Demolition and Reinforcement of Memories in Spin-Glass-like Neural Networks

Joint International PhD in Physics (Tesi in cotutela/Cotutelle de thèse) between Università di Roma La Sapienza and École Normale Supérieure - PSL

Scuola di Dottorato "Vito Volterra" and École Doctorale Physique en Île de France (EDPIF) (XXXVI cycle)

Enrico Ventura

ID number 1699775

Advisor

Prof. Giancarlo Ruocco
Prof. Francesco Zamponi

Academic Year 2022-2023

Demolition and Reinforcement of Memories in Spin-Glass-like Neural Networks
PhD thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author’s email: enrico.ventura@uniroma1.it

Abstract

Statistical mechanics has made significant contributions to the study of biological neural systems by modeling them as recurrent networks of interconnected units with adjustable interactions. Several algorithms have been proposed to optimize the neural connections to enable network tasks such as information storage (i.e. associative memory) and learning probability distributions from data (i.e. generative modeling). Among these methods, the Unlearning algorithm, aligned with emerging theories of synaptic plasticity, was introduced by John Hopfield and collaborators. The primary objective of this thesis is to understand the effectiveness of Unlearning in both associative memory models and generative models. Initially, we demonstrate that the Unlearning algorithm can be simplified to a linear perceptron model which learns from noisy examples featuring specific internal correlations. The selection of structured training data enables an associative memory model to retrieve concepts as attractors of a neural dynamics with considerable basins of attraction. Subsequently, a novel regularization technique for Boltzmann Machines is presented, proving to outperform previously developed methods in learning hidden probability distributions from data-sets. The Unlearning rule is derived from this new regularized algorithm and is showed to be comparable, in terms of inferential performance, to traditional Boltzmann-Machine learning.

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure - PSL.
Dans le cadre d'une cotutelle avec Université de Rome « La Sapienza ».

Démolition et renforcement des états de mémoire dans Modèles de réseaux de neurones de type Spin-Glass

Demolition and Reinforcement of Memories in Spin-Glass-like Neural Networks

Soutenue par
Enrico VENTURA
Le 1 Décembre 2023

Ecole doctorale n° 564
École Doctorale Physique en Île-de-France (EDPIF)

Spécialité
Physique

RÉSUMÉ

La mécanique statistique a apporté des contributions significatives à l'étude des systèmes neuronaux biologiques en les modélisant comme des réseaux récurrents d'unités interconnectées avec des interactions ajustables. Plusieurs algorithmes ont été proposés pour optimiser les connexions neuronales afin de permettre des tâches telles que le stockage d'informations (i.e. la mémoire associative) et l'apprentissage de distributions de probabilités à partir de données (i.e. la modélisation générative). Parmi ces méthodes, l'algorithme du Unlearning, aligné sur les théories émergentes de la plasticité synaptique, a été introduit par John Hopfield et ses collaborateurs. L'objectif principal de cette thèse est de comprendre l'efficacité de l'Unlearning dans les modèles de mémoire associative et les modèles génératifs.

Le premier chapitre sert d'introduction à trois types fondamentaux de modélisation : la mémoire associative, la classification et la modélisation générative (en particulier l'apprentissage par Boltzmann Machine). Il analyse également les différences substantielles entre la mémoire et la classification dans les réseaux neuronaux, ainsi que certaines similitudes formelles.

Le deuxième chapitre examine un algorithme simple de formation par injection de bruit pour les réseaux neuronaux récurrents, connu sous le nom de Training-with-Noise, dans le contexte du bruit structuré. Alors que l'injection d'une quantité maximale de bruit aléatoire pourrait être préjudiciable dans le scénario standard, l'incorporation de dépendances internes entre les caractéristiques des données d'apprentissage bruyantes améliore considérablement le pouvoir d'associativité du réseau. Nous dérivons une recette analytique pour optimiser la structure du bruit et la validons numériquement. En outre, nous élucidons l'émergence de la règle d'Unlearning de l'algorithme de Training-with-Noise en présence de bruit structuré, en examinant divers types de données d'apprentissage, y compris des chiffres manuscrits et des données corrélés dans l'espace.

Le troisième chapitre présente une nouvelle technique de régularisation pour l'apprentissage par Boltzmann Machine. Nous explorons une limite spécifique de cette régularisation, qui conduit à la récupération de l'Unlearning à moyenne thermique. En outre, nous démontrons l'équivalence de la règle d'Unlearning avec une Boltzmann Machine à deux étapes. Enfin, nous établissons une équivalence formelle entre les Boltzmann Machines (i.e. les modèles génératifs), les Support Vector Machines et l'algorithme d'Unlearning (i.e. les modèles à mémoire associative).

Dans la discussion qui suit, nous analysons les résultats et donnons un aperçu des orientations potentielles de la recherche future.

MOTS CLÉS

Systèmes désordonnés, réseaux neuronaux, modèles de mémoire associative, modèles génératifs.

ABSTRACT

Statistical mechanics has made significant contributions to the study of biological neural systems by modeling them as recurrent networks of interconnected units with adjustable interactions. Several algorithms have been proposed to optimize the neural connections to enable network tasks such as information storage (i.e. associative memory) and learning probability distributions from data (i.e. generative modeling). Among these methods, the Unlearning algorithm, aligned with emerging theories of synaptic plasticity, was introduced by John Hopfield and collaborators. The primary objective of this thesis is to understand the effectiveness of Unlearning in both associative memory models and generative models.

Chapter one serves as an introduction to three fundamental types of modeling: associative memory, classification, and generative modeling (specifically Boltzmann Machine learning). It also analyses substantial differences between memory retrieval and classification in neural networks, as well as some formal similarities.

The second chapter delves into an examination of a simple noise-injection training algorithm for recurrent neural networks, known as Training-with-noise, within the context of structured noise. While injecting a maximal amount of random noise could be detrimental in the standard scenario, incorporating internal dependencies among the features of the noisy training data significantly enhances the network's associativity power. We derive an analytical recipe for optimizing the noise structure and validate it numerically. Furthermore, we elucidate the emergence of the Unlearning rule from the Training-with-noise algorithm in the presence of structured noise, investigating various types of training datasets, including handwritten digits and spatially correlated patterns.

Chapter three introduces a novel regularization technique for Boltzmann Machine learning. We explore a specific limit of this regularization, which leads to the recovery of a thermally averaged Unlearning. Additionally, we demonstrate the equivalence of the Unlearning rule with a two-step Boltzmann Machine. Ultimately, we establish a formal equivalence between Boltzmann Machines (i.e., generative models), Support Vector Machines, and the Unlearning algorithm (i.e., associative memory models).

In the subsequent discussion, we will analyze the results and offer insights into potential future research directions.

KEYWORDS

Disordered systems, neural networks, associative memory models, generative models.

Contents

Chapter 1

Introduction: modeling intelligence with Recurrent Neural Networks

Understanding the functioning of the human brain has been a central topic of investigation since a long time [1, 2, 3]. Among the main themes that have attracted the attention of scientists there are: the capability of the brain to store information and retrieve it from external stimulation (i.e. *associative memory*), its ability at separating stimuli into different categories (i.e. *classification*) and understanding the underlying structure that is necessary to generate new coherent information (i.e. *generation*). All these tasks belong to the broader concept of intelligence, which is responsible for adaptation and survival in animals [4].
Early studies on the physiology of the brain [1, 2, 3] pointed out its particular composition: an ensemble of neuronal cells linked by wires (i.e. the *synaptic apparatus*) reciprocally exchanging electric signals at deferred times (i.e. the *postsynaptic potentials*). The underlying network structure is generally sparse, asymmetric in the connections, and it can contain closed loops. We call this type of system *recurrent* neural network [3]. Given the experimental observations and the study of the performance of the brain, one might be encouraged to consider recurrent neural networks as a successful starting point to describe intelligence.
No wonder that mathematicians, simultaneously with the development of neuroscience, discovered that some optimization problems could be mapped into graphical representations sharing important similarities with real neural networks. In this case stimuli experienced by the system were replaced by data-points belonging to a data-set. The most emblematic example is Rosenblatt's perceptron and its generalizations [5, 6]. The perceptron problem consists in separating clusters of data-points into different classes depending on reciprocal similarities: this operation is called *classification* and the perceptron is a *classifier*. One of the contact points with biology is the way artificial and real neurons work: the neuronal spike, i.e. the abrupt emission of a postsynaptic potential by a neuron, is triggered when the sum of all signals received by its neighbours overcomes a threshold in tension; at the same way, the perceptron assigns a class to an input vector depending on whether the full incoming field to an output unit overcomes a fixed value or not. Another aspect

