what the statistics of the data are and what the relevant features are without biasing it with our prior beliefs. Although this approach is, in principle, the safest one (we want the model to tell us something new about the data, and we had better avoid the risk of fooling it with our possibly wrong assumptions), in practice it happens that some features that *we know* are present in the data

are not detected by the model. This can happen because the model has not been trained enough, or the hyper-parameters were sub-optimal (learning rate, number of hidden units, batch size, etc.), or maybe simply because the expressive power of the model is not enough for catching what we are interested in. Another possibility is that the internal classification of data learned by the machine is not the one we were looking for: it could be either too fine or too coarse. Finally, there is also the non-trivial issue of how to extract the learned features encoded in the parameters of the model once the learning is complete.

In many cases, though, we are not completely agnostic about the data, and we may have specific targets that we hope our model detects. For instance, if we train the RBM on the handwritten digits dataset, we expect our model to be able to "understand" that two images of the same digit are *closer* (in a sense that will be clearer in the next chapters) than two images of distinct digits. Or again, if we are studying a protein and we have experimental functional annotations on some sequences of the dataset, we want that two sequences having the same annotations to be recognized as "similar". In all those cases in which we have a well-defined objective in mind, it would be helpful to provide the RBM with those *labels* while it is learning the statistics of the data. In this way, we might nudge the model to recognize the features we are interested in, without with this compromising the *exploration* of unknown properties of the data.

Eventually, if the trained machine learned how to associate labels to the data, we would like to define a simple procedure to enquire the RBM about the probability that a given piece of data has a certain label.

It turns out that there is a simple and elegant way of doing all this with RBMs. If we assume to have N_ℓ possible classes classifying our data, we can introduce a new categorical variable $\ell \in \{1, \dots, N_\ell\}$ that interacts with the hidden layer through a $N_\ell \times N_h$ matrix $\boldsymbol{d}$ and that is subject to the local field $\boldsymbol{c}$. A simplified sketch of the implementation is represented in Figure 1.5.

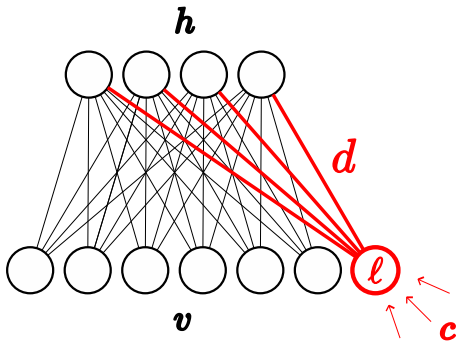

Figure 1.5: Sketch of the RBM in semi-supervised mode: the label ℓ selects the row of the matrix $\boldsymbol{d}$ that has to interact with the hidden layer. The net effect is an additional current in the activation of the hidden nodes.

The Hamiltonian of the Potts RBM in its semi-supervised version is

$$E(\boldsymbol{v}, \boldsymbol{h}, \ell; \boldsymbol{\theta}) = -\sum_{iq} \delta_{v_i,q} a_i^q + \sum_\mu \mathcal{U}_\mu(h_\mu; \boldsymbol{\vartheta}_\mu) - \sum_n \delta_{\ell,n} c_n - \sum_{i\mu q} \delta_{v_i,q} w_{i\mu}^q h_\mu - \sum_{n\mu} \delta_{\ell,n} d_{n\mu} h_\mu. \tag{1.53}$$

In practice, the effect of the label is to introduce a new current that steers the activation of the hidden units toward the desired direction. The functional form of the conditional distributions for visible and hidden variables remains the same, but now we have to account for a new contribution to the hidden current:

$$\begin{aligned} p(v_i = q|\boldsymbol{h}) &= \text{softmax}_q\left(a_i^q + I_i^q(\boldsymbol{h})\right) \\ p(h_\mu = 1|\boldsymbol{v}, \ell) &= \text{sigmoid}\left(b_\mu + I_\mu(\boldsymbol{v}, \ell)\right) \\ p(\ell = l|\boldsymbol{h}) &= \text{softmax}_l\left(c_l + I_l(\boldsymbol{h})\right) \end{aligned} \tag{1.54}$$

where

$$I_\mu(\boldsymbol{v}, \ell) = \sum_{iq} w_{i\mu}^q \delta_{v_i,q} + \sum_n \delta_{\ell,n} d_{n\mu} \quad \text{and} \quad I_l(\boldsymbol{h}) = \sum_\mu d_{l\mu} h_\mu. \tag{1.55}$$

If every data point brings a label, the new dataset will be made of the set of pairs $\mathcal{D} = \{(\boldsymbol{v}_d^{(k)}, \ell_d^{(k)})\}$ for $k = 1, \dots, N_{\text{data}}$, and we can compute the gradient of the LL using (1.10), where the empirical distribution is now

$$p_{\text{data}}(\boldsymbol{v}, \ell) = \frac{1}{N_{\text{data}}} \sum_{k=1}^{N_{\text{data}}} \delta\left(\boldsymbol{v} - \boldsymbol{v}_d^{(k)}\right) \delta\left(\ell - \ell_d^{(k)}\right). \tag{1.56}$$

Now, in a realistic and interesting setting, it happens that only a fraction of the data is associated with a label. For example, experimentally classifying proteins according to their biological function is a difficult, time-consuming and expensive task, and therefore only a few proteins among the whole genome have been classified in a laboratory. How can we handle the fact that only a fraction of the training data is endowed with a label?

When the label is missing, we have to infer it together with the latent representation of the data. The gradient of the LL with respect to the parameter θ_i is now written as

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \left\langle \langle \mathcal{C}_{\theta_i} \rangle_{p(\boldsymbol{h}, \ell | \boldsymbol{v})} \right\rangle_{\mathcal{D}} - \langle \mathcal{C}_{\theta_i} \rangle_E \tag{1.57}$$

where the negative term of the gradient is computed as usual, while the positive term is averaged over

$$p(\boldsymbol{h}, \ell | \boldsymbol{v})\ p_{\text{data}}(\boldsymbol{v}) = \frac{1}{N_{\text{data}}} \sum_{k=1}^{N_{\text{data}}} p(\boldsymbol{h}, \ell | \boldsymbol{v}_d^{(k)})\ \delta\left(\boldsymbol{v} - \boldsymbol{v}_d^{(k)}\right). \tag{1.58}$$

In order to sample from $p(\boldsymbol{h}, \ell | \boldsymbol{v}_d^{(k)})$, we can run a MCMC using the alternating Gibbs Sampling method:

$$\begin{aligned} h_\mu^{(k)}(t+1) &\sim p(h_\mu | \ell^{(k)}(t), \boldsymbol{v}_d^{(k)}) = \text{sigmoid}\left(b_\mu + I_\mu(\boldsymbol{v}_d^{(k)}, \ell^{(k)}(t))\right), \\ \ell^{(k)}(t+1) &\sim p(\ell | \boldsymbol{h}^{(k)}(t+1)) = \text{softmax}_\ell\left(c_\ell + I_\ell(\boldsymbol{h}^{(k)}(t+1))\right), \end{aligned} \tag{1.59}$$

for $t = 1, \dots, N_{\text{Gibbs}}$.

The update rule for the centered gradient, valid only for the case of a Bernoulli hidden potential, becomes

$$\begin{cases} \Delta w_{i\mu}^q = \left\langle (\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}})(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}) \right\rangle_{\mathcal{D}} - \left\langle (\delta_{v_i,q} - \langle \delta_{v_i,q} \rangle_{\mathcal{D}})(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}) \right\rangle_E \\ \Delta d_{\ell\mu} = \left\langle (\delta_{\ell,\ell_d} - \langle \delta_{\ell,\ell_d} \rangle_{\mathcal{D}})(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}) \right\rangle_{\mathcal{D}} - \left\langle (\delta_{\ell,\ell_g} - \langle \delta_{\ell,\ell_d} \rangle_{\mathcal{D}})(h_\mu - \langle h_\mu \rangle_{\mathcal{D}}) \right\rangle_E \\ \Delta a_i^q = \langle \delta_{v_i,q} \rangle_{\mathcal{D}} - \langle \delta_{v_i,q} \rangle_E - \sum_\mu \Delta w_{i\mu}^q \langle h_\mu \rangle_{\mathcal{D}} \\ \Delta b_\mu = \langle h_\mu \rangle_{\mathcal{D}} - \langle h_\mu \rangle_E - \sum_{iq} \Delta w_{i\mu}^q \langle \delta_{v_i,q} \rangle_{\mathcal{D}} - \sum_\ell \Delta d_{\ell\mu} \langle \delta_{\ell,\ell_d} \rangle_{\mathcal{D}} \\ \Delta c_\ell = \langle \delta_{\ell,\ell_d} \rangle_{\mathcal{D}} - \left\langle \delta_{\ell,\ell_g} \right\rangle_E - \sum_\mu \Delta d_{\ell\mu} \langle h_\mu \rangle_{\mathcal{D}} \end{cases} \tag{1.60}$$

where $\langle \delta_{\ell,\ell_d} \rangle_{\mathcal{D}}$ represents the empirical frequency of the label ℓ in the data, while $\left\langle \delta_{\ell,\ell_g} \right\rangle_E$ is the empirical frequency of the label ℓ computed on the labels generated by the model through Equation (1.59).

Once the RBM has been trained with the labelled dataset, there are two things that we might want to do. First, given a real-world example $\boldsymbol{v}_d^{(k)}$, we would like to ask the model what label to assign to it. This is readily obtained by iterating Equations (1.59) conditioned only on $\boldsymbol{v}_d^{(k)}$. Notice that this method, by construction, is only capable of yielding a category among those used for the training. In other words, no new categories are found by the algorithm.

The second task we aim to tackle is, given a label $\tilde{\ell}$, we want the model to generate a sample belonging to that category. To do this, we can iterate the following simple Algorithm 3.

Algorithm 3 Conditioned sampling of data

Input: Initial random state: $\boldsymbol{v}_0$, Number of steps: N_{Gibbs}
Output: Final state: $\boldsymbol{v}|\tilde{\ell} \sim p(\boldsymbol{v}|\tilde{\ell})$
for $t = 1, \dots, N_{\text{Gibbs}}$ **do**
  sample $\boldsymbol{h}_t \sim p(\boldsymbol{h}|\boldsymbol{v}_{t-1}, \tilde{\ell})$
  sample $\boldsymbol{v}_t \sim p(\boldsymbol{v}|\boldsymbol{h}_t)$
end for

1.5 Theoretical Insights

In this section, we want to summarize some theoretical results that the physics of disordered systems' community has achieved toward the understanding of the learning in RBMs. First of all, we will prove that the introduction of hidden variables into the model allows, in principle, to reproduce the empirical correlations in the data potentially up to any k-body order. Secondly, we will present a study based on the replica theory that permits investigating the thermodynamic properties of the RBM and the different phases in which the model can be found. Finally, in the third and fourth subsections, we will focus on the linear regime of the training for understanding what happens when the RBM starts to learn from the dataset.

1.5.1 Expressivity of the RBM

First of all, let use justify the claim we made in the previous chapter about the improvement brought to the expressivity of the model by the introduction of hidden variables. For simplicity, let us consider the binary-binary RBM without local biases defined by the energy function

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{w}) = -\sum_{i\mu} v_i w_{i\mu} h_\mu, \tag{1.61}$$

with $v_i \in \{0, 1\}$ and $h_\mu \in \{0, 1\}$. We are interested in the probability distribution that the model assigns to the physical variables only, the visible ones, which is obtained by marginalizing the joint probability distribution over the hidden variables:

$$\begin{aligned} p_{\boldsymbol{w}}(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p_{\boldsymbol{w}}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z_{\boldsymbol{w}}} \sum_{\boldsymbol{h}} \exp\left[\sum_\mu h_\mu \left(\sum_i w_{i\mu} v_i\right)\right] = \\ = \frac{1}{Z_{\boldsymbol{w}}} \prod_\mu \sum_{h=0,1} \exp\left[h\left(\sum_i w_{i\mu} v_i\right)\right] = \frac{1}{Z_{\boldsymbol{w}}} \prod_\mu \left[1 + \exp\left(\sum_i w_{i\mu} v_i\right)\right]. \end{aligned} \tag{1.62}$$

The exponential can be expandend in Taylor series,

$$\exp\left(\sum_i w_{i\mu} v_i\right) = 1 + \sum_i w_{i\mu} v_i + \frac{1}{2}\left(\sum_i w_{i\mu} v_i\right)^2 + \frac{1}{6}\left(\sum_i w_{i\mu} v_i\right)^3 + \dots,$$

and this yields a probability distribution $p_{\boldsymbol{w}}(\boldsymbol{v}) \propto \exp(-E_{\text{eff}}(\boldsymbol{v}; \boldsymbol{w}))$ defined by the effective Hamiltonian

$$\begin{aligned} E_{\text{eff}}(\boldsymbol{v}; \boldsymbol{w}) &= -\sum_\mu \log\left[2 + \sum_i w_{i\mu} v_i + \frac{1}{2}\left(\sum_i w_{i\mu} v_i\right)^2 + \frac{1}{6}\left(\sum_i w_{i\mu} v_i\right)^3 + \dots\right] \\ &\approx -\frac{1}{2}\sum_{i\mu} w_{i\mu} v_i - \frac{1}{8}\sum_{ij\mu} w_{i\mu} w_{j\mu} v_i v_j + \frac{1}{192}\sum_{ijkl\mu} w_{i\mu} w_{j\mu} w_{k\mu} w_{l\mu} v_i v_j v_k v_l + \dots \end{aligned} \tag{1.63}$$

As we can see, as opposed to the Hamiltonian (1.4), the effective energy function (1.63) contains infinite terms of interaction in the series expansion. This means that the associated probability distribution is able to catch correlations among visible variables (present in the data) up to any order. This result that we have heuristically justified here has been rigorously proven in [17].

1.5.2 Phase Diagram

In this part, we want to get some insights on the thermodynamics of the RBM. To this aim, we will follow the work presented in [6] and we remand to the main article for further details. Let us consider the Bernoulli-Bernoulli RBM defined by the Hamiltonian (1.15)

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\sum_i a_i v_i - \sum_\mu b_\mu h_\mu - \sum_{i\mu} v_i w_{i\mu} h_\mu.$$

The idea is to study the statistics of an ensemble of RBMs, each one characterized by a random matrix $\boldsymbol{w}$, that will provide us with a phase diagram describing the macroscopic behaviour of the model. The characterization of the phase diagram is based on the determination of the free energy in the thermodynamic limit, which can be obtained through replica computations [21]. To simplify the derivation, both the visible and the hidden units are interpreted as spin variables $v_i \in \{-1, 1\},\ h_\mu \in \{-1, 1\}$.

The per-node free energy of the system is given by

$$f(\boldsymbol{o}) = \lim_{L\to\infty} \frac{1}{L} \left(-\mathbb{E}_{\boldsymbol{w}} \log Z_{\boldsymbol{\theta}}\right), \tag{1.64}$$

where $\boldsymbol{o}$ are some order parameters, $L = \sqrt{N_h N_v}$ and $\mathbb{E}_{\boldsymbol{w}}$ indicates the average over the quenched disorder represented by the matrix $\boldsymbol{w}$. At the core of the replica computations there is the so-called *replica trick*, namely the observation that we can express the average of the logarithm in terms of the average of a replicated partition function:

$$\mathbb{E}_{\boldsymbol{w}} \log Z_{\boldsymbol{\theta}} = \lim_{n\to 0} \frac{\mathbb{E}_{\boldsymbol{w}} Z_{\boldsymbol{\theta}}^n - 1}{n}. \tag{1.65}$$

The typical approach to carry out the computations is to assume the entries of the matrix $\boldsymbol{w}$ to be i.i.d. according to the same probability distribution. However, this assumption does not take into account the fact that, during the learning, the weights become correlated between each other. Thus, as a better assumption, the authors consider instead a low-rank decomposition of the weight matrix of the form

$$w_{i\mu} = \sum_{\alpha=1}^{K} x_i^\alpha w_\alpha y_\mu^\alpha + r_{i\mu}, \tag{1.66}$$

where $K \ll L$ and $\boldsymbol{r}$ is a random matrix of i.i.d. elements distributed according to a centered Gaussian of variance σ^2/L. In this decomposition, we are assuming that the eigenvalues w_α encode the information learned by some dataset, while $\boldsymbol{x}^\alpha$ and $\boldsymbol{y}^\alpha$ are random vectors approximately corresponding to the left and right eigenvectors of the matrix $\boldsymbol{w}$. Hence, $\boldsymbol{r}, \boldsymbol{x}$ and $\boldsymbol{y}$ represent the quenched disorder of the system, and we have to average the replicated partition function over them.

The replicated partition function reads

$$Z_{\boldsymbol{\theta}}^n = \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left[\sum_{p=1}^{n} \left(\sum_i v_i^p a_i + \sum_\mu h_\mu^p b_\mu + \sum_{i\mu\alpha} v_i^p x_i^\alpha w_\alpha y_\mu^\alpha h_\mu^p + \sum_{i\mu} v_i^p r_{i\mu} h_\mu^p\right)\right], \tag{1.67}$$

where p is the replica index. The average over $\boldsymbol{r}$ gives a term

$$\begin{aligned}\mathbb{E}_{\boldsymbol{r}} \exp\left[\sum_{i\mu} r_{i\mu}\left(\sum_p v_i^p h_\mu^p\right)\right] &= \prod_i \prod_\mu \left[\frac{1}{\sqrt{2\pi\sigma}} \int_{-\infty}^{+\infty} \mathrm{d}r\ \exp\left(-\frac{L}{2\sigma^2} r^2 + r \sum_p v_i^p h_\mu^p\right)\right] = \\ &= \exp\left[\frac{\sigma^2}{2L} \sum_{i\mu} \left(\sum_p v_i^p h_\mu^p\right)^2\right] = \exp\left[\frac{\sigma^2}{2L}\left(nL^2 + \sum_{i\mu} \sum_{p\neq q} v_i^p v_i^q h_\mu^p h_\mu^q\right)\right].\end{aligned} \tag{1.68}$$

We can handle the second term in the exponent through a Hubbarad-Stratanovich (HS) transformation

$$\exp\left[\frac{\sigma^2}{2L}\sum_{i\mu}\sum_{p\neq q}v_i^p v_i^q h_\mu^p h_\mu^q\right] = \\ = \int\prod_{p\neq q}\frac{\mathrm{d}Q_{pq}\mathrm{d}\bar{Q}_{pq}}{2\pi}\exp\left[-\frac{L\sigma^2}{2}\sum_{p\neq q}\left(Q_{pq}\bar{Q}_{pq}-\frac{Q_{pq}}{N_v}\sum_i v_i^p v_i^q-\frac{\bar{Q}_{pq}}{N_h}\sum_\mu h_\mu^p h_\mu^q\right)\right]. \tag{1.69}$$

Similarly, we can disentangle the interaction term by means of another HS transformation:

$$\exp\left(\sum_{i\mu\alpha}v_i^p x_i^\alpha w_\alpha y_\mu^\alpha h_\mu^p\right) = \exp\left[L\sum_\alpha\left(\frac{1}{\sqrt{L}}\sum_i v_i^p x_i^\alpha\right)w_\alpha\left(\frac{1}{\sqrt{L}}\sum_\mu h_\mu^p y_\mu^\alpha\right)\right] = \\ = \exp\left(L\sum_\alpha v_\alpha^p w_\alpha h_\alpha^p\right) = \\ = \int\prod_\alpha\frac{\mathrm{d}m_\alpha^p\mathrm{d}\bar{m}_\alpha^p}{2\pi}\ \exp\left[-L\sum_\alpha w_\alpha(m_\alpha^p\bar{m}_\alpha^p-m_\alpha^p v_\alpha^p-\bar{m}_\alpha^p h_\alpha^p)\right], \tag{1.70}$$

where we introduced the projections of the states along the modes of the weight matrix

$$v_\alpha^p\equiv\frac{1}{\sqrt{L}}\sum_i v_i^p x_i^\alpha\quad\text{and}\quad h_\alpha^p\equiv\frac{1}{\sqrt{L}}\sum_\mu h_\mu^p y_\mu^\alpha. \tag{1.71}$$

By denoting, for simplicity,

$$\mathcal{D}\boldsymbol{m}\equiv\prod_\alpha\frac{\mathrm{d}m_\alpha^p\mathrm{d}\bar{m}_\alpha^p}{2\pi}\quad\text{and}\quad\mathcal{D}\boldsymbol{Q}\equiv\prod_{p\neq q}\frac{\mathrm{d}Q_{pq}\mathrm{d}\bar{Q}_{pq}}{2\pi}, \tag{1.72}$$

the average over the replicated partition function takes the form, up to constant terms,

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{r}}Z_{\boldsymbol{\theta}}^n = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}\sum_{\boldsymbol{v},\boldsymbol{h}}\int\mathcal{D}\boldsymbol{m}\mathcal{D}\boldsymbol{Q}\exp\left[\sum_{ip}v_i^p a_i+\sum_{\mu p}h_\mu^p b_\mu-L\sum_{\alpha p}w_\alpha(m_\alpha^p\bar{m}_\alpha^p-m_\alpha^p v_\alpha^p-\bar{m}_\alpha^p h_\alpha^p)+\right. \\ \left.-\frac{L\sigma^2}{2}\sum_{p\neq q}\left(Q_{pq}\bar{Q}_{pq}-\frac{Q_{pq}}{N_v}\sum_i v_i^p v_i^q-\frac{\bar{Q}_{pq}}{N_h}\sum_\mu h_\mu^p h_\mu^q\right)\right]. \tag{1.73}$$

Let us introduce the projections of the fields along the principal directions of the weights matrix:

$$a_\alpha\equiv\frac{1}{\sqrt{L}}\sum_i a_i x_i^\alpha\quad\text{and}\quad b_\alpha\equiv\frac{1}{\sqrt{L}}\sum_\mu b_\mu y_\mu^\alpha. \tag{1.74}$$

For simplicity, we will neglect the orthogonal components. Then, let's consider the part that depends on the visible variables. We can write:

$$\log\mathbb{E}_{\boldsymbol{x}}\sum_{\boldsymbol{v}}\prod_i\exp\left[\sum_p v_i^p a_i+\sqrt{L}\sum_{\alpha p}w_\alpha m_\alpha^p v_i^p x_i^\alpha+\frac{L\sigma^2}{2N_v}\sum_{p\neq q}Q_{pq}v_i^p v_i^q\right] = N_v A(\boldsymbol{m},\boldsymbol{Q})$$

with

$$A(\boldsymbol{m},\boldsymbol{Q})\equiv\log\left[\sum_{\boldsymbol{s}^p\in\{-1,1\}}\mathbb{E}_{\boldsymbol{x}}\exp\left(\frac{\sqrt{\kappa}\sigma^2}{2}\sum_{p\neq q}Q_{pq}s^p s^q+\kappa^{\frac{1}{4}}\sum_{\alpha p}(m_\alpha^p w_\alpha+a_\alpha)x^\alpha s^p\right)\right]. \tag{1.75}$$

Similarly, the contribution of the hidden variables is:

$$\log\mathbb{E}_{\boldsymbol{y}}\sum_{\boldsymbol{h}}\prod_\mu\exp\left[\sum_p h_\mu^p b_\mu+\sqrt{L}\sum_{\alpha p}w_\alpha\bar{m}_\alpha^p h_\mu^p y_\mu^\alpha+\frac{L\sigma^2}{2N_h}\sum_{p\neq q}\bar{Q}_{pq}h_\mu^p h_\mu^q\right] = N_h B(\bar{\boldsymbol{m}},\bar{\boldsymbol{Q}})$$

with

$$B(\bar{\boldsymbol{m}}, \bar{\boldsymbol{Q}}) \equiv \log \left[\sum_{\boldsymbol{s}^p \in \{-1,1\}} \mathbb{E}_{\boldsymbol{y}} \exp \left(\frac{\sqrt{\kappa}\sigma^2}{2} \sum_{p \neq q} \bar{Q}_{pq} s^p s^q + \kappa^{-\frac{1}{4}} \sum_{\alpha p} (\bar{m}_\alpha^p w_\alpha + b_\alpha) y^\alpha s^p \right) \right]. \tag{1.76}$$

In the previous equations, we denoted $\kappa \equiv N_h/N_v$, and we have rescaled the eigenvectors $\boldsymbol{x} \mapsto \boldsymbol{x}/\sqrt{N_v}$, $\boldsymbol{y} \mapsto \boldsymbol{y}/\sqrt{N_h}$ so as to consider an orthonormal base. In short, we can express the averaged replicated partition function as

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{r}} Z_{\boldsymbol{\theta}}^n = \int \mathcal{D}\boldsymbol{m} \mathcal{D}\boldsymbol{Q} \; e^{-nL\tilde{f}(\boldsymbol{m},\bar{\boldsymbol{m}},\boldsymbol{Q},\bar{\boldsymbol{Q}})} \tag{1.77}$$

with

$$\tilde{f}(\boldsymbol{m}, \bar{\boldsymbol{m}}, \boldsymbol{Q}, \bar{\boldsymbol{Q}}) = \frac{1}{n} \left[\sum_{\alpha p} m_\alpha^p w_\alpha \bar{m}_\alpha^p + \frac{\sigma^2}{2} \sum_{p \neq q} Q_{pq} \bar{Q}_{pq} - \frac{1}{\sqrt{\kappa}} A(\boldsymbol{m}, \boldsymbol{Q}) - \sqrt{\kappa} B(\bar{\boldsymbol{m}}, \bar{\boldsymbol{Q}}) \right]. \tag{1.78}$$

At this point, we first have to take the thermodynamic limit $L \to \infty$ with κ fixed and, subsequently, the replica limit $n \to 0$. In the thermodynamic limit we can evaluate the integral (1.77) through a saddle-point approximation, i.e. by approximating the integral with the maximum value of the integrand. Hence, the per-node free energy is formally given by

$$f(\boldsymbol{o}) = -\lim_{n \to 0} \lim_{L \to \infty} \frac{1}{nL} \left[e^{-nL\tilde{f}(\boldsymbol{m}^*, \bar{\boldsymbol{m}}^*, \boldsymbol{Q}^*, \bar{\boldsymbol{Q}}^*)} - 1 \right] = \tilde{f}(\boldsymbol{m}^*, \bar{\boldsymbol{m}}^*, \boldsymbol{Q}^*, \bar{\boldsymbol{Q}}^*), \tag{1.79}$$

where the order parameters are given by the integration variables computed at the saddle-point: $\boldsymbol{o} = (\boldsymbol{m}^*, \bar{\boldsymbol{m}}^*, \boldsymbol{Q}^*, \bar{\boldsymbol{Q}}^*)$.

