POLITECNICO
DI TORINO

International Master Course in Physics of Complex Systems

Tesi di Laurea Magistrale

Representations of cortical activity using Restricted Boltzmann Machine

Supervisors :
Prof. Rémi MONASSON
École normale supérieure
Prof. Alessandro PELIZZOLA
Politecnico di Torino
Co-Supervisor :
Dott. Sébastien WOLF
École normale supérieure

Candidate:
Leonardo AGUECI

ACADEMIC YEAR 2018-2019

A alla mia famiglia
e a Noemi

Abstract

Restricted Boltzmann Machines (RBM) are neural network models that learn a probability distribution and a representation of data; they belong to the class of Boltzmann Machines. With respect to the lasts, in RBM the learning procedure is simpler and faster. Furthermore, once properly trained, final couplings will directly show the main correlations between visible units.
In this work we applied RBM to a specific cortical neural data set and demonstrate its usefulness in revealing important properties of this brain region.
Starting from the studies of A. Peyrache et al. [17] on medial Prefrontal Cortex's neural activity in a rat, in which, using simple PCA, a **cell assembly** coding for a particular learned rule was found, we tried to reproduce their result, deepening the description of such phenomenon. G. Tavoni et al. [24][25] showed that an Ising model can faithful reproduce such activity quite well, improving what a simple PCA is able to attain. This suggests a possible efficacy of Boltzmann machines, and then of RBM, since they are equivalent to an Ising model.
Even though good results were obtained, the limitations of Kullback-Leibler Divergence (D_{KL}) in such contest forced us to modify the learning rule: in fact, RBM training is based on the principle of Maximal Likelihood Expectation, equivalent to the minimum of such divergence. This is the reason why we redefined our machinery using a metric coming from optimal transport theory, called the Wasserstein distance, instead of D_{KL}, that has the virtues to be smoother and finite in cases where the previous one, conversely, diverges.
The success of the project could represent a simple way to analyze bigger datasets of neural activity measurements, speeding up the research in this field, both for a fast analysis and a simple, graphical representation of the learning mechanisms taking place in the brain.

Contents

List of Figures

Chapter 1

Introduction

In the field of computational neuroscience, one of the main goals is the study of the mechanism of learning and its related sub-processes.
It is widely known, together with other important features that we will overlook, that learning is one of the consequences of plasticity of neural couplings, called synapses, that modify their structure in some ways in order to tune the signal passing through them and, then, the interaction between neurons and the final response of the system to a given stimulus.
One of the main qualities of neural tissue is its high connectivity: each neuron can have thousands of synapses. It allows a particular subgroup of cells to store a big amount of information: as firstly reported by Hopfield [7], even if on a really simplified model compared to the reality, these complex structures can contain many different sequences corresponding to memories; these correspond to minima of the energy function associated to the model and called Attractors. This result can be extended, with some restrictions, to biological neural networks. Despite being essential, high connectivity also makes analysis of neural activity and definition of a model describing it very challenging, especially if summed to the noisy nature of biological cells.
In the context of spatial temporal learning, a well supported belief is that learned rules acquired in some events are finally coded inside a small group of cells, called 'cell assembly', through interaction of them with the hippocampal cortex. Furthermore, during the subsequent sleep of the animal, some mechanism probably associated with consolidation, results in the replay of the spiking patterns associated to the learning period. These two activities can be somehow compared and used to better characterize the cell assembly [17]. However, detection of these structures still remain hard. It then becomes then necessary to use more sophisticated techniques allowing to treat such complex structures.

Applying Statistical Mechanics on the same collection of data , Tavoni et al. [24] [25] better characterized the phenomenon: they treated the case as an Inverse Ising model, achieving an advanced view on the assembly's properties through the definition of neurons' susceptibility, the coactivation function of the assembly, and a better characterization of the reply.
The aim of our project was the realization of an automatized procedure able to detect and characterize the underlying interactions between neurons, through the definition of a simple representation. To achieve such result we used a Restricted Boltzmann Machine [6], particular case of Boltzmann Machines. The last is a graphical model representing a powerful, as well as simple, tool that allows to treat Ising models. It is made by visible and hidden units, respectively associated to input data and to additional variable of the model; from this framework, the Restricted version comes simply allowing only connections between units of different kind. Such a change entails, after a proper training procedure on the dataset using Likelihood maximization, a direct way to visualize the main interactions, simply looking at the coupling matrix. Furthermore, it also determines a strong simplification in the training and sampling procedures.
Although important results were obtained, some detected problems suggested that a different train rule has to be used. For this reason, the last part of the project was to start the definition of a new training framework, based on minimization of the Wasserstein distance. As reported in literature, this choice avoid different stability problems, and direct the machine toward a different, and hopefully better, probability distribution.

Chapter 2

The RBM model

A restricted Boltzmann Machine is a *generative stochastic artificial neural network*, straightforward to train and to treat mathematically, but able to learn the probability distribution of data given as input, even if with some limitations. After their first introduction by Paul Smolensky in 1986 [21], under the name of **Harmonium**, they were largely popularized and further studied by Geoffrey Hinton, who used them in many different applications, with really astonishing performances for such a simple model.
From classification to dimensional reduction, there are many applications of such model, as overall for neural network. A general task performed by artificial neural network is to *extract features*, also applying transformations on data or *decreasing their dimensionality*, in order to find useful *representations* of them.

A simple way to do both of them is Principal Component Analysis: using eigenvectors associated to the highest eigenvalues, you can find a slow dimensional subspace in which *much information* is contained, i.e. selecting axes on which data have a bigger variance and neglecting the others. Though powerful, such method does not often provide a simple interpretation, and further analysis become necessary.
Among all, an important class of models, and by far the most suitable for statistical mechanics treatment, are the well known *Hopfield models*, rather some of its extensions.

2.1 From Hopfield model to Boltzmann machine

A Hopfield network is a form of *single layer Recurrent Neural Network* (RNN), structures able to exhibit temporal dynamic behavior thanks to the presence of *feedback*,

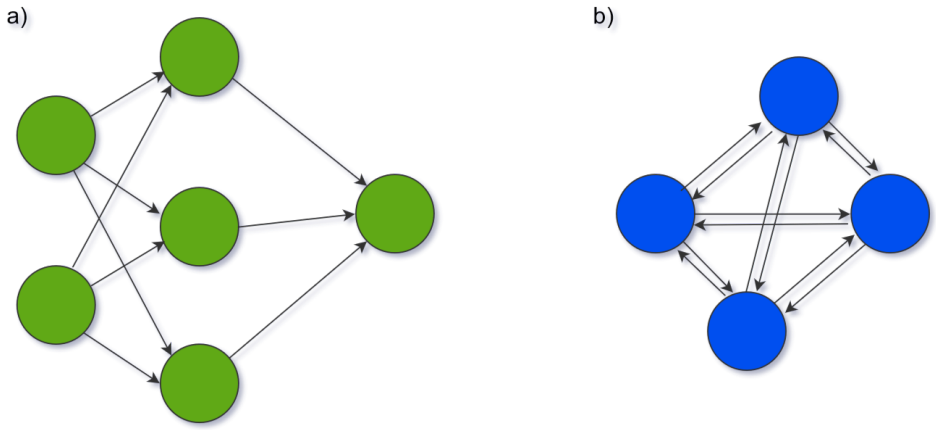

Figure 2.1: a) A feedforward network. b) a feedback network

since it is characterized by interconnections within the layer (then differently from *feedforward networks*, see Fig. 2.1). These units are defined as *binary threshold units*, taking outuput values $s_i = \{+1, -1\}$, with *i=1,..,N* the unit's label, which is determined by whether or not the units' input exceeds a certain threshold $\theta_i \in \mathbb{R}$.
RNN are generally arduous to analyze, since they may, depending on the couplings distribution, have stable states, but even oscillatory regimes [23] and Chaotic behaviors [20]. In the case of Hopfield net, connections are described by *symmetric* coupling constants, $w_{ij} = w_{ji} \in \mathbb{R}$, where *(i,j)* is any couple of connected nodes. This will ensure absence of oscillations and chaos, allowing to define a global energy function and to treat analitically the problem.
Indeed, to each state we can associate an energy, determined by the Hamiltonian:

$$\mathcal{H}(\boldsymbol{s}|\boldsymbol{h}, \{w_{ij}, i, j \in\}) = -\frac{1}{2}\sum_{i,j} w_{ij}s_i s_j + \sum_i \theta_i s_i \tag{2.1}$$

where $w_{ij} = 0$ if there is no connection. The system will evolve minimizing the above-mentioned energy function, as it usually happens in physics. The system will evolve in the energy landscape toward an energy minimum, usually called *Attractor*, where it will remain, and that will constitute a **memory**. Actually, the main application of Hopfield nets is as *content-addressable memory*: differently from usual hardware memories, for which an *address* is required to extract memory content, in such networks information is retrieved through its 'attributes', rather properties. This is a very simplified way to model biological memory, but simple to understand: given the name of a person, you can recall its face. For this reason they are also called *associative nets*[12].
In order to contain such memories, our network has to be trained, and many are

the 'algorithms' that can be applied in order to implement this procedure. In the case of Hopfield network, learning results to be **unsupervised**.
A famous paradigm of biological learning is the **hebbian rule**: the strength of the coupling between two neurons will be proportional to the entity of their simultaneous coactivation.

Hopfield model can be extended adding **hidden** units (see Fig. 2.7): visible units states (*green*) are connected to a group of units (*blue*) whose output will represent an *interpretation* of the incoming sensory input and the system's total energy a sort of *badness* of the interpretation.
One of the main issues with Hopfield nets is the *searching problem*: though the system will evolve through until convergence to a minimum, it may be possible that it gets trapped on a sub-optimal one. This can be even worse in presence of *spurious minima*.
It would then become then necessary to introduce noise: in this way, the probability to escape from any attractor will then be finite, allowing the system to explore the whole configuration space.
A general procedure to find a minimum using noise is **Simulated Annealing** [10]: starting with a major amount of noise, that will allow to explore the energy landscape, you gradually reduce it in order to ends up in a deep minimum (see Fig. 2.3).

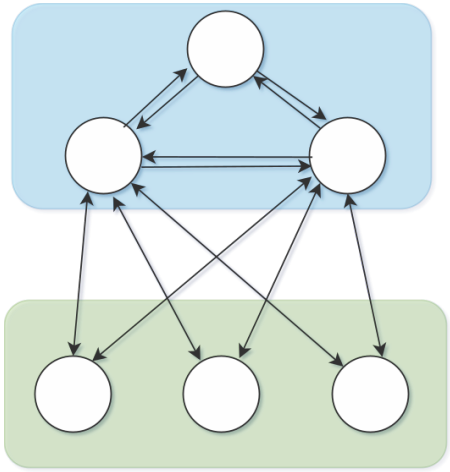

Figure 2.2: Example of Hopfield network with hidden units

There are different ways in which noise can be introduced in Hopfield net: noise can be directly inserted defining a *Temperature* that, if increased, will flatten the energy function and increase probability to change state, or intrinsically added to units; the last case is equivalent to the Boltzmann Machine model.

2.2 Boltzmann Machines and Restricted Boltmann Machines

A Boltzmann Machine (BM) is an *undirected graphical model*, a network of stochastic units associated to a particular energy function and to the related Boltzmann probability distribution, that can be used to fit some observed data. Units are of

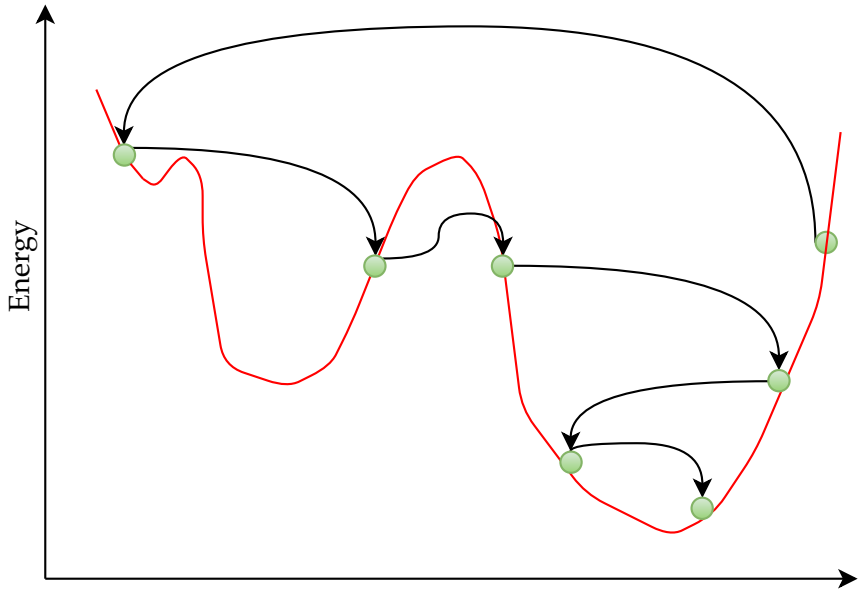

Figure 2.3: Pictorial description of the system dynamics using Simulated Annealing

two different nature: **visible units** are the ones standing for the observed variables that you want to describe; **hidden units**, instead, represent some latent variables catching features from the visible ones.
In the usual framework, the observed data are applied to such visible units and the learning procedure will modify the model parameters in order to reproduce the underlying probability distribution. Defining with $\boldsymbol{v} = (v_1, v_2, ..., v_N)$ the vector of visible variables and with $\boldsymbol{h} = (h_1, h_2, ..., h_M)$ the hidden one, and using Bernoulli units for both ($v_i, h_j = \{0, 1\}$), the BM probability distribution will be:

$$\mathcal{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z}, \qquad Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$$
$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{N} g_i v_i - \sum_{j=1}^{M} g_j h_j - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} v_i h_j - \sum_{i<k\in V} w_{i,k} v_i v_k - \sum_{j<l\in H} w_{j,l} h_j h_l \tag{2.2}$$

where V represent the set of visible units, $N = |V|$, H stands for the set hidden units, $M = |H|$; g_i and g_j are the local fields of, resp., visible and hidden units, readable as mean frequencies or, looking at 2.1, pseudo-thresholds.
Such a general property can be, in fact, useful to inferring the underlying probability distribution describing data, as detecting unusual properties in complex