involves what neuroscientists refer to as *synaptic plasticity* of the network [1, 2]: synapses tend to modify their conductivity in time, in response of both external and internal endogenous stimuli. Such modifications are associated to the act of learning to accomplish a particular task [7].
At this point, it came natural to statistical physicists, who are specialists in applying probability and statistics for the study of interacting systems, to start contributing to the theory of neural networks. At first, it was a simplification effort [8]: real systems of neurons were reduced to simple recurrent networks, where a number of binary variables, representing the two possible *active/silent* states of the neuron, are mutually coupled by pairwise interactions. These models strongly resembled what people called *spin glasses* [9]: the competition among the different strength of the interactions implies a dynamic multistability of the neural activity and a complex variety of possible equilibrium configurations. Most likely, a boost in the study of these very ancestral types of *complex systems* [10], taking place across the 70s, encouraged John Hopfield, with his pioneering work about Hebbian networks [11] published in 1982, to set a bridge between the statistical mechanics of disordered systems and neuro-physiology. According to these models, the more correlated neurons are, while reacting to external stimuli, the stronger is the synapse that connects them. Three years later, Daniel Amit, Hanoch Gutfreund and Haim Sompolinsky applied the physics of spin glasses to compute the thermodynamics and the critical capacity of a plausible memory model [12]. Further progresses on this line were made until the most recent years, giving an important boost in the field of theoretical neuroscience and the study of associative memory.
In parallel with John Hopfield, Geoffrey Hinton and Terrence Sejnowski (a former Hopfield's doctoral student) proposed another spin glass-like model that encoded the statistics of the stimuli in its parameters (i.e. the interactions and the external fields)[13, 14]. From sampling the typical configurations of the model one could create a new data-set being perfectly indistinguishable from the learnt one. This system, known as Boltzmann-Machine, was the first example of neural network-based generative model, and also a notable case of physics connecting the biological learning with the artificial one.
Nevertheless, the early most important success in connecting artificial with biological networks, was made by Elizabeth Gardner, in her late works from 1988-1989 [15, 16], where she mapped the Rosemblatt's perceptron into a recurrent neural network, computed its critical capacity and advanced a training algorithm for the synaptic strength. The type of computation that she proposed is based on optimizing the interactions rather than fixing them a-priori. Once again, this optimization mechanism reflects the synaptic plasticity of a natural neural system.

This thesis is centered on the Unlearning algorithm, a training procedure for recurrent neural networks introduced by John Hopfield and collaborators in 1983 to enhance the associativity of a memory model [17]. The original idea, from which the algorithm is generated, consists in pruning the system from spurious states, i.e. local attractors of the dynamics responsible for disturbing memory retrieval [3, 18]. This removal of the spurious attractors is performed by iterating an anti-Hebbian learning rule over the synapses: the more neurons are correlated across spurious states, the the more their connection will be weakened by the algorithm. Since

the moment that this technique was proposed, interesting similarities between the Unlearning procedure and neuroscience emerged, in particular concerning the way sleep is supposed to affect the functionality of the brain [19]. Even Hinton himself noticed that a mechanism similar to the Unlearning rule contributed to the training of a Boltzmann Machine [20].
The goal of this work is thus threefold:

1. Showing that the Unlearning rule naturally emerges from a perceptron algorithm regularized through noise injection, and that it reaches an optimal associative memory performance. Unlearning appears to be a valuable unsupervised alternative to the training of a maximally stable perceptron, i.e. a Support Vector Machine. These results are published into [21, 22].

2. Proposing a new type of regularization for Boltzmann Machines, that can be generalized to the Unlearning rule. We also investigate the inferential power of the standard Unlearning procedure, and the importance of the initialization of the parameters in a Boltzmann Machine. These results are contained into [23].

3. Using the Unlearning algorithm as a connection between two learning frameworks of artificial intelligence: associative memory (with its formal mapping to classification problems) and generative modeling, specifically Boltzmann Machine learning. In particular, we want to show a formal equivalence between three learning algorithms: Unlearning, Support Vector Machines, and Boltzmann Machines. These results are contained into [24].

This thesis is placed at the interface between statistical mechanics, theoretical neuroscience and artificial intelligence, and it provides some useful insights for the unification of these three fields of knowledge.

The manuscript is structured as follows:

- Chapter 1: an introduction to various learning algorithms associated to different tasks performed by neural networks. A distinction is made between three fundamental types of modeling: associative memory, classification and the generative one.

- Chapter 2: a simple noise-injection training algorithm for recurrent neural networks, named Training-with-noise, is studied in the case of structured noise. While injecting a maximal amount of random noise would be deleterious in the standard scenario, including internal dependencies among the features of the noisy training data significantly improves the associativity power of the network. An analytical recipe for the best noise structure is derived and tested numerically. We display the emergence of the Unlearning rule from the Training-with-noise algorithm in presence of structured noise and study the cases of different types of training data-sets.

- Chapter 3: a new regularization for Boltzmann Machine learning is proposed. A particular limit of the regularization, that recovers a thermally averaged Hebbian Unlearning, is studied and the equivalence of the Unlearning rule with a two-steps Boltzmann Machine is displayed. Eventually, we show a formal equivalence between Boltzmann Machines (i.e. generative models), Support Vector Machines and the Hebbian Unlearning algorithm (i.e. associative memory models).

- Chapter 4: we discuss the results and give some future perspectives of the research.

Each chapter starts with an introduction explaining the research goals and ends with a summary listing the main points discussed in the text. Each section of the thesis ends with a *checkpoint* paragraph briefly summarizing the main results of the analysis.

For what concerns this chapter, its structure will be the following. The idea of associative memory modeling is introduced in section 1.1 together with the description of a recurrent neural network prototype that will be utilized in our analysis. The main observables representing the quality of the memory retrieval are also presented in detail. Three learning rules regarding associative memory are then described: Hebbian Learning, Hebbian Unlearning (HU), Linear perceptrons and Support Vector Machines (SVMs).
Furthermore, section 1.2 defines what is meant for classification in statistical learning and discusses the differences and analogies between classifiers and associative memory models.
Eventually, section 1.3 treats the generative modeling approach and gives an example of a celebrated energy-based model, namely the Boltzmann Machine (BM).

1.1 Associative Memory Task

With the term *associative memory* we define the capability of the neural system of recalling a given concept (i.e. a *memory*) when a corrupted version of it is displayed as an input [3, 25]. Associative memory is extensively studied by neuroscience, in the terms of the capability of precise regions of the brain (e.g. prefrontal cortex, hippocampus) to perform *long term* and *short term* storage of information [2, 26]. We will limit ourselves to the main class of simple models implemented by physicists i.e. a network of N Ising variables S_i mutually interacting through the couplings J_{ij} with $J_{ii} = 0$. Fig. 1.1 depicts an example of fully connected recurrent neural network with symmetric couplings, which is similar to the ones employed in our further analysis. The memory retrieval operation implies an evolution of the network

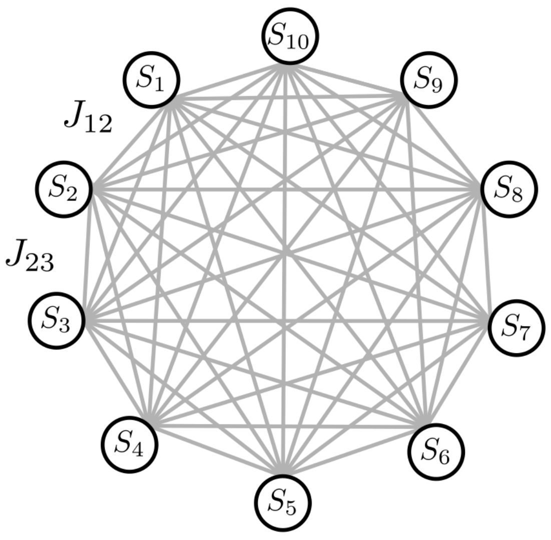


Figure 1.1. Graphical representation of a simple fully connected recurrent neural network of $N = 10$ neurons with symmetric couplings $J_{ij} = J_{ji}$. Autapses J_{ii} are absent.