To move further, we need to make an assumption over the configurations' space of the system. In particular, we restrict ourselves to the Replica Symmetric (RS) case, where we assume that the *overlap* between two replicas does not depend on the replica indices. In practice, this means assuming $Q_{pq} = q$, $\bar{Q}_{pq} = \bar{q}$ $\forall p, q$ and $m_\alpha^p = m_\alpha$, $\bar{m}_\alpha^p = \bar{m}_\alpha$ $\forall p$. Under this assumption, the A term of Equation (1.78) becomes

$$A(\boldsymbol{m}, \boldsymbol{q}) = \log \left[\sum_{\boldsymbol{s}^p \in \{-1,1\}} \mathbb{E}_{\boldsymbol{x}} \exp \left(\frac{\sqrt{\kappa}\sigma^2 q}{2} \sum_{p \neq q} s^p s^q + \kappa^{\frac{1}{4}} \sum_\alpha (m_\alpha w_\alpha + a_\alpha) x^\alpha \sum_p s^p \right) \right]. \tag{1.80}$$

We can disentangle the replicas in the quadratic term through a HS transformation. This yields:

$$\begin{aligned}
\exp \left(\frac{\sqrt{\kappa}\sigma^2 q}{2} \sum_{p \neq q} s^p s^q \right) &= \exp \left[\frac{1}{2} \left(\kappa^{\frac{1}{4}} \sigma \sqrt{q} \sum_p s^p \right)^2 \right] \exp \left[\frac{\sqrt{\kappa}\sigma^2 q}{2} \underbrace{\sum_p (s^p)^2}_{=n} \right] = \\
&= \exp \left(\frac{n\sqrt{\kappa}\sigma^2 q}{2} \right) \frac{1}{\sqrt{2\pi}} \int \mathrm{d}z \; \exp \left[-\frac{z^2}{2} + z\kappa^{\frac{1}{4}} \sigma \sqrt{q} \sum_p s^p \right] = \\
&= \exp \left(\frac{n\sqrt{\kappa}\sigma^2 q}{2} \right) \mathbb{E}_z \exp \left(z\kappa^{\frac{1}{4}} \sigma \sqrt{q} \sum_p s^p \right),
\end{aligned} \tag{1.81}$$

where $\mathbb{E}_z$ represents the average over a standard normal random variable z. Substituting this back into Equation (1.80), we get

$$\begin{aligned}
A(\boldsymbol{m}, \boldsymbol{q}) &= \frac{n\sqrt{\kappa}\sigma^2 q}{2} + n \; \mathbb{E}_{\boldsymbol{x},z} \log \left\{ \sum_{s \in \{-1,1\}} \exp \left[\kappa^{\frac{1}{4}} s \left(z\sigma\sqrt{q} + \sum_\alpha (m_\alpha w_\alpha + a_\alpha) x^\alpha \right) \right] \right\} = \\
&= \frac{n\sqrt{\kappa}\sigma^2 q}{2} + n \; \mathbb{E}_{\boldsymbol{x},z} \log \left[2 \cosh g(\boldsymbol{x}, z) \right],
\end{aligned} \tag{1.82}$$

where

$$g(\boldsymbol{x}, z) = \kappa^{\frac{1}{4}} \left(z\sigma\sqrt{q} + \sum_{\alpha} (m_\alpha w_\alpha + a_\alpha) x^\alpha \right). \tag{1.83}$$

Similarly, the B term becomes

$$B(\bar{\boldsymbol{m}}, \bar{\boldsymbol{q}}) = \frac{n\sqrt{\kappa}\sigma^2\bar{q}}{2} + n\ \mathbb{E}_{\boldsymbol{y},z} \log\left[2\cosh \bar{g}(\boldsymbol{y}, z)\right], \tag{1.84}$$

with

$$\bar{g}(\boldsymbol{y}, z) = \kappa^{\frac{1}{4}} \left(z\sigma\sqrt{\bar{q}} + \sum_{\alpha} (\bar{m}_\alpha w_\alpha + b_\alpha) y^\alpha \right). \tag{1.85}$$

Finally, the free energy of the system can be written as

$$f(\boldsymbol{o}) = \sum_{\alpha} w_\alpha m^*_\alpha \bar{m}^*_\alpha - \frac{\sigma^2}{2} q^* \bar{q}^* + \frac{\sigma^2}{2}(q^* + \bar{q}^*) - \frac{1}{\sqrt{\kappa}} \mathbb{E}_{\boldsymbol{x},z} \log\left[2\cosh g^*(\boldsymbol{x}, z)\right] - \sqrt{\kappa}\mathbb{E}_{\boldsymbol{y},z} \log\left[2\cosh \bar{g}^*(\boldsymbol{y}, z)\right]. \tag{1.86}$$

The saddle-point conditions yield the following self-consistent equations:

$$\begin{aligned} m^*_\alpha &= \kappa^{\frac{1}{4}} \mathbb{E}_{\boldsymbol{y},z} \left[y^\alpha \tanh\left(\bar{g}^*(\boldsymbol{y}, z)\right)\right] \quad & q^* &= \mathbb{E}_{\boldsymbol{y},z} \left[y^\alpha \tanh^2\left(\bar{g}^*(\boldsymbol{y}, z)\right)\right], \\ \bar{m}^*_\alpha &= \kappa^{-\frac{1}{4}} \mathbb{E}_{\boldsymbol{x},z} \left[x^\alpha \tanh\left(g^*(\boldsymbol{x}, z)\right)\right] \quad & \bar{q}^* &= \mathbb{E}_{\boldsymbol{x},z} \left[x^\alpha \tanh^2\left(g^*(\boldsymbol{x}, z)\right)\right], \end{aligned} \tag{1.87}$$

which can be solved numerically. In absence of biases, $\boldsymbol{a} = \boldsymbol{0},\ \boldsymbol{b} = \boldsymbol{0}$, we can identify three different phases:

- **A paramagnetic phase**: which corresponds to the solution $q = \bar{q} = 0$ and $m_\alpha = \bar{m}_\alpha = 0$. In this case the system is not magnetized toward any direction and the configurations have typically zero overlap. The free energy presents a single minimum.
- **A ferromagnetic phase**: where $m_\alpha, \bar{m}_\alpha \neq 0,\ q, \bar{q} \neq 0$. The configurations of the system condensate along one or more principal directions of the weight matrix, and the free energy presents multiple minima. This is the *recall* phase, where the RBM learns patterns from the data and is able to sample them.
- **A spin glass phase**: with $m_\alpha = \bar{m}_\alpha = 0$ but $q, \bar{q} \neq 0$. The system is trapped in one of the (exponentially) many minima of the free energy, and the states are totally uncorrelated with the principal modes of the weights matrix.

The lines separating different phases correspond to *second-order phase transitions*, and can be detected by performing a stability analysis of the linearized mean field equations. We do not reproduce here the study of the transition lines – which can be found in the reference article – and we limit ourselves at reporting a sketch of the phase diagram, which is shown in Figure 1.6. By looking at the phase diagram, we can now justify the already mentioned need of initializing the weights matrix with small couplings, in such a way to start the training outside the spin glass phase in which the system would find itself trapped otherwise.

Let us conclude this part by observing that the self-consistency equations (1.87) depend on the chosen distribution for the principal components of the weight matrix. In [6] the authors show that the properties of the ferromagnetic phase, the most interesting one, are indeed affected by the kurtosis of the chosen distribution. If we denote by γ thee relative kurtosis with respect to the normal distribution, we find that for $\gamma = 0$ only the strongest mode of the weights matrix is stable, and the system will condense along this single mode. If $\gamma < 0$ the weaker modes can be metastable, but the strongest mode remains the stable one. Finally, if $\gamma > 0$ the weaker modes become stable, and the system can condensate along many directions at the same time. This should correspond to the *compositional phase* described by Tubiana et al in [29], and it is characterized by having many attractors not too far from each other, so that one can easily move from one to the other by slightly changing the hidden representation.

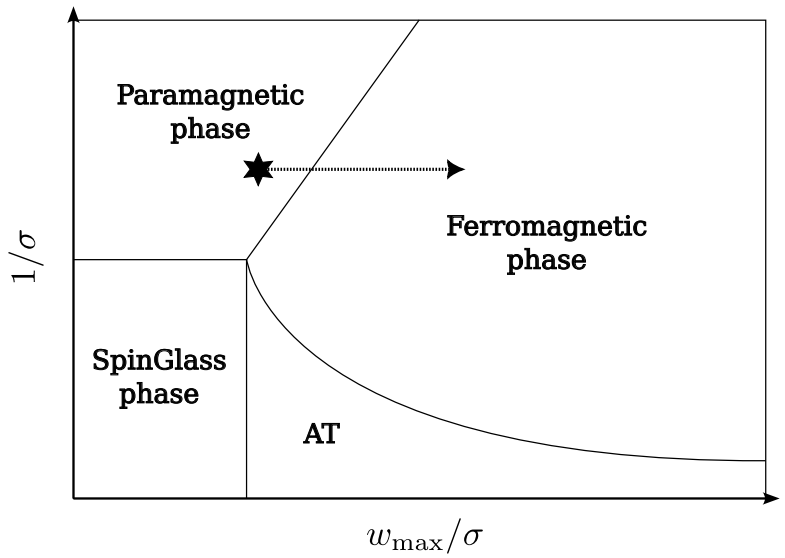

Figure 1.6: Phase diagram of the Bernoulli-Bernoulli RBM without biases. The two control parameters are $1/\sigma$, which can be interpreted as a temperature, and $w_{\max}/\sigma$, i.e. the rescaled maximum eigenvalue of the weights matrix, which plays the role of the couplings among the variables. The dotted arrow represents a typical trajectory of the learning, where the systems starts in the paramagnetic phase and, under the effect of the training, moves into the ferromagnetic region. For completeness, we also reported the AT region, where the RS assumption breaks down and one has to recover to the *One-step Replica Symmetry Breaking* (1RSB) ansatz. The figure is a reproduction of the one present in [6].

1.5.3 Learning Threshold of the RBM

In the previous part we have seen that the RBM starts to learn the features of the dataset when it moves from the paramagnetic to the ferromagnetic phase. Is there a marker that could tell us when this phase transition takes place? Let us consider a binary-binary RBM with variables $\{0, 1\}$. In section 3.1 we will show that, at the beginning of the training, the system can be described by the visible and hidden magnetizations, respectively denoted as $\boldsymbol{m}^v$ and $\boldsymbol{m}^h$, that satisfy the self-consistent equations

$$\begin{aligned} m_i^v &= \text{sigmoid}\left(a_i + \sum_\mu w_{i\mu} m_\mu^h\right), \\ m_\mu^h &= \text{sigmoid}\left(b_\mu + \sum_i w_{i\mu} m_i^v\right), \end{aligned} \tag{1.88}$$

which correspond to the traditional small-weights approximation of the Ising model. For simplicity, let us consider the case in which there are no external fields, $\boldsymbol{a} = \boldsymbol{b} = \boldsymbol{0}$, but the computation with the biases is not much more complicated. Again, we can express the weights matrix in terms of its singular values as

$$w_{i\mu} = \sum_\alpha x_i^\alpha w_\alpha y_\mu^\alpha, \tag{1.89}$$

and define the projected deviations from the mean of the magnetizations of the system along the mode α:

$$\delta\hat{m}_\alpha^h = \sum_\mu y_\mu^\alpha \left(m_\mu^h - \frac{1}{2}\right) \quad \text{and} \quad \delta\hat{m}_\alpha^v = \sum_i x_i^\alpha \left(m_i^v - \frac{1}{2}\right). \tag{1.90}$$

In this way, we get the following self-consistency equations projected along the mode α:

$$\begin{aligned}\delta\hat{m}^v_\alpha &= \sum_i x_i^\alpha \left[\text{sigmoid}\left(\sum_\beta x_i^\beta w_\beta \hat{m}^h_\beta\right) - \frac{1}{2}\right], \\ \delta\hat{m}^h_\alpha &= \sum_\mu y_\mu^\alpha \left[\text{sigmoid}\left(\sum_\beta y_\mu^\beta w_\beta \hat{m}^b_\beta\right) - \frac{1}{2}\right].\end{aligned} \tag{1.91}$$

In the early stage of the learning, we can assume the eigenvalues of the weight matrix to be small and expand the sigmoid function at the first order as

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}} \overset{x\to 0}{\approx} \frac{1}{2}\frac{1}{1-\frac{x}{2}} \overset{x\to 0}{\approx} \frac{1}{2} + \frac{x}{4}, \tag{1.92}$$

so that we end up with

$$\delta\hat{m}^v_\alpha = \frac{1}{4} w_\alpha \hat{m}^h_\alpha \quad \text{and} \quad \delta\hat{m}^h_\alpha = \frac{1}{4} w_\alpha \hat{m}^v_\alpha. \tag{1.93}$$

From these equations, we can see that as soon as $w_\alpha > w^{\text{th}} = 4$ the iterative map diverges exponentially, and the system polarizes along the mode α. On the contrary, if w_α is below this threshold, we have $\delta\hat{m}^h_\alpha = \delta\hat{m}^v_\alpha = 0$, and the system is not polarized. Figure 1.7 shows 20 iterations of the map (1.93) for values of w_α right below and above the threshold. The same reasoning shows that, in the case of spin variables $\{-1, 1\}$, the learning threshold is instead $w^{\text{th}} = 1$.

To sum up, in this subsection we showed that, for a binary RBM, the learning is triggered when $w_\alpha > 4$, in the sense that the model starts to encode patterns. We discuss these early stages in the next subsection.

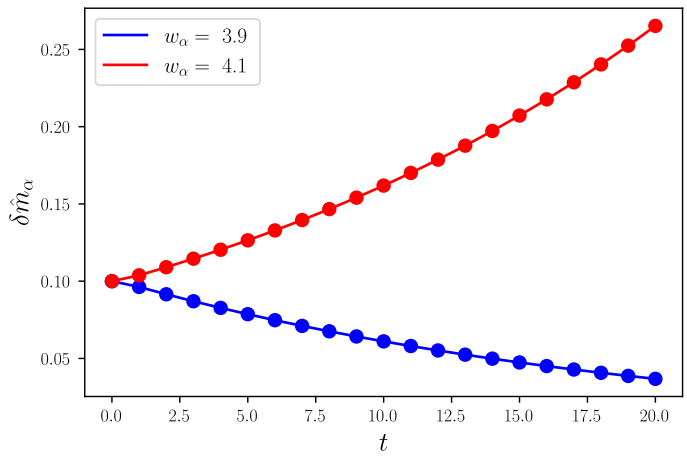

Figure 1.7: 20 steps of the linearized map (1.93) for two values of the eigenvalue w_α below and above the learning threshold $w^{\text{th}} = 4$. The x-axis represents the iteration time t, while the y-axis represent the deviation from the mean of average magnetization along the mode α, $\delta\hat{m}_\alpha = (\delta\hat{m}^h_\alpha + \delta\hat{m}^v_\alpha)/2$.

1.5.4 Early Dynamics of the Learning

In this part, we want to analyze the first stage of the learning in order to shed light on some aspects of how the RBM learns from the data. The analysis we present here is taken from [5].
At the beginning of the training, the machine has not learnt yet how to couple the variables, and therefore the eigenvalues of the weight matrix are small. In this sense, a proper initialization of the weight matrix is also needed, as can be seen by looking at the phase diagram of Figure 1.6. As will be stated more clearly in chapter 3, this is equivalent to assuming a high-temperature regime $\beta \to 0$

for which a mean field free energy can be derived.
At high temperature, the system finds itself in the paramagnetic phase, and whatever distribution over the variables will present one single mode. If we expand the distribution around this mode up to the second order, the magnetizations will have Gaussian fluctuations with covariance matrix

$$C = \begin{pmatrix} 1/\sigma_v^2 & -\boldsymbol{w} \\ -\boldsymbol{w}^T & 1/\sigma_h^2 \end{pmatrix}^{-1}, \tag{1.94}$$

where σ_v^2 and σ_h^2 are the prior variances of visible and hidden nodes. Using this approximation is equivalent to considering a Gaussian-Gaussian RBM, in which both visible and hidden units take real values and interact through the Hamiltonian

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\sum_i \frac{v_i^2}{2\sigma_v^2} - \sum_\mu \frac{h_\mu^2}{2\sigma_h^2} - \sum_{i\mu} v_i w_{i\mu} h_\mu, \tag{1.95}$$

where we are not considering any external field for simplicity. The model defined by the energy function (1.95) represents a special case in which we can write down the gradient of the LL exactly. The conditioned distribution of the hidden variables is a shifted Gaussian,

$$p(\boldsymbol{h}|\boldsymbol{v}) \propto \prod_\mu \exp\left(-\frac{h_\mu^2}{2\sigma_\mu^2} + h_\mu \sum_i v_i w_{i\mu}\right), \tag{1.96}$$

which is centered at $\langle h_\mu \rangle_{p(\boldsymbol{h}|\boldsymbol{v})} = \sigma_h^2 \sum_i v_i w_{i\mu}$.

The marginalized distribution of the visible units can be obtained as

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{v}) &= \frac{1}{Z} \exp\left(-\sum_i \frac{v_i^2}{2\sigma_v^2}\right) \prod_\mu \int \mathrm{d}h \exp\left(-\frac{h^2}{2\sigma_h^2} + h \sum_i v_i w_{i\mu}\right) = \\ &= \frac{1}{Z} \exp\left(-\sum_i \frac{v_i^2}{2\sigma_v^2}\right) \exp\left[\frac{\sigma_h^2}{2} \sum_{ij} v_i \left(\sum_\mu w_{i\mu} w_{j\mu}\right) v_j\right] = \\ &= \frac{1}{Z} \exp\left[-\boldsymbol{v}^T \left(\frac{1}{2\sigma_v^2} - \frac{\sigma_h^2}{2} \boldsymbol{w}\boldsymbol{w}^T\right) \boldsymbol{v}\right] = \frac{1}{Z} \exp\left(-\boldsymbol{v}^T \boldsymbol{A} \boldsymbol{v}\right), \end{aligned} \tag{1.97}$$

where we defined the precision matrix

$$\boldsymbol{A} = \frac{1}{2\sigma_v^2} - \frac{\sigma_h^2}{2} \boldsymbol{w}\boldsymbol{w}^T. \tag{1.98}$$

At this point, we rewrite the weights matrix $\boldsymbol{w}$ using the *Singular Value Decomposition* (SVD):

$$w_{i\mu} = \sum_\alpha x_i^\alpha w_\alpha y_\mu^\alpha, \tag{1.99}$$

where w_α are the singular values of the matrix and $\boldsymbol{x}^\alpha$, $\boldsymbol{y}^\alpha$ are the left and right eigenvectors. If we introduce the change of variables that projects the states of the system along the principal directions of $\boldsymbol{w}$,

$$\hat{v}_\alpha = \sum_i x_i^\alpha v_i \quad \text{and} \quad \hat{h}_\alpha = \sum_\mu y_\mu^\alpha h_\mu, \tag{1.100}$$

the Gaussian measure (1.97) factorizes as

$$p_{\boldsymbol{\theta}}(\hat{\boldsymbol{v}}) = \frac{1}{Z} \prod_\alpha \exp\left(-\frac{\hat{v}_\alpha^2}{2} \frac{1 - \sigma_v^2 \sigma_h^2 w_\alpha^2}{\sigma_v^2}\right). \tag{1.101}$$

The gradient of the LL projected along the $\alpha - \beta$ modes of the SVD maintains the usual form:

$$\begin{aligned}(\Delta \boldsymbol{w})_{\alpha\beta} &= \sum_{i\mu} x_i^\alpha \frac{\partial \mathcal{L}}{\partial w_{i\mu}} y_\mu^\beta = \sum_{i\mu} x_i^\alpha \left(\langle v_i h_\mu \rangle_{\mathcal{D}} - \langle v_i h_\mu \rangle_E \right) y_\mu^\beta = \\ &= \left\langle \hat{v}_\alpha \hat{h}_\beta \right\rangle_{\mathcal{D}} - \left\langle \hat{v}_\alpha \hat{h}_\beta \right\rangle_E . \end{aligned} \tag{1.102}$$

If we neglect the fluctuations of the stochastic gradient and we consider a vanishing learning rate such that the time interval between two updates can be considered infinitesimal, we can approximate $\Delta w_{i\mu} \sim \frac{\mathrm{d}w_{i\mu}}{\mathrm{d}t}$. This allows us to consider the time derivative of the weights matrix projected along its eigenmodes:

$$\begin{aligned}\left(\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} \right)_{\alpha\beta} &= \sum_{i\mu} x_i^\alpha \left(\frac{\mathrm{d}}{\mathrm{d}t} \sum_\gamma x_i^\gamma w_\gamma y_\mu^\gamma \right) y_\mu^\beta = \\ &= \sum_{i\mu\gamma} \left(x_i^\alpha x_i^\gamma \frac{\mathrm{d}w_\gamma}{\mathrm{d}t} y_\mu^\gamma y_\mu^\beta + x_i^\alpha \frac{\mathrm{d}x_i^\gamma}{\mathrm{d}t} w_\gamma y_\mu^\gamma y_\mu^\beta + x_i^\alpha x_i^\gamma w_\gamma \frac{\mathrm{d}y_\mu^\gamma}{\mathrm{d}t} y_\mu^\beta \right) = \\ &= \delta_{\alpha,\beta} \frac{\mathrm{d}w_\alpha}{\mathrm{d}t} + (1 - \delta_{\alpha,\beta}) \left[\left(\frac{\mathrm{d}\boldsymbol{x}^\beta}{\mathrm{d}t} \right)^T \boldsymbol{x}^\alpha w_\beta + w_\alpha \left(\frac{\mathrm{d}\boldsymbol{y}^\alpha}{\mathrm{d}t} \right)^T \boldsymbol{y}^\beta \right] = \\ &= \delta_{\alpha,\beta} \frac{\mathrm{d}w_\alpha}{\mathrm{d}t} + (1 - \delta_{\alpha,\beta}) \left(\Omega^x_{\beta\alpha} w_\beta + \Omega^y_{\alpha\beta} w_\alpha \right), \end{aligned} \tag{1.103}$$

where we introduced the rotation matrices

$$\Omega^x_{\alpha\beta} = \left(\frac{\mathrm{d}\boldsymbol{x}^\alpha}{\mathrm{d}t} \right)^T \boldsymbol{x}^\beta \quad \text{and} \quad \Omega^y_{\alpha\beta} = \left(\frac{\mathrm{d}\boldsymbol{y}^\alpha}{\mathrm{d}t} \right)^T \boldsymbol{y}^\beta, \tag{1.104}$$

which are anti-symmetric under the index exchange $\alpha \leftrightarrow \beta$. This equation shows that the gradient update of the weight matrix can be decomposed into two contributions: the evolution of the eigenvalues, and a rotation of the principal directions $\boldsymbol{x}$ and $\boldsymbol{y}$. The time evolution of the modes can be obtained through Equation (1.102):

$$\begin{aligned}\frac{\mathrm{d}w_\alpha}{\mathrm{d}t} &= \left(\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} \right)_{\alpha\alpha} = \left\langle \hat{v}_\alpha \hat{h}_\alpha \right\rangle_{\mathcal{D}} - \left\langle \hat{v}_\alpha \hat{h}_\alpha \right\rangle_E = \\ &= \sigma_h^2 w_\alpha (\langle \hat{v}_\alpha^2 \rangle_{\mathcal{D}} - \langle \hat{v}_\alpha^2 \rangle_E) = \\ &= \sigma_h^2 w_\alpha \left(\langle \hat{v}_\alpha^2 \rangle_{\mathcal{D}} - \frac{\sigma_v^2}{1 - \sigma_v^2 \sigma_h^2 w_\alpha^2} \right). \end{aligned} \tag{1.105}$$

If we imagine to keep $\boldsymbol{x}$ and $\boldsymbol{y}$ fixed, we see that the machine tries to match the variance of the visible units with the one of the dataset along the direction of the mode α. The stationary value of the mode is

$$w_\alpha^2 = \begin{cases} \frac{1}{\sigma_h^2} \left(\frac{1}{\sigma_v^2} - \frac{1}{\langle \hat{v}_\alpha^2 \rangle_{\mathcal{D}}} \right) & \text{if} \quad \langle \hat{v}_\alpha^2 \rangle_{\mathcal{D}} > \sigma_v^2 \\ 0 & \text{if} \quad \langle \hat{v}_\alpha^2 \rangle_{\mathcal{D}} \leq \sigma_v^2 \end{cases}, \tag{1.106}$$

from which we can see that, if the empirical variance of the data along a certain mode is smaller than the prior variance of the visible units, that mode gets suppressed.

To compute the time evolution of the rotation matrices, let us first notice that we can use Equation (1.103) for writing

$$\begin{aligned}\Omega^x_{\alpha\beta} &= -\frac{1}{w_\alpha + w_\beta} \left(\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} \right)^{\mathrm{A}}_{\alpha\beta} + \frac{1}{w_\alpha - w_\beta} \left(\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} \right)^{\mathrm{S}}_{\alpha\beta}, \\ \Omega^y_{\alpha\beta} &= \frac{1}{w_\alpha + w_\beta} \left(\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} \right)^{\mathrm{A}}_{\alpha\beta} + \frac{1}{w_\alpha + w_\beta} \left(\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} \right)^{\mathrm{S}}_{\alpha\beta}, \end{aligned} \tag{1.107}$$

where the superscripts A and S indicate respectively the anti-symmetric and symmetric part of the matrix. Let's now compute the right-hand side of Equation (1.102). The average over the data distribution for $\alpha \neq \beta$ is

$$\begin{aligned}\left\langle \hat{v}_\alpha \hat{h}_\beta \right\rangle_{\mathcal{D}} &= \left\langle \hat{v}_\alpha \left\langle \hat{h}_\beta \right\rangle_{p(\boldsymbol{h}|\boldsymbol{v})} \right\rangle_{p_{\text{data}}(\boldsymbol{v})} = \left\langle \hat{v}_\alpha \sum_\mu y_\mu^\beta \left\langle h_\mu \right\rangle_{p(\boldsymbol{h}|\boldsymbol{v})} \right\rangle_{p_{\text{data}}(\boldsymbol{v})} = \\ &= \sigma_h^2 \left\langle \hat{v}_\alpha \sum_\mu y_\mu^\beta \sum_i w_{i\mu} v_i \right\rangle_{p_{\text{data}}(\boldsymbol{v})} = \sigma_h^2 \left\langle \hat{v}_\alpha \sum_{\mu i \gamma} y_\mu^\beta y_\mu^\gamma w_\gamma x_i^\gamma v_i \right\rangle_{p_{\text{data}}(\boldsymbol{v})} = \\ &= \sigma_h^2 \left\langle \hat{v}_\alpha \sum_\gamma \delta_{\beta,\gamma} w_\gamma \left(\sum_i x_i^\gamma v_i \right) \right\rangle_{p_{\text{data}}(\boldsymbol{v})} = \sigma_h^2 w_\beta \left\langle \hat{v}_\alpha \hat{v}_\beta \right\rangle_{\mathcal{D}}, \end{aligned} \tag{1.108}$$

where

$$\left\langle \hat{v}_\alpha \hat{v}_\beta \right\rangle_{\mathcal{D}} = \sum_{ij} x_i^\alpha \left(\frac{1}{N_{\text{data}}} \sum_{k=1}^{N_{\text{data}}} v_i^{(k)} v_j^{(k)} \right) x_j^\beta. \tag{1.109}$$

Similarly, the average over the model distribution can be written as

$$\left\langle \hat{v}_\alpha \hat{h}_\beta \right\rangle_E = \sigma_h^2 w_\beta \left\langle \hat{v}_\alpha \hat{v}_\beta \right\rangle_E, \tag{1.110}$$

where the last average is over the distribution (1.101). However, since this distribution factorizes, there are no entries out of the diagonal, and this term do not enter in the definition of the $\boldsymbol{\Omega}$. From the expressions (1.107) is now easy to obtain

$$\begin{aligned}\Omega^x_{\alpha\beta} &= (1-\delta_{\alpha,\beta}) \frac{\sigma_h^2}{2} \left(\frac{w_\alpha - w_\beta}{w_\alpha + w_\beta} + \frac{w_\alpha + w_\beta}{w_\alpha - w_\beta} \right) \left\langle \hat{v}_\alpha \hat{v}_\beta \right\rangle_{\mathcal{D}}, \\ \Omega^y_{\alpha\beta} &= (1-\delta_{\alpha,\beta}) \frac{\sigma_h^2}{2} \left(-\frac{w_\alpha - w_\beta}{w_\alpha + w_\beta} + \frac{w_\alpha + w_\beta}{w_\alpha - w_\beta} \right) \left\langle \hat{v}_\alpha \hat{v}_\beta \right\rangle_{\mathcal{D}}. \end{aligned} \tag{1.111}$$

When inserting these expressions into (1.103), we find that the steady state is such that

$$0 = \sigma_h^2 w_\beta \left\langle \hat{v}_\alpha \hat{v}_\beta \right\rangle_{\mathcal{D}} \overset{(1.108)}{=} \left\langle \hat{v}_\alpha \hat{h}_\beta \right\rangle_{\mathcal{D}} \quad \text{for} \quad \alpha \neq \beta. \tag{1.112}$$

To recap, Equations (1.105) and (1.112) tell us that, in the early stages of the learning, the RBM:

1. Rotates the principal axes of the weight matrix, $\boldsymbol{x}$ and $\boldsymbol{y}$, so as to diagonalize the empirical covariance matrix of the dataset. In other words, the eigenvectors of $\boldsymbol{w}$ align with the principal directions of the dataset;
2. Along the principal directions, it suppresses those modes for which the empirical variance is smaller than the prior variance and adjusts those modes above the threshold until the limit value given by Equation (1.106) is reached.

This analysis allows us to say that, in the first part of the learning, the RBM is performing a sort of SVD of the dataset by just keeping the modes above a certain threshold. Then, as the learning evolves, it leaves the linear regime and the model starts to learn non-linear transformations of the data over which we cannot say much. For what concerns us, this results tell us that, at the beginning of the learning, the weight matrix "learns" the PCA of the dataset starting with the strongest modes, because the vectors of the SVD of $\boldsymbol{w}$ align with those of the PCA.

Chapter 2

Training and Testing RBMs

This chapter is dedicated to presenting the RBM from the point of view of the learning. The goal is to show how the properties of this model allow us to get a partial theoretical understanding of it and to have control over many aspects of the training. In the first section, we describe the three datasets used during this work and we justify their adoption. Sections 2, 3 and 4 are more technical, and there we discuss more in detail how to train an RBM and how to assess a trained model. Finally, in the last section, we present some experiments done with RBMs, with the aim of showing the capabilities of the model and its versatility in carrying out different tasks.

2.1 Datasets Description

In this section, we present the three datasets that we used for carrying out our numerical experiments during this work. We chose to consider real-world datasets with different properties. The MNIST dataset [18], which consists of images of handwritten digits, allows us to visualize the generated samples, hence allowing for a by-eye inspection of the results. The Human Genome dataset [1] represents genomic mutations with respect to a reference sequence, and it provides us with two-level labelling that gives us the possibility to check if the RBM is able to learn even sub-structures of the data (ethnic or population-level differences). Finally, the WW dataset [23, 14, 26], consisting of amino acidic sequences of proteins, is the most interesting for practical applications and gives us the opportunity of testing the Potts version of the RBM with more than two colors.