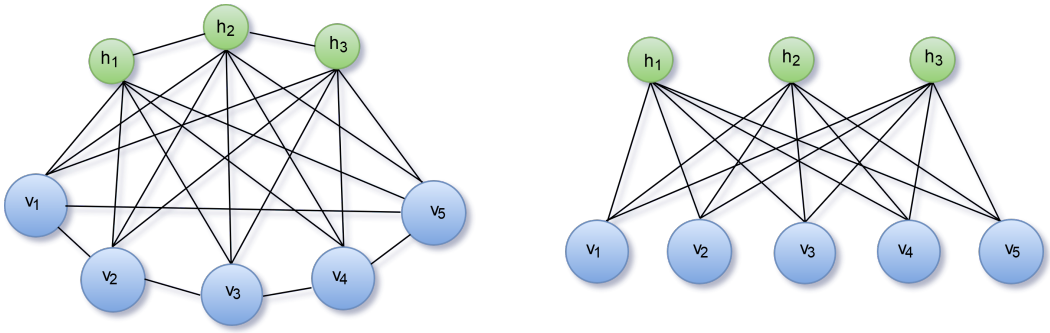

Figure 2.4: *Left)* example of BM: hidden units are in green, visible in blue. *Right)* example of RBM, same coloring

systems and to compute the Posterior.

In the last decade, the fitting of this model was largely studied by the statistical physics community in the last years, under the names of *inverse Ising problem* [16] or *Maximum Entropy modelling* [9], since BM is actually equivalent to the so called Ising model.
Differently from the usual framework in statistical physics, in which macroscopic observables are derived from microscopic laws governing the system, in the inverse problem our attempt is to extract, from observations of the behaving system, the microscopic laws.
Inverse problems are particularly useful in the study of large systems of interacting units, when their microscopic structure is well known: the functional differentiation between units does not often arise from the unit itself, rather from its interaction with the rest of the population. This is the case of Neuroscience: it is not the single neuron that has the ability to codify for a particular function, but the whole neural structure and its interactions to play such a role.

Thanks to its nature, BM can be easily used as a **generative model**: extracting randomly hidden variables, visible units will then follow the distribution

$$P(\boldsymbol{v}) = \sum_{\boldsymbol{h}} P(\boldsymbol{h}) P(\boldsymbol{v}|\boldsymbol{h}) \tag{2.3}$$

that will resemble the empirical one, obviously if the model properly learned the real distribution.
The main problem of BM is the high correlation present between units: such property complicate its analysis, as it happens also for procedures like sampling or

learning. This is the reason why Restricted Boltzmann Machines are usually preferred: in this particular realization of BM, only couplings between units of different nature are present, defining a *bipartite graph* (Fig. 2.4). This allow to simplify the correlation description, as far as the learning procedure. Furthermore, with such a graph thermal equilibrium can be reached in a single step: whereas in BM, imposing from data the visible vector, or through sampling the hidden one, we need to let the system evolve until reaching its stable distribution, with RBM configuration of the opposite layer (visible or hidden) will take directly a configuration following Boltzmann distribution in 2.2, since the update of all the units of a particular layer can be computed in parallel (consequence of the bipartition property).
We stress that, although visible units of RBM are not directly connected, they still contain correlations because of common inputs from the hidden layer, that act as a bridge. Hidden units will represent, as a matter of fact, collective modes of variation between visible units, present within the data.

2.3 Going ahead with RBM

In order to better understand the functioning, properties and potentialities of RBM, we will proceed now with a detailed explanation of the essential steps that are necessary to properly use it. Even though the Bernoulli case is a good model in many cases, in many situations it could be preferable to use units of different nature. The visible unit variables choice will be dictated by the data type on which you do inference, whereas for the hidden ones different potentials can be used, depending on what type of correlations you expect or which properties you want to study; as you can easily understand, there is no exact rule apart from intuition. The RBM probability distribution takes the general form:

$$\begin{aligned} \mathcal{P}(\boldsymbol{v},\boldsymbol{h}) &= \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z} \\ E(\boldsymbol{v},\boldsymbol{h}) &= -\sum_{i=1}^{N} g_i v_i + \sum_{j=1}^{M} \mathcal{U}_j(h_j) - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} v_i h_j \end{aligned} \tag{2.4}$$

where we used the same notation as in 2.1, and $\mathcal{U}_j(h_j)$ are unary potentials controlling marginal distributions of variables h_j.Some hidden potentials $\mathcal{U}_j(h_j)$ examples are:

- Bernoulli: $\mathcal{U}(x) = -gx, \quad x \in \{0,1\}$,

- Potts: $\mathcal{U}(x) = -g(x), \quad x \in \{1, ..., K\}$,
- Gaussian potential: $\mathcal{U}(x) = \frac{1}{2}\gamma x^2 + \theta x, \quad x \in \mathbb{R}$,
- ReLU potential: $\mathcal{U}(x) = \frac{1}{2}\gamma x^2 + \theta x, \quad x \in \mathbb{R}^+$,
- dReLU potential: $\mathcal{U}(x) = \frac{1}{2}\gamma^+ x^{+2} + \frac{1}{2}\gamma^- x^{-2} + \theta^+ x^+ + \theta^- x^-$

As you will see, it is useful to compute the probability distribution over visible units only, marginalizing then over hidden ones:

$$\mathcal{P}(\boldsymbol{v}) = \int \prod_{j=1}^{M} dh_j \mathcal{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} exp\Big(-\sum_{i=1}^{N} \mathcal{U}_i(v_i) + \sum_{j=1}^{M} \Gamma_j(I_j(\boldsymbol{v})) \Big) \equiv \frac{1}{Z} e^{-E_{eff}(\boldsymbol{v})} \quad (2.5)$$

where I_j is the total input of hidden unit h_j and Γ_j the cumulant generating function associated to $\mathcal{U}_j$:

$$I_j(\boldsymbol{v}) = \sum_i w_{ij} v_i, \qquad \Gamma_j(I) = log\Big[\int dh e^{-U_j(h) + hI} \Big] \quad (2.6)$$

2.3.1 Sampling

In the case of RBM, similarly to what happens with BM, the sampling cannot be done directly. In fact, a usual method is the **Gibbs sampling**, a Markov Chain Monte Carlo algorithm that extracts approximated samples from the probability distribution described by the model. Starting from an initial configuration, the algorithm updates, one at a time, each unit in a random order, drawing it from its conditional distribution $\mathcal{P}(x_k | \{x_j\}_{j \neq k})$. This procedure satisfies detailed balance, aperiodicity and irreducibility, i.e. there is non zero transition probability between any couple of states. All these requirements are fundamental to do the sampling correctly, since they assure convergence toward Boltzmann distribution for sufficiently large time.
Gibbs sampling is used both for BM and RBM, but with RBM the restricted connectivity allows to simplify the procedure, since the units of one layer are conditionally independent among themselves (see Fig. 2.5) :

1. Compute the hidden units input $I_j = \sum_i w_{ij} v_i$;
2. Sample each hidden unit in parallel, since independent

$$\mathcal{P}(h_j | I_j) \propto exp\big[-\mathcal{U}_j(h_j) + h_j I_j \big];$$

3. Compute the visible layer input: $I_i = \sum_j w_{ij} h_j$;

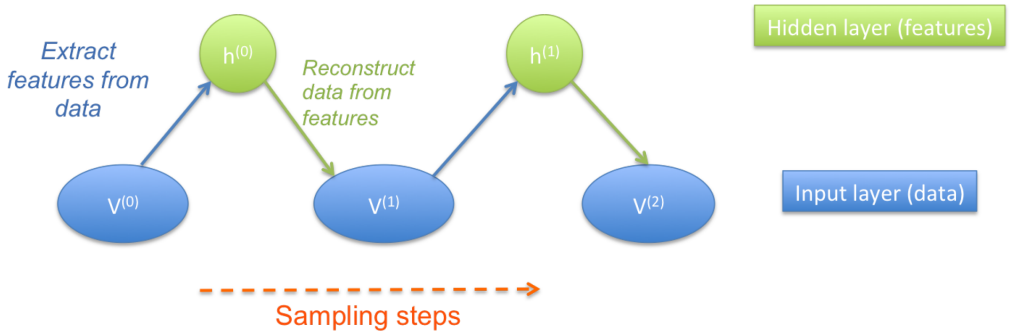

Figure 2.5: Sketch of the Gibbs sampling procedure on RBM. Taken from [27]

4. Sample each visible unit independently

$$\mathcal{P}(v_i|I_i) \propto exp\big[-(g_i + I_i)v_i\big].$$

Given a visible layer, we can define its most probable configuration as

$$\begin{aligned} h_j^*(\boldsymbol{v}) =& arg\,\max_{h_j}\Big[\mathcal{P}(h_j|\boldsymbol{v})\Big] = H_j(I_j(\boldsymbol{v})) \qquad H_i = (\mathcal{U}_j')^{-1} \\ =& \langle h_j|\boldsymbol{v}\rangle \equiv \frac{\partial \Gamma_j}{\partial I}(I_j(\boldsymbol{v})) \end{aligned} \tag{2.7}$$

since Γ is the cumulant generating function [28].
We anticipate that we will use mostly dReLU potential: ReLU is widely used in neuroscience as non-linearity, and dReLU is a more general form, proved to significantly outperform the other potentials reported above [14] in the context of image recognition . In the case of many hidden units, a cumbersome part will be the computation of the partition function Z : for this reason the usual way is to set a Markov Chain Monte Carlo simulation, thanks to the Markovianity of such a system. In fact, in RBM the present state is determined only by the previous one (Markov property). Such a property can be exploited using Monte Carlo sampling, where each accepted new configuration will represent a point in the evolution of the system, then defining the Markov chain. If properly used, such evolution can be interpreted as the dynamics of the system in the configuration space, ad used as an 'empirical' distribution from which you can extract a probability distribution.

2.3.2 Learning

Both in BM and RBM training consists in maximizing the log-likelihood on the training data-set $\mathcal{L} = \langle log\, P(\boldsymbol{v})\rangle_{data}$ using Gradient Ascent algorithm, updating

each parameter according to this rule:

$$\theta^{(t+1)} = \theta^{(t)} + \epsilon^t \cdot \nabla_\theta L \tag{2.8}$$

where ϵ^t is the *learning rate*, usual parameter in machine learning. The last term, with some simple calculations (see appendix A), takes the form:

$$\nabla_\theta L = -\langle \nabla_\theta E(\mathbf{v}, \theta) \rangle_d + \langle \nabla_\theta E(\mathbf{v}, \theta) \rangle_m \tag{2.9}$$

where the first expectation is computed on the training dataset distribution and the second one on the current probability distribution of the model. Informally, we can see the update as a way to diminish the energy on the real configurations, in accordance to memory storaging in Hopfield model, as to the usual hebbian rule, but raising the energy on the current model distribution (see Fig. 2.6), avoiding blowing up of the model and getting rid of spurious minima, as a sort of *unlearning*.

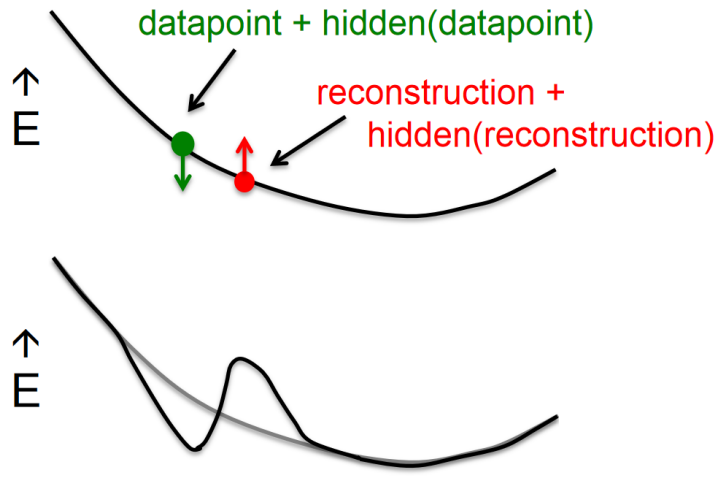

Figure 2.6: Schematic representation of a possible interpretation of the lerning procedure.