in time, i.e. a rule for the neural dynamics. An emblematic rule is

$$S_i(t+1) = \text{sign}\left(\sum_{j=1}^{N} J_{ij} S_j(t)\right), \qquad i = 1, .., N \tag{1.1}$$

which can be run either in parallel (i.e. *synchronously*) or in series (i.e. *asynchronously* in a random order) over the i indices [27]. We will mainly concentrate on asynchronous dynamics, in which case equation (1.1) can only converge to fixed points, when they exist [3]. This kind of network can be used as an associative memory device, namely for reconstructing a number p of configurations $\{\vec{\xi}^{\mu}\}_{\mu=1}^{p}$ called *memories*, when the dynamics is initialized on configurations similar enough to them. Memories have binary entries $\xi_i^{\mu} = \pm 1$, $\mu \in [1, ..., p]$. In this work, we will concentrate on Rademecher random memories, generated with a probability

$$P(\xi_i^{\mu} = +1) = P(\xi_i^{\mu} = -1) = 1/2 \qquad \forall i, \mu \tag{1.2}$$

With an appropriate choice of the couplings, the model can store an extensive number of memories $p = \alpha N$, where α is called *load* of the network. This category of models, that aim at retrieving memories as stable fixed points of the dynamics, are called *attractor neural networks.*
We generally want to benchmark the neural network performance, specifically in terms of the dynamic stability achieved by the memory vectors and the ability of the system to retrieve them from blurry examples. Thus, some useful definitions and observables are now introduced for this purpose.

We define *perfect-retrieval* as the capability to perfectly retrieve each memory when the dynamics is initialized on the memory itself. It is here convenient to define a quantity, called *stability*, defined as

$$\Delta_i^\mu = \frac{\xi_i^\mu}{\sqrt{N}\sigma_i} \sum_{j=1} J_{ij}\xi_j^\mu, \qquad \sigma_i = \sqrt{\sum_{j=1}^{N} J_{ij}^2/N}. \tag{1.3}$$

We call n_{SAT} the fraction of ξ_i^μ units that satisfy the following inequality

$$\Delta_i^\mu > 0, \tag{1.4}$$

i.e. that are stable according to one step of the dynamics (1.1). Perfect-retrieval is reached when $n_{SAT} = 1$. It is thus important to remark the following implication of statements

$$n_{SAT} = 1 \iff \Delta_i^\mu > 0 \quad \forall i, \mu. \tag{1.5}$$

The label SAT derives from the nomenclature used in celebrated optimization problems [28, 6] computing the limits of satisfiability of the condition in eq. (1.5). The phase of the model where eq. (1.5) is satisfied was called SAT phase, as opposed to the UNSAT phase.

We furthermore define *robustness* as the capability to retrieve the memory, or a configuration that is strongly related to it, by initializing the dynamics on a noise-corrupted version of the memory. This property of the neural network is related to the size of the basins of attraction to which the memories belong, and does not imply $n_{SAT} = 1$. A good measure of the performance in this sense is the *retrieval map*

$$m_f(m_0, J) := \overline{\Big\langle \frac{1}{N} \sum_{i=1}^{N} \xi_i^\mu S_i^\mu(\infty) \Big\rangle}. \tag{1.6}$$

Here, $\vec{S}^\mu(\infty)$ is the stable fixed point reached by the network having couplings J, when it exists, when the dynamics is initialized on a configuration $\vec{S}^\mu(0)$ having overlap m_0 with a given memory $\vec{\xi}^\mu$. The symbol $\overline{\cdot}$ denotes the average over different realizations of the memories and $\langle \cdot \rangle$ the average over different realizations of $\vec{S}^\mu(0)$. In the perfect-retrieval regime, one obtains $m_f = 1$ when $m_0 = 1$. The analytical computation of the retrieval map might be challenging for some networks. Hence one can introduce another indicative observable for the robustness, i.e. the *one-step retrieval map* $m_1(m_0)$ [29], defined by applying a single step of *synchronous* dynamics

(1.1):

$$m_1(m_0, J) := \overline{\frac{1}{N}\sum_{i=1}^{N}\Big\langle \xi_i^\mu \,\mathrm{sign}\Big(\sum_{j=1}^{N} J_{ij} S_j^\mu(0)\Big)\Big\rangle}\,, \tag{1.7}$$

We now provide a list of notable learning prescriptions in attractor neural networks that will be useful for the rest of the dissertation.

1.1.1 Hebbian Learning

Hebb's (or Hebbian) learning prescription [11, 30] consists in building up the connections between the neurons as an empirical covariance of the memories, i.e.

$$J_{ij}^H = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu. \tag{1.8}$$

This rudimentary yet effective rule allows to retrieve memories up to a critical capacity $\alpha_c^H = 0.138$. [12]. Notably, when $\alpha < \alpha_c^H$ memories are not perfectly recalled, but only reproduced with a small number of errors. In this phase, named *retrieval* phase, memories show some robustness since they belong to large basins of attraction. On the other hand, when $\alpha > \alpha_c^H$ the statistical interference between the random memories impedes the dynamics to retrieve them, shifting the system into an oblivion regime. Notably, the landscape of attractors given by this rule is rugged and disseminated with *spurious states*, i.e. stable fixed points of the dynamics barely overlapped with the original memories [18].
An important notion to keep in mind for the rest of this work is that, whenever the network interactions are symmetric, as in the current case, the Lyapunov function of the dynamics, that is minimized by the attractor states, coincides with the *energy* function of the model. Consequently, attractors are local minima of the energy landscape. For recurrent neural networks, the energy can be defined as

$$E[\vec{S}|J] = -\sum_{i,j>i} S_i J_{ij} S_j, \tag{1.9}$$

equivalently to Ising spin systems in statistical mechanics [3, 25].

1.1.2 Hebbian Unlearning

Inspired by the brain functioning during REM sleep [19], the Hebbian Unlearning algorithm (HU) [17, 19, 31, 32, 21, 33, 34] is a training procedure leading to perfect-retrieval and good robustness in a symmetric neural network. This paragraph contains an introduction to the algorithm as well as a description of the gained performance in terms of perfect-retrieval. The robustness capability of the algorithm will be adressed further in the work.

Training starts by initializing the connectivity matrix according to the Hebb's rule eq. (1.8) (i.e. $J^{(0)} = J^H$). Then, the following procedure is iterated at each time step d:

1. Initialize the network on a random neural state.
2. Run the asynchronous dynamics (1.1) until convergence to a stable fixed point $\vec{S}^*$.
3. Update couplings according to:

$$\delta J_{ij}^{(d)} = -\frac{\lambda}{N} S_i^* S_j^* \qquad J_{ii} = 0 \quad \forall i. \tag{1.10}$$

This algorithm was first introduced to prune the landscape of attractors from proliferating spurious states, i.e. fixed points of (1.1) not coinciding with the memories [18, 3]. Such spurious states are only weakly correlated with the memories. Even though this pruning action leads to the full stabilization of the memories, the exact mechanism behind this effect is not completely understood.
HU is an *unsupervised* algorithm, in the sense that it does not need to be provided explicitly with the memories $\{\vec{\xi}^\mu\}_{\mu=1}^p$, and only exploits the information encoded in Hebbian initialization (see eq. (2.44)). The total number of iterations D is a parameter of the algorithm, and must be chosen as to maximize the recognition performance at a given load α. We report here the analysis of the perfect-retrieval properties of the resulting network, with an estimate of the critical capacity and the amount of iterations for an effective early-stopping of the algorithm, contained in [21].
The performance of the algorithm has been studied in terms of the stabilities Δ_i^μ. This approach has already been attempted [35], but the following analysis pushes it further and reveals new unexpected features. Fig. 1.2 shows the typical behavior of the minimum stability Δ_{min}, the average one Δ_{av} and the maximum stability Δ_{max} as the Unlearning procedure unfolds. The horizontal axis represents the number of steps d performed by the algorithm, rescaled by a factor λ/N. Focusing on Δ_{min}, we can see a non-monotonic behavior: the minimal stability grows to positive values, peaks at some value $d = D_{top}$ and then decreases back to negative values. Between D_{in} and D_{fin} every stability is positive or, equivalently, every memory is a fixed point for the dynamics. As we increase α, the interval $[D_{in}, D_{fin}]$ shrinks, and the height of the peak at D_{top} lowers, until we reach a critical load α_c, above which Δ_{min} never goes above zero. Collecting data for networks of size $N = 300$, 400, 500, 600, 800 and different values of α and ϵ it is possible to extrapolate the position of D_{in}, D_{top} and D_{fin} as a function of α, λ and N, as well as the critical capacity α_c. By fitting the data with respect to the model parameters one can find that the number of iterations is in every case linear in N and $1/\lambda$. Moreover, D_{top} also depends linearly on α. At the critical capacity,