2.1.1 MNIST dataset

The MNIST dataset [18] consists of grey-scale images representing handwritten digits, from 0 to 9, of dimension 28×28 pixels. The full training set contains 60000 images, while the full test set is composed of 10000 images. Both datasets are completely annotated, namely, we have a label for each sample indicating the represented digit. For our experiments, we extracted 10000 samples from the training set and 2000 images from the test set. The images have been flattened and converted in binary format by setting to 1 the pixels with a value above a certain threshold and to 0 otherwise. In this way, the dataset we used was made of 784-dimensional binary vectors. Although dealing with binary variables, instead of using the binary version of the RBM, we employed the Potts RBM with a number of categories $N_q = 2$. The advantage of using this dataset is that, since it is made of images, we can visualize the generated samples and check their quality through a by-eye inspection. This is also a classical benchmark in Machine Learning. Some data extracted from the MNIST dataset are shown in Figure 2.1 A). In Figure 2.1 B) we show the projection of the dataset along the two main principal directions of the PCA. This plot shows that the MNIST dataset is not very much clusterized under linear projections, in the sense that different categories tend to overlap. Hence, a correct classification cannot be achieved from a clustering of the PCA projections.

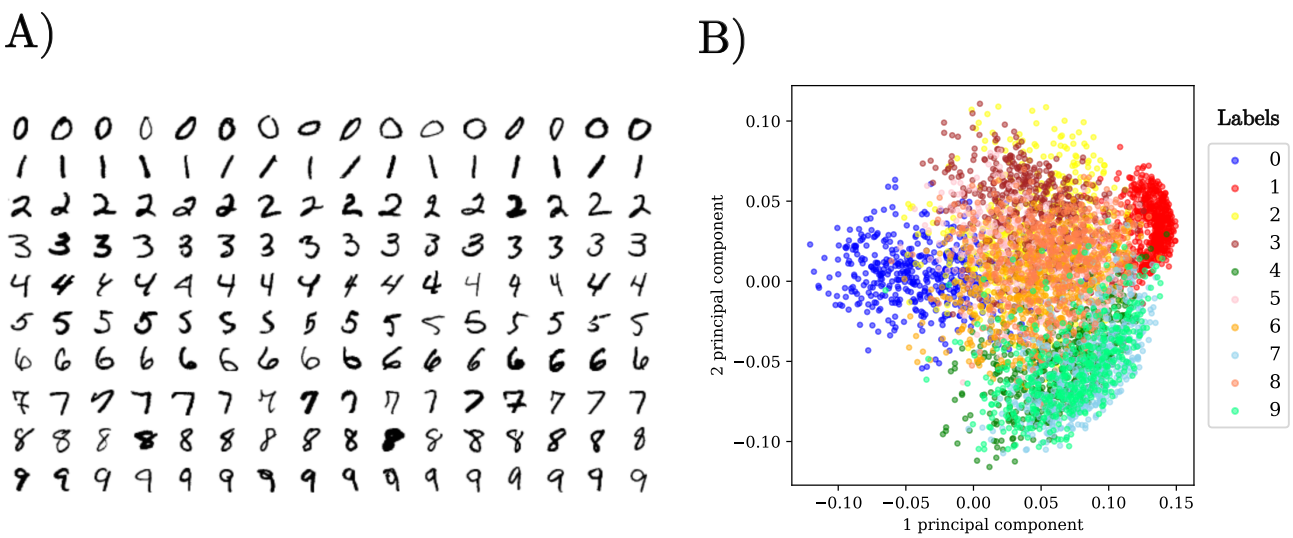

Figure 2.1: A): some examples extracted from the MNIST dataset. B): projection of 5000 samples from the training set along the firsts two principal components of the PCA. Samples corresponding to different digits are colored in different colors.

2.1.2 Human Genome (HG) Dataset

This dataset [1] represents human genetic variations of a population of 5008 individuals sampled from 26 populations in Africa, east Asia, south Asia, Europe and the Americas. Each sample is a sequence of 805 binary variables, $v_i \in \{0, 1\}$, representing the alteration or not of a gene with respect to a reference genetic sequence. These data present two interesting aspects. The first one is that, as visible in Figure 2.2, the dataset is quite clusterized, and this is often challenging for MCMC-based models [8, 2, 1]. The other peculiarity is that, for these data, we have a complete two-level hierarchical labelling. At the higher level, the sequences are classified based on the continental origin of the individuals, while at a finer level we also have a label for the particular ethnic population the data was sequenced from. This allows us to test the discriminative capabilities of the RBM at different resolutions. We reserved 90% of the data for the training set ($N_{\text{data}} = 4508$) and we kept out 500 samples for the test set.

2.1.3 WW domain dataset

The WW dataset (Pfam Domain accession code: PF00397) is the most interesting one from the point of view of the applications, because it allows us to test our RBM on a partially classified protein dataset. From a structural point of view, proteins are chains of amino acids that fold into a well-defined 3d structure. The structure of the protein determines its biological function, and it is thoroughly encoded into the amino acidic sequence. This means that, in principle, one should be able to determine the folding of a protein by just knowing the physical properties of its building blocks and how they interact. For instance, some amino acids are hydrophobic, and they tend to position internally into the structure, while others, which are hydrophilic, prefer to stay more exposed to the intracellular medium. However, in practice, deriving the folding of a protein by just stating from its amino acid sequence is a formidable task, and only recently it has been addressed with good accuracy by means of machine learning methods [15]. This paradigmatic example shows that, instead of trying to derive the properties of a protein by first principles, it is often convenient to exploit the statistics of large datasets of *homologous* (proteins of common evolutionary ancestry) sequences to learn some emergent properties. In fact, two homologous proteins can differ in more than 70% of the amino acidic sequence, and yet present a very similar 3d structure. This means that, in nature, we can find a huge number of different sequences that, once folded, show up the same protein. Those sets of sequences constitute *protein families*, and they encode the evolutionary constraints that keep the protein functional [3].

Usually, large proteins are obtained by concatenating *domains*, which are functional units that can fold independently of the rest of the sequence. The WW is one of the smallest domains, and it serves

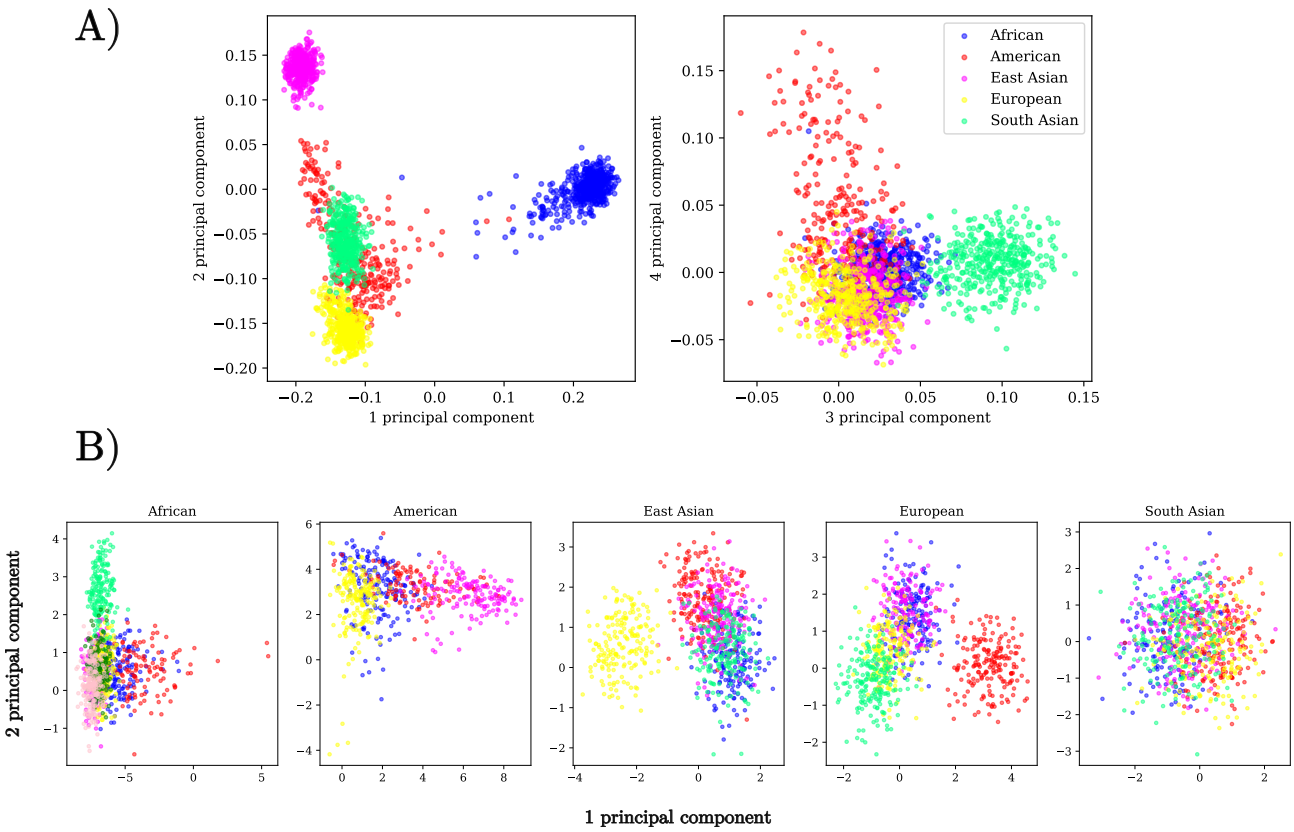

Figure 2.2: A): Projection of the HG dataset along the 1st and 2nd principal directions of the PCA (left) and along the 3rd and 4th principal directions (right). Samples coming from different regions are shown in different colors. The dataset appears to be clustered, and some ethnic groups are well distinguishable already at the PCA level. B) Again, we project the data along the first two principal components but we show separately data sampled in different regions. We highlight different ethnics in in different colors.

as a protein-protein interaction mediator. The domain does not show up in every protein the same, because evolution produced a huge family of variants of the ancestor sequence (homologous). Those sequences differ from one another for some substitutions or insertions and deletions of amino acids. In particular, the WW domain can be classified into different groups depending on the binding affinity it has with different motifs.

Apart from reconstructing protein folding, machine learning can be used in this context for several other purposes, such as protein annotation. In fact, even if we had the 3d structure, still we are not able to determine the exact function of the protein and its specificity, such as the binding affinity in the case of the WW domain. Traditionally, proteins' annotation based on their biological function has been carried out in a laboratory, with lots of costs in terms of time and resources. In the last years though, machine learning models, among which RBMs, started entering the tools of bioinformatics with the purpose of classifying proteins based on the statistical similarity among sequences [29].

To be processed by the RBM, all sequences have first to be aligned through a procedure called *Multiple Sequence Alignment* (MSA). In fact, because of the presence of insertions and deletions of sites, the length of the sequences may change, whereas the RBM's structure requires fixed-length input vectors. There are several methods for performing the MSA, but the most common works by starting from the most similar pair of sequences, i.e. those with the largest number of shared sites, and progressing until the most distantly related. In this process, the algorithm inserts *gaps* within the sites of the sequences in order to best match the position of the amino acids with the ones of the growing pile. To perform the alignment, we used the MAFFT software (version 7) [25]. After performing the MSA, the dataset was made of 15752 sequences of $N_v = 37$ sites. Among these, we kept 10% of the data for the test set, and we trained the model over the remaining 90%. Each site is a categorical variable that can take one of the $N_q = 21$ possible values, where $q = 0$ is reserved for the gaps and each of

the remaining values $q \in \{1, \dots, 20\}$ is associated to an amino acid. For this dataset, we have some sequences annotated by experimental works [23, 14, 26] and some labels which have been inferred by a previous study using an algorithm called *ProfileView* [31]. Figure 2.3 shows some of the data represented using the PCA with super-impressed the experimentally annotated data, which are a very reduced fraction of the total number of sequences (61 experimentally-classified sequences in total).

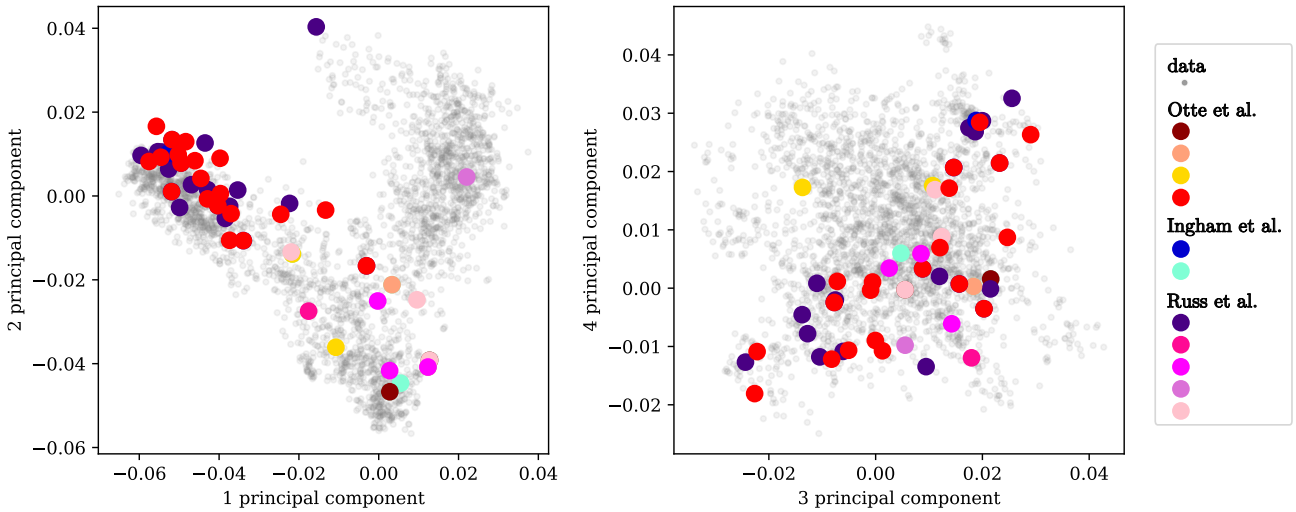

Figure 2.3: Projection of the WW dataset on 1st vs 2nd (left) and 3rd vs 4th (right) principal directions of its PCA. The data points are in grey, while the colored dots represents the experimental annotations made by the various authors.

2.2 Approximation of the Negative Term

In Algorithm 2 we outlined the procedure for sampling configurations from the joint probability distribution of the RBM, $p_{\boldsymbol{\theta}}(\boldsymbol{v}, \boldsymbol{h})$, in order to estimate the negative term of the gradient of the LL (or, alternatively, to generate samples once the training is finished). In the SGA scheme, for each minibatch of data we initialize N_{chains} MCMC chains and we update them in parallel using the block sampling method for k steps. The set of final points of the chains will be then used for evaluating the average in the negative term. It remains unsaid, though, how to choose the initial conditions of the Markov Chains, $\{(\boldsymbol{v}_0, \boldsymbol{h}_0)_r\}_{r=1,\dots,N_{\text{chains}}}$, at each gradient's update. The classical choices are:

- **Contrastive Divergence (CD)-k**: This method was first proposed by Hinton [12], and it consists in initializing the MC chains on the data of the minibatch and then performing $k \sim \mathcal{O}(1)$ MCMC updates. This means that N_{chains} must correspond to the chosen batch size. The idea is that the data should be close to the equilibrium distribution of the model, and therefore starting from them should allow to cover the entire distribution. The problem with this method is that it prevents the chains from exploring regions of the states' space that lay far from the dataset, thus limiting the generalization capabilities of the learned RBM. It has been shown [9] that RBMs trained with CD-k do not produce good quality samples, especially when k is small. The reason is that this method creates local free energy wells around the datapoints, which get separated by high free energy barriers. This decreases progressively the ergodicity of the Markov Chains as training progresses, thus degrading the quality of the approximation of the LL's gradient.

- **Persistent CD (PCD)-k**: In this scheme, N_{chains} chains are initialized randomly at the beginning of the training. Then, at each gradient's update, the chains are initialized with the final values of the previous update, and k new steps are performed. This is equivalent of collecting samples every k steps from a very long MCMC chain that runs throughout all the learning. It has been observed that this method is able to generate samples very similar to the data of the dataset, with the advantage with respect to the previous method to allow for a broader exploration of the configurations' space. This method seems to be quite good in the first part of the training, when the mixing time of the chains is small. However, in the advanced phase of

the training, the energy landscape becomes steeper and the mixing time increases. This means that the chains might get stuck in the local minima of the free energy, hence over-representing certain modes of the distribution and introducing a distortion in the estimate of the negative term of the gradient that undermines training. Running many chains in parallel mitigates this problem, but it does not completely solve it.

- **Parallel Tempered MCMC (PT)-k**: This method was first introduced in [9] for overcoming the aforementioned problem with PCD. The idea is to introduce an ordered set of temperatures $T_0 < T_1 < \cdots < T_{K-1} < T_K$, with $T_0 = 1$, and to put a replica of the model at each of those temperatures. Then, we run in parallel a PCD chain for each of them. At high temperatures it is easier to explore the energy landscape and the PCD chains mix well, but the probability distribution we want to sample from is the one defined at T_0. One can thus define a new Markov Chain MC process where the temperature of neighbouring chains, T_k and T_{k+1}, is interchanged once in a while. Detailed balance is guaranteed if the proposal is accepted with the following rule

$$r = \exp[(\beta_k - \beta_{k+1})(E(\boldsymbol{v}_k, \boldsymbol{h}_k; \boldsymbol{\theta}) - E(\boldsymbol{v}_{k+1}, \boldsymbol{h}_{k+1}; \boldsymbol{\theta}))], \tag{2.1}$$

where $\beta_k = 1/T_k$. Hence, this algorithm gives to the chains at low temperatures the possibility to escape from local minima of the free energy and forget the past history. Eventually, the particles used for evaluating the negative term of the gradient are those at T_0.

Although in many cases this algorithm performs very well and allows for a fast thermalization of the MC chains, it nevertheless presents some drawbacks. First of all, the algorithm is quite heavy to run, since we have to simulate $K + 1$ processes in parallel among which only one (the RBM at $T_0 = 1$) is used for extracting the samples. In the second place, PT performs very poorly in highly-clustered datasets, namely when samples gather in few high-density regions of the data space separated by large empty regions. The reason for this is that, for those kinds of datasets, the annealing in temperature passes through a *first-order phase transition* that separates very different configurations of the system. In that case, the exchange probability becomes too small and the performances of the algorithm drop down. A pictorial representation of this phenomenon is depicted in Figure 2.4.

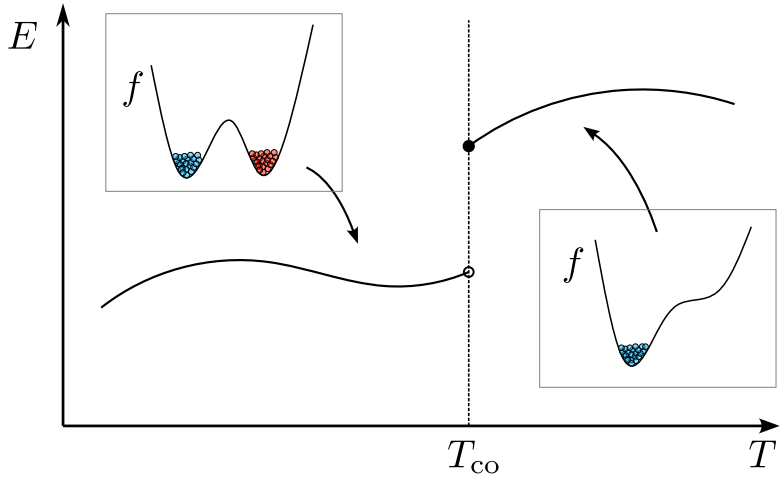

Figure 2.4: Sketch of a first-order phase transition in 1d appearing when changing the temperature of the model. In correspondence with a (possibly non-unique) coexistence point T_{co}, the energy function presents a discontinuity, which separates two regions in which the configurations of the system are very different. When crossing the transition temperature, the free energy of the system, f, abruptly presents a new stable minimum which is separated by the others by high free energy barriers. The colored dots represent MC chains running in parallel for a model at a given temperature. The model at temperature $T < T_{\text{co}}$ displays a high probability region of the data space that is not present in the models at $T > T_{\text{co}}$, hence determining a low exchange probability since the configurations are very different. Another way of saying this is that chains at the two sides of the transition line do no longer interchange temperatures, because what is stable at one side is no good at the other side. The "thermalizing" flow gets broken and there is no net improvement.

- **Random (Rdm)-k**: This setting consists on initializing the MC chains with random samples at each gradient's update and then performing k steps. In [8] it has been shown that an RBM trained with this scheme learns a dynamical process in which, starting from the same initial conditions and performing k MC steps, the model generates good samples similar to the data. However, if the initialization of the chain or the rules of the dynamics are different from the ones used for the training, the quality of the generated samples gets sensitively worse. The reason of this behaviour is that, as the training proceeds, the mixing time of the Markov Chains increases. This means that quite soon the number of MC steps k is not sufficient for reaching equilibrium configurations, and the learning becomes a process out of equilibrium in which the RBM memorizes the particular dynamics that allows it to produce good-quality samples.

2.3 Scores for Monitoring the Learning

In order to assess the quality of the learned model, we have to introduce some observables that compare the properties of the training/test datasets with the ones of the samples generated by the RBM during the learning. Traditionally, the most used measure for evaluating the performance of a trained RBM is the Log-Likelihood. As we already discussed, this cannot be computed exactly, but it is still possible to approximate it by estimating the partition function using the Annealed Importance Sampling method [16]. The vast majority of the works in the literature until recently relied uniquely on this measure. However, in [8] it has been shown that the LL alone is quite unreliable for assessing the generative power of an RBM, since it might happen that images generated with models with a high LL are quite bad under visual inspection, and the opposite might happen too (low LL machines can produce high-quality samples). On top of this, in unsupervised machine learning it is generally hard to assess the quality of the generative model. In the Machine Learning community, people nowadays use measures such as the *Frechet Inception Distance* [11] or the *Inception Score* [27]. While these two measures can have benefits and drawbacks, we excluded them because they rely on yet another trained deep neural network and they are in general used for images. Since we will not be dealing only with images, these measures would not be very useful, nor we would be satisfied with a measure based on an uninterpretable complex function. For these reasons, in the experiments we performed we decided not to use the LL nor the aforementioned scores as quality indicators, and we propose instead the following observables.

- **Error in the energy $\boldsymbol{\epsilon^E}$**: It is given by the Mean Squared Error (MSE) between the energy function of the model computed on the data and the one evaluated on the generated samples. Since at equilibrium the generated set and the dataset should be statistically indistinguishable, we expect this score to go to zero for properly trained models and adequate generation times.

- **Error of the first moment $\boldsymbol{\epsilon^{(1)}}$**: We want to check that the empirical frequency of the color $q \in \{1, \dots, N_q\}$ at the site $i \in \{1, \dots, N_v\}$ is the same between the dataset and the generated set. This score is the MSE between the two sets of frequencies.

- **Error of the second moment $\boldsymbol{\epsilon^{(2)}}$**: With this observable we want to see if the empirical second moment of the dataset has been learned properly. It is given by the MSE between the covariance matrix of the dataset and the one computed on the generated samples.

- **Error on the spectrum $\boldsymbol{\epsilon^{\mathbf{S}}}$**: After performing the SVD of the dataset and the generated set, we compute the MSE between the two ordered sets of eigenvalues.

- **Error in the Adversarial Accuracy Indicator (AAI) $\boldsymbol{\epsilon^{\mathbf{AAI}}}$**: This score [4] aims at assessing the ability of the generative model to produce data which are statistically identical to the ones of the dataset, but without merely copying them. The idea is to compute a matrix of (Euclidean) distances on the set of points obtained by merging the dataset and the generated set. Then, for each point, we check whether its nearest neighbor belongs to the dataset or to the generated set. In this way we can obtain two indicators: $\epsilon^{\mathrm{AAI}}_{\mathrm{gen}}$, which measures the probability that the nearest neighbor of a generated sample is a generated sample, and $\epsilon^{\mathrm{AAI}}_{\mathrm{data}}$, which represents the probability of a sample from the dataset to have as a nearest neighbor another sample from the

dataset. The optimal condition is when $\epsilon_{\text{gen}}^{\text{AAI}} = \epsilon_{\text{data}}^{\text{AAI}} = 0.5$, because it means that the two sets of points are statistically very similar, but not overlapping. Notice that this score is also a good indicator for detecting overfitting, because if the generated samples were identical to the dataset samples we would have $\epsilon_{\text{gen}}^{\text{AAI}} = \epsilon_{\text{data}}^{\text{AAI}} = 0$, which is far from optimality.

2.4 Equilibrium and Non-equilibrium Properties

In Figure 2.5 we show the aforementioned scores as a function of the generation time and for different ages of the RBM in two different settings: one in which the training was carried out in Rdm mode, and the other in which the PCD mode was used instead. The first thing that catches the eye, is the very different profile that the score curves display in the two cases.

In the Rdm case, all scores indicate a well-defined generation time in which the performances of the machine are bests, and it corresponds to the number of Gibbs steps, N_{Gibbs}, that has been used for training the RBM ($t_{\text{G}} = 10$, in the present case). Generating the data earlier or later would result in worse outcomes. This phenomenon is a clear sign that the RBM has learnt a non-equilibrium process in which the LL had to be maximized in a specific number of steps starting from random initial conditions. In Figure 2.6, on the left, we show the generated images after $t_{\text{G}} = 10$ for different ages of the machine. Notice how the oldest RBM ($t_{\text{age}} = 5000$ epochs) is able – with only 10 MCMC steps – to generate very good images (such as the 5 and 6 digits in the 3^{rd} and 4^{th} rows) but also samples that do not resemble to any existing digit at younger ages.

For the RBM trained in PCD mode, instead, we see how, after an initial transient period in which the scores improve in an almost monotonic way, the MC Markov chains thermalize and the scores become stationary around their equilibrium value. The curves in Figure 2.5 assure us that, after $\mathcal{O}(10^4)$ updates, the chains are independent of the random initial conditions and the generated images are correctly sampled from the equilibrium probability distribution. In Figure 2.6, on the right, we report some images generated with $t_{\text{G}} = 10^5$ chain updates for different ages of the RBM, and one can appreciate how the oldest machine is capable of generating really good images. From Figure 2.5 we can also notice how the thermalization time increases with the age of the RBM, and this is devoted to the free energy barriers that become higher and higher as the training proceeds, so that the chains struggle exploring the data space. To help visualizing this point, we reported in Figure 2.7 an example of 5 chain's trajectories initialized at random. Clearly, the number of steps performed is too small for this RBM, because the chains are mostly confined in specific regions and did not have the time to visit the whole data space. The message is that we can improve the quality of the generated samples by training the machine for longer times, but this comes with the cost of having to wait longer and longer in the generation phase before sampling at equilibrium.

Let us observe how the oldest machines typically improve the scores we considered, but this is not always the case. In particular, it might happen that the majority of the scores assign a preference for certain models (normally the oldest ones), but some of them reward others instead. That is why it is important to use multiple metrics for assessing the quality of the training and to have a broad overview of the process.
As a final remark, we stress the fact that a properly balanced RBM must display one of the behaviours shown in Figure 2.5, depending on the training mode chosen. Atypical trends, such as the appearance of local minima at different generation times in the PCD mode, must be considered as an alarm bell for a wrong choice of hyper-parameters. Changing the number of hidden units or the learning rate properly should fix the problem.