A standard learning algorithm in multiplayer neural networks is *back-propagation*, that consists of the repeated application of the following two passes: the *Forward step*, the network is activated on one example and the error of the output layer is computed; then, in the *Backward pass*, the network error is used for updating the weights.
Since in RBM the process of settling to thermal equilibrium corresponds to prop-

agating information about weights, we don't need back-propagation.
Considering the case of RBM, the likelihood gradients will be:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial g_i} &= \langle v_i \rangle_d - \langle v_i \rangle_m \\ \frac{\partial \mathcal{L}}{\partial \xi_j} &= \Big\langle \frac{\partial \Gamma_j(I_j(\boldsymbol{v}))}{\partial \xi_j} \Big\rangle_d - \Big\langle \frac{\partial \Gamma_j(I_j(\boldsymbol{v}))}{\partial \xi_j} \Big\rangle_m \\ \frac{\partial \mathcal{L}}{\partial w_{ij}} &= \langle v_i \langle h_j | \boldsymbol{v} \rangle \rangle_d - \langle v_i \langle h_j | \boldsymbol{v} \rangle \rangle_m\end{aligned} \tag{2.10}$$

where again $\langle h_j|\boldsymbol{v}\rangle = \frac{\partial \Gamma_j(I_j(\boldsymbol{v}))}{\partial I}$.
In the first term, gradients, computed with respect to fields and couplings, are non-linear functions that has to be computed at each parameters' update; in our algorithm we used Stochastic Gradient Ascent, deepen in 2.5.1. The difficult part is the right term $\langle\rangle_m$, because we need to estimate the Model probability distribution, and neither analytical nor direct sampling evaluation are possible; a standard way is to use Markov Chain Monte Carlo , constructing a Markov Chain that matches the correct one in the limit of infinite sampling time. It is usually slow and sometimes deceitful, as it can happen in case of bad-connected low energy regions; for this reason, different method were developed, depending on the problem. In our work we used *Augmented Parallel Tempering* (APT), that shares properties of Contrastive Divergence and Parallel Tempering, explained below.

2.3.3 Likelihood estimation

As seen, RBM learning consists in finding parameters changes that increase the loglikelihood $\mathcal{L}$ on the dataset, quantity that we need to estimate in some way. Since the partition function Z is intractable in both BM and RBM, the loglikelihood $\mathcal{L} = log\, \mathcal{P}(v)$ cannot be computed directly. For that reason Annealed Importance Sampling (AIS) algorithm can be used for estimating the partition function and, then, the likelihood. For more details see [15]. In the usual *Importance sampling* you to estimate partition function ratios.
Let $\mathcal{P}_1(\boldsymbol{x}) = \frac{\mathcal{P}_1^*(\boldsymbol{x})}{Z_1}$, $\mathcal{P}_0(\boldsymbol{x}) = \frac{\mathcal{P}_0^*(\boldsymbol{x})}{Z_0}$ two probability distributions. The next identity holds:

$$\Big\langle \frac{\mathcal{P}_1^*(\boldsymbol{x})}{\mathcal{P}_0^*(\boldsymbol{x})} \Big\rangle_{\boldsymbol{x}\sim\mathcal{P}_0} = \sum_{\boldsymbol{x}} \frac{\mathcal{P}_1^*(\boldsymbol{x})}{\mathcal{P}_0^*(\boldsymbol{x})} \frac{\mathcal{P}_0^*(\boldsymbol{x})}{Z_0} = \frac{1}{Z_0} \sum_{\boldsymbol{x}} \mathcal{P}_1^*(\boldsymbol{x}) = \frac{Z_1}{Z_0} \tag{2.11}$$

This result can, in fact, be used in the evaluation of the Z_1 partition function: choosing a well known, or at least simple to compute, model with partition function Z_0, it will be possible to estimate Z_1 by Monte Carlo. The major drawback

lies in a possible variance of the estimator: if P_1 and P_0 are very different from one another, some configurations are very likely for P_1 and very rare P_0; such configurations almost never appear in the Monte Carlo estimate of the average, but the probability ratio can be exponentially large.
In Annealed Importance Sampling, we address this problem by constructing a continuous path of interpolating probability distributions $\mathcal{P}_\beta = \mathcal{P}_1^\beta \mathcal{P}_0^{1-\beta}$, then we can estimate the wanted partition as

$$Z_1 = \frac{Z_1}{Z_{\beta_{l_{max}}}} \frac{Z_{\beta_{l_{max}}}}{Z_{\beta_{l_{max}-1}}} ... \frac{Z_{\beta_1}}{Z_0} \times Z_0 \tag{2.12}$$

Intuitively given two distributions, which might be disjoint in their support, we create intermediate distributions that are *bridging* from one to another. Then we do MCMC to move from one distribution to the next, ending up in our target distribution.
The best way to use such a method is to choose $\mathcal{P}_\beta$ as the closest independent model, in terms of D_{KL}, to the data distribution $\mathcal{P}_d$, using a linear set of β_l: starting from $\mathcal{P}_0$ you draw samples using Gibbs sampling and gradually pass toward $\mathcal{P}_{\beta_{l_{max}}}$.

2.4 Pseudo-likelihood

Before starting to describe how learning takes place it can be useful to define a new quantity, that will substitute the log-likelihood during the most of the computation.
Although likelihood is the main quantity we want to maximize during the training, its evaluation is often computationally demanding: it involves the computation of the Partition function, and it has to be computed each time a parameter is updated in order to be reliable.
RBM can also be trained using different objectives than maximum likelihood, such as minimum probability flow [22] or pseudo-likelihood maximization (PLM) [3]. In our work, we used the last during training, evaluating log-likelihood only after many steps, getting a reasonably good approximation.

In PLM, the learning criterion is to maximize the conditional probability of observing one variable given all the others:

$$P\mathcal{L} = \langle log\, \mathcal{P}(v_i | \{v_j, j \neq i\}) \rangle \tag{2.13}$$

Pseudo-likelihood maximization gives in principle less accurate predictions, but it is instead efficiently computable without further approximations, required instead for the likelihood estimation. Furthermore, in case of large datasets, maximizing the exact conditional likelihood would give the same result as maximizing full exact likelihood. Then pseudo-likelihood maximization is a *consistent estimator*, which is an important theoretical advantage of this approach.

2.5 Learning Methods

2.5.1 Stochastic Gradient Ascent

The usual Gradient Ascent algorithm is an important tool in finding functions maxima.
The main idea is that the gradient of a parameter has to be directed in the same direction in which the function derivative is positive. Defining such a gradient locally can gives problems in case of multiple maxima present: depending on the initial point, different result will be achieved, even if only one will be the true global maximum. This is the reason why full-batch gradient ascent is usually preferred, in which the chosen gradient will be an average of the one computed on the full dataset.
Going forward, a "mini-batch" way can be preferred: in this version, the mean gradient is computed only on a randomly chosen subset of the full dataset. The change can be useful in cases where many local maxima are present, since they can have a global weight larger than the global maxima one. This procedure is the so called Stochastic Gradient Ascent [1].

2.5.2 Contrastive Divergence (CD) and Persistence Contrastive Divergence (PCD)

CD is a MCMC method introduced by Hinton [4]: instead of random initialize the markov chain, we use data samples, from which evolving the MCs for few steps only.
Intuitively, the final, wanted result is that Gibbs sampling leaves as invariant as possible the initial configuration: instead of start a MCMC at random, waiting for thermalization, and then computing the gradient, we can do few Gibbs sampling (2.3.1) and consequently compute the gradient to reduce the tendency of the chain to run away from the correct distribution. Heuristically, the gradient quantifies a 'contrast' between the initial configuration and the model 'divergence'. To implement it, we will use as many chains as the number of batches, with few MC steps,

using the same data sample for SGA.

Since CD is biased, it will not have good *generalization* capabilities, having unexplored region possibly containing spurious minima, as seen for learning in the Hopfield model.

In PCD [26], MC are not initialized at each gradient update, but they are used through the whole evolution; this can be justified by the fact, assuming that samples are at equilibrium and probability varies slowly between updates, only few steps are required to the new equilibrium distribution (at least true in the case when the learning rate decays to zero).

Although these two methods are faster than normal MCMC, they are biased and they fail in presence of small mixing rates between rare configurations, besides possible divergence of the likelihood, partially avoidable decreasing the learning rate.
A widely used method to speed up the sampling is Parallel tempering; we will now introduce it, since it is the basis of a new method introduced in [27] and that we used in our simulations.

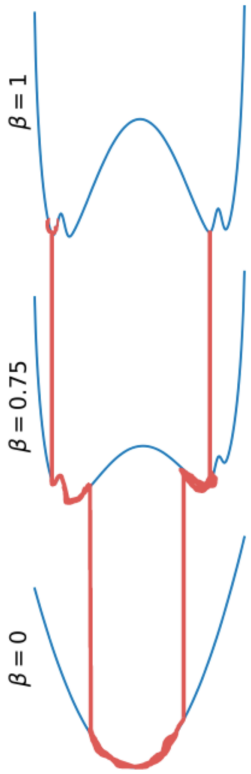

Figure 2.7: Illustration of the Parallel Tempering principle. Taken from [27]

2.5.3 Parallel Tempering

Also called Replica Exchange Method [11], PT simulates at the same time several replica of the system, drawn from the Boltzmann distribution at *inverse temperature* $\beta_i \in [0,1]$, $i \in 1,...,N_R$:

$$P_{\beta_i}(\mathbf{v},\mathbf{h}) = \frac{1}{Z_{\beta_i}} e^{-\beta_i E(\mathbf{v},\mathbf{h}) - (1-\beta_i)E_0(\mathbf{v},\mathbf{h})}$$

with β_i ordered from 0 to 1 and $E_0 = \sum_i \mathcal{U}_i^0(v_i) \sum_j \mathcal{U}_j^0(h_j)$ is the energy of a non-interacting system, related to P_0, that can be chosen as the independent model closest to the data, i.e. the one that minimizes the Kullback-Leibler Divergence

$D_{KL}(P_d|P_0)$. Denoting with $\mathbf{v}^{(t)} = (\boldsymbol{v}_1^{(t)}, ..., \boldsymbol{v}_i^{(t)}, ..., \boldsymbol{v}_{N_R}^{(t)})$ the current population of replica $\mathbf{v}_i^{(t)}$, the sampling occurs repeating the following steps:

1. compute a parallel *Gibbs update* of each replica $\mathbf{v}_i^{(t)}$ using the Metropolis-Hastings algorithm

2. Propose configuration swapping: choose a replica i and try to exchange it with one of its neighbors ($i \pm 1$, prob. a half for both)

3. Accept the move with probability

$$AR_i = min\left\{1, \frac{P_{\beta_i}(\mathbf{v}_{i\pm1})P_{\beta_{i\pm1}}(\mathbf{v}_i)}{P_{\beta_i}(\mathbf{v}_i)P_{\beta_{i\pm1}}(\mathbf{v}_{i\pm1})}\right\}$$

In this way, a particle trapped in a local energy minimum at $\beta = 1$ can, moving it to higher temperature, diffuse to a different state through an energy barrier lower than the initial one, effectively escaping from the initial mode. This method allows to speed up the transition time between modes, preserving the transition probability between states thanks to the acceptance rule, and ensuring unbiased sampling. In order to have reasonable acceptance, temperatures need to be chosen carefully, as in presence of phase transitions; furthermore, this sampling becomes wrong in case of entropic barriers: although at $\beta = 0$ the energy landscape becomes flat, modes containing a bigger number of equivalent configurations will prevail the evolution, possibly breaking ergodicity, with the Markov Chain unable to overcome this barrier (like in first order phase transitions, or in different cases with RBM, since this entropic unbalance is usual).

2.5.4 Augmented Parallel Tempering (APT)

This new method, introduced in [27], join features of CD, PCD and PT, but overcoming their major drawbacks.
Even though PT can be used to explore unknown energy regions, what we need is to evaluate the unknown probability distribution of our parametrized model near the known data, since it is a generative model. This information can be used as in the case of Contrastive Divergence, but through an unbiased model. Bringing back into play the idea of Parallel Tempering, we can use as P_0 a *Mixture of Independent model* (MoI) fo mode z

$$\mathcal{P}_0(\boldsymbol{v}) = \sum_{z=1}^{Z} \prod_i \mathcal{P}_0(v_i|z)\mathcal{P}_0(z) \tag{2.14}$$

able to learn multimodal distributions: such improvement allow direct sampling during the MCMC, ensuring ergodicity and multimodality.
Using MoI instead of independent model, it is possible to heavily speed up the trajectory trhough the configuration space, as to use fewer replica thanks to the lesser distance between P_0 and P_1. In this way you will also sort out the inability of usual MCMC-based learning to fit multimodal and bad connected datasets.
A general problem with RBM training is to find the minimal number of Monte Carlo steps required to correctly learn the model, a quantity determined by the specific dataset. Using APT, we can associate to each data point its mode $z = arg\,max\,\mathcal{P}_0(z|\boldsymbol{v})$ and computing the transition time to a new mode, defining then a transition matrix: the effective convergence time, related to the Markov Chain mixing rate, can be defined as $\tau = -\frac{1}{log\,\lambda_2}$ where λ_2 is the second higher eigenvalue: a large τ corresponds to a transition matrix with small off-diagonal terms, then slow in changing the mode, while a small τ is related to well mixing systems. This parameter can then be used to define such length, method that will define the *Adaptive APT*.
J. Tubiana shown that *Adaptive APT* outperform the previous ones: lower-Temperature replica are much closer to the target distribution, resulting in higher swap rates.
(write appendix on implementation and results)

Chapter 3

Restricted Boltzmann machine applied on cortical activity

We presented in the last chapter the Restricted Boltzmann Machine model, the model we used in our analysis. As we reported, such a model can be used to fit any type of dataset from which, if properly trained, it extracts the probability distribution that better describe the empirical one.
The aim of our project was to define the best procedure enlightening, using RBM, new functional neural structures arising in the medial Prefrontal Cortex of rat after task-learning, with a wide quantity of possible applications in many other situations.
First, after introducing the concept of *cell assembly* in the field of neuroscience, we summarize the recent methods developed in order to extract these neural ensembles from the recorder activity of the Prefrontal Cortex. Then, we will present the main results of our study.