$$\alpha_c^{HU} = 0.589 \pm 0.003 \ ,$$

and the value of $\Delta_{min}(D_{top})$ approaches zero. The value of the critical capacity α_c^{HU} as well as the linear dependence of D_{in}, D_{top}, D_{fin} on N/λ are consistent with evidence provided by past literature [32, 31, 35].
Because the $\Delta_{min}(D)$ curve is quadratic around D_{top}, as illustrated in fig. 1.2, D_{in} and D_{fin} both tend to D_{top} at the critical capacity with a critical exponent $1/2$.

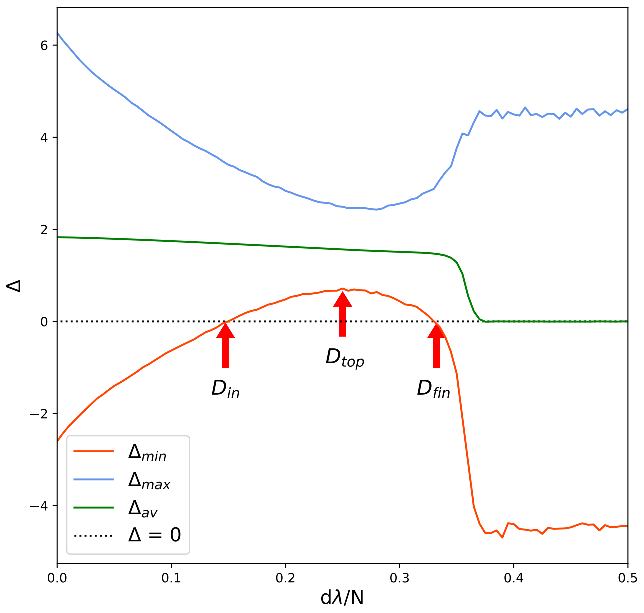


Figure 1.2. Values of the minimal stability Δ_{min} (orange), maximal stability Δ_{max} (blue) and average stability Δ_{av} (green) computed during Hebbian unlearning, averaged over 50 realizations of the memories at $N = 800$, $\alpha = 0.3$, $\lambda = 10^{-2}$. The black dotted line represents the zero-stability threshold to be overcome by Δ_{min} to have all memories perfectly recalled. We denote the corresponding value of the number of iterations D by D_{in}. D_{top} is for the point where the algorithm reaches the maximum value of Δ_{min}, while D_{fin} is the end of the perfect perfect-retrieval regime of the network. We used red arrows to point at D_{in}, D_{top} and D_{fin}.

The resulting scaling relations are:

$$D_{top}(\lambda, \alpha, N) = \frac{N}{\lambda}(a \cdot \alpha + b) \ , \tag{1.11}$$

$$D_{in}(\lambda, \alpha, N) = D_{top} - \frac{N}{\lambda}(c \cdot \alpha + d)^{1/2} \ , \tag{1.12}$$

$$D_{fin}(\lambda, \alpha, N) = D_{top} + \frac{N}{\lambda}(e \cdot \alpha + f)^{1/2} \ , \tag{1.13}$$

with

$$a = 1.02 \pm 0.02 \ , \qquad b = -0.05 \pm 0.01 \ ,$$
$$c = -0.039 \pm 0.003 \ , \qquad d = 0.023 \pm 0.002 \ ,$$
$$e = -0.022 \pm 0.001 \ , \qquad f = 0.013 \pm 0.001 \ .$$

All the statistical errors have been evaluated using the jackknife method [36].

The study of the evolution of the basins of attraction during Unlearning is one of the main contribution of this manuscript, and it will be deepened in detail in the next Chapter.

1.1.3 Linear Perceptrons & Support Vector Machines

The linear perceptron algorithm [15, 37, 6], which is one of the pillars of modern artificial intelligence, is an iterative procedure allowing to fully stabilize the memories and tune their robustness capabilities. Specifically, we are going to refer to *linear perceptron* as the adaptation of the classical perceptron to recurrent neural networks, already introduced in [15, 37, 38, 39, 40]. In this case a N-dimensional input layer is fully connected by the connections J to a N-dimensional output layer. This architecture can thus be mapped into a biologically inspired recurrent neural network. Further details about this mapping will be provided in the next paragraph about classification.
Given the fully connected architecture, we want to find a set of couplings that satisfy the constraints

$$\Delta_i^\mu > k, \qquad \forall \mu, i \ . \tag{1.14}$$

with k being a parameter called *margin*. An elegant way to interpret this problem is to recast it in terms of a linear regression [6] in the J. In terms of the network dynamics the larger $k \geq 0$ is, the more robust memories will be under perturbation of the dynamics. Specifically, given a value of α all memories will be stable up to a maximum value of $k_{max}(\alpha)$. The solution of the problem such that $k = k_{max}(\alpha)$ can be proved to be unique, given one realization of the memories. The maximum capacity achievable by the network is $\alpha_c^P = 2$ such that $k_{max}(2) = 0$. Following previous work, we call SAT phase the region in the space (α, k) such that eq. (1.14) is satisfied, while the rest of the phase space will be said to be UNSAT.
Inside these limits all constraints in (1.14) will be satisfied after a number of iterations of the following serial update for the couplings

$$\delta J_{ij}^{(d)} = +\lambda \sum_{\mu=1}^{p} \epsilon_i^\mu(d) \xi_i^\mu \xi_j^\mu, \quad J_{ii} = 0, \quad \lambda > 0 \tag{1.15}$$

$$\epsilon_i^\mu(d) = \frac{1}{2}\left(1 + \text{sign}(k - \Delta_i^\mu(d))\right),$$

where λ is the learning rate, considered to be small, d is the descrete algorithm time on which the mask ϵ_i^μ depends. One can also symmetrize equation (1.15) to train symmetric couplings by redefining the mask ϵ_i^μ as

$$\epsilon_i^\mu \to \epsilon_{ij}^\mu = \frac{1}{2}(\epsilon_i^\mu + \epsilon_j^\mu), \tag{1.16}$$

where the dependence on d has been removed for clarity. The perceptron algorithm is *supervised*, because it needs to be provided explicitly with the memories $\{\vec{\xi}^\mu\}_{\mu=1}^p$ to update the couplings. In the symmetric case, that will be shortened as SP for the rest of the manuscript, the function $k_{max}(\alpha)$ has been determined analytically [37] for slightly diluted recurrent networks, i.e. networks with an average connectivity scaling as $\log N$. Numerical results from the study of the algorithm defined in eq. (1.15) on networks that are both fully connected and fully symmetric suggest that, for the same degree of symmetry, k_{max} at a given α is located slightly above the one predicted by [37]. This finding, discussed in fig. 1.3, suggests to reconsider previous interesting analyses [42] and opens the road to further investigations of the