2.5 Experiments in Semi-Supervised Mode

In this section, we want to test the capabilities of an RBM trained in semi-supervised mode and we want to study how the performances depend on the amount of labels that we use during the training. The results we present are obtained on the MNIST dataset using a Potts-Bernoulli RBM. We will

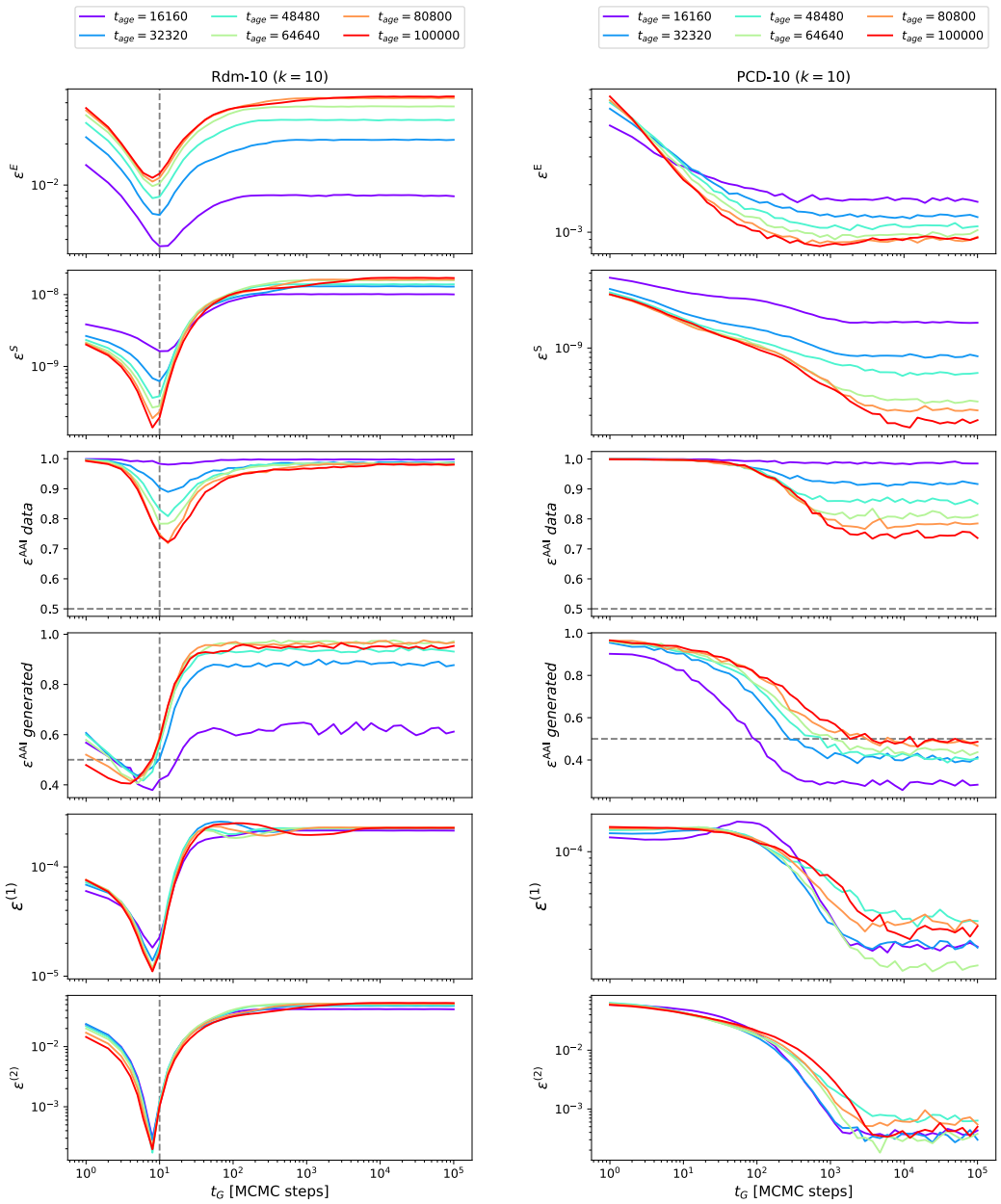

Figure 2.5: Scores profiles for two Potts-Bernoulli RBMs trained on the MNIST dataset, one in Rdm mode (leftmost column) and the other in PCD mode (rightmost column). The two machines share the same hyper-parameters: learning rate $= 10^{-3}$, $N_h = 500$, $N_{\text{Gibbs}} = 10$ ($k = 10$), minibatch size $= 500$. Each row shows the behaviour of one of the scores described in the main text with respect to the generation time t_G (number of MCMC steps for generating the images). The different colored lines correspond to different ages of the RBM expressed in gradient updates. The scores have been computed by comparing 2000 generated samples with the same number of images extracted from the test set, and all the chains have been initialized with random initial conditions.

profit from this part to also present some of the observables that can be monitored during the training of an RBM.

2.5.1 Observables of the Training

In Figure 2.8 we show how the eigenvalues of the weight matrix and of the label matrix evolve during the training. After about 100 epochs the first mode gets expressed, and the RBM enters the ferromagnetic phase. Then, also the other eigenvalues reach the learning threshold and the other modes of the distribution are learnt, first gradually and then many at the same time. We underlie the fact that the learning threshold is not 4 as derived in subsection 1.5.3, because here are present also

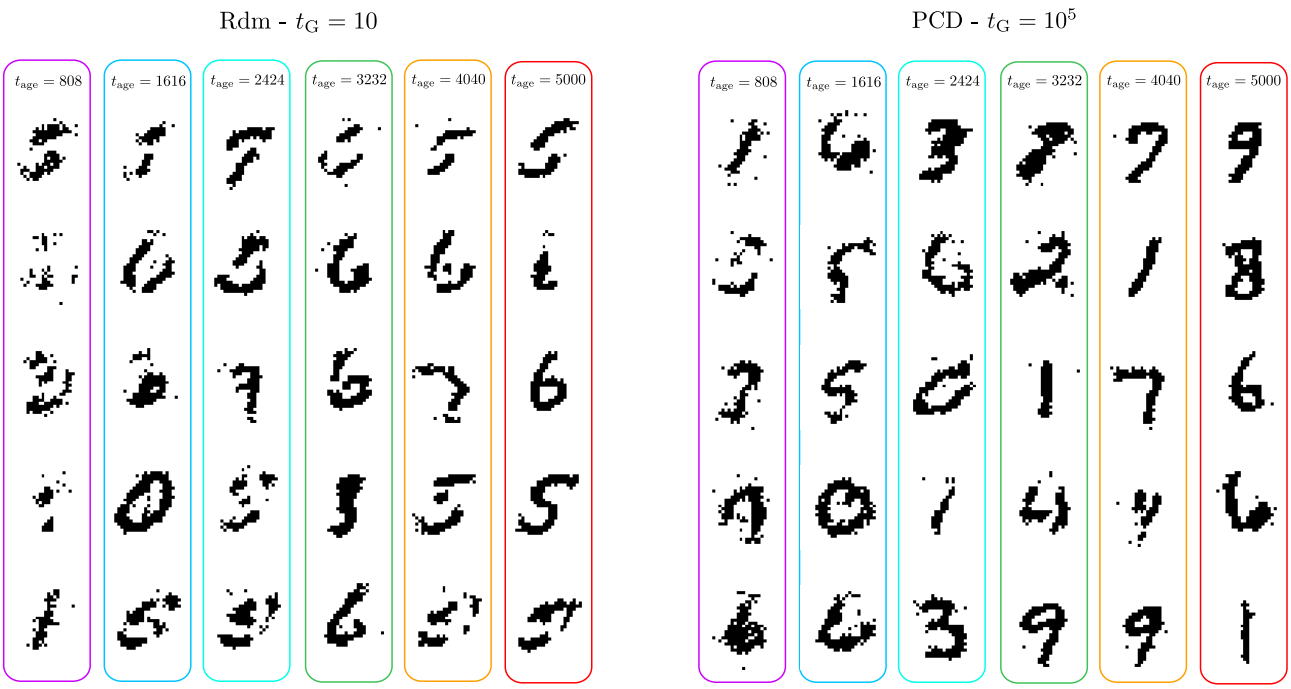

Figure 2.6: Examples of 5 images generated by the RBM trained in Rdm mode (left) and the one trained in PCD mode (right) at different ages. In the Rdm case, we generated the images using $t_G = 10$ MCMC steps, which is where the scores are bests, while in the PCD case we waited for $t_G = 10^5$ MCMC steps, when we are sure that all machines have thermalized. The age of the RBM, t_{age}, is here expressed in terms of epochs. Each epoch corresponds to 20 gradient updates for the MNIST dataset.

the external fields and the structure of the weight matrix is different since we are using Potts variables with 2 colors.

The features learnt by the machine can be visualized into the 20 randomly selected feature maps shown in Figure 2.9, which represent the filters that act on the visible layer to produce the hidden representation. We can recognize several profiles of digits impressed on top of each other. Also, we can notice how some of the features are quite localized since they are characterized by small intense regions. Comparing the two rows, we see that there is a clear correlation between $w^0_{i\mu}$ and $w^1_{i\mu}$.

2.5.2 Data Clustering

A trained RBM can be used for subdividing data into different categories. The idea is the same as projecting the dataset along the principal directions of the PCA, but this time we can use the principal components of the weight matrix instead. Moreover, rather than doing this directly with the data, we might use instead the hidden representation that the machines assigned to the dataset. In fact, in many cases the hidden variables appear more clustered than the visible ones, hence allowing for a better classification. As we demonstrated in subsection 1.5.4, the result obtained for early ages of the RBM should be the same as performing a simple PCA of the data. This is nicely proved by comparing the panel B) of Figure 2.10 with the first plot in panel A). As the learning advances, the machine starts to learn non-linear transformations of the data, and the projections along the weight matrix start to differ from the PCA. At the very late stages of the learning, the projected data merge together and the clusters are almost indistinguishable. Empirically, one observes that the best performances in the data classification are reached when applying a clustering algorithm at the intermediate stages of the learning. For instance, this strategy has been applied in [29] for inferring protein annotations.

2.5.3 Label Inference and Conditioned Sampling

Once the RBM has been trained in semi-supervised mode, we can enquire it on the labels to assign to a presented image as described in section 1.4. In Figure 2.11 we tested the accuracy of the RBM

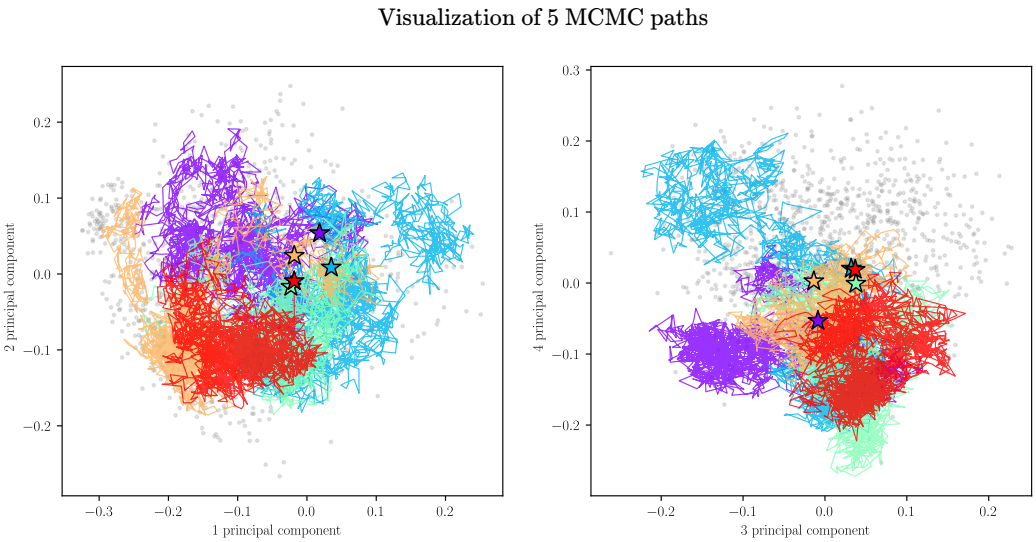

Figure 2.7: Five MCMC trajectories of 1000 steps with random initialization visualized on the PCA projection along the first and second principal components (left) and third and fourth principal components (right). The starting point of the each chain is indicated with a star. The grey points are the data, and the different plots represent the projections of the dataset onto the spaces defined by couples of PCA principal components. The figure shows a sampling that cannot be at equilibrium, because the chains visited only a small volume of the data space and are still correlated with their initial conditions.

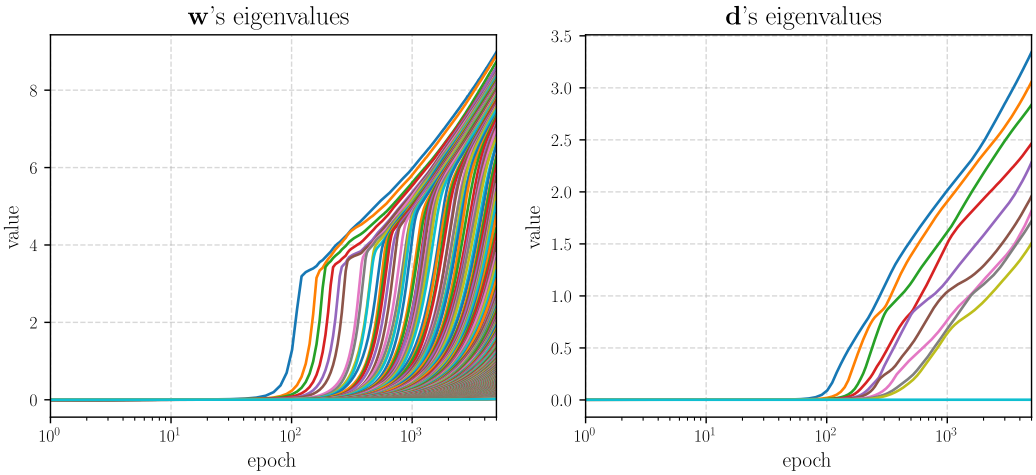

Figure 2.8: Evolution of the eigenvalues of the weight matrix (on the left) and of the label matrix (on the right) as a function of the training epochs. Each epoch correspond to 20 gradient updates.

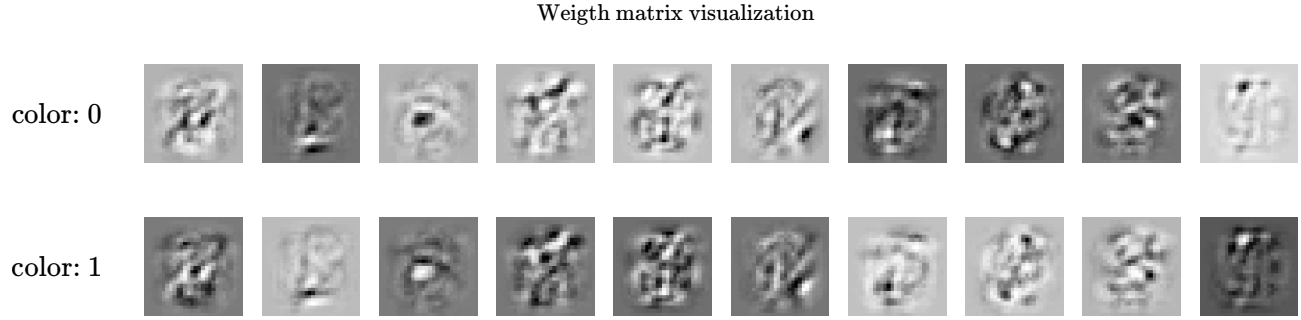

Figure 2.9: Representation of some of the feature maps learned from the RBM after 5000 epochs. Each feature map corresponds to the weights that connect a randomly selected hidden unit with the visible layer once we reshape the vector to form a 28×28 image. The first row shows the connection with the "0" Potts state of the visible layer, while the second row represents the connections with the state "1" for the same hidden unit.

as a function of the age for models trained with a different amount of labels. We can appreciate how

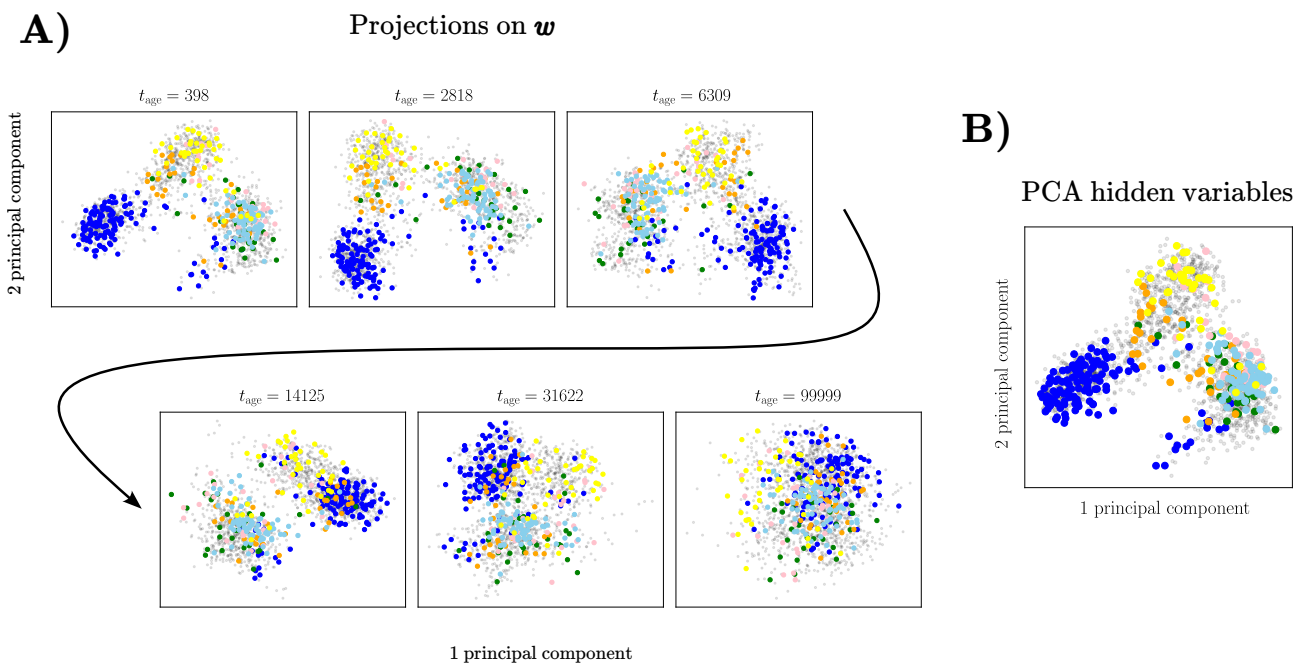

Figure 2.10: A) Projections of the hidden representation of the data along the first two principal components of the weight matrix for different ages of the RBM. The age of the machine is indicated in terms of epochs (1 epoch = 20 gradient updates in this case). The grey points represent the data, while the colored ones are the data classified by [31]. B) First two components of the PCA of the hidden variables. The RBM used here was trained in unsupervised mode for a total of 10^4 epochs.

slightly increasing the number of labels – from 1% to 5% – results in a sensitive improvement of the performances, approaching an impressive 80% of accuracy on the test set with just 5% of the labels. In the most favourable case, where we had at disposal 50% of the labels, the mostly trained machine reaches a remarkable 90% of accuracy.

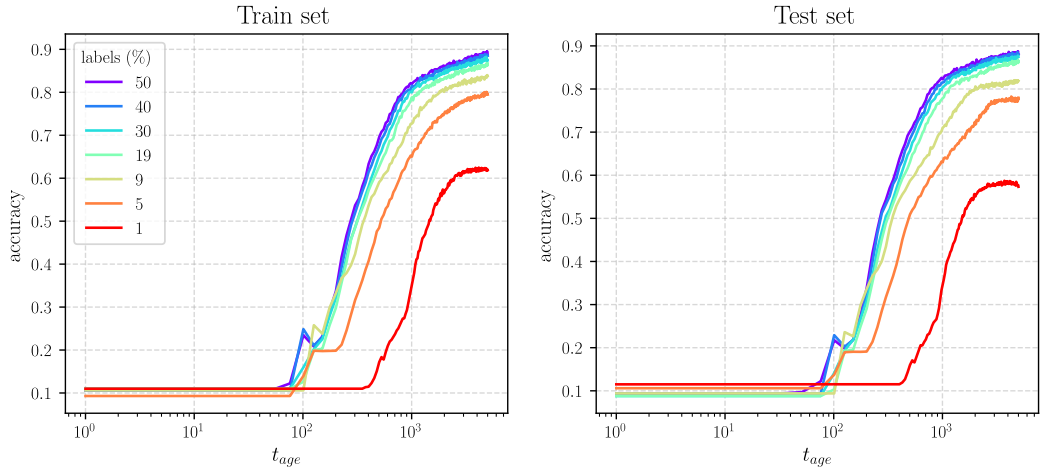

Figure 2.11: Accuracy reached on inferring the correct labels of the train set (left plot) and of the test set (right plot) for different RBMs as a function of the age. The RBMs differ for the percentage of labelled data that were used during the training, from 1% to 50%.

The other interesting task that this type of model can tackle is to generate samples *conditioned* on a given label specified by us. We described how to implement this feature into Algorithm 3. In Figure 2.12 we show the images generated by a model for different target labels. The machine is mostly able to produce coherent images, although in some cases the returned digit does not fulfill the delivery. This is very likely due to the fact that the Gibbs Markov chains, sometimes, might not be able to overcome the free energy barriers in the limited amount of updates that we set, hence remaining stuck in a representation that is not the desired one.

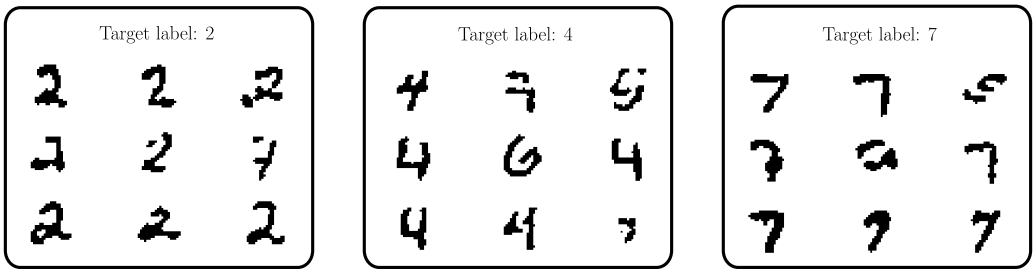

Figure 2.12: Some conditioned samples generated by an RBM trained with 100% of the labels after 10^4 Gibbs updates.

Chapter 3

Mean Field Approach

In the previous chapters, we already demonstrated how a physics-inspired computational model offers the possibility of applying methods developed in the Statistical Physics community in order to unveil many aspects of the functioning of the model, paving the way to a *explainable machine learning* approach.

In this chapter, we present yet another method borrowed from Statistical Physics that allows us to estimate the intractable free energy of the model through the so-called *mean field* (MF) approximation. This approach can be interpreted as an expansion of the system's free energy up to a certain order in the high temperature limit $\beta \to 0$. Since the temperature regulates the relative contribution of the entropy versus the energy in the free energy, $F = E - TS$, we can interpret the high-temperature regime as a stage in the training dynamics in which the energy, that mediates the interactions among variables, is negligible compared to the entropy of the system. This approximation is therefore valid in the early stages of the learning, where the couplings among the nodes of the RBM are small.

This derivation is not just instructive for a better understanding of the model at the beginning of the learning, but it is also at the core of the algorithm for constructing relational trees of data that we will present in the next chapter.

In the first section of this chapter, we apply one of the standard MF techniques in order to derive a first-order approximation of the RBM model that we refer to as *naive MF*. In the second section, instead, we present a generalization of this approach that allows – in principle – to compute the mean field approximation of the free energy up to any order, the so called *Plefka expansion* [24].

3.1 Naive Mean Field

In this section, for clarity and simplicity, we derive the naive mean field equations for the case of an RBM with Potts variables in the visible layer and binary variables $\{0, 1\}$ with Bernoulli potential in the hidden layer. We recall the Hamiltonian of the model:

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\sum_{iq} a_i^q \delta_{v_i,q} - \sum_{\mu} b_\mu h_\mu - \sum_{i\mu q} \delta_{v_i,q} w_{i\mu}^q h_\mu, \tag{3.1}$$

and we introduce the inverse temperature parameter $\beta = 1/T$, so that the probability distribution of the model is written as

$$p(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \frac{1}{Z} e^{-\beta E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})}. \tag{3.2}$$

Our aim is to make use of the variational method in order to approximate the probability distribution (3.2) with a simpler one – which we call *test distribution* – parametrized by some variational parameters $\boldsymbol{\pi}$. We arbitrary choose the test distribution to be in a Boltzmann-like form:

$$p_0(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\pi}) = \frac{1}{Z_0} e^{-\beta E_0(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\pi})}, \tag{3.3}$$

where E_0 is the test-Hamiltonian of a system put at the same temperature T of the original model. The Kullback-Leibler divergence between the probability distribution of the model and the test distribution is

$$D_{\mathrm{KL}}(p_0||p) = \sum_{\boldsymbol{v},\boldsymbol{h}} p_0(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\pi}) \left[\log p_0(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\pi}) - \log p(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})\right] = -S_0 + \beta \left\langle E \right\rangle_0 + \log Z, \tag{3.4}$$

where S_0 is the Shannon entropy of the test distribution and we denoted as $\langle\cdot\rangle_0$ the average over (3.3). Now we can use the fact that the KL divergence is a non-negative function, and write a variational equation for the free energy of the model:

$$F = -\frac{1}{\beta}\log Z \leq \langle E\rangle_0 - \frac{1}{\beta} S_0 \equiv F_{\mathrm{var}}. \tag{3.5}$$

If we rewrite $S_0 = \beta \left\langle E_0 \right\rangle_0 - \beta F_0$, with $F_0 = -1/\beta \log Z_0$ being the free energy of the test model, the inequality (3.5) can be written in the equivalent form

$$F \leq F_{\mathrm{var}} = \langle E - E_0\rangle_0 + F_0. \tag{3.6}$$

At this point we have to choose the test Hamiltonian. The simplest choice is to use a model in which all variables are independent, namely

$$E_0(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\pi}) = -\sum_{iq} \delta_{v_i,q}\phi_i^q - \sum_{\mu} h_\mu \psi_\mu, \tag{3.7}$$

where $\boldsymbol{\pi} = (\{\phi_i^q\}, \{\psi_\mu\})$ are the variational parameters of the test model. The goal is to find the optimal values of the variational parameters such that the variational free energy is as close as possible to the real one:

$$F \leq F^*_{\mathrm{var}} \equiv \min_{\boldsymbol{\pi}} \big\{ F_{\mathrm{var}}(\boldsymbol{\pi},\boldsymbol{\theta}) \big\}. \tag{3.8}$$

For simplicity, let us define the magnetizations $f_i^q = \langle \delta_{v_i,q} \rangle_0$ and $m_\mu = \langle h_\mu \rangle_0$. Then, the variational free energy (3.6) becomes

$$F_{\mathrm{var}} = -\sum_{iq} (a_i^q - \phi_i^q) f_i^q - \sum_{\mu} (b_\mu - \psi_\mu) m_\mu - \sum_{i\mu q} f_i^q w_{i\mu}^q m_\mu + F_0. \tag{3.9}$$

We can explicitly compute the magnetizations:

$$\begin{aligned} m_\mu = \langle h_\mu \rangle_0 &= -\frac{\partial F_0}{\partial \psi_\mu} = \frac{\sum_{h_\mu=0,1} h_\mu e^{\beta\psi_\mu h_\mu}}{\sum_{h_\mu=0,1} e^{\beta\psi_\mu h_\mu}} = \frac{1}{1+e^{-\beta\psi_\mu}} = \mathrm{sigmoid}\left(\beta\psi_\mu\right), \\ f_i^q = \langle \delta_{v_i,q} \rangle_0 &= -\frac{\partial F_0}{\partial \phi_i^q} = \frac{\sum_{v_i} \delta_{v_i,q}\, e^{\beta\phi_i^{v_i}}}{\sum_p e^{\beta\phi_i^p}} = \frac{e^{\beta\phi_i^q}}{\sum_p e^{\beta\phi_i^p}} = \mathrm{softmax}_q\left(\beta\phi_i^q\right). \end{aligned} \tag{3.10}$$

The variational parameters $\{\phi_i^q\}$ and $\{\psi_\mu\}$ must minimize the variational free energy, hence we have

$$\begin{aligned} 0 = \frac{\partial F_{var}}{\partial \phi_i^q} &= -\frac{\partial f_i^q}{\partial \phi_i^q}\left(a_i^q - \phi_i^q + \sum_\mu w_{i\mu}^q m_\mu\right), \\ 0 = \frac{\partial F_{var}}{\partial \psi_\mu} &= -\frac{\partial m_\mu}{\partial \psi_\mu}\left(b_\mu - \psi_\mu + \sum_{iq} w_{i\mu}^q f_i^q\right), \end{aligned} \tag{3.11}$$

which yield the optimal values:

$$\begin{aligned} \phi_i^q &= a_i^q + \sum_\mu w_{i\mu}^q m_\mu, \\ \psi_\mu &= b_\mu + \sum_{iq} w_{i\mu}^q f_i^q. \end{aligned} \tag{3.12}$$

If we substitute these expressions into Equations (3.10), we obtain the self-consistent equations

$$f_i^q = \text{softmax}_q \left(\beta a_i^q + \beta \sum_\mu w_{i\mu}^q m_\mu \right), \tag{3.13}$$

$$m_\mu = \text{sigmoid} \left(\beta b_\mu + \beta \sum_{iq} w_{i\mu}^q f_i^q \right). \tag{3.14}$$

At this point we can evaluate the variational free energy at its minimum. To do so, we first need to express the auxiliary fields in terms of the magnetizations by inverting equations (3.14) and (3.13). The first one gives straightforwardly

$$\beta\psi_\mu = \log \left(\frac{m_\mu}{1 - m_\mu} \right), \tag{3.15}$$

while the second case is less trivial. Indeed, the direct inversion of equation (3.13) is defined up to an arbitrary constant c

$$\beta\phi_i^q = \log f_i^q + c, \tag{3.16}$$

which is a consequence of the gauge freedom of the independent model (3.7). To uniquely define the constant, we need to specify a convenient gauge choice, which in this case turns out to be the lattice-gas gauge:

$$\phi_i^{N_q} = 0 \Rightarrow c = -\log f_i^{N_q}. \tag{3.17}$$

In this way we obtain the inverse relation of the softmax function (3.13) as

$$\beta\phi_i^q = \log \left(\frac{f_i^q}{f_i^{N_q}} \right). \tag{3.18}$$

Substituting these relations into (3.9) and using the normalization constraint $\sum_q f_i^q = 1$ we get

$$\begin{aligned} F_{\text{var}}^* = &- \sum_{iq} a_i^q f_i^q - \sum_\mu b_\mu m_\mu - \sum_{iq\mu} f_i^q w_{i\mu}^q m_\mu + \\ &- \frac{1}{\beta} \left[- \sum_{iq} f_i^q \log f_i^q - \sum_\mu \left[m_\mu \log m_\mu + (1 - m_\mu) \log(1 - m_\mu) \right] \right], \end{aligned} \tag{3.19}$$

where in the first line we recognize the average energy of the model over the non-interacting model and in the second line, in the brackets, there is the entropic contribution of categorical variables (first term) and of binary variables (second term). Let us point out that the variational free energy (3.19) is a valid approximation of the true free energy only when the magnetizations $\{m_\mu\}$ and $\{f_i^q\}$ satisfy the self-consistent conditions (3.13) and (3.14). To obtain those values, we can start with a generic initial condition $(\{m_\mu(0)\}, \{f_i^q(0)\})$ and iterate

$$\begin{aligned} f_i^q[t] &\leftarrow \text{softmax}_q \left(\beta a_i^q + \beta \sum_\mu w_{i\mu}^q m_\mu[t-1] \right), \\ m_\mu[t] &\leftarrow \text{sigmoid} \left(\beta b_\mu + \beta \sum_{iq} w_{i\mu}^q f_i^q[t] \right), \end{aligned} \tag{3.20}$$

for $t = 1, \ldots, N$. The recursion rule (3.20) is guaranteed to converge to one of the possible fixed points of F_{var}^* [24].