3.1 The cell assembly and its description

One of the main peculiarity of mammals, as well as the majority of living beings, is the capability to learn from experience: it is probably the main resource of the individual, and one of the principal features determining the strength and success of a species in nature.
Tough such ability was, and presently is, largely studied, huge questions remain open yet. One of these is how, effectively, the learning actually take place.
Nevertheless, given the solid belief that the brain works as a distributed system of interacting units, it seems quite reasonable that such mechanism would consist in small changes in the neural connectivity, probably associated to local modifica-

tions in the neuron itself.
With the term **cell assembly**, neuroscientists usually identifies putative subgroups of cells coding for particular functions, as memories or learned abilities: such neurons interact, in an precise way, showing particular temporal patterns, that could be associated to such encoded information.

3.2 Rule-learning

A well established model concerning spatio-temporal experience-related learning is the Standard Model of Consolidation, for which, roughly saying, information is stored as short-term memory in the *Hippocampus* and after, mainly during sleep, such information flow towards prefrontal cortex. Such phenomenon takes place as strong coactivation of the two structures. In their work, Peyrache et al. [17] analyzed reactivation strength of neural activity during *Slow-Wave Sleep* (SWS) period, preceding (*Pre*) and following (*Post*) rule-learning experiences of rats: in particular, each rat was asked to perform a task in a Y-maze in which it had to learn how to select the rewarded arm using one of four possible rules: left arm, right arm, illuminated arm and nonilluminated arm; during each trial, one target arm was illuminated at random; such period is referred as the *Awake* epoch.
The data analysis was mainly based on Principal Component Analysis (PCA): it is an orthogonal linear transformation that transforms the data to a new coordinate system. Indeed, it identifies directions with larger variance, then containing much information about the points distribution. They found that the highest eigenvalues of the *Awake* correlation matrix were associated to rat's strategies, and that the first principal component's higher elements, selected using the theoretical upper bound for random spike trains, correspond to the notorious *cell assembly* (Fig. 3.1, *Top*). Furthermore, through the definition of a *Reactivation strength* function, estimate of the similarity between awake and sleep activity, it was also observed a clear emergence of the cell assembly, from Sleep Pre to Sleep Post. Reactivation peaks occurred together with SPWR hippocampal waves, via large synchronous oscillations of the two cortical structures, according to the Standard model of Consolidation. Finally, as it is possible to see in Fig. 3.2 (*Bottom*), the probability distribution of such reactiovation resulted to change across the two rest periods, with an increased tail in the second one.

The dataset provided by Peyrache's group was further studied; in particular, some interesting outcomes were obtained through statistical inference, which will justify and enrich our work.

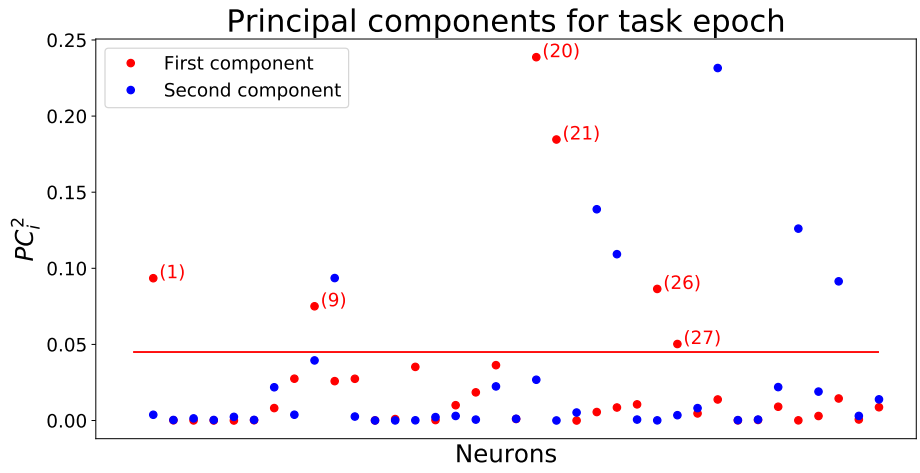

Figure 3.1: First two principal components of the correlation matrix related to the Awake epoch; the red line represent the upper bound for random spikes train.

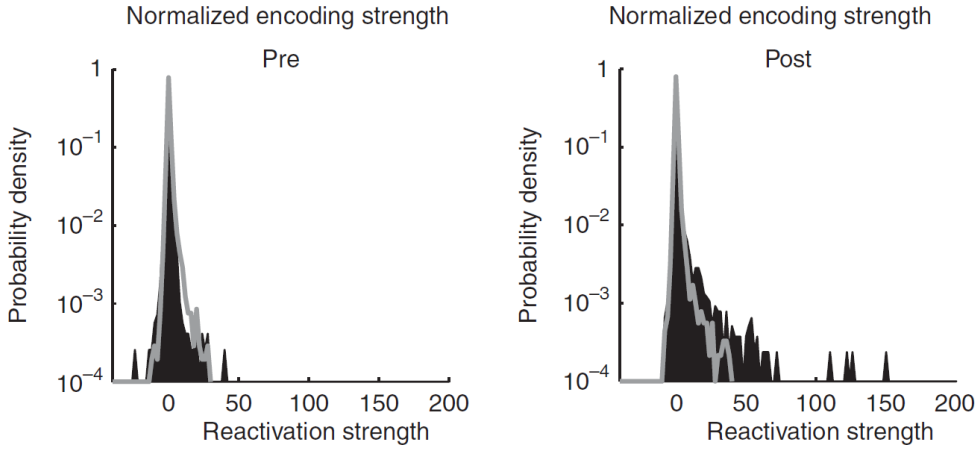

Figure 3.2: Incidence of reactivation strengths during rest periods. Black filled zones indicate SWS periods and the gray trace indicates non-SWS. The post epoch SWS histogram has a heavy tail, reflecting strong transient reactivation events. Taken from [17]

3.3 Study of neural activity through inverse Ising problem

To better characterize the knowledge on such cell assembly and its replay, Tavoni et al. [25][24] inferred functional interaction networks reproducing low-order statistics of 'snapshots' of the neural population studied. Among the two articles they extracted, using maximum-entropy, a BM model associated to the real system, whose parameters reproduce recorded frequencies and pairwise cross-correlations of neurons.

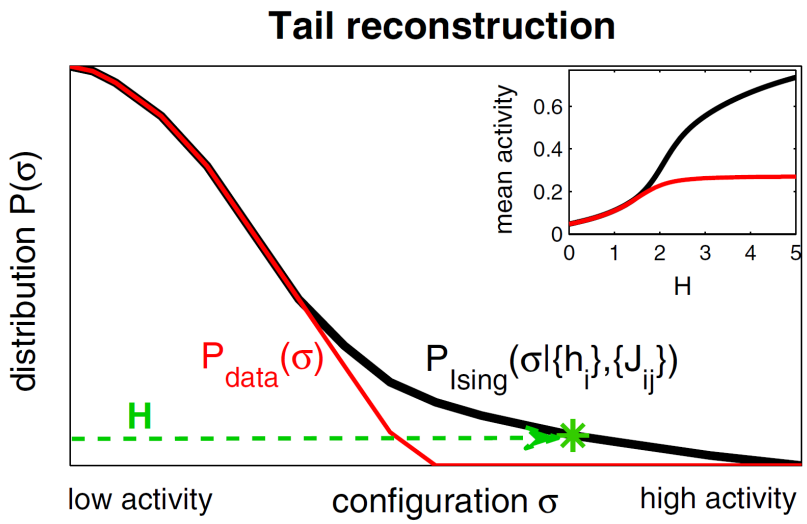

Figure 3.3: In contradistinction with the data distribution (*red*), the Ising distribution (*black*) reconstructs accurately the tail of the distribution of all activity patterns potentially generated in such a network. Inset: mean global activity as a function of the external drive H, computed from MCMC of Ising probability distribution. Taken from [24]

In [25] they mainly analyzed structural changes in the networks between Sleep epochs, then looking at correlation of these with the Awake one. Extracting the probability distribution and such related graphical model allowed to refine the analysis, with multiple neuron correlation, exact definition of the modifications of the inferred functional couplings as the reactivation of the enhanced group. In the second one [24], the same model was instead used to the identification of coactivating neurons (cell assemblies) in single epochs: once the model was inferred, a *neural susceptibility* to an external drive was defined; using it, the cell assembly was defined as the group of cells with the highest susceptibility peak at the drive value. Besides the other outcomes, the important thing, at least in our point of view, is the power of such inferring procedure: as schematically explained in Fig. 3.3, the inferred model reconstructs the tail of the distribution of all observable configurations; rare and high-activity configurations, resulted to include also the cell assembly activation, that emerges from structural properties of the functional network obtained.

3.4 The machinery

In this project we try to define a new RBM model in order to study the mPFC data of [17]. We start describing the physical model's details; after, we will define the final training procedure used, followed by the parametrization.

3.4.1 Graphical model

The RBM used is made by 37 Bernoulli visible units, each one associated to a particular recorded neuron. These are connected with a variable number of hidden units (from 1 to 60). Of course, many more units can be used, possibly improving the final likelihood. Nevertheless, what we want to define is a **simple** and **interpretable** model, with each hidden unit and its related couplings representing particular interactions between neurons, easily linkable to biological phenomena. Using too many units then, even though it could better reproduce spiking activity, would not be useful for the purpose we addressed. The best results, in fact, were obtained not exceeding 10 units.
A general feature that we forced along learning was sparsity: this feature allows to get, for each hidden unit, only few strongly coactivating visible units, and also to differentiate them in subgroups of higher interacting cells. Sparsity can be obtained in many ways, using Regularization terms.
Concerning the nature of hidden units, as anticipated we used dReLU potentials:

$$\mathcal{U}(x) = \frac{1}{2}\gamma^{+}x^{+2} + \frac{1}{2}\gamma^{-}x^{-2} + \theta^{+}x^{+} + \theta^{-}x^{-}$$

In order to understand this choice, we compare it with the other two most used potentials, i.e. Bernoulli and quadratic (Gaussian) (see 2.3): with respect to Bernoulli, dReLU takes continuous values, allowing to encode much information than in the other case. Also, after marginalization, quadratic potentials create pairwise effective interactions whereas Bernoulli and dReLU does not. It was shown in the context of image processing [14] and text mining that non-pairwise models are more efficient in practice, and theoretical arguments also highlight the importance of high-order interactions.

3.4.2 Learning Methods

Following section 2.3.2, the learning procedure consists in updating the model parameters using Gradient Ascent rule in order to maximize the likelihood 2.8. We used Stochastic Gradient Ascent (2.5.1), in which data are divided in *mini-batches* of $300 \sim 500$ data-vectors.

As usual in Machine Learning, we randomized the dataset ordering, dividing it in training and test set, in order to check *generalization* ability of our net: such procedure will confirm the effective learning of the probability distribution describing the data, representing also a way to detect possible overfitting.
For the sake of performance, gradients were instead computed using *Pseudo-likelihood* (see section 2.4): such quantity is simpler to be evaluated and, although it is less stable and accurate than the likelihood estimation, it turned out to be a valid quantity to use, supported by occasional likelihood evaluation, both on train and test set, making use of Annealed Importance Sampling (2.3.3).

The learning task was accomplished by means of Adaptive Augmented Parallel Tempering (2.5.4), using two replica of the system and a single Markov step in the Monte Carlo evolution, in accordance with the usual PCD procedure (2.5.2). The mixture model was fitted on the data at the beginning, and kept fixed along the whole evolution. Finally, the so called *batch normalization* procedure [8] was used. During learning, each time a parameter changes, a coviarate shift phenomenon takes place [19], consisting in a spurious change in the mean activity of the hidden unit, requiring a lowering in the learning rate and a consequent slow down of the training. The idea is to reparametrize the network, such that all intermediate activities have zero mean and unit variance. Such procedure was tested in many cases [27], resulting in much better performances.

3.4.3 Parametrization

RBM can obtain good performances only with careful choices of parameters.