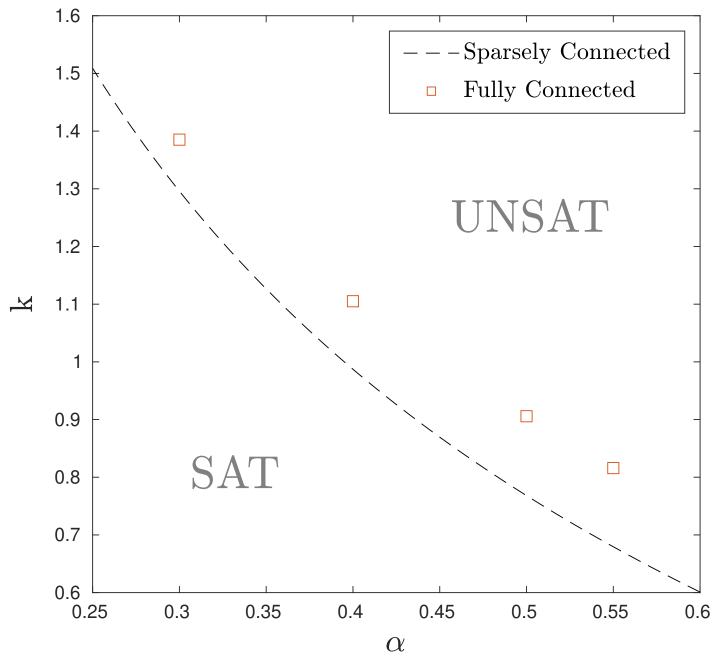


Figure 1.3. Phase diagram of the linear symmetric perceptron in the plane defined by the load parameter α and the margin k. The dashed line is the analytical result for $k_{max}(\alpha)$ obtained for slightly diluted networks [37]. Squares show numerical results for $k_{max}(\alpha)$ in a fully connected model at $\alpha \in \{0.3, 0.4, 0.5, 0.55\}$. Simulations have been run at different sizes of the network to measure the probability for the algorithm to converge before 10^3 steps of the training (hence providing a lower bound to the actual value of stability). A standard finite size scaling analysis [41, 28] has been used to extrapolate the value of $k_{max}(\alpha)$ to the thermodynamic limit.

critical capacity as a function of the network connectivity [43]. It has been proved numerically that, given α, the larger k, with $0 < k < k_{max}(\alpha)$, the wider the basins of attraction [44, 21]. In line with previous literature [38, 45] we call a maximally stable perceptron, such that $k = k_{max}(\alpha)$, a Support Vector Machine (SVM).

Checkpoint

In this section we have seen that:

- Associative memory, i.e. the ability of a system to store information and retrieve it when stimulated, can be modeled by recurrent neural networks.
- The definition of memory performance used in this work is based on two properties: perfect-retrieval, i.e. the capability of the network to retrieve a memory vector with no errors; robustness, the capability to associate corrupted versions of a memory to the memory itself.
- Learning algorithms can build the neural network from a specific realization of the memory in an iterative way: Hebbian learning does not reach perfect-retrieval, yet its retrieval phase extends up to $\alpha_c^H \simeq 0.14$; Hebbian Unlearning

(HU) reaches perfect-retrieval up to $\alpha_c^{HU} \simeq 0.6$ and it is unsupervised; the linear perceptron gains perfect-retrieval up to $\alpha_c^{P} = 2$ and it is supervised.

1.2 Classification Task

In the classification problem we have a set of N dimensional data-points $\{\vec{S}^\mu\}_{\mu=1}^M$ assigned to C possible classes by an unknown function $\phi(\vec{S^\mu})$, i.e.

$$\vec{S}^\mu \xrightarrow{\phi} \xi^\mu \qquad \text{with} \qquad \xi^\mu \in \{\text{class 1}, \text{class 2}, ..., \text{class C}\}, \quad \forall \mu. \tag{1.17}$$

Generally speaking, each class is associated to a number, as data-points are also encoded in vector of numbers (i.e. the *features*), allowing proper statistical calculations to be performed. The goal of a classifier, as a method to solve the classification problem, is to learn the class to which each data-point belongs and how to assign new unseen data to their most suitable classes. Literature usually refer to the act of learning as *training* of the model: the data used for this purpose are named *training-set* while the ones used to test the classification performance form the *testing-set.*
Let us consider the case of only $C = 2$ classes, that we translate into two possible values for ξ^μ, i.e. $\xi^\mu \in \{-1, +1\}$, $\forall \mu$. This problem can be translated into a simple neural network problem, with nodes and couplings. Yet in this case there is not recurrency in the graph and the simplest classification method is called *linear regression.* Fig. 1.4 provides a graphical representation of a classification problem

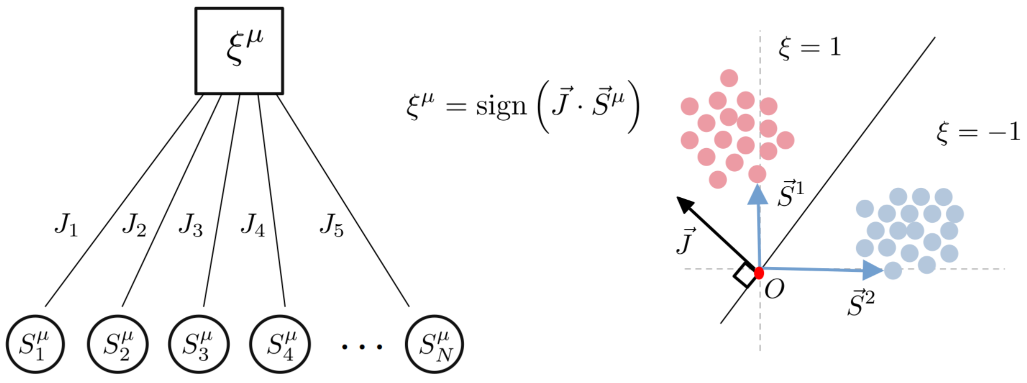


Figure 1.4. Graphical representation of a simple classification problem with two classes. Data-points are separated by a hyperplane in the space of the data features.

translated into a neural network. Given the classes for each data-point, we can search for the realization of the parameters $\vec{J}$ such that

$$\xi^\mu = \text{sign}\left(\vec{J} \cdot \vec{S}^\mu\right), \qquad \forall \mu. \tag{1.18}$$

It is now evident that the $N - 1$ dimensional hyperplane orthogonal to $\vec{J}$ separates the two classes in the space of the features. In principle there is not one single solution to the problem (i.e. one realization of $\vec{J}$ and the hyperplane), and one can modify the classification rule to make the separation even more robust to new data to be showed to the system. In statistical learning the capability of a network to classify unseen data belonging to the testing-set is called *generalization.* The

absence of generalization is called *overfitting.* A broader description of the concepts of overfitting and generalization will be provided further in the manuscript.
This model coincides with the *linear perceptron* that we previously introduced in the context of associative memory. Though, as a difference with the previous case, now we have an input layer that converges into an output variable through a single vector $\vec{J}$. The practical method to learn $\vec{J}$ from the knowledge of the classes is analogous to the one introduced in eq. (1.15), i.e.

$$\delta J_i^{(d)} = +\lambda \sum_{\mu=1}^{M} \epsilon_i^\mu(d) S_i^\mu \xi^\mu \qquad \lambda > 0, \tag{1.19}$$

$$\epsilon_i^\mu(d) = \frac{1}{2}\left(1 - \text{sign}(\xi^\mu \vec{J}(d) \cdot \vec{S}^\mu)\right),$$

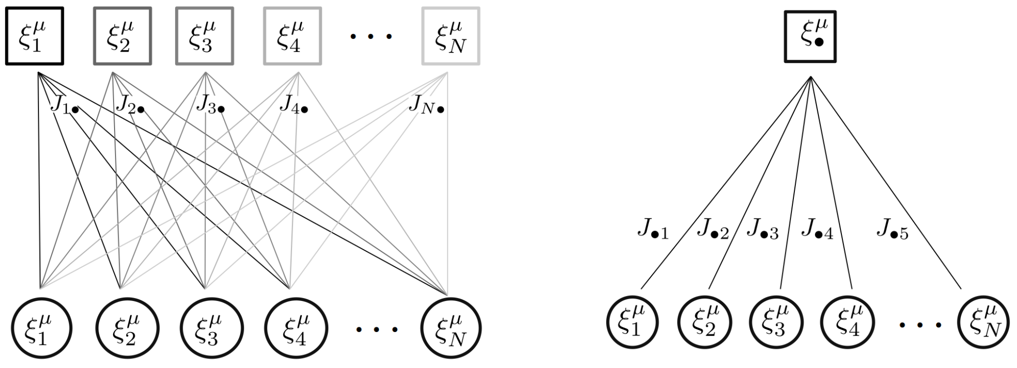
where λ is the learning rate, considered to be small, and d is the discrete algorithm time, on which the mask ϵ_i^μ depends. It can be proved that, after a suitable number of iterations, and up to a maximum amount of training data that is the double of the number of neurons, the matrix J converges to one configuration that satisfies eq. (1.18) [15, 46].