The derivation of the naive mean-field free energy does not make explicit that this approach is equivalent to approximating the real free energy at the first order in the high-temperature regime $\beta \to 0$; this will become clearer in the next section. Notice, however, that in this limit the entropic term of equation (3.19) becomes the dominant one, and the interactions among variables are negligible.

3.2 Plefka Expansion and TAP Equations

The so-called *Plefka expansion* was first introduced in [24] to show how the Thouless, Anderson and Palmer (TAP thereafter) equations for the Sherrington–Kirkpatrick model could be derived, and generalized, as a second-order expansion in the couplings of a proper Gibbs free energy. In the following section, we derive the TAP equations for the Potts RBM with generic hidden potential by generalizing the approach presented in [28] to the case of categorical variables.

At the core of the Plefka expansion there is the aforementioned observation that a series expansion of the free energy in the high-temperature regime is equivalent to an expansion in the weak-couplings limit, that is $\boldsymbol{w} \to \mathbf{0}$. Hence, instead of interpreting β as an inverse temperature, it is convenient to define an Hamiltonian in which β plays the role of an expansion parameter coupled with the weights tensor:

$$-\beta E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \sum_{iq} a_i^q \delta_{v_i,q} - \sum_{\mu} \mathcal{U}_\mu(h_\mu; \boldsymbol{\vartheta}_\mu) + \beta \sum_{i\mu q} \delta_{v_i,q} w_{i\mu}^q h_\mu. \tag{3.21}$$

The free energy of the model is then defined as

$$-\beta A_\beta = \log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left(-\beta E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})\right). \tag{3.22}$$

Since this is just a mathematical trick for expanding in the $\boldsymbol{w} \to \mathbf{0}$ limit, after the derivation of the mean-field free energy we will set $\beta = 1$.

The strategy of the derivation is the following. First, we define an extended system by introducing some auxiliary fields. We then derive a Gibbs free energy by taking the Legendre transform of the Helmholtz free energy of the extended system. This will allow us to introduce a new set of variational parameters, the magnetizations, that must match the first two moments of the probability distribution of the system. Since the obtained Gibbs free energy is as intractable as the true free energy, we rather consider a power series expansion in the $\beta \to 0$ limit. If we stop at the second-order and we impose the stationary conditions to the approximated Gibbs free energy, we obtain a set of recursive relations that are the TAP equations we are looking for.

First of all, let us introduce the temperature-dependent auxiliary fields $\boldsymbol{\Lambda} = \{\Lambda_i\} = (\{\phi_i^q\}, \{\psi_\mu\}, \{\xi_\mu\})$ and write the Helmholtz free energy of an extended system:

$$-\beta F_\beta(\boldsymbol{\Lambda}) = \log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left(-\beta E(\boldsymbol{v}, \boldsymbol{h}) + \sum_{iq} \phi_i^q(\beta)\delta_{v_i,q} + \sum_{\mu} \psi_\mu(\beta) h_\mu + \sum_{\mu} \xi_\mu(\beta) h_\mu^2\right). \tag{3.23}$$

Notice that when the auxiliary fields disappear we recover the true free energy of the system: $F_\beta(\mathbf{0}) = A_\beta$. It can be shown that (3.23) is a convex function of $\boldsymbol{\Lambda}$ [28]. We can therefore take its Legendre transform, which is again a convex function (and thus it has a unique minimum), obtaining a Gibbs free energy that depends on the conjugate variables $\boldsymbol{\Omega} = \{\Omega_i\} = (\{f_i^q\}, \{m_\mu\}, \{V_\mu\})$,

$$\begin{aligned}\Gamma(\beta) \equiv -\beta G_\beta(\boldsymbol{\Omega}) &= -\beta \sup_{\boldsymbol{\Lambda}} \left[F_\beta(\boldsymbol{\Lambda}) + \frac{1}{\beta}\sum_{iq} \phi_i^q(\beta) f_i^q + \frac{1}{\beta}\sum_{\mu} \psi_\mu(\beta) m_\mu + \frac{1}{\beta}\sum_{\mu} \xi_\mu(\beta)(V_\mu + m_\mu^2)\right] = \\ &= -\beta F_\beta(\boldsymbol{\Lambda}^*(\boldsymbol{\Omega})) - \sum_{iq} \phi_i^q(\boldsymbol{\Omega}, \beta) f_i^q - \sum_{\mu} \psi_\mu(\boldsymbol{\Omega}, \beta) m_\mu - \sum_{\mu} \xi_\mu(\boldsymbol{\Omega}, \beta)(V_\mu + m_\mu^2)\end{aligned} \tag{3.24}$$

where, by definition of Legendre transform,

$$\Omega_i = \frac{\partial(-\beta F_\beta)}{\partial \Lambda_i}(\boldsymbol{\Lambda}^*) \quad \to \quad \Lambda_i^*(\boldsymbol{\Omega}) = \left(\frac{\partial(-\beta F_\beta)}{\partial \Lambda_i}\right)^{-1}(\boldsymbol{\Omega}), \tag{3.25}$$

and we indicated as $\boldsymbol{\Lambda}^*$ the supremum values of the auxiliary fields.

Equation (3.25) implies that the following relations among conjugate variables must hold:

$$\begin{aligned} f_i^q &= \frac{\partial}{\partial \phi_i^q}\left(-\beta F_\beta(\mathbf{\Lambda}^*)\right) = \langle \delta_{v_i,q} \rangle\,, \\ m_\mu &= \frac{\partial}{\partial \psi_\mu}\left(-\beta F_\beta(\mathbf{\Lambda}^*)\right) = \langle h_\mu \rangle\,, \\ V_\mu &= \frac{\partial}{\partial \xi_\mu}\left(-\beta F_\beta(\mathbf{\Lambda}^*)\right) - m_\mu^2 = \left\langle h_\mu^2 \right\rangle - \langle h_\mu \rangle^2\,, \end{aligned} \tag{3.26}$$

Here, the average $\langle\cdot\rangle$ are taken at $\mathbf{\Lambda}^*$. From now on, for ease of notation, we will omit the superscript *. The relations (3.26) tell us that the Gibbs free energy (3.24) is valid only when the conjugate variables $\mathbf{\Omega}$ represent the firsts two moments of the marginal distribution of the extended system, and for that reason we will refer to them as *magnetizations*.

We are interested in the case in which the auxiliary fields vanish, since $F_\beta(\mathbf{\Lambda} = \mathbf{0})$ correspond to the true free energy of the model. It turns out that this corresponds to finding the stationary conditions of Γ with respect its arguments. In fact:

$$\begin{aligned} 0 = \frac{\partial \Gamma}{\partial \Omega_i} &= \frac{\partial}{\partial \Omega_i}\left[-\beta F_\beta(\mathbf{\Lambda}(\mathbf{\Omega})) - \sum_k \Lambda_k(\mathbf{\Omega})\Omega_k\right] = \\ &= \sum_k \underbrace{\frac{\partial(-\beta F_\beta)}{\partial \Lambda_k}}_{\Omega_k} \frac{\partial \Lambda_k}{\partial \Omega_i} - \sum_k \frac{\partial \Lambda_k}{\partial \Omega_i}\Omega_k - \Lambda_i(\mathbf{\Omega}) = \\ &= -\Lambda_i(\mathbf{\Omega}), \end{aligned} \tag{3.27}$$

which implies that, at the minimum of the Gibbs free energy, $\phi_i^q = \psi_\mu = \xi_\mu = 0\ \forall i, q, \mu$. But this means that, in that point, the *inverse* Legendre transform gives $F_\beta(\mathbf{0}) = A_\beta$, and we obtain the true free energy of the system. This suggests that, rather than try to directly evaluate the free energy of the model, we can work with the system $\Gamma(\beta, \mathbf{\Omega})$ instead. However, evaluating Γ is as difficult as evaluating A_β, so we may rather study a high-temperature expansion of the Gibbs free energy around $\beta = 0$. To do so, let us first rewrite (3.24) in the equivalent form

$$\begin{aligned} \Gamma(\beta) &= \log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-\beta E - \mathcal{K}(\beta)) = \\ &= \log \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\Bigg(\sum_{iq} a_i^q \delta_{v_i,q} - \sum_\mu \mathcal{U}_\mu(h_\mu; \boldsymbol{\vartheta}) + \beta \sum_{i\mu q} \delta_{v_i,q} w_{i\mu}^q h_\mu + \\ &\qquad + \sum_{iq} \phi_i^q(\beta)\left(\delta_{v_i,q} - f_i^q\right) + \sum_\mu \psi_\mu(\beta)\left(h_\mu - m_\mu\right) + \sum_\mu \xi_\mu(\beta)\left(h_\mu^2 - V_\mu - m_\mu^2\right)\Bigg), \end{aligned} \tag{3.28}$$

where we distinguished between the original Hamiltonian and the part coming from having extended the system:

$$\mathcal{K}(\beta) \equiv -\sum_{iq} \phi_i^q(\beta)\left(\delta_{v_i,q} - f_i^q\right) - \sum_\mu \psi_\mu(\beta)\left(h_\mu - m_\mu\right) - \sum_\mu \xi_\mu(\beta)\left(h_\mu^2 - V_\mu - m_\mu^2\right). \tag{3.29}$$

Our aim is to find a second-order expansion in β of the Gibbs free energy around its minimum:

$$\Gamma(\beta) = \Gamma(0) + \beta \left.\frac{\partial \Gamma}{\partial \beta}\right|_{\beta=0} + \frac{\beta^2}{2} \left.\frac{\partial^2 \Gamma}{\partial \beta^2}\right|_{\beta=0} + \mathcal{O}(\beta^3). \tag{3.30}$$

Zeroth Order The zeroth order is readily evaluated as

$$\Gamma(0) = -\sum_{iq}\phi_i^q(0)f_i^q - \sum_{\mu}\psi_\mu(0)m_\mu - \sum_{\mu}\xi_\mu(0)(V_\mu + m_\mu^2) + \\ + \sum_i \log\left[\sum_v \exp\left(\sum_q \delta_{v,q}(a_i^q + \phi_i^q(0))\right)\right] + \sum_\mu \log\left[\sum_h P_\mu(h,\boldsymbol{\vartheta}_\mu)e^{\psi_\mu(0)h+\xi_\mu(0)h^2}\right] \tag{3.31}$$

where we expressed the hidden potential in terms of the probability distribution $P = e^{-\mathcal{U}}$.

First Order Let us denote the interaction term as

$$\mathcal{I} \equiv -\sum_{i\mu q}\delta_{v_i,q}w_{i\mu}^q h_\mu. \tag{3.32}$$

Then, the first-order term can be computed as

$$\left.\frac{\partial\Gamma}{\partial\beta}\right|_{\beta=0} = \left\langle \sum_{iq\mu} w_{i\mu}^q\delta_{v_i,q}h_\mu + \sum_{iq}\partial_\beta\phi_i^q(\beta)(\delta_{v_i,q} - f_i^q) + \sum_\mu \partial_\beta\psi_\mu(\beta)(h_\mu - m_\mu) + \right. \\ \left. + \sum_\mu \partial_\beta\xi_\mu(\beta)(h_\mu^2 - V_\mu - m_\mu^2)\right\rangle\Big|_{\beta=0} = \\ = \langle -\mathcal{I} - \partial_\beta\mathcal{K}(\beta)\rangle|_{\beta=0} = -\langle\mathcal{I}\rangle = \sum_{i\mu q} f_i^q w_{i\mu}^q m_\mu, \tag{3.33}$$

where we have used $\langle\partial_\beta\mathcal{K}\rangle = 0$, which comes directly from (3.26).

Second Order To compute the second derivative we have first to evaluate the derivative of the probability distribution with respect to β. We have

$$\frac{\partial}{\partial\beta}p(\boldsymbol{v},\boldsymbol{h}) = \frac{\partial}{\partial\beta}\left(\frac{e^{-\beta E-\mathcal{K}}}{Z(\beta)}\right) = (-\mathcal{I} - \partial_\beta\mathcal{K})\frac{e^{-\beta E-\mathcal{K}}}{Z(\beta)} - \frac{1}{Z^2(\beta)}\frac{\partial Z(\beta)}{\partial\beta}e^{-\beta E-\mathcal{K}}, \tag{3.34}$$

where

$$\frac{\partial Z(\beta)}{\partial\beta} = \frac{\partial}{\partial\beta}\sum_{\boldsymbol{v},\boldsymbol{h}} e^{-\beta E-\mathcal{K}} = \sum_{\boldsymbol{v},\boldsymbol{h}}(-\mathcal{I} - \partial_\beta\mathcal{K})e^{-\beta E-\mathcal{K}} = Z(\beta)(-\langle\mathcal{I}\rangle - \underbrace{\langle\partial_\beta\mathcal{K}\rangle}_{=0}). \tag{3.35}$$

Thus

$$\left.\frac{\partial}{\partial\beta}p(\boldsymbol{v},\boldsymbol{h})\right|_{\beta=0} = (\langle\mathcal{I}\rangle - \mathcal{I} - \partial_\beta\mathcal{K})\; p(\boldsymbol{v},\boldsymbol{h})|_{\beta=0}\,. \tag{3.36}$$

In general, if we have a function $f(\beta,\boldsymbol{v},\boldsymbol{h})$, then

$$\frac{\partial}{\partial\beta}\langle f\rangle = \left\langle\frac{\partial f}{\partial\beta}\right\rangle + \langle f\rangle\langle\mathcal{I}\rangle - \langle f\mathcal{I}\rangle - \langle f\partial_\beta\mathcal{K}\rangle\,. \tag{3.37}$$

Now we can proceed by computing the second derivative of the free energy:

$$\frac{\partial^2\Gamma}{\partial\beta^2} = \frac{\partial}{\partial\beta}\sum_{\boldsymbol{v},\boldsymbol{h}}(-\mathcal{I} - \partial_\beta\mathcal{K}(\beta))p(\boldsymbol{v},\boldsymbol{h}) = \\ = \underbrace{\langle\partial_\beta^2\mathcal{K}\rangle}_{=0} + \sum_{\boldsymbol{v},\boldsymbol{h}}(-\mathcal{I} - \partial_\beta\mathcal{K}(\beta))\frac{\partial}{\partial\beta}p(\boldsymbol{v},\boldsymbol{h}) = \\ = \left\langle(\mathcal{I} + \partial_\beta\mathcal{K})^2\right\rangle - \langle\mathcal{I}\rangle^2 = \left\langle\mathcal{I}^2\right\rangle - \langle\mathcal{I}\rangle^2 + \left\langle(\partial_\beta\mathcal{K})^2\right\rangle + 2\langle\mathcal{I}\;\partial_\beta\mathcal{K}\rangle\,. \tag{3.38}$$

When expanding the terms in (3.38), we need to compute the pair correlation functions with respect to the extended probability function at $\beta = 0$. In the infinite-temperature limit the interaction term disappears, so the variables of the system becomes independent. For $\{0, 1\}$ variables, this yields:

$$\begin{aligned} \langle h_\mu h_\nu \rangle &= m_\mu \delta_{\mu\nu} + (1 - \delta_{\mu\nu}) m_\mu m_\nu, \\ \left\langle \delta_{v_i,q} \delta_{v_j,p} \right\rangle &= f_i^q \delta_{ij} \delta_{qp} + (1 - \delta_{ij}) f_i^q f_j^p. \end{aligned} \tag{3.39}$$

After some computation, we arrive at the expression

$$\begin{aligned} \left.\frac{\partial^2 \Gamma}{\partial \beta^2}\right|_{\beta=0} = &\sum_\mu \left(\langle h_\mu^2 \rangle - m_\mu \right) \left[\sum_{iq} (w_{i\mu}^q)^2 f_i^q + \sum_{iqp} w_{i\mu}^q w_{i\mu}^p f_i^q f_i^p + (\partial_\beta \psi_\mu(0))^2 + 2 \sum_{iq} w_{i\mu}^q (\partial_\beta \psi_\mu(0)) f_i^q \right] + \\ &+ \sum_{iqp} (f_i^q \delta_{pq} - f_i^q f_i^p) \left[\sum_{\mu\nu} w_{i\mu}^q w_{i\nu}^p m_\nu m_\mu + (\partial_\beta \phi_i^q(0))(\partial_\beta \phi_i^p(0)) + 2 \sum_\mu w_{i\mu}^q m_\mu (\partial_\beta \phi_i^q(0)) \right]. \end{aligned} \tag{3.40}$$

To compute the derivative of the auxiliary fields at $\beta = 0$ we can exploit the Schwarz's theorem for the Gibbs free energy:

$$\begin{aligned} \partial_\beta \psi_\mu(0) &= -\frac{\partial}{\partial \beta} \left.\frac{\partial \Gamma}{\partial m_\mu}\right|_{\beta=0} = -\frac{\partial}{\partial m_\mu} \left.\frac{\partial \Gamma}{\partial \beta}\right|_{\beta=0} = -\frac{\partial}{\partial m_\mu} \left[\sum_{i\mu q} f_i^q w_{i\mu}^q m_\mu \right] = -\sum_{iq} f_i^q w_{i\mu}^q, \\ \partial_\beta \phi_i^q(0) &= -\frac{\partial}{\partial \beta} \left.\frac{\partial \Gamma}{\partial f_i^q}\right|_{\beta=0} = -\frac{\partial}{\partial f_i^q} \left.\frac{\partial \Gamma}{\partial \beta}\right|_{\beta=0} = -\frac{\partial}{\partial f_i^q} \left[\sum_{i\mu q} f_i^q w_{i\mu}^q m_\mu \right] = -\sum_\mu m_\mu w_{i\mu}^q, \\ \partial_\beta \xi_\mu(0) &= -\frac{\partial}{\partial \beta} \left.\frac{\partial \Gamma}{\partial (V_\mu + m_\mu^2)}\right|_{\beta=0} = -\frac{\partial}{\partial (V_\mu + m_\mu^2)} \left.\frac{\partial \Gamma}{\partial \beta}\right|_{\beta=0} = -\frac{\partial}{\partial (V_\mu + m_\mu^2)} \left[\sum_{i\mu q} f_i^q w_{i\mu}^q m_\mu \right] = 0. \end{aligned} \tag{3.41}$$

After inserting these expressions into (3.40), we obtain

$$\left.\frac{\partial^2 \Gamma}{\partial \beta^2}\right|_{\beta=0} = \sum_\mu V_\mu \left[\sum_{iq} (w_{i\mu}^q)^2 f_i^q - \sum_i \left(\sum_q f_i^q w_{i\mu}^q \right)^2 \right]. \tag{3.42}$$

From now on, to lighten the notation, we drop the $\beta = 0$ specification in the auxiliary fields. Putting all pieces together, the Gibbs free energy of the model can be expanded at the second order in β as

$$\begin{aligned} \Gamma^{(2)}(\beta) = &- \sum_{iq} \phi_i^q f_i^q - \sum_\mu \psi_\mu m_\mu - \sum_\mu \xi_\mu (V_\mu + m_\mu^2) + \\ &+ \sum_i \log \left[\sum_v \exp \left(\sum_q \delta_{v,q} (a_i^q + \phi_i^q) \right) \right] + \sum_\mu \log \left[\sum_h P_\mu(h; \boldsymbol{\vartheta}_\mu) e^{\psi_\mu h + \xi_\mu h^2} \right] + \\ &+ \beta \sum_{i\mu q} f_i^q w_{i\mu}^q m_\mu + \\ &+ \frac{\beta^2}{2} \sum_\mu V_\mu \left[\sum_{iq} (w_{i\mu}^q)^2 f_i^q - \sum_i \left(\sum_q f_i^q w_{i\mu}^q \right)^2 \right]. \end{aligned} \tag{3.43}$$

In order to obtain the auxiliary fields, we can impose the stationary conditions to equation (3.43) with

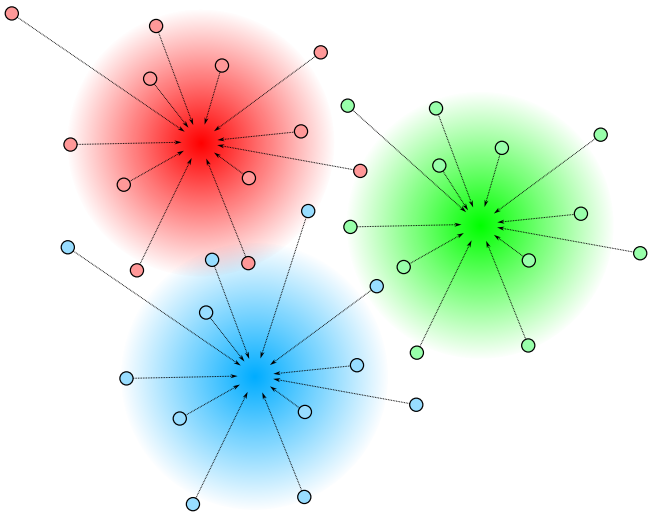

Figure 3.1: Sketch of the effect of propagating the TAP equations by starting from different initial conditions. In the drawing we assume the free energy to have 3 local minima, whose basins of attraction are represented by the colored shaded areas. All the points that belong to a given basin of attractions are dragged toward the local minima by the effect of the TAP dynamics.

respect to f_i^q, V_μ and m_μ. At $\beta = 1$ this yields, respectively,

$$\begin{aligned}
\phi_i^q &= \sum_\mu w_{i\mu}^q m_\mu + \frac{1}{2}\sum_\mu (w_{i\mu}^q)^2 V_\mu - \sum_{\mu p} f_i^p w_{i\mu}^p w_{i\mu}^q V_\mu, \\
\xi_\mu &= \frac{1}{2}\sum_{iq} (w_{i\mu}^q)^2 f_i^q - \frac{1}{2}\sum_i \left(\sum_q f_i^q w_{i\mu}^q\right)^2, \\
\psi_\mu &= \sum_{iq} f_i^q w_{i\mu}^q - 2 m_\mu \xi_\mu.
\end{aligned} \tag{3.44}$$

Similarly, since the Gibbs free energy does not depend on the auxiliary fields, we can obtain the magnetizations by differentiating equation (3.43) with respect to respectively ϕ_i^q, ψ_μ and ξ_μ. We obtain:

$$\begin{aligned}
f_i^q &= \frac{\partial}{\partial \phi_i^q}\sum_i \log\left[\sum_v \exp\left(\sum_q \delta_{v,q}(a_i^q + \phi_i^q)\right)\right] = \mathrm{softmax}_q(a_i^q + \phi_i^q), \\
m_\mu &= \frac{\partial}{\partial \psi_\mu}\sum_\mu \log\left[\sum_h P_\mu(h, \boldsymbol{\vartheta}_\mu) e^{\psi_\mu h + \xi_\mu h^2}\right] = \langle h_\mu \rangle_{\tilde{P}_\mu}, \\
V_\mu &= \frac{\partial}{\partial \xi_\mu}\sum_\mu \log\left[\sum_h P_\mu(h, \boldsymbol{\vartheta}_\mu) e^{\psi_\mu h + \xi_\mu h^2}\right] - m_\mu^2 = \mathrm{Var}_{\tilde{P}_\mu}(h_\mu),
\end{aligned} \tag{3.45}$$

where we defined the modified probability distribution

$$\tilde{P}_\mu(h; \boldsymbol{\vartheta}_\mu, \psi_\mu, \xi_\mu) \equiv \frac{P_\mu(h, \boldsymbol{\vartheta}_\mu) e^{\psi_\mu h + \xi_\mu h^2}}{\sum_h P_\mu(h, \boldsymbol{\vartheta}_\mu) e^{\psi_\mu h + \xi_\mu h^2}}. \tag{3.46}$$

By inserting Equations (3.44) into (3.45) we obtain the self-consistency equations that allow us to find the fixed points of the approximated free energy of the RBM. At the second order of the expansion, those are called *TAP equations*.

Now, while the exact Gibbs free energy is a convex function of the magnetizations $\boldsymbol{\Omega}$, the TAP free energy may present several stationary points that increase with β. For fixed $\beta = 1$ this still holds true as the training advances, because the growing variance of the weights acts as an inverse temperature.

This means that the learning produces the same effect as if we were cooling the system. The interesting observation is that, if we start from a given initial condition $(f_i^q[0], m_\mu[0], V_\mu[0])$ and we run the TAP equations, the magnetizations will converge toward the nearest solution of the TAP free energy. In other words, each solution of the TAP free energy corresponds to a mode of the associated distribution, and all initial conditions that lay in the basin of attraction of this solution will be dragged by the TAP dynamics toward it (Figure 3.1). At the very beginning of the training, the RBM is in a paramagnetic phase in which all initial conditions converge toward the unique mode corresponding to the unique minimum of the TAP free energy. As the learning proceeds, several other modes are learnt and the number of TAP solutions increases, until the point in which the number of minima is comparable with the number of data samples. This last regime might be interpreted as overfitting the data, but we have also to consider that, at this stage, the TAP approximation starts to fail in reproducing the real free energy of the system, and some fixed points might be *spurious solutions*. Figure 3.2 shows the fixed points of the TAP free energy at different stages of the learning.

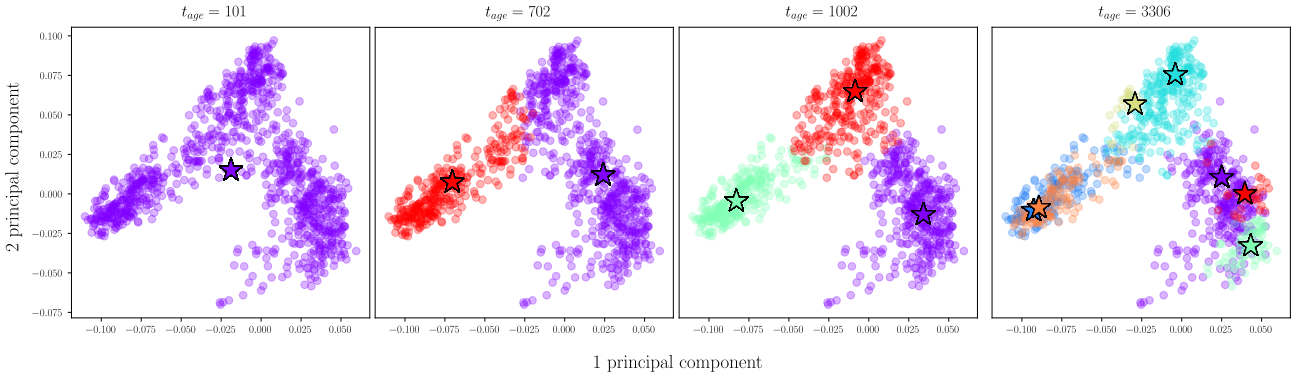

Figure 3.2: Projection of the WW dataset along the first two principal components of the PCA (colored points) for different ages (expressed in terms of epochs) of the training. Each epoch correspond to 28 gradient updates. The fixed points (indicated with the colored stars) are obtained by using as initial conditions the data (colored dots) and propagating the TAP equations until convergence. All points with the same color end up in the same fixed point. Under the effect of the TAP equations, the points collapse toward the "closest" solution of the TAP equations, which can be interpreted as the must resembling internal representation of the machine for that data. We can notice how the number of modes increases with the age of the RBM. This will be used later first to identify all the main clusters visible in the PCA and then to find finest sub-structures of the data.