Particular care has to be taken for the system initialization. We initialized visible fields from the independent model, i.e. imposing them equal to the corresponding mean activity of the correspondent cell, corresponding to Maximum entropy inference [9]; weights are extracted from $\mathcal{N}(0, \sigma^2 = \frac{0.01}{N})$, usual choice with such model, that avoid to have initial hidden units with too high values, fact that can slow or get worse the generalization performance.

An important parameter defining training is the **learning rate**, a coefficient that determine the magnitude of the effective gradient applied on the single parameter (ϵ^t in 2.8). As reported in [5], it has to be chosen in such a way that the total change in weights does not exceed the empirical rule:

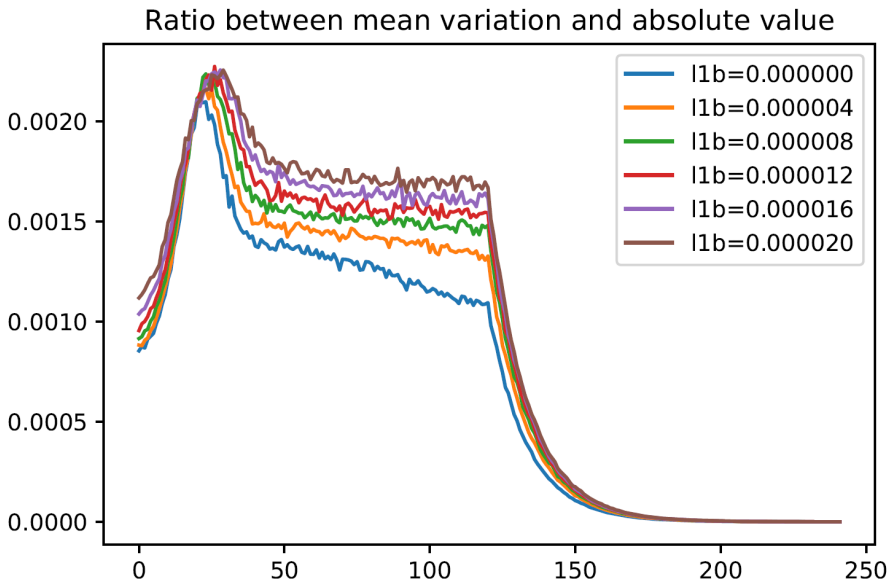

Figure 3.4: Evolution of the ratio between the mean weights variation and mean weights amplitude, plotted for different simulations varying the regularization term. It is clearly visible the effect of the learning rate's exponential decay on the weights variation.

$$\Delta w_{tot} \sim 10^{-3} w_{tot}; \quad \Delta w_{tot} = \sum_{i,j} |\Delta W_{ij}|, \quad w_{tot} = \sum_{i,j} W_{ij} \tag{3.1}$$

Furthermore, it is usually suggested to use a dynamic parameter, following an *Annealing* procedure, according to the idea of Simulated Annealing [10]. For these reasons, we used an initial learning rate $lr = 0.01$, with exponential decay on the second half of the training, till a final value of 10^{-6}. In Fig. 3.4 an example of the dynamics followed by the ratio of the two quantities in 3.1, where it is possible to notice the correct change and the effect of the Annealing.

A decisive parameter is the **regularization**.
In machine learning, training is carried out by minimization of a *loss function*, to which some additional terms can be added: they will add supplementary information, routing the model on a particular direction that minimize such a term:

$$\min_f \sum_{i=1}^{n} V(f(x_i), y_i) + \lambda R(f)$$

where λ is a parameter controlling the strength of the regularization function R(f).

This new term can be used to impose particular properties on couplings, giving preference to low weights instead of large one, for example.
The main reason to include it is that it is a powerful way to avoid overfitting, i.e. when the high number of fitting parameters, despite corresponding closely or exactly to a particular set of data, brings the system to fail to fit additional data or predict future observations reliably. If the regularization works properly in such a direction, the likelihood, seen as a function of this parameter, will show first an increase, after which it will start to go down again, meaning that the model will start to overfit data.
In our system, we used the so called L_1^2 *penality*, since we focus on **sparse weights**: we addressed to find new representations that are interpretable, and sparse weights are then crucial. Furthermore, such property will not arise naturally in RBM learning, then such regularization penality is required. L_1^2 is defined as:

$$L_1^2 \propto \frac{1}{2N} \sum_{\mu} (\sum_{i,v} |w_{i\mu}(v)|)^2 \tag{3.2}$$

Notice that in our case we set up the training on likelihood instead of loss, trying to *maximize* it: this results in a change of sign in the final gradient computation. In contrast with our expectations, the described likelihood shape, presenting a well defined maximum, was not found: as you can see in Fig. 3.5, before the fast decrease caused by overfitting, only a plateau is present.

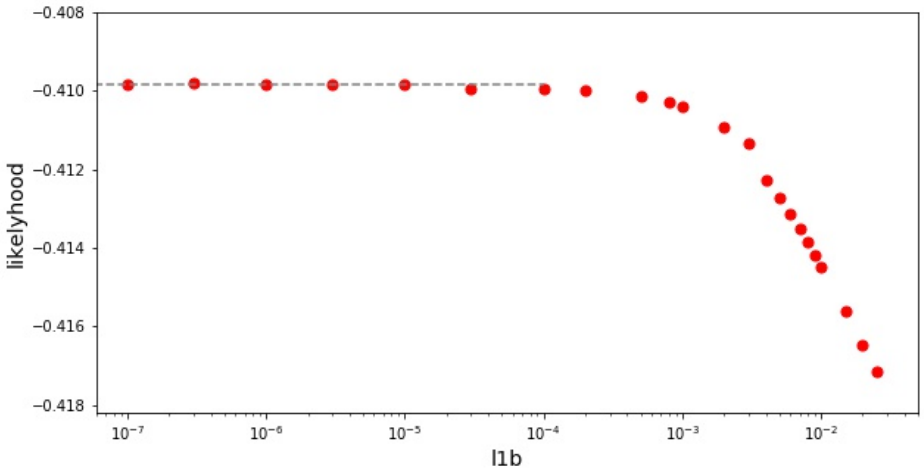

Figure 3.5: Final evaluation of the likelihood as a function of l_{1b}, computed on test set. Simulations were done with 60 hidden units, in order to confirm that the model has enough parameters. l1b$\equiv L_1^2$)

In order to validate our set up, we used a second *generation* of RBM: using the

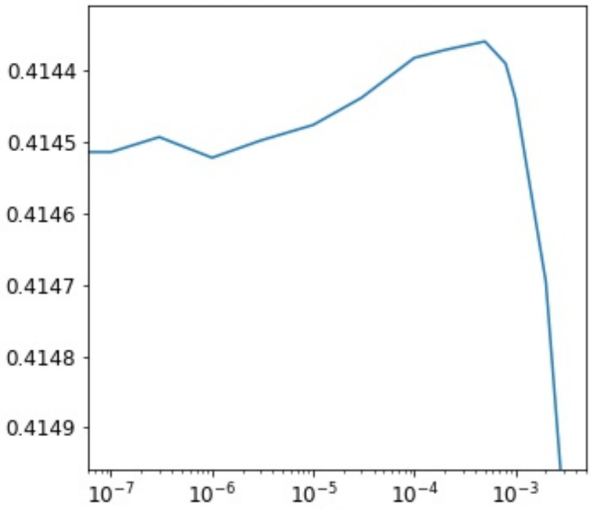

Figure 3.6: Same function of Fig. 3.5, but computed on a second generation of RBM, trained on *artificial* data generated by the first generation of RBM, i.e trained on real data.

fact that RBM are generative models (see 2.2), we trained a first RBM on real data, and generated a new dataset used as training set on which a second RBM was used. In this second case the expected behavior was found, as you can see in Fig. 3.6. We will face the reasons of these two cases in the next section.

A final parameter involved in our analysis, even if not directly implemented in the RBM, is the data binning: the Peyrache's recordings are time series of cell-events, independently recorded one each other. Since, for their structure, RBM are not able to correlate activation over time, then learning temporal patterns, the only way to find correlations in arbitrary large time window is to apply a binning procedure. The simple way, that is the one we used, is to simply look at fixed time windows, and consider the neurons spiking inside them as activated for the whole period.

The choice of the binning size was take according to some outcomes of the articles presented above, then equal to 100ms, i.e. a bit larger than the mean time of coactivation of the cell assembly.

Although useful, such a procedure will negatively affect the final result. In fact, the cut applied on data could split in two these patterns, and has also the drawback of losing information about how much each active cell effectively spikes.

3.4.4 Evolution

Before presenting the results, we will show an example of model's parameters evolution during the training.
As explained, learning is accomplished by likelihood maximization: as you can see in Fig. 3.7, the behavior is the one expected, with a first increase, followed by a stabilization. The learning rate starts to decrease in the first half of the procedure (1100 over 2200 epochs), same point in which strong variations precede a final increase: it seems to be exactly the effect of the annealing of the parameter, as pictorially showed in 2.3, interpreting the parameter as a noise on the dynamics and the raising as the final arrival to a better minimum. The picture report the log-likelihood of the model, divided by the number of visible units, then corresponding to the single neuron. Other two quantities monitored during the evolution were weights *sparsity* S and the average sum of the weights squared N_W associated with each hidden unit

$$\begin{aligned} S &= \frac{1}{NM}\sum_{\mu}^{M}\frac{(\sum_i^N w_{\mu i}^3)^2}{(\sum_i^N w_{\mu i}^6)} \\ N_W &= \frac{1}{M}\sum_{i\mu} w_{i\mu}^2 \end{aligned} \tag{3.3}$$

with N visible units and M hidden ones. The observations 3.8 and 3.9 graph confirm the effect of the L_1^2 regularization of decreasing sparsity, also figuring again the effect of the learning rate annealing. The reported curves are related to

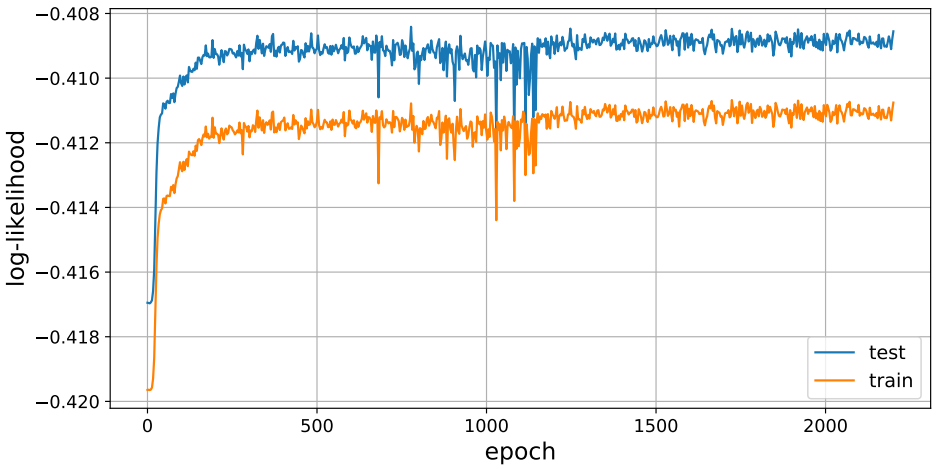

Figure 3.7: Evolution of log-likelihood of the RBM, computed on training and test set, during learning.

simulations with the same likelihood final distribution as in Fig. 3.5, using some selected points: the association allows to understand that probably the decreasing is not due to overfitting, but rather to an excessive effect of the regularization, that blocks the growth of the weights.