1.2.1 The formal analogies between associative memory and classification

The strong similarity between the concepts of associative memory and classification now looks clear from the previous sections. Nevertheless, it is important to recognize the mutual differences as well as their points of contact.
Associative memory is a dynamic process, because the retrieval of a memory from an example relies on the existence of a basin of attraction, which is a pure dynamic entity. Hence we might think to use basins of attraction as *dynamic classes*, areas of influence of the memories to which new unseen neural configurations belong. In other words, we might treat each memory as a distinct dynamic class. If this is the case, each class would be no more encoded into one numerical variable, as it was for classification, but rather into a vector, having the same dimension of the neural configurations. Fig. 1.5a graphically represents an associative memory model as a feed-forward neural network with a N dimensional input layer and a N dimensional output layer being fully connected by the elements of the couplings matrix J. The output is no more a single label but, instead, a collection of labels $\vec{\xi}^\mu$. This feed-forward representation can be mapped into a recurrent neural network, of the same kind as in fig. 1.1.
Furthermore, an associative memory model is built by learning J directly from the memories, i.e. the dynamic classes are learnt by means of the dynamic classes themselves. This means that the classification rule is no more the one expressed in eq. (1.17) but it is instead given by

$$\xi_i^\mu = \text{sign}\left(\vec{J}_i \cdot \vec{\xi}^\mu\right) \qquad \forall i, \mu, \tag{1.20}$$

where $\vec{J}_i$ is the i line of the connectivity matrix of the recurrent neural network. This picture can be interpreted as a Cartesian product of N classifiers where each



(a) Associative memory model **(b)** Gardner's linear perceptron

Figure 1.5. Graphical representations of associative memory models, in analogy with traditional classification problems.

class is a memory: perfect-retrieval is obtained when each memory is fitted by the intersection of the hyper-planes found by the rows of the connectivity matrix J, i.e. when N parallel classification problems are correctly solved. In principle, if no assumptions about the structure of the couplings are made, such classification problems are independent, but this is not always the case, especially when the symmetry of the connections is constrained by the network model.

By contrast with the associative memory picture explained above, classifiers are not dynamic machines. In fact, once the classes have been learned, new data to be classified are instantaneously associated to one or the other side of the hyperplane (or to one of the multiple half-planes if more than two classes are provided). Hence they are not classified by any dynamic process.
We might propose two points of contact between the two types of problems. Classifiers usually rely on a number of classes C that is much smaller than the dimension of the data-points, i.e. $C \ll N$. On the other hand, associative memory models have been developed with the explicit aim of dealing with a number of memories $p = \mathcal{O}(N)$: this implies the emergence of spurious attractors and similar dynamic phenomena that, apparently, have nothing to share with classification problems. However, when $p \ll N$ memories are well separated with each other, their basins of attraction are large, and new unseen configurations are rapidly associated to one memory. In this manner we can relate the concepts of *perfect-retrieval* in memory models and *classification* as well as the ones of *robustness* and *generalization*.
Another case where the two tasks overlap significantly is when we map a linear perceptron into a recurrent neural network, i.e. when we solve the perceptron problem à la Gardner [15, 40]. In this case we want to perfectly retrieve p randomly generated memories. To do so, we build N independent perceptrons: each of them receives one memory as an input and returns one of the entries of the same memory as an output. This picture is depicted, once again, in fig. 1.5a. However, if perceptrons are independent, and memories are generated by the same random process, we can treat the N parallel problems as one single perceptron classification problem, represented

in fig. 1.5b. It is not a case that, by construction, Gardner's perceptron algorithm [15, 37, 44] forces memories to be retrieved in one step of the dynamics, when the network is initialized elsewhere in their basins of attraction (see eq. (1.15)). This problem is thus a clear connection between dynamic classes embodied by basins of attraction and standard, static classes used in classifiers.

Checkpoint

In this section we have seen that:

- Classification can be performed by feed-forward neural networks. Classic perceptron models are examples of classifiers.
- Classification and Associative memory are different tasks since the former relies on one step of the network dynamics, the latter needs convergence into a fixed point. Associative memory resembles classifiers when the number of memories is small and dynamics easily converges onto attractors.

1.3 Data Generation Task

By referring to *generative modeling* we describe the capacity of specific neural networks to learn the probability distribution of a data-set and generate brand new data that exhibit maximum coherence with the same statistics [47, 46]. Imagine to have a collection of N variables in a vector $\vec{S} = (S_1, ..., S_N)$. Data are realizations of such a vector grouped into a set $\{\vec{S}^\mu\}_{\mu=1}^M$ and sampled from a joint distribution $P_{true}(\vec{S})$. Given M data, we have access to the frequency of occurrence of a certain variable, i.e. the empirical distribution

$$P_{data}(\vec{S}) = \frac{1}{M}\sum_{\mu=1}^{M} \delta\left(\vec{S}^\mu - \vec{S}\right). \tag{1.21}$$

The generative approach finds the model described by the joint distribution $P_{mod}(\vec{S}|\hat{\theta})$ where the parameters $\hat{\theta}$ are inferred from the training data such that P_{mod} is the closest possible to P_{data}. However, P_{data} might still differ from the ground-truth distribution P_{true}. As a remedy, it is useful to reduce the number of degrees of freedom of the problem by designing the model depending on some hints that we have about P_{true}. For instance one might choose a *graphical model*, where variables are sketched as the nodes of graph [46] with interactions to be inferred, rather than setting a prior distribution for the variables as a *mixture of Gaussians* of unknown means and unit variances [48]. *Regularization* techniques are used for this purpose, i.e. to reduce the number of parameters, and thus allowing P_{mod} to get closer to P_{true}.
The generative approach is largely implemented across several disciplines such as computational neuroscience [49, 50], bio-informatics [51], animal behaviour [52], physical simulations [53], image and text synthesis [54, 55, 56].

1.3.1 Boltzmann Machine Learning

We now describe a specific graphical model of generative neural networks which will be particularly relevant to the rest of our dissertation. It is inspired by the statistical mechanics at the equilibrium and it is called Boltzmann Machine (BM) [57, 20, 58, 46].
Consider a fully connected network of binary N Ising variables $\vec{S} \in \{-1, +1\}^N$ with the following energy function

$$E[\vec{S}|J, \vec{h}] = -\sum_{i,j>i}^{N} S_i J_{ij} S_j - \sum_{i}^{N} h_i S_i, \tag{1.22}$$

where J_{ij} are symmetric couplings and h_i are the fields acting on each neuron site. We know, from statistical mechanics, that such a system at equilibrium at a temperature β^{-1} will obey the following joint probability density function

$$P_{mod}(\vec{S}|J, \vec{h}, \beta) = \frac{1}{Z_\beta}\exp\left(-\beta E[\vec{S}|J, \vec{h}]\right), \qquad Z_\beta = \sum_{\vec{S}} \exp\left(-\beta E[\vec{S}|J, \vec{h}]\right), \tag{1.23}$$

which is the *Gibbs-Boltzmann distribution*, where β can be set to one without any effect in the training. Imagine a data-set $\{\vec{S}^\mu\}_{\mu=1}^M$ satisfying the following empirical distribution

$$P_{data}(\vec{S}) = \frac{1}{M}\sum_{\mu=1}^{M}\prod_{i=1}^{N}\delta_{S_i^\mu,S_i}. \tag{1.24}$$

Training a Boltzmann Machine means finding the parameters J and $\vec{h}$ that minimize the distance between P_{data} and P_{mod}. Thus it comes natural to impose the minimization of the Kullback-Leibler divergence between P_{data} and P_{mod}, i.e.

$$\mathcal{L}(J,\vec{h}) = -\sum_{\vec{S}} P_{data}(\vec{S})\log\left(\frac{P_{mod}(\vec{S}|J,\vec{h})}{P_{data}(\vec{S})}\right), \tag{1.25}$$

which is equivalent to maximize the cross-entropy of P_{mod} with respect to the empirical distribution P_{data}. Derivating eq. (1.25) with respect to J_{ij} and h_i we obtain the gradient of the Loss, i.e.