In the following part, we derive the TAP equations for the specific cases of Bernoulli and Truncated Gaussian hidden potentials and we generalize the approximated Gibbs free energy to the case of semi-supervised learning.

3.2.1 Bernoulli Hidden Potential

Let's start by considering binary hidden variables, $h \in \{0, 1\}$, distributed according to the Bernoulli distribution

$$P_\mu^{\mathrm{B}}(h; \rho_\mu) = \rho_\mu^h \, (1 - \rho_\mu)^{1-h} \quad \text{with} \quad \rho_\mu \in [0, 1]. \tag{3.47}$$

By reparametrizing $\rho_\mu = \frac{1}{1+e^{-b_\mu}}$, we obtain the equivalent form

$$P_\mu^{\mathrm{B}}(h; b_\mu) = \frac{e^{h b_\mu}}{1 + e^{b_\mu}} \quad \text{where} \quad b_\mu \in \mathbb{R}. \tag{3.48}$$

Using $h^2 = h$ for binary variables, the modified probability distribution (3.46) remains a Bernoulli distribution with a shifted parameter, $\tilde{P}_\mu^{\mathrm{B}}(h; \tilde{b}_\mu)$, where

$$\tilde{b}_\mu = b_\mu + \psi_\mu + \xi_\mu = b_\mu + \sum_{iq} f_i^q w_{i\mu}^q - \left(m_\mu - \frac{1}{2}\right) \left[\sum_i \left(\sum_q f_i^q w_{i\mu}^q \right)^2 - \sum_{iq} (w_{i\mu}^q)^2 f_i^q \right]. \tag{3.49}$$

Then, using the expressions for the moments of a Bernoulli distribution, we get

$$\begin{aligned} m_\mu &= \langle h_\mu \rangle_{\tilde{P}_\mu^{\mathrm{B}}} = \tilde{\rho}_\mu = \mathrm{sigmoid}(\tilde{b}_\mu), \\ V_\mu &= \mathrm{Var}_{\tilde{P}_\mu^{\mathrm{B}}}(h_\mu) = \tilde{\rho}_\mu \, (1 - \tilde{\rho}_\mu) = m_\mu - m_\mu^2. \end{aligned} \tag{3.50}$$

Inserting these expressions into Equations (3.45), we get the TAP equations for the Potts RBM with binary hidden variables:

$$\begin{aligned} m_\mu[t] \leftarrow \mathrm{sigmoid}\Bigg(& b_\mu + \sum_{iq} f_i^q[t-1] w_{i\mu}^q - \left(m_\mu[t-1] - \frac{1}{2} \right) \left[\sum_i \left(\sum_q f_i^q[t-1] w_{i\mu}^q \right)^2 + \right. \\ & \left. - \sum_{iq} (w_{i\mu}^q)^2 f_i^q[t-1] \right] \Bigg), \\ f_i^q[t] \leftarrow \mathrm{softmax}_q & \left(a_i^q + \sum_\mu w_{i\mu}^q m_\mu[t] + \sum_\mu (m_\mu[t] - m_\mu^2[t]) \left[\frac{1}{2} (w_{i\mu}^q)^2 + w_{i\mu}^q \sum_p f_i^p[t-1] w_{i\mu}^p \right] \right), \end{aligned} \tag{3.51}$$

for $t = 1, \dots, N$. Notice that, in the case of a Bernoulli distribution, we do not need to compute also the magnetization V_μ, because the second moment of the distribution is fully determinated by the first moment.

3.2.2 Truncated Gaussian Hidden Potential

Let's now consider a Truncated Gaussian distribution in the form (1.46):

$$P_\mu^{\mathrm{TG}}(h; b_\mu, \gamma_\mu) = \sqrt{\frac{2\gamma_\mu}{\pi}} \frac{e^{-\frac{1}{2}\gamma_\mu h^2 - b_\mu h}}{1 - \mathrm{erf}\left(\frac{b_\mu}{\sqrt{2\gamma_\mu}} \right)} \mathbb{1}_{[0, +\infty]}. \tag{3.52}$$

We immediately see that the modified distribution (3.46) is again a Truncated Gaussian with parameters:

$$\tilde{\gamma}_\mu = \gamma_\mu - 2\xi_\mu \quad \text{and} \quad \tilde{b}_\mu = b_\mu - \psi_\mu. \tag{3.53}$$

The expressions for the firsts two moments of the Truncated Gaussian yield:

$$\begin{aligned} m_\mu = \langle h \rangle_{\tilde{P}_\mu^{\mathrm{TG}}} &= -\frac{\tilde{b}_\mu}{\tilde{\gamma}_\mu} + \frac{1}{\sqrt{\tilde{\gamma}_\mu} \Phi\left(\frac{\tilde{b}_\mu}{\sqrt{\tilde{\gamma}_\mu}} \right)}, \\ V_\mu = \mathrm{Var}_{\tilde{P}_\mu^{\mathrm{TG}}}(h) &= \frac{1}{\tilde{\gamma}_\mu} \left[1 + \frac{\tilde{b}_\mu}{\sqrt{\tilde{\gamma}_\mu} \Phi\left(\frac{\tilde{b}_\mu}{\sqrt{\tilde{\gamma}_\mu}} \right)} - \left(\frac{1}{\Phi\left(\frac{\tilde{b}_\mu}{\sqrt{\tilde{\gamma}_\mu}} \right)} \right)^2 \right], \end{aligned} \tag{3.54}$$

where Φ is the auxiliary function (1.48). The iteration of the TAP equations for the Potts RBM with TG hidden potential can be broke down as:

$$\begin{aligned}
2\xi_\mu[t] &\leftarrow \sum_{iq}(w^q_{i\mu})^2 f^q_i[t-1] - \sum_i \left(\sum_q f^q_i[t-1] w^q_{i\mu}\right)^2, \\
\psi_\mu[t] &\leftarrow \sum_{iq} f^q_i[t-1] w^q_{i\mu} - m_\mu[t-1] 2\xi_\mu[t], \\
\tilde{\gamma}_\mu[t] &\leftarrow \gamma_\mu - 2\xi_\mu[t], \\
\tilde{b}_\mu[t] &\leftarrow b_\mu - \psi_\mu[t], \\
m_\mu[t] &\leftarrow -\frac{\tilde{b}_\mu[t]}{\tilde{\gamma}_\mu[t]} + \frac{1}{\sqrt{\tilde{\gamma}_\mu[t]}\Phi\left(\frac{\tilde{b}_\mu[t]}{\sqrt{\tilde{\gamma}_\mu[t]}}\right)}, \\
V_\mu[t] &\leftarrow \frac{1}{\tilde{\gamma}_\mu[t]}\left[1 + \frac{\tilde{b}_\mu[t]}{\sqrt{\tilde{\gamma}_\mu[t]}\Phi\left(\frac{\tilde{b}_\mu[t]}{\sqrt{\tilde{\gamma}_\mu[t]}}\right)} - \left(\frac{1}{\Phi\left(\frac{\tilde{b}_\mu[t]}{\sqrt{\tilde{\gamma}_\mu[t]}}\right)}\right)^2\right], \\
\phi^q_i[t] &\leftarrow \sum_\mu w^q_{i\mu} m_\mu[t] + \sum_\mu V_\mu[t]\left[\frac{1}{2}(w^q_{i\mu})^2 - w^q_{i\mu}\sum_p f^p_i[t-1] w^p_{i\mu}\right], \\
f^q_i[t] &\leftarrow \mathrm{softmax}_q(a^q_i + \phi^q_i[t]),
\end{aligned} \tag{3.55}$$

for $t = 1, \dots, N$. In practice, though, dealing with the Truncated Gaussian case is quite tricky. Indeed, when iterating the TAP equations it can happen that $\tilde{\gamma}_\mu = \gamma_\mu - 2\xi_\mu$ becomes negative. To handle this case, one has to write the TG distribution in terms of the *imaginary error function*

$$\mathrm{erfi}(x) = -i\mathrm{erf}(ix) = \frac{2}{\sqrt{\pi}}\int_0^x \mathrm{d}t\ t^2. \tag{3.56}$$

We remand the details of the application to the Appendix of [28]. In the present work, we have not implemented yet the TAP equations for the ReLU RBM.

3.2.3 TAP Equations in Semi-Supervised Mode

Similarly to what we have done so far, we can derive a TAP Gibbs free energy also in the case of the Potts RBM trained in semi-supervised mode. To do this, we simply have to assume a high temperature / small weights expansion also for the label matrix $\boldsymbol{d}$. We therefore consider the Hamiltonian

$$-\beta E(\boldsymbol{v}, \boldsymbol{h}, \ell; \boldsymbol{\theta}) = \sum_{iq}\delta_{v_i,q} a^q_i - \sum_\mu \mathcal{U}_\mu(h_\mu; \boldsymbol{\vartheta}_\mu) + \sum_n \delta_{\ell,n} c_n + \beta\sum_{i\mu q}\delta_{v_i,q} w^q_{i\mu} h_\mu + \beta\sum_{n\mu}\delta_{\ell,n} d_{n\mu} h_\mu \tag{3.57}$$

and we introduce an auxiliary field $\{\lambda_\ell\}$ and a magnetization $\{g_\ell\}$ representing the average frequency of the label ℓ. Eventually, we end up with the following TAP Gibbs free energy:

$$\begin{aligned}\Gamma^{(2)}_{\rm ssl}(\beta) = &-\sum_{iq}\phi_i^q f_i^q - \sum_{\mu}\psi_\mu m_\mu - \sum_{\ell}\lambda_\ell\ g_\ell - \sum_{\mu} V_\mu(\xi_\mu + m_\mu^2) + \\ &+ \sum_i \log\left[\sum_v \exp\left(\sum_q \delta_{v,q}(a_i^q + \phi_i^q)\right)\right] + \sum_\mu \log\left[\sum_h P_\mu(h;\boldsymbol{\vartheta}_\mu) e^{\psi_\mu h + \xi_\mu h^2}\right] + \\ &+ \log\left[\sum_\ell \exp\left(\lambda_\ell + c_\ell\right)\right] + \\ &+ \beta\left[\sum_{i\mu q} f_i^q w_{i\mu}^q m_\mu + \sum_{\ell\mu} d_{\ell\mu}\ g_\ell\ m_\mu\right] + \\ &+ \frac{\beta^2}{2}\sum_\mu V_\mu\left[\sum_{iq}(w_{i\mu}^q)^2 f_i^q - \sum_i\left(\sum_q f_i^q w_{i\mu}^q\right)^2 + \sum_\ell (d_{\ell\mu})^2 g_\ell - \left(\sum_\ell d_{\ell\mu} g_\ell\right)^2\right]. \end{aligned} \tag{3.58}$$

By imposing the stationary conditions with respect to the fields and the magnetizations we obtain the corresponding TAP equations. We do not report them here because the results are analogous o those of section 3.2.

Chapter 4

Generating Relational Trees by Exploiting the Dynamics of the Learning

In chapter 2 we discussed the use of RBM models for data classification. The central idea is that projecting the data (or their hidden representation) along the principal components of the weight matrix might provide us with a separation of the data into feature-related clusters that may vary along the learning. Starting from this representation, we can then exploit one of the main clustering algorithms (e.g. DBSCAN) in order to put the labels on the data without specifying the number of clusters. Such an approach was exploited by Tubiana et al in [30] in the context of functional characterization of protein sequences. However, this method presents a number of shortcomings, among which the principal are:

1. It is not clear how much trained the RBM should be in order to maximize the accuracy of the classification. As we justified theoretically in subsection 1.5.4, in the first part of the training the RBM learns how to perform the PCA of the data, and to classify the data at this stage it makes no point using an RBM. On the other hand, a very old machine has learnt a highly non-linear transformation of the data that makes all the clusters overlap when we visualize the data projections using the PCA of the weight matrix. Experimentally, one finds that the best results in the classification task are obtained for intermediate ages of the RBM, which by the way have to be determined by later inspection. Still, the use of intermediate ages makes sense because we are not seeking a very fine classification, but a separation over broad and general classes;
2. Given that one has found a nearly optimal age of the model for performing the classification, a new arbitrariness comes into play when choosing the parameters of the clustering algorithm. For instance, if using the K-means algorithm, one has to decide a priori how many clusters are present. This choice is instead automatized in density-based clustering algorithms such as DBSCAN, where one has nevertheless to tell the algorithm when to consider two points to be "close together".

The main contribution of this thesis work goes in the direction of overcoming those issues and pushing even further the goal of data characterization. In particular, we want to address the following question: can we gather information by looking at the whole learning history in order to construct a relational tree that could tell us which are the relations among data in a hierarchical way?

4.1 General Idea of the Algorithm

The algorithm we propose here is based on the idea that, when learning from data, the RBM first learns how to recognize the most relevant, high-level features and then, as the training goes on, it

progressively starts recognizing and encoding finer and finer differences between the samples. Then, we can pictorially imagine a sort of "relay race" in which different versions of the same RBM with different ages run for the same team. The youngest RBM starts the race, splits the data doing its best, and then passes the baton to the next RBM, which is older than the previous one. The baton passing can be interpreted as the following question: "Given that I managed to perform this splitting of the data, what more can you tell me about the samples inside each category I made?" If we do this with several RBMs and we keep track of the splittings that each model made, what we obtain at the end is a *relational tree* of the dataset. The first branches of the tree correspond to the most "evident" differences between the data that even the most "immature" RBM could recognize, while the deeper we go into the tree the more the detected features are subtle. In this way, we leverage the information brought by different RBMs at different stages of the learning, which is a more sophisticated approach compared to a naive annealing in β on an old RBM.

To put into practice this idea, we have to address the following problems:

1. Specify a procedure for classifying the data at each stage of the learning;
2. Define how to perform the "baton passing" between two RBMs;
3. Find some criteria to assess the quality of the obtained relational tree;
4. Identify a method for choosing what ages of the training to use for the tree construction.

In the following sections, we will tackle those technical points one by one.

4.1.1 Classification Procedure

The method we used for classifying the data totally relies on the mean-field theory we developed in chapter 3. As we already pointed out in that chapter, the interesting effect of iterating the TAP equations when starting from the data is that the trajectories will converge toward the various local minima of the TAP free energy, which approximate the internal representations of the model for those data. Stated differently, all points that belong to the same basin of attraction of a free energy's local minimum are represented in the same way by the RBM, and under the effect of iterating the TAP equations they collapse in the same point of the magnetizations' space.
Therefore, one can think of first iterating the TAP equations until convergence, and then applying the DBSCAN algorithm to the endpoints of the trajectories in order to classify the data into categories. The main advantage of this procedure is that, in practice, it is parameter-free. In fact, since the data collapse on very restricted regions of the magnetizations' space, they constitute several isolated and high-density clusters. In that situation, the DBSCAN can classify the data without any ambiguity, for we empirically verified that the result of the classification is independent of the algorithm's hyper-parameters under reasonable choices. Finally, from a conceptual point of view, this procedure allows us to know exactly what the RBM "thinks" about each data point at its level of the training and also to visualize the basins of attractions of the free energy minima.

This idea is not a new one, and it can be found in several works, such as [28, 7]. The reasons that might prevent someone from using this approach to classify data are easy to guess. The point is that the Plefka expansion – at any order – is just an approximation of the actual model, valid only in the first stages of the learning when the interactions among variables are weak. Also, to apply this method one has to derive the corresponding self-consistent equations in the first place, which could prove to be a hard task for more complex models than the simple Bernoulli version.

That being said, we argue that often many features of interest in real-world datasets are mostly detectable already by using quite young models. Traditional clustering methods are typically used when the projection of the data (or the hidden representation) along the principal directions of the weight matrix is sufficiently clustered to allow for a reliable classification, but this problem does not show up at all in the mean-field method we discussed. On top of this, we have empirical reasons for arguing that already the second order (TAP) approximation of the free energy does a really good job even at moderate stages of the learning.

This argument justifies our adoption of this clustering method as part of the tree construction algorithm. In addition, we believe that the way we implement this procedure enhances the accuracy of the resulting classification, because (as will be discussed in the next subsection) we do not just rely on a single snapshot of the system, but we integrate also the information coming by other stages of the learning.

4.1.2 The Baton Passing

The relay race analogy we outlined in the introduction of this section is useful for describing the spirit of the tree algorithm, but it is not actually the best strategy to use. Indeed, by strictly interpreting this analogy, we would proceed by categorizing the data with the youngest RBM and then, at the following step, we would enquire the next RBM on the labels to put on the data *restricted* to the data subsets obtained at the previous step. The critical part of this approach is that we are charging with the most important splits of the tree – the creation of the first branches – the youngest, most inaccurate models. As a consequence, it might happen that some data points find themselves in a completely wrong branch of the tree even for a slight inaccuracy in the classification by the first "inexperienced" RBM.

A much better approach is to look at the process *backward* in time. In other words, we start with the oldest machine and we cluster the data into the many fixed points that the free energy will have at that stage of the learning. Then, we move to a younger RBM and we run the clusterization by using as initial conditions not the original data, but the fixed points obtained in the previous step. This is like constructing the tree starting from the leaves and then moving toward the roots rather than the contrary. Proceeding this way assures us that the youngest machines find the initial conditions already close to their fixed points, and there is no issue in the accuracy of classifying points that lay at the boundary between two basins of attraction. The two approaches to the baton passing are depicted in Figure 4.1, while the resulting tree construction is sketched in Figure 4.2.

At this point, a legitimate objection would be to point out that the first clustering steps are performed by old machines that we know are badly approximated by the mean-field free energy. This observation is indeed true but, nevertheless, it only means that we cannot rely too much on the separation of the data at the level of the leaves of the tree. The splittings that are closer to the tree's root are made by machines for which the mean-field free energy provides a good approximation, and very often the relevant features of the data that we are mostly interested in are located at this level.

4.1.3 Scoring the Trees

Once we have built the tree, we need a quality assessment procedure that acts on two different levels. At *inter-tree* level, it has to tell us which tree is better among many generated on the same dataset but with different hyper-parameters, such as the choice of the ages for constructing the tree or the RBM model itself. At *intra-tree* level, instead, it has to provide us with the optimal depth of the tree that better describes the categories of the data. Indeed, if the tree is too shallow we do not have a proper separation of the data into categories and we do not get much useful information out of it. On the other hand, if the tree is too deep we might find substructures of the data that are not really meaningful nor trustworthy, as we discussed above. The optimal depth of the tree will tell us how to identify the categories in the data, and besides that, we will also have a relational structure of the dataset.
Clearly, there is no way of evaluating any kind of inference about the data without having some "ground-truth" information about them. In our case, the assessment procedure we developed relies on the labels of the data that we have beforehand. It is sufficient to have only a fraction of labelled data, although the more labels we have and the more reliable the quality of the measure is.

Before moving further, let us introduce some notation. Suppose that we chose an ordered set of ages, $\{t^k_{\text{age}}\}_{k=1,\dots,N_{\text{T}}}$, such that $t^k_{\text{age}} < t^{k+1}_{\text{age}}$. Those ages correspond to the same RBM model but trained until different times. We call those RBMs *constructors* and we denote the ordered set of constructors as $\mathcal{R}$. Running the tree construction algorithm from t^1_{age} until t^k_{age} generates a tree of depth k that we indicate

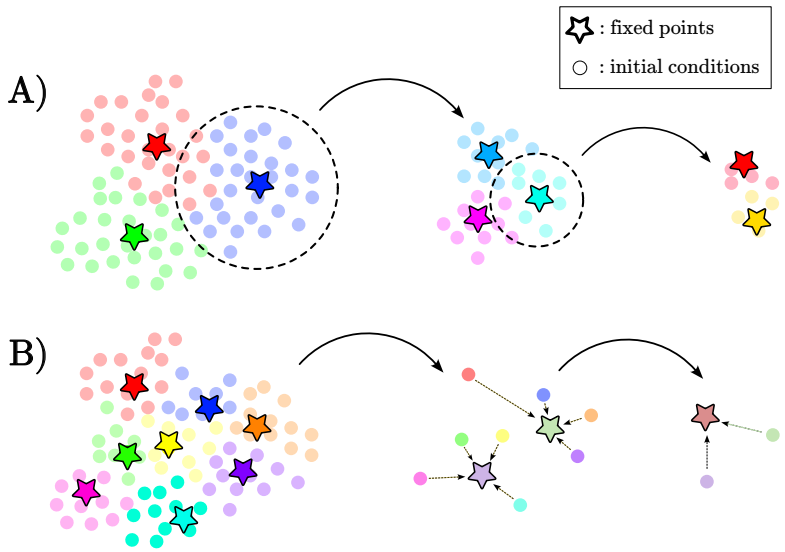

Figure 4.1: A) Wrong method of constructing the tree. The first "young" machine classifies the data into the three colors represented here: blue, red and green. Then we start again from each of those subsets, for example the blue one in this case, and we classify again the data using an older machine. The problem with this method is that the most important splits are the first ones, and they are performed by the most poorly trained machine. B) Correct approach to the baton passing. We start by clustering and classifying the data with the most trained machine, and then we repeat the procedure with the younger RBMs, each time starting from the fixed points obtained at the previous step.

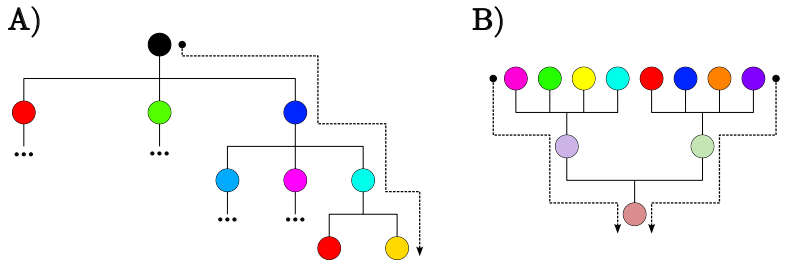

Figure 4.2: A) Wrong method of constructing the tree corresponding to panel A) of Figure 4.1. The tree is constructed starting from the root and proceeding down toward the leaves. B) Correct way of constructing the tree corresponding to panel B) of Figure 4.1. The tree is generated starting from the leaves and progressively aggregating the branches until the root is reached. The colors of the nodes correspond to those of Figure 4.1.

as τ^k. We can therefore define the set of trees that can be generated using $\mathcal{R}$ as $\mathcal{T} = \{\tau_k\}_{k=1,\dots,N_\mathrm{T}}$ with $\tau^k \subset \tau^{k+1}$, meaning that τ^{k+1} is obtained adding one more layer to τ^k. Hence, N_T can be interpreted as the number of levels of the deepest tree in $\mathcal{T}$.

To evaluate the trees $\tau_k \in \mathcal{T}$, we constructed two scores that address the two competing requirements we would like to fulfil.

- **Accuracy Score (AS)**: This score accounts for the wish that all the leaves connected with the same branch should have the same label. Let us define $\mathcal{B} = \{B_i\}_{i=1,\dots,N_\mathrm{B}}$ as the set of N_B branches in the deepest level of the tree τ_k and $\mathcal{L}$ as the set of possible labels. For each $i = 1, \dots, N_\mathrm{B}$, we denote as $N_{i\ell}$ the number of leaves in $B_i \in \mathcal{B}$ that have label $\ell \in \mathcal{L}$. Notice that we are requiring that only a subset of the leaves has to bring the labels, so $\sum_{i\ell} N_{i\ell}$ is the total number of labelled data in the tree, not the total system's size. From the set $\{N_{i\ell}\}_{\ell\in\mathcal{L}}$ we can compute $N_i^* = \max_\ell\{N_{i\ell}\}$ as the number of leaves that have the mostly present label in the branch B_i and $M_i = \sum_\ell N_{i\ell}$, which is the total number of leaves with a label inside the branch. Then, we define the Accuracy Score of the tree as

$$AS(\tau_k) = \frac{\sum_i s_i M_i}{\sum_i M_i} \quad \text{with} \quad s_i = \frac{N_i^*}{M_i} = \frac{N_i^*}{\sum_\ell N_{i\ell}}, \tag{4.1}$$

which is positive definite and bounded from above at 1. Note that weighting the sum with the branch sizes serves for enhancing the role of big branches with respect to the smallest ones. If a branch does not contain any leaves with a label, it does not contribute to the score. Also, it is easy to realize that, by construction, this score is a never-decreasing function of the depth of the tree. To see this, one can consider the extreme case of a tree in which every leaf has its own branch, in which case we have $AS(\tau_k) = 1$. To distinguish this case from the perfect one we need to consider another score that counterbalances the AS.

- **Entropy Score (ES)**: The undesired characteristic of the "one-leaf-one-branch" tree we just mentioned is that all leaves having a given label are spread around into different branches. We therefore need a score that rewards those configurations in which all the data having the same label find themselves in the same branch of the tree.
  Let us consider the label $\ell \in \mathcal{L}$ and let us call N_{L} the total number of categories we have. Then, from the set of branches $\mathcal{B}$ we can define the vector

$$\boldsymbol{r}_\ell = (\{r_{i\ell}\})_{i=1,\dots,N_{\mathrm{B}}} \quad \text{with} \quad r_{i\ell} = \frac{N_{i\ell}}{N_\ell} = \frac{N_{i\ell}}{\sum_i N_{i\ell}}. \tag{4.2}$$

  In words, $r_{i\ell}$ is the fraction of leaves with label ℓ that are attached to the branch B_i. Since $\boldsymbol{r}_\ell$ is a vector belonging to the N_{B}-dimensional simplex, i.e. it is a positive and normalized vector, its squared norm has the interesting property of being maximum when $r_{i\ell} = \delta_{i,j}$ for some $j \in 1, \dots, N_{\mathrm{B}}$ and minimum for $r_{i\ell} = 1/N_{\mathrm{B}}\ \forall i \in 1, \dots, N_{\mathrm{B}}$. But that is exactly the property we were looking for, so we can define the Entropy Score as

$$ES(\tau_k) = \frac{\sum_\ell s_\ell}{N_{\mathrm{L}}} \quad \text{with} \quad s_\ell = ||\boldsymbol{r}_\ell||_2^2 = \sum_i (r_{i\ell})^2. \tag{4.3}$$

  Let us notice that this score is a sort of measure of the entropy of the tree, because in some sense it assesses "how ordered" the tree is. In fact, starting from the vectors $\boldsymbol{r}_\ell$ we could have also computed the Shannon entropy of the tree, but this would have yielded an increasing function of the tree's depth. Instead, the ES we defined decreases as the tree becomes deeper, and this behaviour is needed to contrast the trend of the AS measure.

For a given set of trees $\mathcal{T}$, we can plot the AS and ES scores as a function of the tree depth, as shown in Figure 4.3 for an example of tree construction using MNIST data. By taking the average of the two measures as a global score, we find a nicely peaked function which displays a maximum in correspondence to the optimal depth. Through a by-eye inspection of several examples, we can confirm that the depth prescribed by this scoring procedure seems to be indeed the best compromise between simplicity and resolution in the data classification. We can therefore take as a global score of the tree set $\mathcal{T}$ the maximum value of the average score, and we can classify the data using the branches of the optimal tree with the optimal depth, τ^{opt}.

4.1.4 Choosing the Ages

The tree construction we described above is uniquely defined, i.e. parameter-free, once the set of ages of the RBM to use has been specified. However, the choice of the ages is crucial for the proper functioning of the algorithm, and it has to balance between accuracy requirements and computational costs. Choosing the ages too coarsely brings the risk of losing some important parts of the training dynamics, and the final result might be inaccurate. On the other hand, we cannot afford to use too many ages for constructing the tree, both because saving all those models has a not negligible weight in terms of memory and, above all, because the computational time needed to run the algorithm becomes too long (although we are still talking about a linear increase with the number of ages). Ideally, one would like to find a relation between the geometry of the free energy landscape (e.g. the number of fixed points that are present) and some observable that can be monitored during the training, such as the spectra of the weight matrix, so as to decide optimally and automatically when to save the state of the RBM. Unfortunately, from a first study of the learning dynamics, we were not able to find any such a convincing relation. We leave more extensive studies in that direction for possible future works.

tree	dataset	epochs	minibatch size	total gradient updates[1]	$\mathbf{N}_{\text{Gibbs}}$	learning rate	$\mathbf{N_h}$
Figure 4.4	MNIST	5000	500	10^5	50	10^{-3}	500
Figure 4.5	MNIST	5000	500	10^5	50	10^{-3}	500
Figure 4.7	HG	10000	500	$9 \cdot 10^4$	50	10^{-3}	500
Figure 4.8	HG	10000	500	$9 \cdot 10^4$	50	10^{-3}	500
Figure 4.10	WW	5000	500	$1.4 \cdot 10^5$	50	10^{-3}	500

Table 4.1: Technical specifics of the RBMs used for generating the trees reported below. All machines have been trained in the PCD mode and implement a Bernoulli hidden potential.