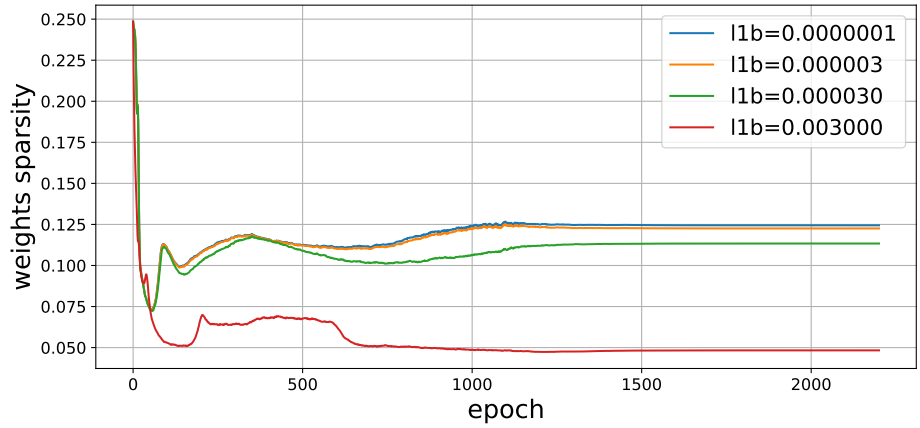

Figure 3.8: Evolution of the RBM weights' sparsity for different regularization parameters. l1b$\equiv L_1^2$

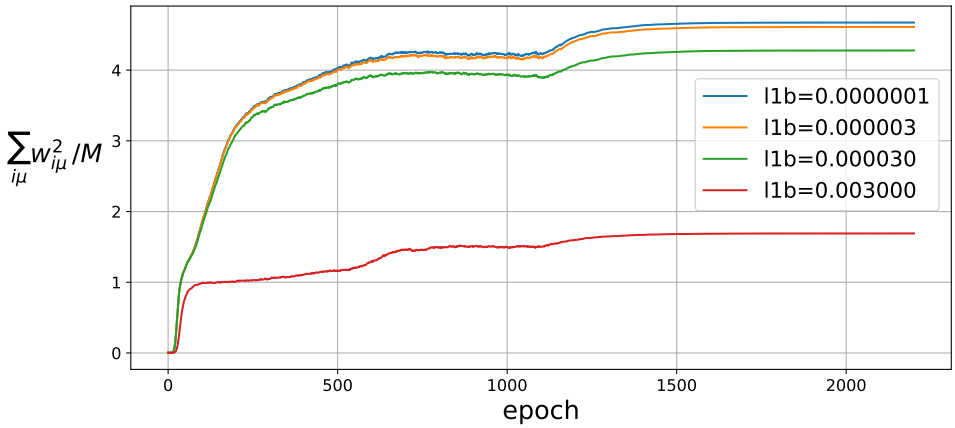

Figure 3.9: Evolution of the RBM weights' norm sum, defined in 3.3, for different regularization parameters. l1b$\equiv L_1^2$

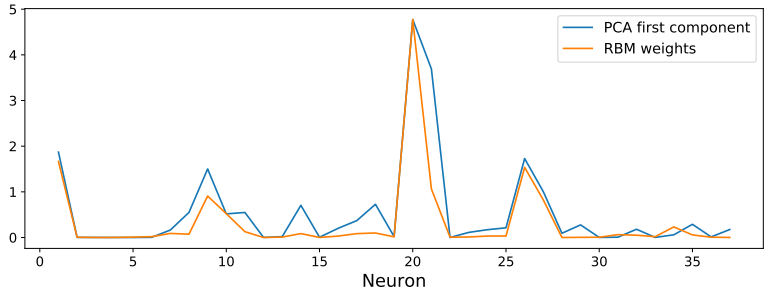

Figure 3.10: Matching between first principal component of PCA and weights of the RBM-1. Both vectors are squared.

3.5 Results

Once the network training was correctly done, we looked for methods to evaluate its efficacy, mainly concerning reactivation.
A first, simple way to compare our procedure with usual PCA is to define an RBM, trained on the Awake epoch dataset, including only one hidden unit (called RBM-1 from now on). As you can see in Fig. 3.10, higher values corresponds in case of neurons belonging to the cell assembly. In the labeling we used, cell assembly will correspond to cells 1, 9, 20, 21, 26 and 27
A second check was to look at the generative properties of our RBM. This can be easily accomplished comparing frequencies and correlation matrices of the original and the artificial dataset. As you can see in Fig. 3.11, the two frequencies distribution are almost identical, as happens for correlations.

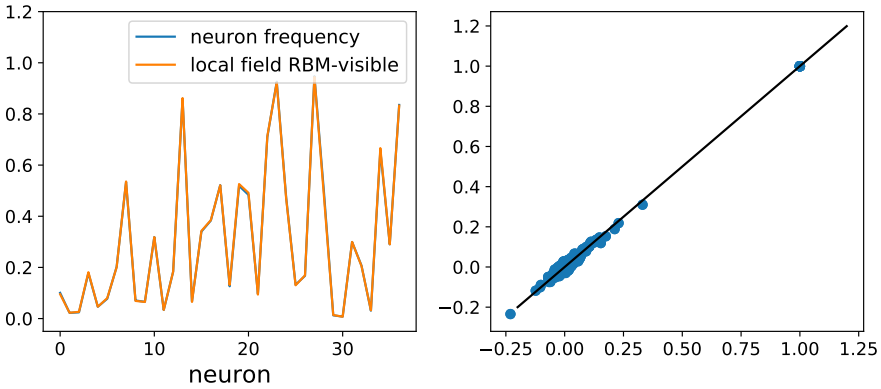

Figure 3.11: *Left*) Comparison between neural frequencies on real and artificial data. *Right*) Scatter plot of correlation matrices: x and y coordinates correspond respectively to real and artificial data correlation matrix elements.

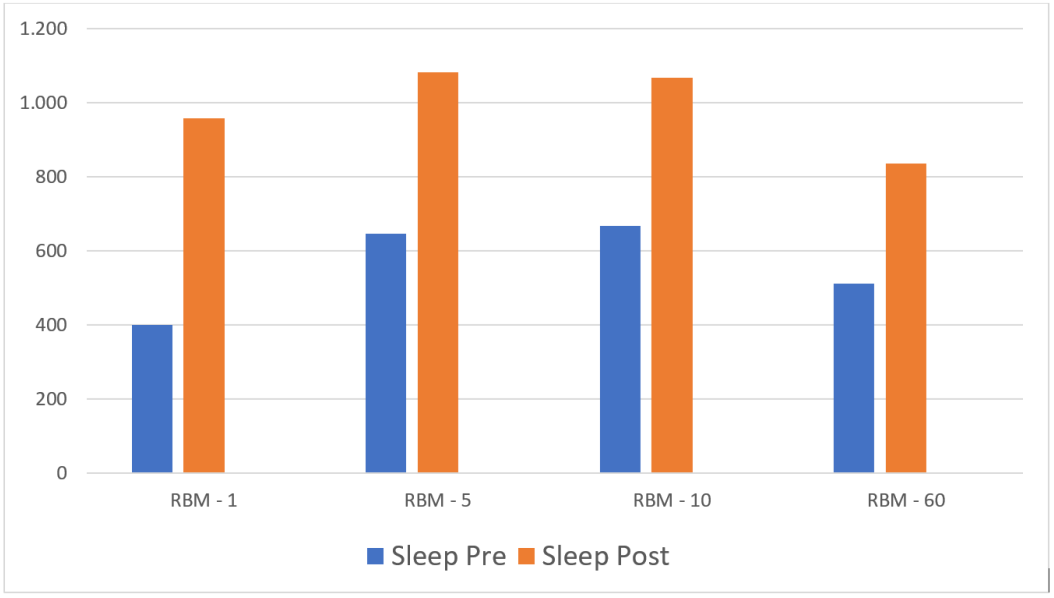

Figure 3.12: Histograms of the Reactivation scores, relative to RBM with 1,5,10 and 60 hidden units

Concerning the replay, a simple way to verify the RBM performances is to define a *Replay score RS* in the following way:

$$\begin{aligned} RS_{Pre} &= \frac{1}{2L_{Pre}}(\boldsymbol{Q_{pre}W})^2 - (\boldsymbol{Q_{pre}})^2(\boldsymbol{W})^2 \\ RS_{Post} &= \frac{1}{2L_{Post}}(\boldsymbol{Q_{post}W})^2 - (\boldsymbol{Q_{post}})^2(\boldsymbol{W})^2 \end{aligned} \tag{3.4}$$

with L the length of the dataset, and Q_i the renormalized datasets of sleep Pre and Post

$$(Q_{Pre})_{i,l} = \frac{D_{i,l} - m_i}{\sigma_i} \qquad i = 1, ..., N \tag{3.5}$$

m_i and σ_i are mean and variance of each neuron computed on the particular dataset. Results are shown in Fig. 3.12: for each hidden units the reply score works well, even if, paradoxically, the best result was obtained with RBM - 1. Going further in this direction, we studied the probability distribution of reactivating sequences, trying to reproduce the Peyrache result in Fig. 3.2, then looking at the emergence of *tail events*. We defined our probability distributions on a new reactivation, simpler than the previous one, multiplying the Sleep Pre and Sleep Post datasets by the couplings of the model. The values will be described by the function

$$R_\mu(\boldsymbol{v}) = \sum_{i=1}^{N} w_{i\mu} * v_i \tag{3.6}$$

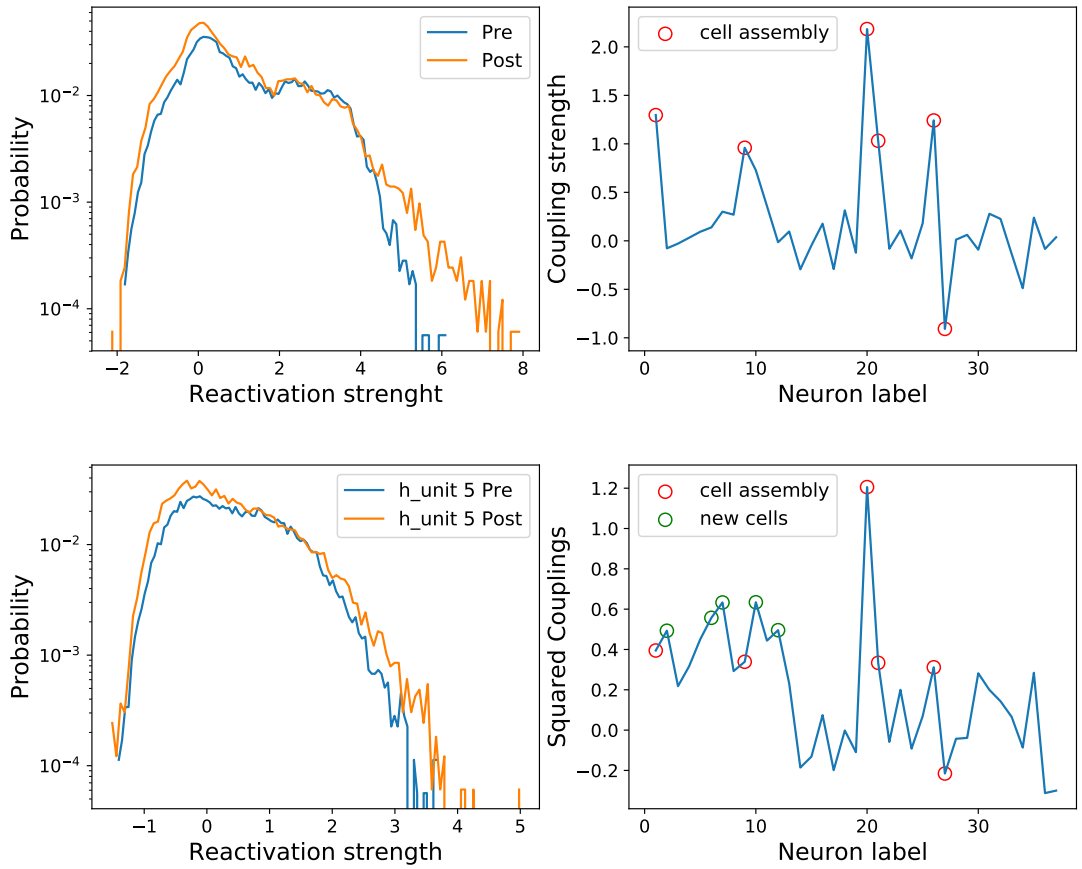

Figure 3.13: *Left*) Probability distribution of reactivation strength for RBM-1 (Top) and RBM-5 (Bottom). *Right*) Distribution of the RBM weights. Red (Top and Bottom) and green circles (Bottom) represents, resp., the affirmed cell assemblies and new cell that could be added.

The results are shown in Fig. 3.13. You can easily see that the reactivation strength probability distribution reveals, as expected, an increase of replay from Sleep Pre to Sleep Post, in accordance with literature.

We then extended the procedure to 5 and 10 hidden units (Fig. 3.14): although most of the distributions remained basically unchanged, some of these showed the wanted tail events. Such a result should mean that the system starts to distinguish between *preserved* and *new* associations of cells.