$$\nabla_{ij}\mathcal{L} = \langle S_iS_j\rangle_{mod} - \langle S_iS_j\rangle_{data}, \tag{1.26}$$

$$\nabla_{i}\mathcal{L} = \langle S_i\rangle_{mod} - \langle S_i\rangle_{data}, \tag{1.27}$$

where $\langle\,\cdot\,\rangle_{data}$ and $\langle\,\cdot\,\rangle_{mod}$ are the averages over the respective probability distributions. Therefore, the parameters can be found by iterating the following gradient descent equations

$$\delta J_{ij} = -\lambda\nabla_{ij}\mathcal{L} = \lambda\left(\langle S_iS_j\rangle_{data} - \langle S_iS_j\rangle_{mod}\right), \tag{1.28}$$

$$\delta h_i = -\lambda\nabla_{i}\mathcal{L} = \lambda\left(\langle S_i\rangle_{data} - \langle S_i\rangle_{mod}\right), \tag{1.29}$$

with λ being a small positive learning rate.
To go technical in the training algorithm, the mean and covariance over the data can be computed upstream, because they only depend on the training data-set; on the other hand, the moments of P_{mod} must be sampled step-by-step during the process, because their exact calculation would involve a sum over all possible $\vec{S}$. Sampling can be performed by a sufficient number of Monte Carlo chains at the equilibrium at $\beta = 1$, implying an algorithm time that is long enough to ensure ergodicity for each chain. Usually the number of chains should be of the same order of magnitude of M, the number of training data-points, in order for the corrections to the empirical averages not to be sub-dominant with respect to the sampled ones.
When the process converges to the fixed points of eq. (1.28) and eq. (1.29) we obtain a condition of *moment matching*, i.e. the first and the second moments of the two probability distributions coincide. In principle the training of a BM is a convex problem [57], however there might be some initial conditions that push the parameters closer to their target configuration. A good choice is, for instance

$$J_{ij}^{(0)} = \langle S_iS_j\rangle_{data} - \langle S_i\rangle_{data}\langle S_j\rangle_{data}, \tag{1.30}$$

and

$$h_i^{(0)} = \operatorname{asinh}\left(\langle S_i\rangle_{data}\right). \tag{1.31}$$

because eq. (1.30) and eq. (1.31) are generally close to the fixed point of equations (1.28) and (1.29).

In order to measure the quality of the training of a BM one can measure whether the moment matching condition is met or not. The observable to measure in this case is the Pearson coefficient between the moments of the P_{mod} and P_{data} distributions at the end of the training. Let us collect the 2-point correlation matrices $\langle S_i S_j \rangle$ in a vector $\vec{c}$ where each entry runs over the indices $i, j > i$. Let us group the means $\langle S_i \rangle$ in a similar vector $\vec{\mu}$ where each entry runs over the index i. Then the Pearson coefficients will be defined as

$$\rho_J = \frac{\sum_i c_i^{mod} c_i^{data}}{\sqrt{\sum_i (c_i^{mod})^2}\sqrt{\sum_i (c_i^{data})^2}} \qquad \rho_h = \frac{\sum_i \mu_i^{mod} \mu_i^{data}}{\sqrt{\sum_i (\mu_i^{mod})^2}\sqrt{\sum_i (\mu_i^{data})^2}}. \tag{1.32}$$

Checkpoint

In this section we have seen that:

- Generative models learn from data a joint probability distribution which can be used to sample new examples. Efficient models generate examples that are indistinguishable from the training data.
- A good generative model does not learn the probability distribution of the visible data, but rather the ground-truth distribution that generated them.
- Boltzmann-Machines are generative models that can be mapped into a recurrent neural network. The parameters of the model, i.e. couplings and fields, help a Gibbs-Boltzmann distribution to fit the statistics of the data. The training of the neural network is unsupervised.

1.4 Summary & Conclusions

In this chapter we have seen that:

- An associative memory model is a recurrent neural network that stores information in form of binary configurations of the neurons, named memories. These models can be characterized in terms of two properties of the memories: perfect-retrieval and robustness. Hebbian networks are simple memory models that never reach the perfect-retrieval condition. Hebbian Unlearning (HU) is an unsupervised learning algorithm that reaches perfect-retrieval up to a critical value of the load $\alpha_c^{HU} \simeq 0.6$. Linear perceptrons, instead, reach both perfect-retrieval and robustness through a full supervised training procedure, also up to a critical capacity $\alpha_c^P = 2$ (when the symmetry of the couplings is not constrained a-priori).

- There are some differences and formal analogies between the associative memory task, accomplished by recurrent neural networks, and the classification one, performed by feed-forward networks. The main difference is: memory models use a neural dynamics, to be iterated for several steps, to associate one given neural state to a memory, when the former state belongs to the basin of attraction of the latter; classifiers associate each data-point to a class in one single step of the neural dynamics. Associative memory tend to behave similarly to classifiers when the number of memories is subdominant with respect to the number of neurons. In this case one can relate perfect-retrieval to accomplished classification and robustness to the generalization properties of classifiers. The mapping of linear perceptrons into recurrent neural networks established by Elisabeth Gardner is a successful bridge between these two learning frameworks.

- Boltzmann Machines (BMs) are generative models that share the same architecture of associative memory models. As a conceptual difference, BMs learn the probability distribution of a data-set, without storing the data themselves. Once the distribution is learned, one can sample new examples that are statistically coherent to the training data. The way a BM can learn the unknown distribution of data can be improved by regularizing the Loss function of the problem.

In the next section we will extensively analyze the associative memory performance of HU, highlighting its excellent robustness properties. We advance a theoretical explanation for the approaching of such wide basins of attraction by employing the theory behind a noise-injection based learning algorithm resembling a supervised linear perceptron. Eventually, the analytical argument is tested on different types of data-set, such as MNIST or spatially correlated Euclidean maps.

Chapter 2

Unlearning as noise injection: approaching maximally stable Perceptrons

An important challenge for modern artificial neural networks is to improve their performance by means of regularization techniques. One class of techniques relies on injecting noise in the training process, with the idea of teaching the machine how to better infer the hidden data structure from errors. Examples of these types of regularization are the drop-out [59, 60, 61, 62] or data-augmentation procedures [63, 64]. The former consists in stochastically deactivating neurons across the network, the latter acts on the training data themselves, by performing transformations (e.g. translations, rotations and other symmetries) to increase the heterogeneity of the data-set.
Even if such procedures are practical and successful in the world of deep networks, there are only few examples of theory-based criteria for choosing how to engineer noise to inject in the training [60, 64]. Our work considers a simple learning algorithm for attractor neural networks (see section 1.1), employing noise-injection during training. We are interested in this type of networks for multiple reasons: they are suitable to be modeled through the tools of statistical mechanics; they constitute a simplified version of modern neural networks, and yet they present a resemblance with biological neuronal systems. In the context of attractor neural networks, *noise-injection* refers to a random alteration of the neuronal activity encoding the memories. The objective of this chapter is twofold:

1. To establish a theoretical criterion for generating the most effective noise to be utilized in training, thereby optimizing the performance of an associative memory model. Specifically, we demonstrate that generating optimal noise translates into producing training data-points whose features adhere to specific constraints, which we refer to as the *structure* of the noise.

2. To show that injecting a maximal amount of noise, subject to specific constraints, turns the learning process into an unsupervised procedure, which is faster and more biologically plausible. We will also delineate some important connections between training procedures with optimally structured noise and

other consolidated learning prescriptions present in literature, such as the Hebbian Unlearning rule. Specifically we will re-interpret the Unlearning algorithm in terms of an effective noise-injection regularization for a Hebbian network.