Instead, we stuck with the following "common sense" approach. First of all, during the training, we saved a certain number of linearly time-separated models (e.g. ~ 100). Then, we performed a first sifting by running the naive mean field self-consistent equations (3.20) until convergence for each age, starting from the oldest model. Each time, we counted the number of fixed points and we added the model to the set of tree constructors $\mathcal{R}$ only if the number of fixed points had decreased with respect to the last constructor. In general, the naive mean field equations converge way faster than the TAP version, but they tend to produce additional spurious solutions quite soon. For that reason, when constructing the tree, we added a new layer only when the number of fixed points found with the TAP equations decreased from the previously added layer.

4.2 Results

In this section, we present the results obtained by applying the proposed tree construction algorithm to the three datasets we described in section 2.1. In Table 4.1 we report the technical specifics of the RBMs used for constructing each tree.

4.2.1 MNIST Dataset

In Figures 4.4 and 4.5 we report the trees obtained by running the tree algorithm on 1000 samples from the test set of the MNIST dataset using two different RBMs: the first was trained in semi-supervised mode using all the labels, while for the second only 1% of the available labels were used. In Figure 4.3, instead, we show the scoring profiles of the two trees. An interesting question here was to check if using a larger fraction of data with the label during the training would improve the quality of the tree and to what extent. Quite surprisingly, the tree "trained" with fewer labels has a global score of 0.74, which is higher than the 0.69 of the other one. A visual inspection of the two trees confirms the score's judgment, in that the tree with 1% of labels seems to better recognize the true categories of the data. However, from a first superficial study of the quality of the trees obtained by changing the labels' proportion during the training, we did not find any clear relation, and further studies in this direction are needed.

If we focus now on the hierarchical relation of the data found by the algorithm, we might gain some qualitative information on how the machine discriminates the different digits. Moreover, the internal nodes of the tree represent actual points in the magnetizations' space, and thus we can visualize them to see how the data representation evolves during the training. In Figure 4.6, for clarity of visualization, we summarized the splittings done by the algorithm on the tree in Figure 4.5.
We observed that the first thing the RBM learns is how to represent the digit 0. Then, when the second mode emerges, it divides the dataset into either the representation of the digit 0 or 1. This splitting does not seem to have topological nature, because the digit 9 – which has a hole – is classified differently than the digit 0. This might be reasonable if we think that the images enter the RBM

[1] total updates = train set size × epochs / minibatch size.

flattened, and the machine does not have any convolutional layer that might effectively catch local features such as holes. A hypothesis is that the RBM is rather sensitive to the symmetries of the digits which, once the image is flattened, translate into repeated patterns of the nodes' values 0 and 1. Proceeding down the tree, it becomes quite risky to make a hypothesis on how the model discriminates the data based only on a few trees. For instance, in Figure 4.4 the digit 1 is nicely isolated, while the digits 5 and 3 are often confused. On the other hand, in the tree of Figure 4.5 the digit 3 is readily recognized (as illustrated in Figure 4.6), while the 5 and the 1 are separated only at a deeper level of the tree. This suggests that semi-supervised learning might alter the way the machine would naturally subdivide the data, and not always in the correct direction.

4.2.2 Human Genome Dataset

In Figure 4.7 we report the trees obtained on the test set and part of the training set for the Human Genome dataset with an RBM trained in semi-supervised mode with 100% of the major labels given. Concerning the ethnic groups, in all cases, our RBM was not able to discriminate between Americans and Europeans at any level of the tree. This result seems quite plausible, at least from a superficial analysis. On the other hand, in both cases, there is a subgroup of Americans that are classified differently from the rest, and those individuals seem to belong to the same one or two sub-populations. Apart from this, both machines give impressive accuracy in identifying the different ethnic groups, with a perfect classification in the most favourable case of the training set. Concerning the sub-labels, apart from the already mentioned group of Americans, the machine can identify a part of the members of an East Asian community. The resolution of the model is not enough to find any other sub-structure at this stage.
In Figure 4.8 we show the result yielded by an RBM trained in a purely unsupervised way. We recover almost all the information we get from the machines trained with the labels, although this time the model is not able to properly distinguish between South Asian and East Asian. Hence, in this case, using the labels for the training seems to enhance the discriminative power of the RBM. As a final comment on this part, we notice that the semi-supervised training has been carried out with only the information on the major labels. It would be straightforward, though, to incorporate also the sub-labels in the training process and see if this can be decisive for increasing the resolution of the model.

4.2.3 WW Dataset

In Figure 4.10 we report the tree obtained on all the WW protein domain sequences already classified by previous works for a direct comparison. We remark that the RBM employed was trained without any supervision, so no labels were used. The score curves in Figure 4.11 define, for each set of labels, which is the most compatible classification inside the tree. By looking at the scores over the experimental labels, in all three cases we find quite good results, although these measures are not very robust given the scarce amount of data. The main comparison has to be done with the classification provided by ProfileView [31], whose results, however, have to be taken carefully since those labels are inferred and not obtained experimentally.
Of the six categories identified by ProfileView, two of them (the yellowish and the blueish ones) are mostly classified accordingly. A third category (indicated in light blue) is clearly recognizable, but it is discovered deeper in the tree. A future improvement of the current method might go in the direction of allowing the recognition of categories at different levels in the tree. Apart from these three classes, the remaining three are not clearly recognized by our method, and are mostly grouped in a branch constituting a fourth category. Also Tubiana et al in [30] managed to clearly recognize three classes but, as opposed to our method, they needed prior knowledge to be able to do this.

4.3 Discussion

In this chapter, we presented a new approach for generating relational trees of data and identifying different categories by leveraging the learning dynamics of RBMs. As always, each newly introduced

method brings its own strengths and limitations. We discuss here the pros and the cons of the method we developed, and eventually we sketch some possible improvements that can be implemented in future works.

4.3.1 Strengths of the Method

Among the advantages of this method we have:

- As silly as it may seem as an observation, our method allows us to construct a relational tree of the data. This means that, even in the most favourable conditions in which we train the RBM with 100% of data having a label and we apply the algorithm to the training set, still the information that we get about the relational structure of the dataset is valuable.
- The tree construction conceptually resembles the learning process of intelligent beings, where increasing levels of abstraction are added on top of each other.
- Once the particular RBM model has been chosen and the ages for the construction of the tree have been specified, the algorithm is totally parameter-free and no further choices are needed. In particular, we do not have to worry about tuning the parameters of the clustering algorithm, as opposed to other classification methods.
- The algorithm is based on the estimation of the free energy of the model, which embeds all the information extracted from the data by the RBM. In particular, this means that the clustering of the data makes use also of the information provided by the learned external fields, which is not considered in those methods that uniquely rely on the projections of the dataset along the principal components of the weight matrix.
- As opposed to other hierarchical clustering algorithms, there is no notion of *distance* entering in the proposed method[2], hence we do not have to make any assumption over the geometrical properties of the data space.
- As far as the tree is concerned, the RBM does not need to be trained for too long. The interesting splittings of the data are found quite early in the learning.
- The method is very general, and it applies to any type of dataset. Other algorithms, such as ProfileView, are instead domain-specific.
- The internal nodes of the tree are not just abstract constructions, but they represent actual points in the magnetization space. This means that we can generate samples that represent intermediate states of the data, and this might have possible interesting applications in biology.

4.3.2 Limitations of the Method

The main limitations of this method are instead:

- At the actual state, there is no optimal nor unique recipe for choosing the ages of the RBMs that will construct the tree. It also seems that a proper choice of the ages is important for obtaining good results. The method we proposed here yields satisfactory results, but it might be improved in the future.
- Considering the time needed to sufficiently train the RBM (approximately from 1 to 6 hours on a GPU, depending on the dataset and the chosen number of Gibbs steps, k, for the gradient evaluation) and the time required for the tree construction (from few minutes to half an hour on a GPU for a batch of ~ 1000 samples, depending on the dataset), the overall procedure is quite time-consuming. However, we observe that it is still competitive with other proposed methods

[2]More precisely, the DBSCAN algorithm we use to recognize the different fixed points of the free energy internally uses the euclidean distance. However, since at this stage the points are collapsed in very small regions of the space, whatever metrics on the data manifold can be nicely approximated by the euclidean one.

such as ProfileView [31] (around 9 hours for classifying the whole WW dataset, according to the authors).

The fact that one has to save several models during the training (~ 100 in our implementation) requires the allocation of several Gigabytes of disk memory for running the algorithm. Also, executing the code on a GPU is needed for having acceptable processing times.

- The accuracy of the method depends on the degree of approximation with which we can approximate the free energy of the model at different ages. In all the datasets we tested, the second-order expansion of the free energy was enough for properly classifying the data based on the main features of interest. However, for catching finer structures of the dataset and improving the accuracy and reliability of the generated trees, it might be necessary a further step in the Plefka expansion of the free energy.
- The approach presented here is based on the computation of the TAP equations starting from a Hamiltonian of the form (3.21). We developed our implementation based on the simplest case of a Bernoulli potential in the hidden layer, and we presented a general formula at the second order that applies to any distribution of the hidden variables. However, each time we aim at generalizing the model (3.21) (for instance, by introducing a regularization term), we have also to re-derive the corresponding TAP equations, and this might not be an easy task. Also, we found that dealing with more sophisticated distributions, such as the Truncated Gaussian one, might bring computational problems that have to be handled properly.
- The splittings that the RBM performs at the various stages, as far as we can tell, are not related to any universal concept (e.g. a distance between data points), but rather they depend on the way the machine learns from the data. This means that to better understand the criteria of the tree construction, we should investigate further the properties of the learning dynamics of the RBM model. On top of this, the tree that we get is the result of a *specific* trajectory of the learning, and the result might change across different training episodes. To what extent this might influence the tree's structure remains unexplored.

4.4 Future Developments

To conclude, let us delineate some possible directions of improvement for the presented method. First of all, we would like to study more closely the dynamics of the learning with the aim of finding a way to relate some observables to the desired properties we want to have. In particular, this relates to our wish to find an optimal way of selecting the constructors to save during the training. Second, the Plefka expansion might be computed up to the third order, so as to improve the accuracy of the method in estimating the true free energy. Another feature we would like to implement is a criterion for identifying classes of data at different levels in the tree. Indeed, as exemplified in Figure 4.10, not all the features of interest are found at the same "level of abstraction" in the hierarchical structure of the dataset. Finally, it might be interesting to construct trees from an ensemble of RBMs and compare them to see how the result is sensitive to the particular trajectory of the learning.

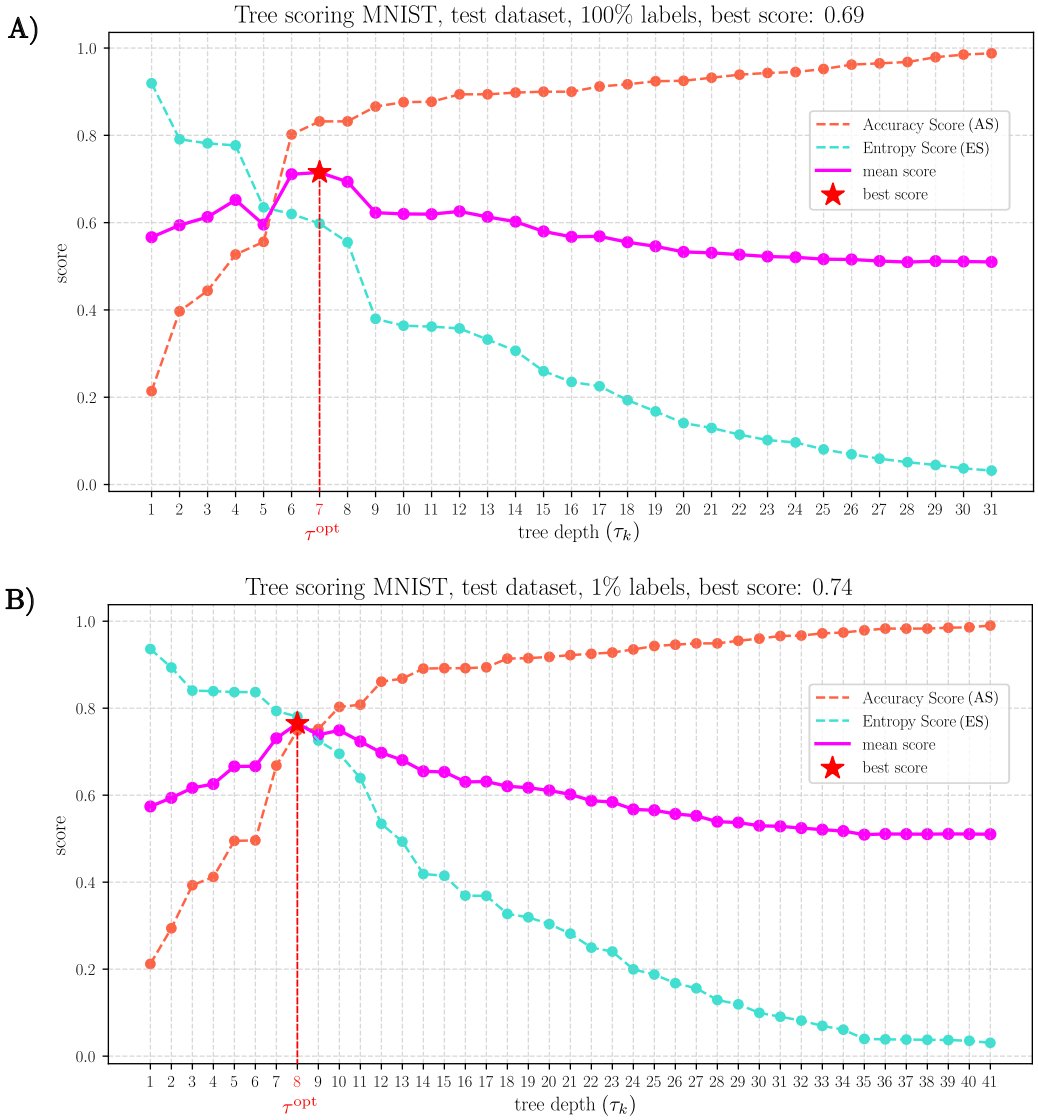

Figure 4.3: Scoring profiles of the trees shown in Figure 4.4 (plot A) and Figure 4.5 (plot B). The Entropy Score tends to decrease as we go deeper into the tree, while the Accuracy Score tends to increase. The best compromise is found as the maximum of the mean between the two scores, and it is indicated with a red star.

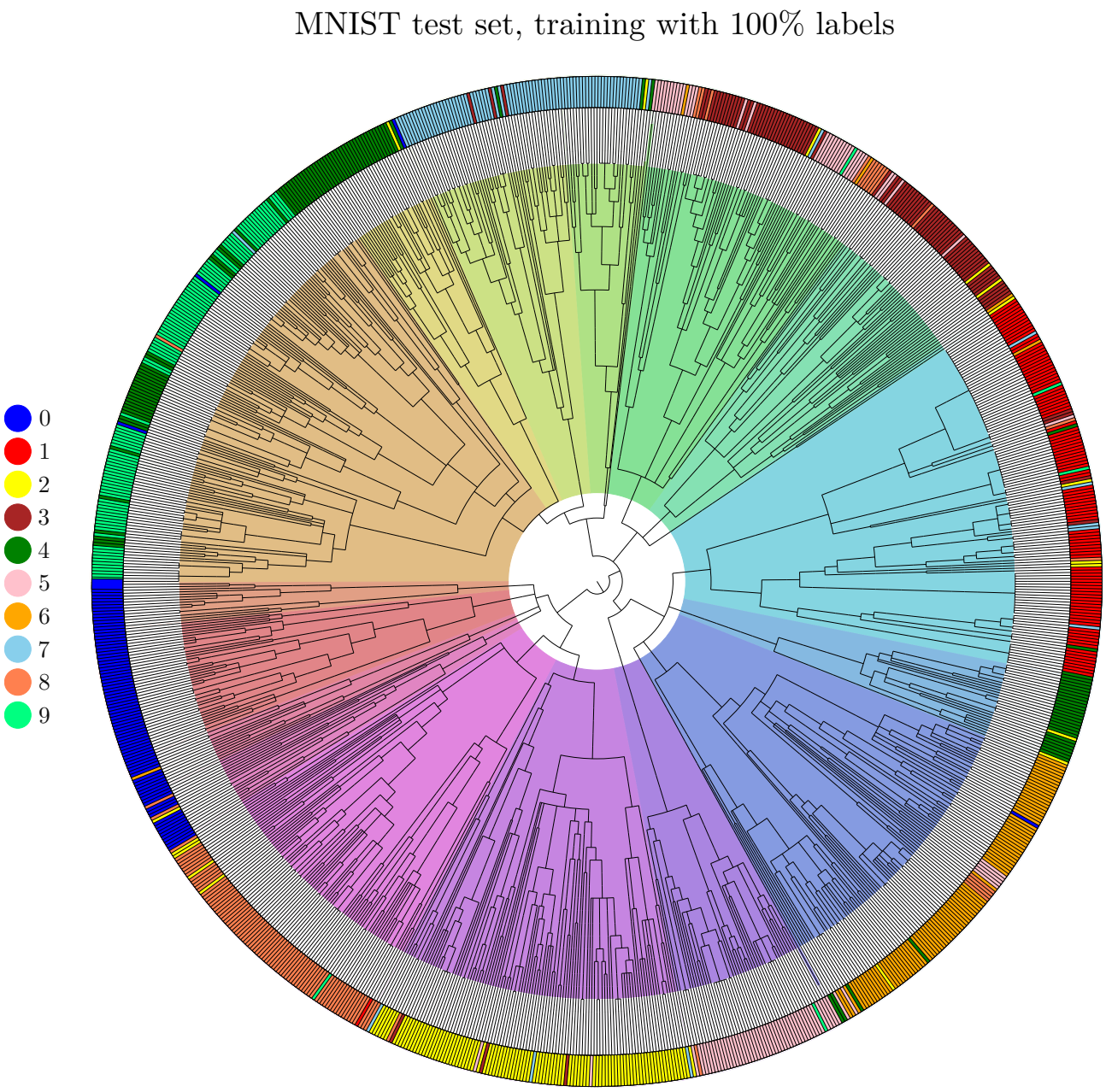

Figure 4.4: Tree of 1000 MNIST images taken from the test dataset. It is obtained using a Bernoulli RBM trained in semi-supervised mode with 100% of the labels. The external colored tags represent the true labels, while the colored regions inside the tree correspond to the classification inferred by the algorithm. The depth chosen for the classification is the best one prescribed by the scoring profile of Figure 4.3 A).

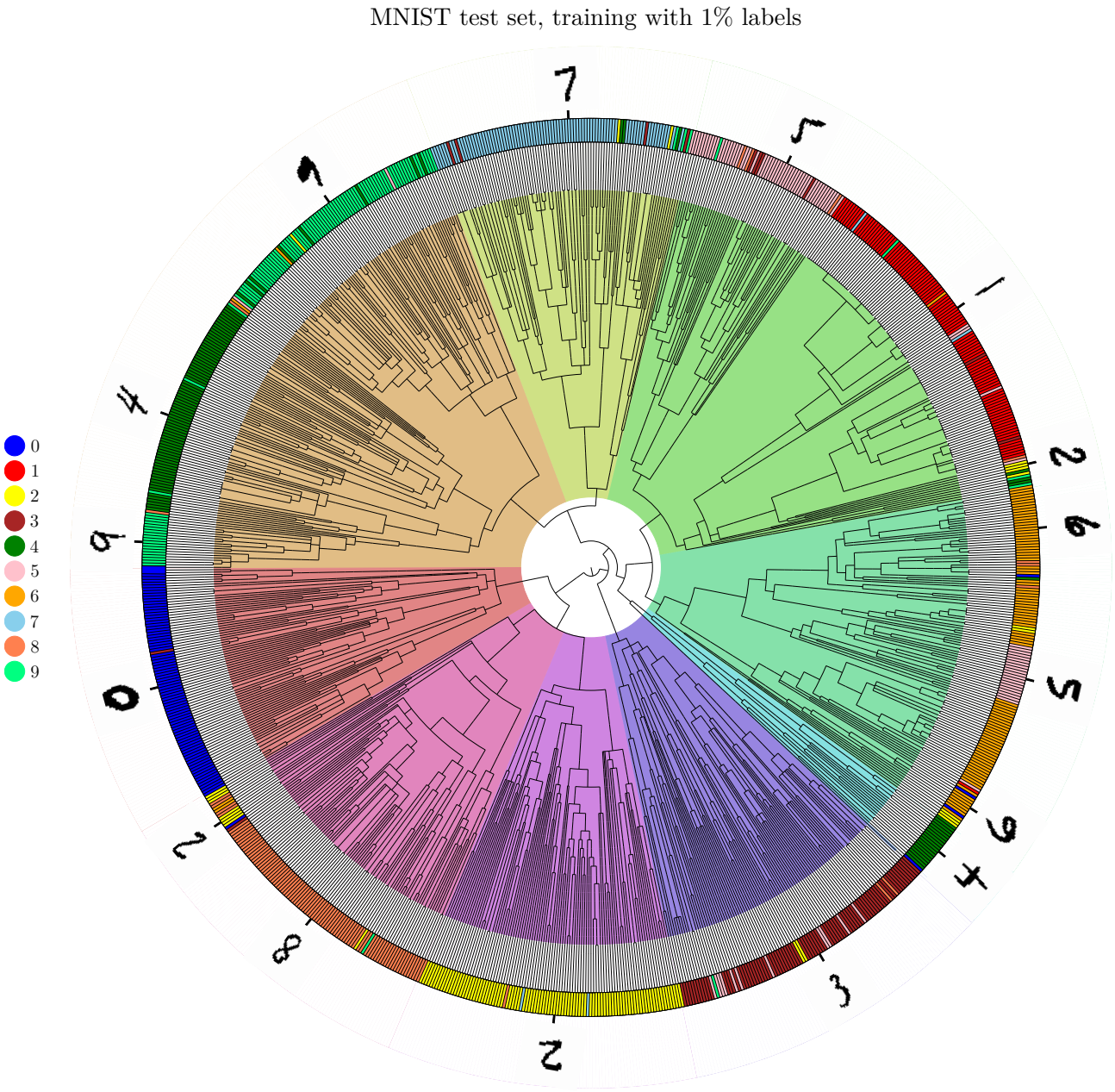

Figure 4.5: Tree of 1000 MNIST images taken from the test dataset. It is obtained using a Bernoulli RBM trained in semi-supervised mode with 1% of the labels. The external colored tags represent the true labels, while the colored regions inside the tree correspond to the classification inferred by the algorithm. The depth chosen for the classification is the best one prescribed by the scoring profile of Figure 4.3 B). We also reported some images of the data at different branches of the tree for a visual comparison.

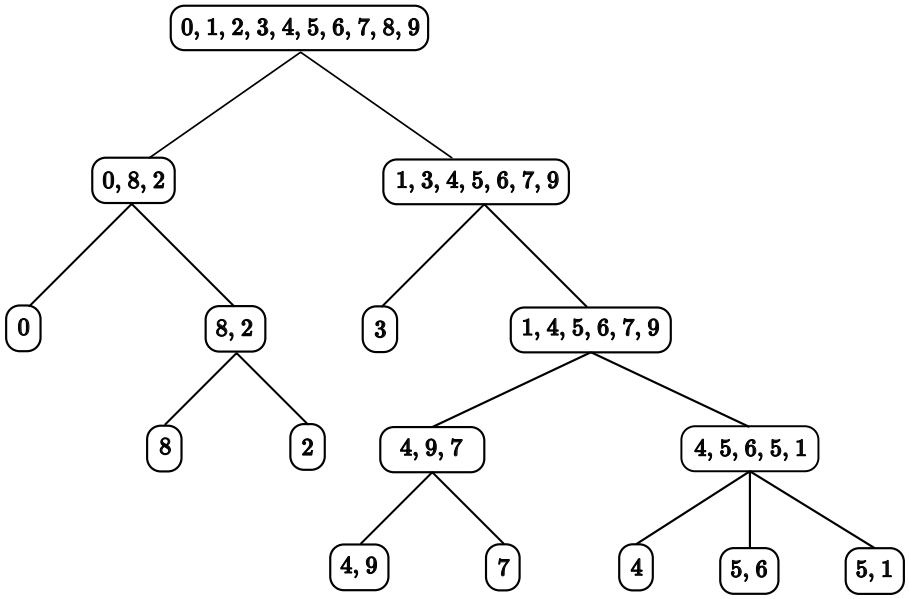

Figure 4.6: Simplified representation of the tree in Figure 4.5 extracted from the training of an RBM using 1% of the labels.

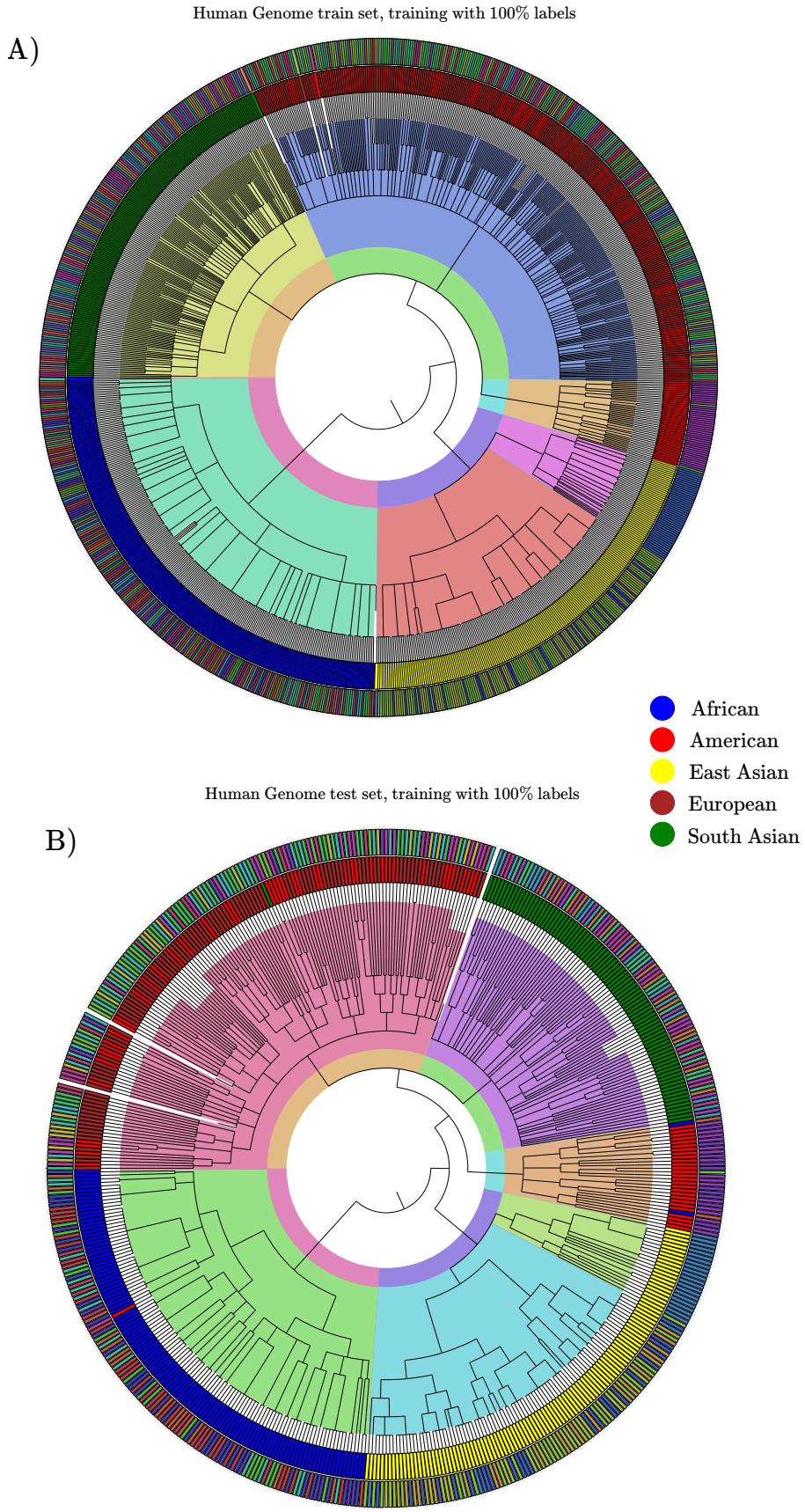

Figure 4.7: Trees obtained on the Human Genome dataset using an RBM trained in semi-supervised mode with 100% of the labels. Plot A is obtained on 1000 samples from the training set, while plot B is obtained on the test set. The two external rings of colored tags represent, from the inner to the outer one, the true labels of the coarser and finer categorization. There are two multi-coloured regions inside the tree. The most internal one corresponds to the optimal depth for the categorization of the coarser labels, as prescribed by the score curves of the left column in Figure 4.9, Plots B and C. The external region is instead the best categorization for the sub-features given by the best scores in the right column of Figure 4.9, Plots B and C. In the legend, we indicated only the meaning of the major labels.