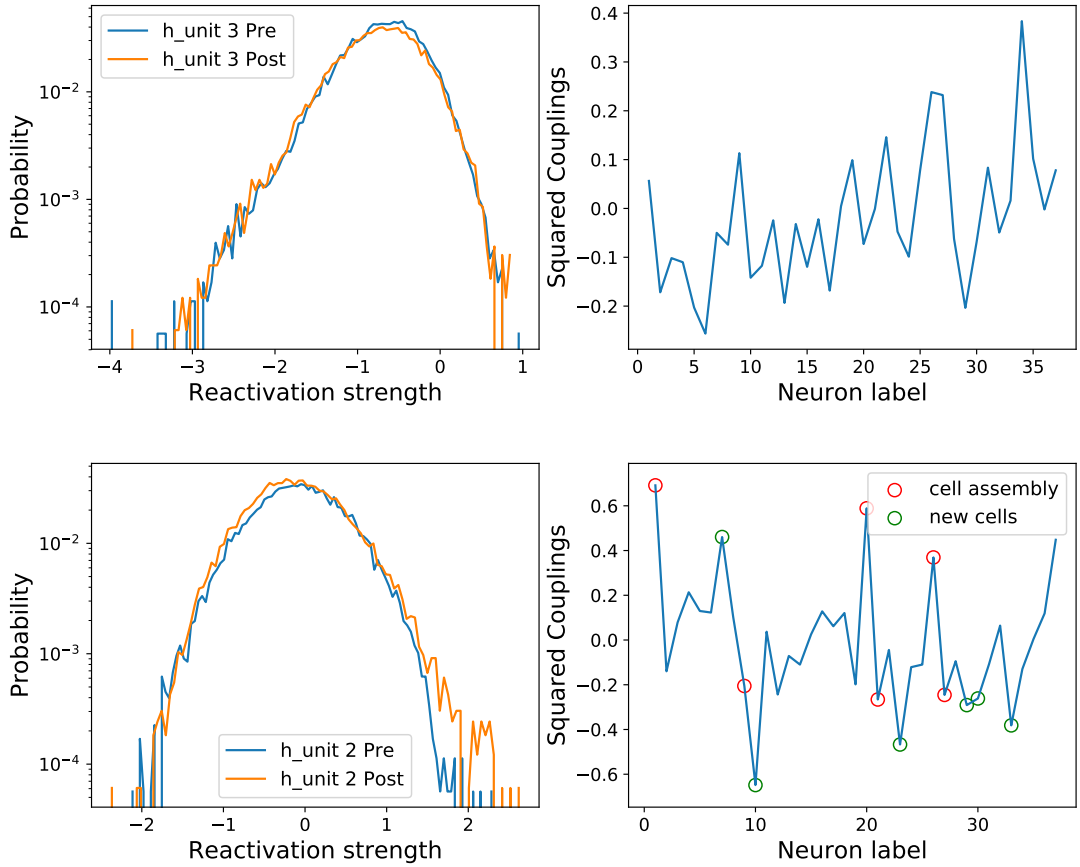

Figure 3.14: *Left*) Probability distribution of reactivation strength for RBM-10. *Right*) Distribution of the RBM-10 weights

These findings confirm the efficacy of the defined method and its possible application on larger data, also allowing to enlarge the cell assembly with new cells. Nevertheless, the absence of a maximum in the likelihood-L_1^2 plot, and then the inefficacy of the regularization on its improving, suggests the inefficacy of our machine: increasing the units allows to better represent interactions, but does not improve the likelihood, even with an high number of hidden units used. Furthermore, modeling the system with *independent neurons*, then inferring local fields only, resulted in a log-likelihood level around -0.48, close to the best case found using our procedure, that never exceeded -0.41.
In our opinion, the D_{KL} divergence minimization, equivalent to Likelihood maximization, might not be the good quantity to look at, since, as widely known, it is not a well-behaved quantity in case of high dimensions, in which the possibility to have no shared support between the two distribution that you are comparing can be finite. For this reason we started to redefine the learning procedure on Wasserstein distance minimization, that, instead, has important properties that could overcome the problem.

Chapter 4

Wasserstein

In the framework of Boltzmann Machines, and generally in machine learning, learning is usually seen as Maximal Likelihood Estimation, equivalent to minimization of the Kullback-Leibler Divergence. Even if it is a powerful tool in quantifying how much *close* two probability distributions are (that cannot be defined as *distance* because not symmetric), going to high dimensions, it can be problematic to handle with, in particular when the distribution that has to be achieved live in a lower dimensional manifolds. This will increase the possibility to get in some regions where the two distribution do not overlap. For this reason, different groups started to use **Wasserstein distance**, also known as *optimal transport distance*, a fundamental distance that, instead, is symmetric and smooth, then endowed by better generative properties.

4.1 Background

Consider two continuous probability distributions p and q in $\mathcal{P}(\mathcal{X})$, probability distribution space defined on a metric space $(\mathcal{X}; d)$. Consider π as a joint probability measure on $\mathcal{X}\times\mathcal{X}$ with marginals p and q, also called a *transport plan*:

$$\begin{aligned} \int dx'\, \pi(x, x') &= p(x) \quad \forall x \in A \subseteq \mathcal{X} \\ \int dx\, \pi(x, x') &= q(x') \quad \forall x \in B \subseteq \mathcal{X} \\ \pi(x, x') &\geq 0 \end{aligned} \tag{4.1}$$

The Wasserstein distance between p and q is defined as

$$\begin{aligned} W(p,q) &= \min_{\pi\in\Pi(p,q)} E_\pi d(x,x') \\ &= \min_{\pi\in\Pi(p,q)} \left\{ \int dx dx' \pi(x,x') d(x,x') \right\} \\ s.t. \quad & \int dx' \, \pi(x,x') = p(x), \quad \int dx \, \pi(x,x') = q(x') \end{aligned} \tag{4.2}$$

with $\Pi(p,q)$ the marginal polytope of p and q.
We define *optimal* the one which represent such minimum. To this quantity it can be associated a dual form (**Kantorovich duality**), that will be useful in the distance computation:

$$\begin{aligned} W(p,q) = \max_{\alpha,\beta} & \left\{ \int dx \, \alpha(x) p(x) + \int dx' \, \beta(x') q(x') \right\} \\ s.t. \quad & \alpha(x) + \beta(x') \leq d(x,x') \end{aligned} \tag{4.3}$$

Following the pictorial exegesis of Villani [29], we can interpret *p(x)* as the amount of bread produced in a bakery located at x, *q(x')* as the amount of bread sold in café located at x', *d(x,x'* the cost transport per unit from x to x', and π(x,x') as the amount of bread to transport. In this picture, the first formulation can be interpreted as the *minimum* cost to transport all bread from bakeries to cafés.
Imagine, instead, that an external agent wants to compete for business with the transportation department, buying bread in x at price -$\alpha(x)$ and selling it to cafés located in x' at price *β(x')*. In order to be competitive, it has to ensure $\alpha(x) + \beta(x') \leq d(x,x')$: the second formulation interpretation will correspond to maximization of the agent's profit. The gain will be the same if both will choose the *optimal transport plan*.

4.2 Wasserstein distance in the maximum entropy framework

Since such computation can be unfeasable to do, especially in non euclidean high dimensional spaces, approximations are necessary. Following the Cuturi's idea [2] , we can use a γ-smoothed Wasserstein distance, i.e. consider the maximum-entropy principle. This allow to gain differentiability of the function:

$$W_\gamma(p,q) = \min_{\pi\in\Pi(p,q)} E_\pi d(x,x') - \gamma H(\pi)$$

with $H(\pi) = -\int dx\, dx' \pi(x,x')\, log\, \pi(x,x')$ the Shannon entropy.
Following the same approach of (), we can define a Lagrangian for W_γ containing the constraints for p and q (4.2):

$$\begin{aligned} L = &\int dx\, dx'\, \pi(x,x')\Big(d(x,x') + \gamma\, log\, \pi(x,x')\Big) + \\ &\int dx\, \alpha(x)\Big(p(x) - \int dx'\, \pi(x,x')\Big) + \\ &\int dx'\, \beta(x')\Big(q(x') - \int dx\, \pi(x,x')\Big) \end{aligned} \tag{4.4}$$

where $\alpha(x)$ and $\beta(x')$ are Lagrange multipliers.

$$\begin{aligned} \frac{\partial L}{\partial \pi(x,x')} =& d(x,x') + \gamma(1 + log\pi(x,x')) - \alpha(x) - \beta(x') = 0 \\ \Rightarrow \quad & \pi(x,x') = e^{\frac{1}{\gamma}(\alpha(x)+\beta(x')-d(x,x'))-1} \end{aligned} \tag{4.5}$$

Defining $u(x) = e^{\frac{\alpha(x)}{\gamma}}$, $v(x') = e^{\frac{\beta(x')}{\gamma}}$ and $K(x,x') = e^{-\frac{d(x,x')}{\gamma}-1}$ the contraints will require:

$$u(x)\int dx'\, K(x,x')v(x') = p(x) \qquad \int dx\, u(x)K(x,x')v(x') = q(x') \tag{4.6}$$

Starting from an initial guess for $u(x)$ these two equations can be solved iteratively, until convergence, finding

$$\begin{gathered} \alpha^* = \gamma\, log\, u(x) \qquad \beta^*(x') = \gamma\, log\, v(x') \\ W_\gamma(p,q) = \int dx\, \alpha^*(x)p(x) + \int dx'\, \beta^*(x')q(x') - \gamma \int dx\, dx'\, e^{\frac{1}{\gamma}(\alpha(x)+\beta(x')-d(x,x'))-1} \end{gathered} \tag{4.7}$$

It has to be noticed that the optimal dual variables α, β have an extra degree of freedom, since from (4.3) $(\alpha^*(x) - c, \beta^*(x') + c)$ are also optimal, for any c. For this reason we will set c = $\int dx\, \alpha^*(x)p(x)$ (*centered optimal dual variable*), s.t. $\langle \alpha^*(x)p(x) \rangle = 0$
Now we identify in $p \equiv p_\theta$ the probability distribution describing the model (where θ stands for a general parametrization), and in $q \equiv \hat{p}$ the real (empirical) one that has to be achieved

$$p_\theta(\boldsymbol{x}) = \frac{e^{-E_\theta(\boldsymbol{x})}}{Z_\theta}, \qquad \hat{p} = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i} \tag{4.8}$$

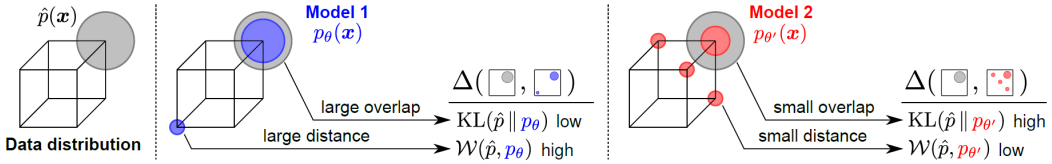

Figure 4.1: Empirical distribution $\hat{p}(\boldsymbol{x})$ (gray) defined on the set of states $0{,}1^d$ with d=3 superposed to two possible models of it defined on the same set of states. Size of the circles indicates probability mass for each state.

In order to define the new learning procedure, we need to determine the ***gradients*** w.r.t. the general parameter θ.

$$\begin{aligned}\frac{\partial W_\gamma(p_\theta, \hat{p})}{\partial \theta} &= \int \boldsymbol{dx}\, \frac{\partial W_\gamma}{\partial p_\theta} \frac{\partial p_\theta(\boldsymbol{x})}{\partial \theta} \\ &= \int \boldsymbol{dx}\, \alpha^*(\boldsymbol{x}) \frac{\partial\, log\, p_\theta(\boldsymbol{x})}{\partial \theta} p_\theta(\boldsymbol{x}) \\ &= \Big\langle \alpha^*(\boldsymbol{x}) \Big(-\nabla_\theta E_\theta(\boldsymbol{x}) \Big) \Big\rangle_{p_\theta}\end{aligned} \tag{4.9}$$

Since an analytical approach for computing such expectation is generally hard, we replace the average with the empirical mean computed by sampling from the model. This leads to

$$\frac{\partial W_\gamma(p_\theta, \hat{p})}{\partial \theta} \approx -\frac{1}{\tilde{N}} \sum_{n=1}^{\tilde{N}} \hat{\alpha}^*(\tilde{\boldsymbol{x}}_n) \nabla_\theta E_\theta(\tilde{\boldsymbol{x}}_n)$$

with $\tilde{N}$ the number of sample extracted from the model, $\hat{\alpha}^*$ the dual solution of the smooth Wasserstein distance, computed between $\hat{p}$ and the approximated distribution of the model $\hat{p}_\theta$.

4.3 D_{KL} vs Wasserstein

To understand the difference between the two metrics, we propose a schematic picture from [13]. As you can see, whereas the D_{KL} looks at overlaps between the real distribution $\hat{p}$ and the model's one p_θ, Wasserstein distance try to compute the *mean* distance between each couple of points of the two distribution. This will corresponds, as in the two example in figure, in different results for the same situation.

Though this new distance appears powerful in usual ML tasks, it has to be noticed that it cannot stand alone: indeed, the gradient depends mainly on the actual

modeled distribution, while the empirical one play only a minor role, through the $\alpha(x)$ computation. For this reason two adjustment will be taken in the implementation of the procedure . We will divide the training in two different evolutions: the first will be done only using the usual MLE gradient, in order to bring our distribution as near as possible to the correct one; in the second part, we will use Wassertein gradient, that will be then computed with better precision. Furthermore, we included in the regularization factor a term that takes into account the KL gradient, in order to ensure a certain *closeness* between the two results. This because, even if the wanted result can be different from the MLE result, the two will be similar.
All this will determine the final expression for the gradient descent:

$$\begin{aligned} d\theta = &- \lambda\Big(\nabla_\theta W_\gamma + \nabla_\theta \mathcal{L} + reg\Big) \\ =&\lambda\big\langle \alpha^*(\boldsymbol{x})\nabla_\theta E_\theta(\boldsymbol{x})\big\rangle_{\hat{p}_\theta} + \\ &\lambda\,\eta\Big(\big\langle \nabla_\theta E_\theta(\boldsymbol{x})\big\rangle_{\hat{p}} - \big\langle \nabla_\theta E_\theta(\boldsymbol{x})\big\rangle_{\hat{p}_\theta}\Big) + \\ &- \lambda(reg) \end{aligned} \tag{4.10}$$

where λ is the learning rate, η the D_{KL} stabilization term and 'reg' the regularization term that can be used.