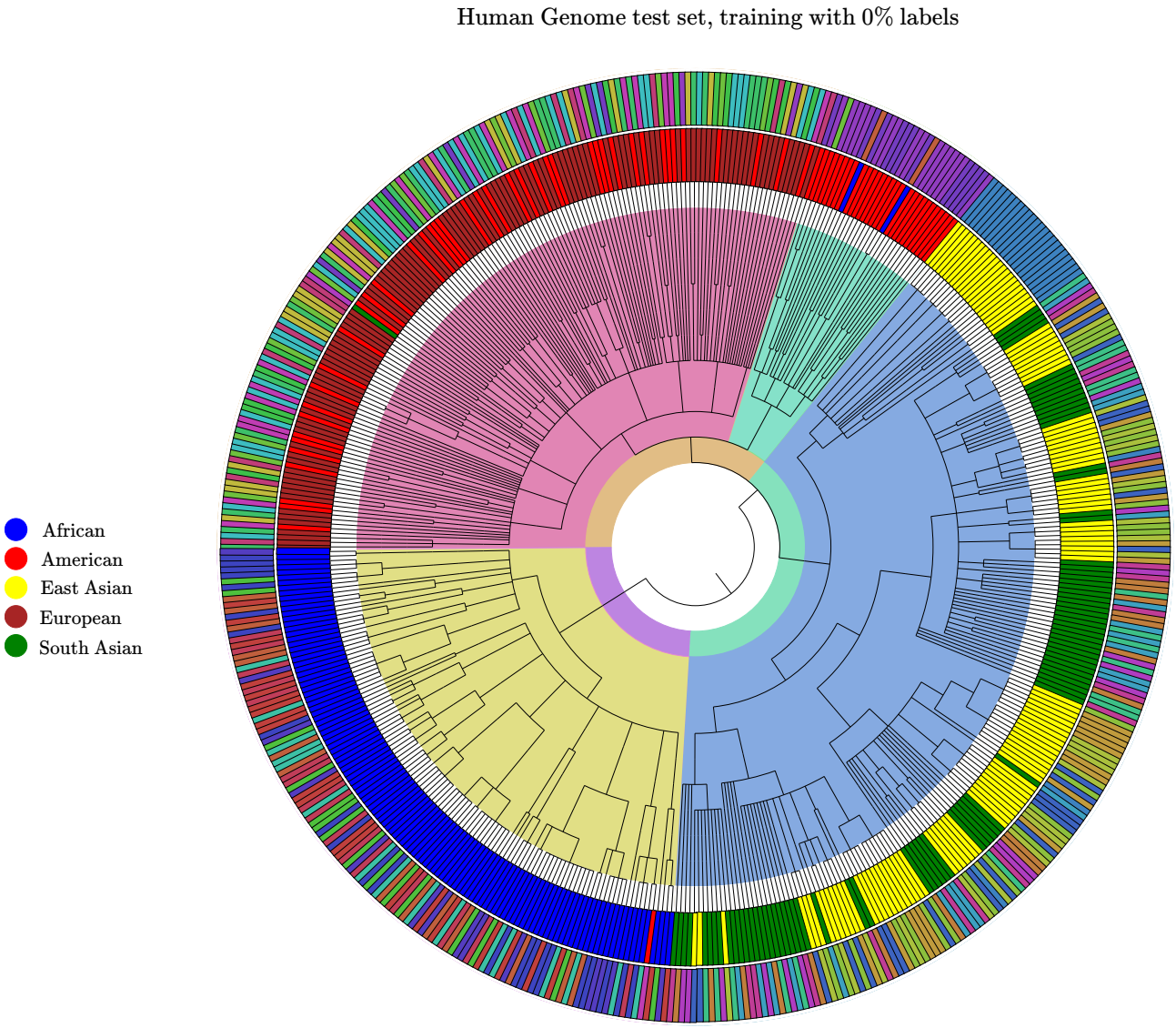

Figure 4.8: Tree obtained from the test set of the Human Genome Dataset using an RBM trained in a purely unsupervised way, i.e. without labels. The two external rings of colored tags represent, from the inner to the outer one, the true labels of the coarser and finer categorization. There are two multi-coloured regions inside the tree. The most internal one corresponds to the optimal depth for the categorization of the coarser labels, as prescribed by the score curves of the left column in Figure 4.9, plot A. The external region is instead the best categorization for the sub-features given by the best scores in the right column of Figure 4.9, Plot A. In the legend, we indicated only the meaning of the major labels.

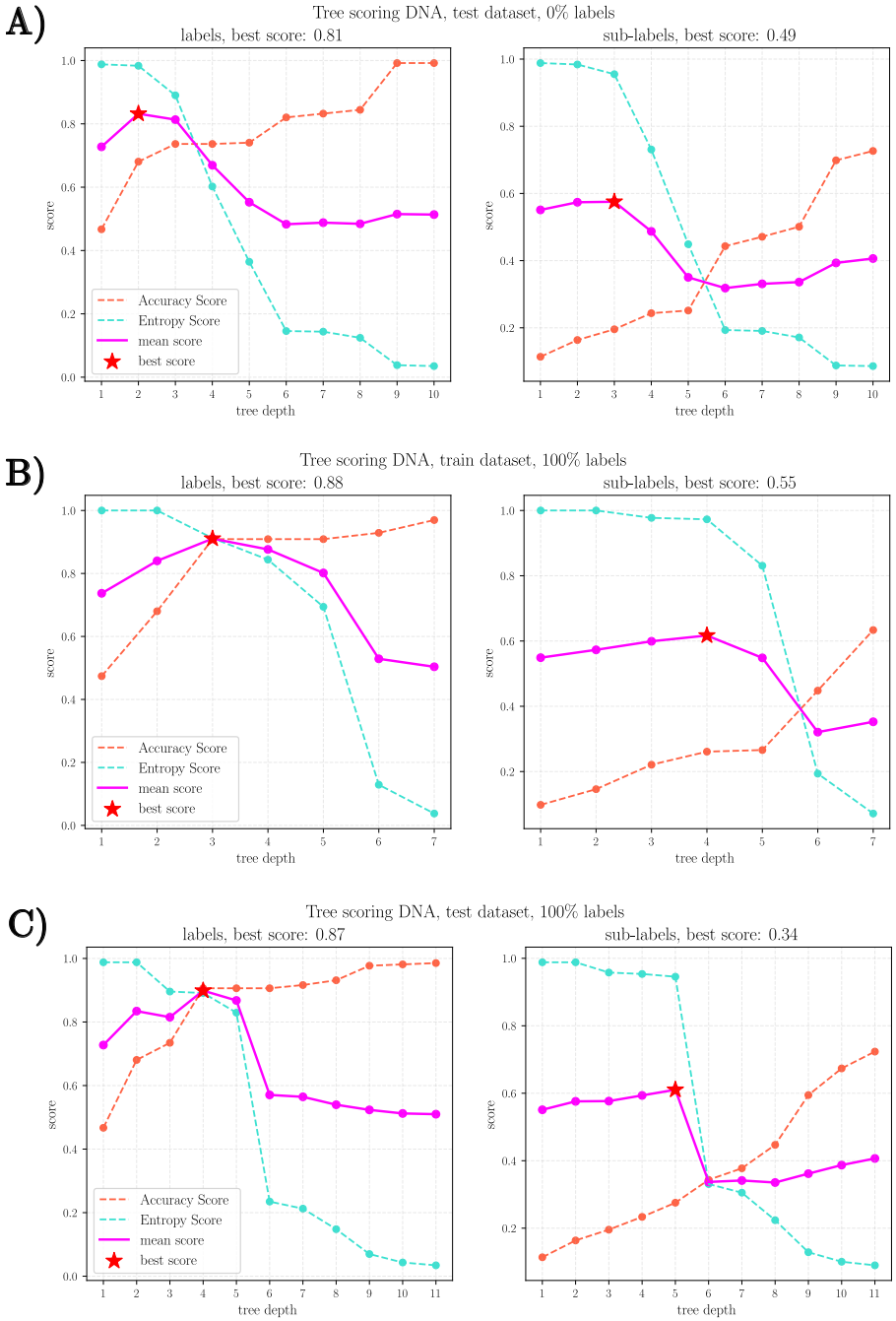

Figure 4.9: Score profiles corresponding to the trees in Figure 4.8 (plot A) and Figure 4.7 (plots B and C). On the leftmost column, we report the scores evaluated on the major labels, namely those concerning the ethnic groups. The rightmost column represents instead the scores obtained on the sub-labels that distinguish the different populations inside the same ethnic groups.

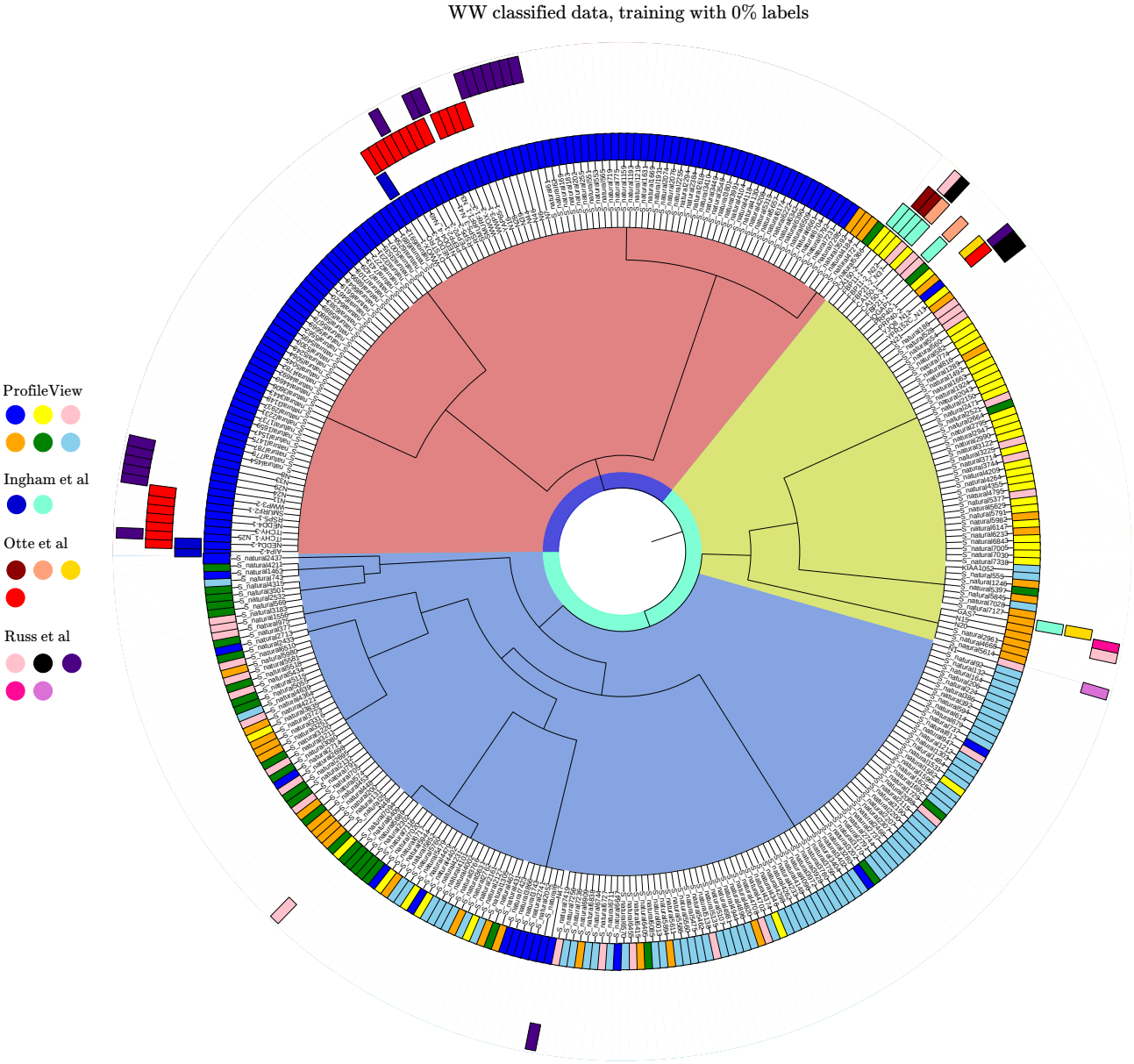

Figure 4.10: Tree obtained on all the previously classified data of the WW dataset, taken both from the test set and training set. The RBM used was trained in a purely unsupervised way (no labels were given). The four external rings of colored tags represent the annotations previously identified by, in order from the inside out, ProfileView [31] (inferred), Ingham et al [14], Otte et al [23], and Russ et al [26] (the last tree are experimental). The two internal multi-coloured regions correspond to the categories identified by our algorithm using the scores in Figure 4.11. The most internal classification gives the best score compatible with Russ et al, while the external one gives the best agreement with the other three works. For each leaf, we also reported the name of the specific WW domain sequence.

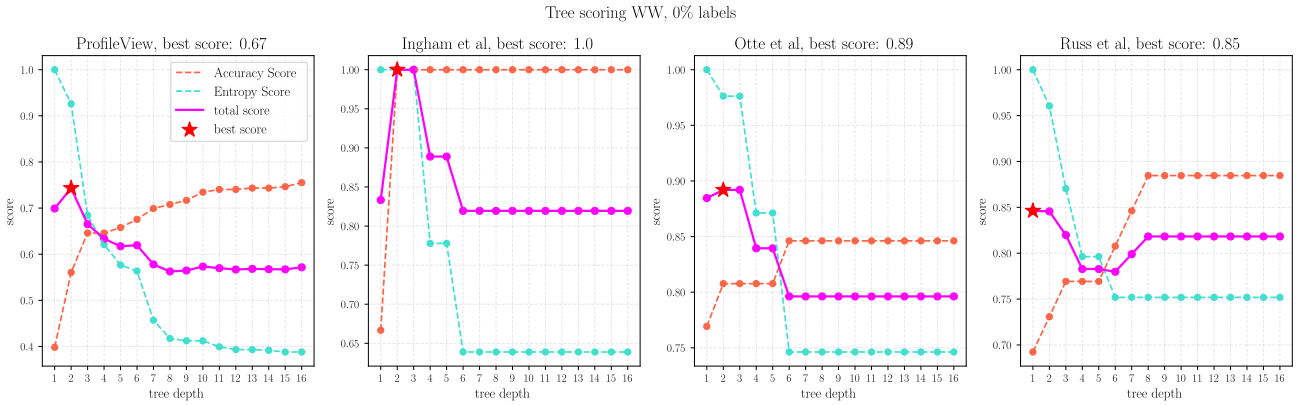

Figure 4.11: Score profiles of the tree in Figure 4.10 based on the labels given by [31, 14, 23, 26].

Conclusions

In this thesis work we presented the Restricted Boltzmann Machine, a generative machine learning model which is unique in combining expressiveness, explainability and flexibility.

In the first part, we tried to condense some of the main theoretical results from the literature that allow us to understand some of the aspects that rule the learning process of the model, and we extended some of them to deal with categorical variables.

Then, we discussed some of the recent advances in the assessment of the quality of the learned models and we presented the advantages and limitations of the main training protocols. We showed how the RBM can be easily modified in order to make it work in a semi-supervised fashion, hence allowing for label inference and conditioned generation of data.

We continued by discussing the use of the mean-field theory for approximating the free energy of the model up to any order in the large-temperature limit, and we derived the TAP equations for the Potts RBM with a generic hidden potential. By iterating the TAP equations until convergence, we showed how it is possible to have a clue about the free energy landscape of the RBM and to cluster the data according to the closest internal representations learned by the model.

Finally, in the last chapter, we presented a novel algorithm based on the mean-field theory that allows us to generate relational trees of the data. The proposed method makes use of the information contained in the learning dynamics of the model, a topic that is still not very well studied in the literature. The interest in this new approach is double. On one side, it represents a new promising way of producing hierarchical descriptions of the datasets under study, finding interesting applications in particular in the field of genomics. On the other hand, the trees obtained through the algorithm can be inspected with the aim of qualitatively understanding what are the guiding principles that enter during the learning process. The proposed method proved promising on all the tested datasets, and there is room left for future improvements.

Appendix A

PCA and SVD

In this work, the concepts of *Principal Component Analysis* (PCA) and *Singular Value Decomposition* (SVD) are used many times, and sometimes the two terms are interchanged. In this appendix we show how the two methods are connected.

Suppose to have a dataset $\mathcal{D}$, which is a $N_{\text{data}} \times N_v$ real valued matrix. For simplicity, suppose that the dataset has been centered, so that the columns of the matrix have zero mean. The PCA consists on decomposing the covariance matrix of the data according to

$$\text{Cov}(\mathcal{D}) = \mathcal{D}^T\mathcal{D} = W\Sigma W^T, \tag{A.1}$$

where Σ is a $N_v \times N_v$ diagonal matrix and W is a $N_v \times N_v$ matrix whose columns are the eigenvectors of the covariance matrix. Projecting a data sample $\boldsymbol{v}_{\text{d}}$ along the principal directions of the covariance matrix means obtaining a vector $\boldsymbol{u}_{\text{d}} = W^T\boldsymbol{v}_{\text{d}}$ whose components are independent from each other. The ordered set of eigenvalues in Σ represent the variability of the dataset along the corresponding components, meaning that we can reduce the dimensionality of the data vectors by just keeping the first few projected components, which normally contain the large part of the information present into the dataset.

The SVD, instead, is a decomposition of the data matrix of the form

$$\mathcal{D} = USV^T, \tag{A.2}$$

where S is a diagonal matrix and U and V are, respectively, a $N_{\text{data}} \times N_v$ and a $N_v \times N_v$ matrices. We can relate the two transformation by computing

$$\mathcal{D}^T\mathcal{D} = (USV^T)^T(USV^T) = VS^T\underbrace{U^TU}_{\mathbb{1}}SV^T = VS^2V^T, \tag{A.3}$$

which compared with equation (A.1) gives $\Sigma = S^2$ and $V = W$. The elements on the diagonal of S are just the square root of the eigenvalues of the covariance matrix, and they are called *singular values*. Since the projection matrices between PCA and SVD are the same, projecting a data sample along the principal directions of $\mathcal{D}$ is the same as projecting it along the eigenvectors of the covariance matrix.

Appendix B

Technical Notes on the Truncated Gaussian Potential

Implementing the TG potential into the RBM brings the disadvantage of having to compute the erf function and the inverse erf function, which is quite of an expensive task in terms of computational time. To overcome this issue, instead of using the exact definitions, we rather used the following approximations that are fast to evaluate and easy to parallelize:

$$c = \frac{8(\pi - 3)}{3\pi(4 - \pi)}, \tag{B.1}$$

$$\mathrm{erf}(x) \approx \mathrm{sgn}(x)\sqrt{\frac{1 - \exp\left(-\frac{4x^2}{\pi + cx^2}\right)}{1 + cx^2}}, \tag{B.2}$$

$$b = -\frac{2}{\pi c} - \frac{1}{2}\log\left(1 - x^2\right), \tag{B.3}$$

$$\mathrm{erf}^{-1}(x) \approx \mathrm{sgn}(x)\sqrt{b + \sqrt{b^2 - \frac{1}{c}\log\left(1 - x^2\right)}}. \tag{B.4}$$

In Figure B.1, insets B) and C), we show that the two approximations are indeed really good if compared with the true functions. However, when computing the Φ function (1.48), we verified that the approximated erf function displays an instability in the interval $2 < x < 7$. For this reason, we used the exact erf function provided by the Pytorch library when we had to compute Φ, and the approximated version when computing the inverse cumulative density function (1.51). In the inset B) of Figure B.1 we show how we can use the approximated version of (1.51) to correctly sampling from a Truncated Gaussian with parameters $b = \sigma = 1$. Using the naive ReLU activation function would instead produce a very high peak at $h = 0$, hence altering the true shape of the distribution.

Another issue that enters when computing Φ is the numerical instability that shows up when $x \gtrsim 7$, as shown in the inset A) of Figure B.1. To handle this problem, we evaluated the function using the definition (1.48) when $x \leq 7$, and the asymptotic expansion

$$\Phi(x) \approx \frac{1}{x} - \frac{1}{x^3} + \frac{3}{x^5} \tag{B.5}$$

when $x > 7$.

The last technical problem of this implementation is about sampling from the Truncated Gaussian using (1.51) when $b - I(\boldsymbol{v}) \gtrsim 6$. In this case, indeed, it may happen that the inverse erf function has to be evaluated very close to the asymptotes $x \in \{-1, 1\}$. In that scenario, the limited precision of the computer prevents us form correctly compute the function (1.51), yielding infinite or negative values of h. We overcame this issue by clamping at $b - I(\boldsymbol{v}) = 6$ the shift parameter of the Truncated Gaussian. The assumption is that the distortion introduced by doing so should not be relevant, since for $b - I(\boldsymbol{v}) = 6$ the probability of sampling far from $h = 0$ is already very small.

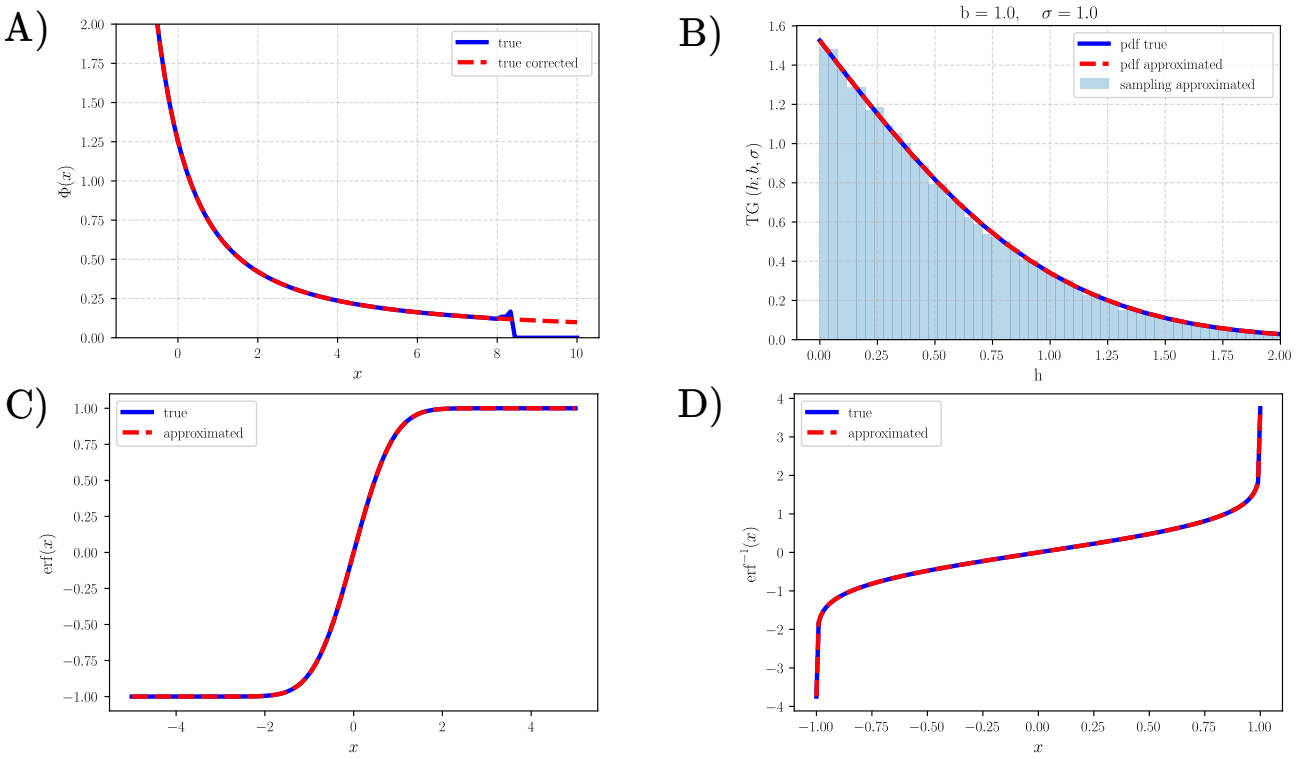

Figure B.1: A): Comparison between the true Φ function (1.48) and the corrected version with the asymptotic expansion (B.5) when $x > 7$. B): The lines represent the exact evaluation of the TG (blue line) and the approximated version using (B.2) (red dashed line). The shaded area represents the histogram obtained by sampling from the TG using (1.51) and the approximated functions (B.2) and (B.4). C) and D): Comparison between the true and approximated erf and inverse erf functions.

Bibliography

[1] 1000 Genomes Project Consortium Corresponding authors Auton Adam adam. auton@ gmail. com 1 b Abecasis Gonçalo R. goncalo@ umich. edu 2 c et al. "A global reference for human genetic variation". In: *Nature* 526.7571 (2015), pp. 68–74.

[2] Nicolas Béreux et al. "Learning a Restricted Boltzmann Machine using biased Monte Carlo sampling". In: *arXiv preprint arXiv:2206.01310* (2022).

[3] Simona Cocco et al. "Inverse statistical physics of protein sequences: a key issues review". In: *Reports on Progress in Physics* 81.3 (2018), p. 032601.

[4] Saloni Dash et al. "Privacy preserving synthetic health data". In: *ESANN 2019-European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2019.

[5] Aurélien Decelle, Giancarlo Fissore, and Cyril Furtlehner. "Spectral dynamics of learning in restricted Boltzmann machines". In: *EPL (Europhysics Letters)* 119.6 (2017), p. 60001.

[6] Aurélien Decelle, Giancarlo Fissore, and Cyril Furtlehner. "Thermodynamics of restricted Boltzmann machines and related learning dynamics". In: *Journal of Statistical Physics* 172.6 (2018), pp. 1576–1608.

[7] Aurélien Decelle and Cyril Furtlehner. "Restricted Boltzmann machine: Recent advances and mean-field theory". In: *Chinese Physics B* 30.4 (2021), p. 040202.

[8] Aurélien Decelle, Cyril Furtlehner, and Beatriz Seoane. "Equilibrium and non-equilibrium regimes in the learning of restricted Boltzmann machines". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5345–5359.

[9] Guillaume Desjardins et al. "Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 145–152.

[10] Matteo Figliuzzi, Pierre Barrat-Charlaix, and Martin Weigt. "How pairwise coevolutionary models capture the collective residue variability in proteins?" In: *Molecular biology and evolution* 35.4 (2018), pp. 1018–1027.

[11] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30 (2017).

[12] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence". In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[13] Geoffrey E Hinton, Terrence J Sejnowski, et al. "Learning and relearning in Boltzmann machines". In: *Parallel distributed processing: Explorations in the microstructure of cognition* 1.282-317 (1986), p. 2.

[14] Robert J Ingham et al. "WW domains provide a platform for the assembly of multiprotein networks". In: *Molecular and cellular biology* 25.16 (2005), pp. 7092–7106.

[15] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[16] Oswin Krause, Asja Fischer, and Christian Igel. "Algorithms for estimating the partition function of restricted Boltzmann machines". In: *Artificial Intelligence* 278 (2020), p. 103195.

[17] Nicolas Le Roux and Yoshua Bengio. "Representational power of restricted Boltzmann machines and deep belief networks". In: *Neural computation* 20.6 (2008), pp. 1631–1649.

[18] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[19] Yann LeCun et al. "Neural networks: Tricks of the trade". In: *Springer Lecture Notes in Computer Sciences* 1524.5-50 (1998), p. 6.

[20] Jan Melchior, Asja Fischer, and Laurenz Wiskott. "How to center deep Boltzmann machines". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 3387–3447.

[21] Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*. Vol. 9. World Scientific Publishing Company, 1987.

[22] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*. 2010.

[23] Livia Otte et al. "WW domain sequence activity relationships identified using ligand recognition propensities of 42 WW domains". In: *Protein Science* 12.3 (2003), pp. 491–500.

[24] Timm Plefka. "Convergence condition of the TAP equation for the infinite-ranged Ising spin glass model". In: *Journal of Physics A: Mathematical and general* 15.6 (1982), p. 1971.

[25] John Rozewicki et al. "MAFFT-DASH: integrated protein sequence and structural alignment". In: *Nucleic acids research* 47.W1 (2019), W5–W10.

[26] William P Russ et al. "Natural-like function in artificial WW domains". In: *Nature* 437.7058 (2005), pp. 579–583.

[27] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016).

[28] Eric W Tramel et al. "Deterministic and generalized framework for unsupervised learning with restricted boltzmann machines". In: *Physical Review X* 8.4 (2018), p. 041006.

[29] Jérôme Tubiana. "Restricted Boltzmann machines: from compositional representations to protein sequence analysis". PhD thesis. Université Paris sciences et lettres, 2018.

[30] Jérôme Tubiana, Simona Cocco, and Rémi Monasson. "Learning protein constitutive motifs from sequence data". In: *Elife* 8 (2019), e39397.

[31] R Vicedomini et al. "Multiple profile models extract features from protein sequence data and resolve functional diversity of very different protein families". In: *Molecular biology and evolution* 39.4 (2022), msac070.