4.4 Implementation and Results

In order to apply the new training rule, some practical aspects has to be clarified. As you can see in 4.9, gradients are computed only on the model distribution, while the data information comes only indirectly through the α computation. This simple, but important detail suggests that a preliminar training using only D_{KL} minimization is necessary, in order to bring the model close enough to the empirical distribution, to correctly compute the gradients.
In the new case, Frobenius regularization was used, with frobenius norm defined as usual:

$$||A||_2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{4.11}$$

We compare now the new procedure against the D_{KL} training previously defined, using an RBM of 10 Bernoulli hidden units. Just looking at the coupling matrix (Fig. 4.2), it results quite clear that the D_{KL} fails in the cell assembly detection, unlike the Wasserstein one. Moreover, using the Reactivation score, the Wasserstein again outperformed the D_{KL} one, that actually failed (Fig. 4.3).

Although the positive results, further analysis are needed to conclude the overall improvement, at least extending the model to dReLu potential.

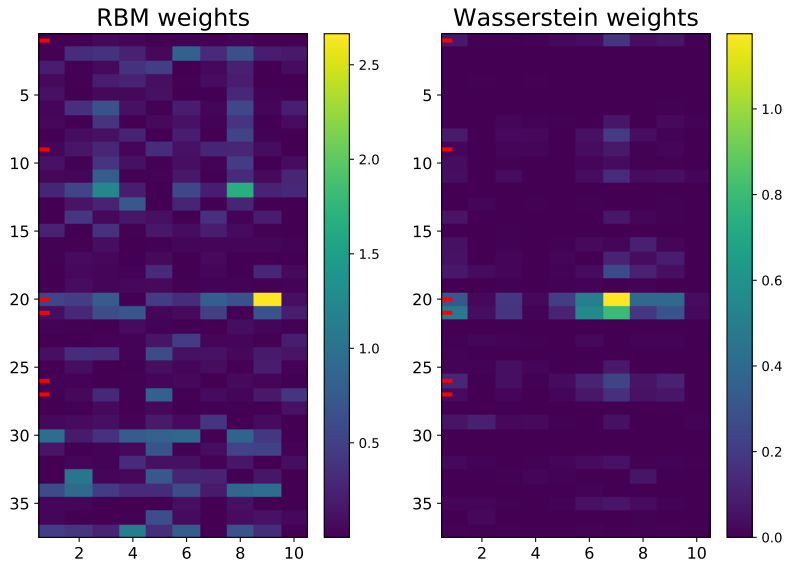

Figure 4.2: Representation of the $D_{KL} - RBM$ final couplings matrix (Left) and of the *Wasserstein-RBM* one (Right)

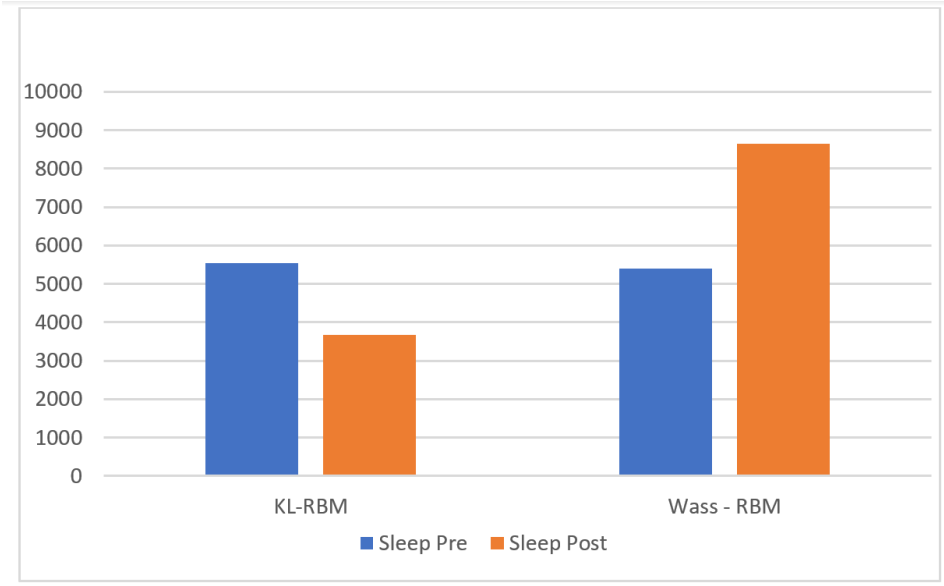

Figure 4.3: Histograms of the Reactivation score for the two RBMs defined above.

Chapter 5

Conclusions

The study of neural spiking activity using RBM, once correctly set, allows to easily find the main correlations between cells, with a clear representation of the underlying structure. Replay analysis supported and finally confirmed such a statement. Once defined the coupling matrix, further analysis can be added to better characterize the system. Nevertheless, it seems quite sure that the D_{KL} training procedure cannot be taken as totally reliable, given the strange behavior; then, further analysis are needed .
Concerning the Wasserstein RBM, a complete implementation, including dReLU potential is necessary before affirming it as a better model. However, the preliminary results showed at least confirm that it is able to obtain some basic results
Further improvements can be easily achieved, using an ensemble of RBMs to correlate spikes in time, avoiding the binning procedure and the associated drawbacks already treated.
All at all, the RBM model, already affirmed in several different circumstances, seems a very powerful tool to be used in inverse inference on this kind of data, and its use could represent an important opportunity to speed up the research in the neuroscience field.

Appendix A

Derivation of the likelihood gradient in RBM

Given the probability distribution described by the RBM model

$$\begin{aligned} \mathcal{P}_\theta(\boldsymbol{v}) &= \sum_{\boldsymbol{h}} \mathcal{P}_\theta(\boldsymbol{v},\boldsymbol{h}) = \frac{\sum_{\boldsymbol{h}} e^{-E_\theta(\boldsymbol{v},\boldsymbol{h})}}{\sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E_\theta(\boldsymbol{v},\boldsymbol{h})}} = \frac{e^{-E_\theta(\boldsymbol{v},\theta)}}{Z_\theta} \\ E(\boldsymbol{v},\boldsymbol{h}) &= -\sum_{i=1}^{N} g_i v_i + \sum_{j=1}^{M} \mathcal{U}_j(h_j) - \sum_{i=1}^{N}\sum_{j=1}^{M} w_{i,j} v_i h_j \end{aligned} \tag{A.1}$$

with θ a generic parameter, the log-likelihood of the model on real data will be $\mathcal{L} = \langle log\, P_\theta(\boldsymbol{v})\rangle_d$.
In order to use Gradient Ascent algorithm we need to compute loglikelihood gradients with respect to each parameter of the model.

$$\begin{aligned} \frac{\partial log\, \mathcal{P}_\theta(\boldsymbol{v})}{\partial\theta} &= \frac{\partial}{\partial\theta}\left[log\left(\frac{\sum_{\boldsymbol{h}} e^{-E_\theta(\boldsymbol{v},\boldsymbol{h})}}{Z_\theta}\right)\right] \\ &= \frac{Z_\theta}{\sum_{\boldsymbol{h}} e^{-E_\theta(\boldsymbol{h},\boldsymbol{v})}} \frac{Z_\theta \sum_{\boldsymbol{h}} \partial_\theta(e^{-E_\theta(\boldsymbol{h},\boldsymbol{v})}) - (\sum_{\boldsymbol{h}} e^{-E_\theta(\boldsymbol{h},\boldsymbol{v})}) \sum_{\boldsymbol{v},\boldsymbol{h}} (\partial_\theta e^{-E_\theta(\boldsymbol{h},\boldsymbol{v})})}{Z_\theta^2} \\ &= \frac{\sum_{\boldsymbol{h}} (-\partial_\theta E_\theta) e^{-E_\theta(\boldsymbol{h},\boldsymbol{v})}}{\sum_{\boldsymbol{h}} e^{-E_\theta(\boldsymbol{h},\boldsymbol{v})}} - \frac{\sum_{\boldsymbol{v},\boldsymbol{h}} (-\partial_\theta E_\theta) e^{-E_\theta(\boldsymbol{v},\boldsymbol{h})}}{\sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E_\theta(\boldsymbol{v},\boldsymbol{h})} Z_\theta} \\ &= -\langle \partial_\theta E_\theta(\boldsymbol{h},\boldsymbol{v})\rangle_d ata + \langle \partial_\theta E_\theta(\boldsymbol{h},\boldsymbol{v})\rangle_m odel \end{aligned} \tag{A.2}$$

where $\langle\rangle_{data}$ and $\langle\rangle_{model}$ are the average computed using respectively the probability distribution followed by the observations and by the model.

References

[1] L. Bottou. *Large-scale machine learning with stochastic gradient descent*. *Proceedings of COMPSTAT Springer*, 2010.

[2] M. Cuturi. *Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances*. *Advances in Neural Information Processing Systems 26, pages 2292–2300, 2013*, 2013.

[3] M. Ekeberg, T. Hartonen, and E. Aurell. *Fast pseudolikelihood maximization for direct-coupling analysis of protein structure from many homologous amino-acid sequences*. *Journal of Computational Physics*, 276:341 – 356, 2014.

[4] G. E. Hinton. *Training Products of Experts by Minimizing Contrastive Divergence 14(8):1711-1800*. *Neural Computation*, 2002.

[5] G. E. Hinton. A practical guide to training restricted boltzmann machines (version 1). Technical report, University of Toronto, 08 2010.

[6] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[7] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. *Proceedings of the National Academy of Sciences of the USA*, 8:2554,2558, 1982.

[8] S. Ioffe and C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. *CoRR*, abs/1502.03167, 2015.

[9] E. T. Jaynes. *Information Theory and Statistical Mechanics*. *Phys. Rev.*, 106:620–630, May 1957.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. *Optimization by Simulated Annealing*. *Science, New Series*, 220:671–680, 1983.

[11] F. Liang, C. Liu, and R. C. Carroll. *Advanced Markov Chain Monte Carlo Methods, chap. 5.4*. Wiley, 2010.

[12] D. J. C. MacKay. *Chapters(38-42): 'Introduction to neural networks' and 'Hopfield model' in "Information Theory, Inference and Learning Algorithm"*. Cambridge, 2003.

[13] G. Montavon, K.-R. Müller, and M. Cuturi. *Wasserstein Training of Restricted Boltzmann Machines*. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3718–3726. Curran Associates, Inc., 2016.

[14] V. Nair and G. E. Hinton. *Rectified linear units improve restricted boltzmann machines*. *Proceedings of the 27th international conference on machine learning (ICML-10), pages 807-814*, 2010.

[15] R. M. Neal. *Annealed importance sampling*. *Statistics and computing*, 11(2):125âĂŞ139, 2008.

[16] H. C. Nguyen, R. Zecchina, and J. Berg. *Inverse statistical problems: from the inverse Ising problem to data science*. *Advances in Physics, 66 (3), 197-261 (2017)*, 2017.

[17] A. Peyrache, M. Khamassi, K. Benchenane, S. I. Wiener, and F. P. Battaglia. *Replay of rule-learning related neural patterns in the prefrontal cortex during sleep*. *Nature Neuroscience*, 12(7):919–926, 2009.

[18] E. T. Rolls. *Cerebral cortex: Principles of operation - Chapter: Attractor networks, energy landscapes, and stochastic neurodynamics*. Oxford University Press, 2016.

[19] H. Shimodaira. *Improving predictive inference under covariate shift by weighting the log-likelihood function*. *Journal of Statistical Planning and Inference*, 90:227–244, 10 2000.

[20] H. T. Siegelmann, B. G. Horne, and C. L. Giles. *Computational Capabilities of Recurrent NARX Neural Networks*. *University of Maryland*, 27(1):208, 1995.

[21] P. Smolensky. *Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory in Rumelhart, David E.; McLelland, James L. (eds.). "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", Volume 1: Foundations*. MIT Press, 1986.

[22] J. Sohl-Dickstein, P. B. Battaglino, and M. R. DeWeese. *New Method for Parameter Estimation in Probabilistic Models: Minimum Probability Flow*. *Phys. Rev. Lett.*, 107:220601, Nov 2011.

[23] H. Sompolinsky and I. Kanter. *Temporal association in Asymmetric Neural networks*. *Physical Review Letters*, 57:22, 1986.

[24] G. Tavoni, S. Cocco, and R. Monasson. *Neural assemblies revealed by inferred connectivity-based models of prefrontal cortex recordings*. *Journal of Computational Neuroscience*, 41(3):269–293, Dec 2016.

[25] G. Tavoni, U. Ferrari, F. P. Battaglia, S. Cocco, and R. Monasson. Functional coupling networks inferred from prefrontal cortex activity show experience-related effective plasticity. *Network Neuroscience*, 1(3):275–301, 2017.

[26] T. Tieleman. *Training restricted boltzmann machines using approximations to the likelihood gradient*. *Proceedings of the 25th international conference on Machine learning, ACM*, page 1064âĂŞ1071, 2008.

[27] J. Tubiana. *Restricted Boltzmann Machines : from Compositional Representations to Protein Sequence Analysis*. PhD thesis, Ecole normale superieure de Paris, 45 Rue de l'Ulm, 75005 Paris, France, 11 2018. Supervisor: Remi Monasson.

[28] J. Tubiana, S. Cocco, and R. Monasson. *Learning Compositional Representations of Interacting Systems with Restricted Boltzmann Machines: Comparative Study of Lattice Proteins*. *Neural Computation*, 2019.

[29] C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.