The structure of the chapter will be the following. Gardner's original *training-with-noise* algorithm is presented in section 2.1 and the consistency of the numerical results with the theory developed by [65, 66] is showed; we will deal in particular with the perfect-retrieval and robustness capabilities of the algorithm on fully connected networks.
We then propose a derivation of the constraints that should be satisfied by maximally noisy training data for the same algorithm in order to mimic SVM learning. This will be treated in section 2.2 where we also show how relevant the initial conditions over the couplings are to sample the good training configurations. Specifically, we conclude that low saddles in an initially Hebbian-shaped energy landscape are the most effective data for the training-with-noise algorithm.
Furthermore, section 2.3 focuses on the use of stable fixed points of the neural dynamics as training data, proving that the celebrated Unlearning rule is contained into the training-with-noise procedure when noise is maximal. Both the perfect-retrieval and robustness properties of the trained network are examined: consistently with the theoretical results, the system correctly reproduces a SVM for α up to $\alpha_c^{HU} \simeq 0.6$. Eventually, the original Unlearning routine is generalized by implementing a training-with-noise procedure with a moderate amount of noise, showing an improvement in the critical capacity.
Section 2.4 deals with the case of correlated memories, such as the pictures from a MNIST data-set and the paramagnetic configurations of a 2-dimensional Ising model.
Section 2.5 follows up with the evaluation of another type of data, i.e. spatial maps in a D-dimensional Euclidean space. The performance of the Unlearning algorithm is yet again compared to the one of a SVM in terms of robustness and diffusive dynamics in the real Euclidean space.
To end up, a sampling procedure for optimal noisy training data in the case of maximal noise is proposed and implemented in section 2.6, proving to outperform both the training-with-noise and the HU learning procedures.

2.1 Training with noise

The concept of learning from noisy examples, introduced for the first time in [67], is at the basis of a study performed by Gardner and co-workers [68], a pioneering attempt to increase and control robustness through the introduction of noise during the training phase of recurrent neural networks. Here, we report the algorithm and characterize, for the first time, its performance over fully connected neural networks. All the observables used for this purpose have been defined in section 1.1.
The training-with-noise (TWN) algorithm [68] consists in starting from any initial coupling matrix $J_{ij}^{(0)}$ with null entries on the diagonal, and updating recursively the couplings according to

$$\delta J_{ij}^{(d)} = \frac{\lambda}{N} \epsilon_i^{\mu_d} \xi_i^{\mu_d} S_j^{\mu_d}, \qquad \delta J_{ii}^{(d)} = 0 \quad \forall i, \tag{2.1}$$

where λ is a small learning rate, $\mu_d \in [1, ..., p]$ is a randomly chosen memory index and the mask $\epsilon_i^{\mu_d}$ is defined as

$$\epsilon_i^{\mu_d} = \frac{1}{2}\Big(1 - \operatorname{sign}\Big(\xi_i^{\mu_d} \sum_{k=1}^{N} J_{ik} S_k^{\mu_d}\Big)\Big). \tag{2.2}$$

In this setting, $\vec{S}^{\mu_d}$ is a noisy memory, generated according to a Bernoulli process

$$P(S_i^{\mu_d} = x) = \frac{(1+m_t)}{2}\delta(x - \xi_i^{\mu_d}) + \frac{(1-m_t)}{2}\delta(x + \xi_i^{\mu_d}). \tag{2.3}$$

The *training overlap* m_t is a control parameter for the level of *noise* injected during training, corresponding to the expected overlap between $\vec{S}^{\mu_d}$ and $\vec{\xi}^{\mu_d}$, i.e.

$$m_t = \frac{1}{N} \sum_{j=1}^{N} \xi_j^{\mu_d} S_j^{\mu_d} + O\Big(\frac{1}{\sqrt{N}}\Big). \tag{2.4}$$

Each noisy configuration can be expressed in terms of a vector of *noise units* $\vec{\chi}$, such that

$$S_i^{\mu_d} = \chi_i^{\mu_d} \xi_i^{\mu_d}. \tag{2.5}$$

In this setting, noise units are i.i.d variables, distributed according to

$$P(\chi_i^{\mu_d} = x) = \frac{(1+m_t)}{2}\delta(x-1) + \frac{(1-m_t)}{2}\delta(x+1). \tag{2.6}$$

The algorithm would converge when every configuration with overlap m_t with a memory generates on each site a local field aligned with the memory itself. Let us define the function

$$\mathcal{L}(m, J) = -\frac{1}{\alpha N^2} \sum_{i,\mu}^{N,p} \operatorname{erf}\left(\frac{m \Delta_i^{\mu}}{\sqrt{2(1-m^2)}}\right). \tag{2.7}$$

Calculations contained in appendix A prove that $-\mathcal{L}(m = m_0, J)$ is equal to the one-step retrieval map for one realization of J and averaged over all the configurations having an overlap m_0 with the memories, that we refer to as $m_1(m_0, J)$. Wong and Sherrington [65, 66] propose an elegant analysis of a network designed to optimize $\mathcal{L}(m, J)$, i.e. whose couplings $J_{WS}(m)$ correspond to the global minimum of $\mathcal{L}(m, J)$. Some of their findings, relevant to this work, are:

1. For any m_0, the maximum value of $m_1(m_0, J_{WS}(m))$ with respect to m is obtained when $m = m_0$. This result is important because it suggests that a network with $J = J_{WS}(m)$ tends to increase the influence of the memory over the surrounding configurations, possibly increasing the basins of attraction to which they belong.

2. When $m \to 1^-$, the minimization of the $\mathcal{L}(m, J)$ trains a linear perceptron with maximal stability, i.e. a SVM. This result translates into having $J_{WS}(m \to 1^-) = J_{SVM}$. This result is not trivial, since $m = 1$ would reproduce a linear perceptron with zero margin.

3. When $m \to 0^+$, the minimization of $\mathcal{L}(m, J)$ leads to a Hebbian connectivity matrix. This result translates into having $J_{WS}(m \to 0^+) \propto J^H$. On the other hand, the case $m = 0$ would be trivial, since no learning would be possible.

Now, we want to examine the TWN procedure defined above in light of the results obtained by Wong and Sherrington. When the network is trained through TWN, the resulting coupling matrix depends on m_t, i.e. $J = J(m_t)$. It is crucial to stress the difference between the variables m and m_t: the former is a parameter of eq. (2.7), the latter is the level of noise used by the training algorithm (2.1). Eq. 2.7 is relevant to the TWN procedure, since eq. (2.1) leads to a reduction of $\mathcal{L}(m, J(m_t))$, for any value of m and m_t. In fact, considering a small variation of the stabilities induced by the algorithm update

$$\Delta_i^\mu \to \Delta_i^\mu + \delta\Delta_i^\mu,$$

and performing a Taylor expansion of (2.7) at first order in $\mathcal{O}(N^{-1/2})$, one obtains (see Appendix B.1)

$$\mathcal{L}' = \mathcal{L} + \sum_{i=1}^{N} \delta\mathcal{L}_i \tag{2.8}$$

where

$$\delta\mathcal{L}_i = -\frac{\epsilon_i^{\mu_d}\lambda}{\alpha\sigma_i N^{5/2}} \frac{\sqrt{2}m \cdot m_t}{\sqrt{\pi(1-m^2)}} \exp\left(-\frac{m^2 {\Delta_i^{\mu_d}}^2}{2(1-m^2)}\right). \tag{2.9}$$

Hence, $\delta\mathcal{L}_i$ is strictly non-positive when $\frac{\lambda}{N}$ is small, so that the Taylor expansion is justified. Moreover, we numerically find that iterating (2.1) with a given value of m_t drives $\mathcal{L}(m, J(m_t))$ to its theoretical absolute minimum computed in [66], as reported in fig. 2.1 for one choice of N, α and m. This means that the performance of the TWN algorithm can be completely described in the analytical framework of [66] and, that $\mathcal{L}(m, J(m_t))$ can be considered as the Loss function optimized by the TWN algorithm. As a technical comment, note that standard deviation along one row of the couplings matrix σ_i (see eq. (1.3)) is a variable quantity over time, and numerics suggest that it is slowly decreasing. As a result, the expansion performed to determine the variation of $\mathcal{L}$ (see eq. (B.2) in Appendix B.1) might not be justified after a certain number of steps, leading to a non-monotonic trend of the Loss function. The non-monotonic trend of $\mathcal{L}(m, J(m_t))$ due to this effect is showed in the inset of fig. 2.1. However, this inconvenience can be overcome by rescaling the learning rate λ into $\lambda_i = \lambda \cdot \sigma_i$ at each iteration, as also the curves in fig. 2.1 display.

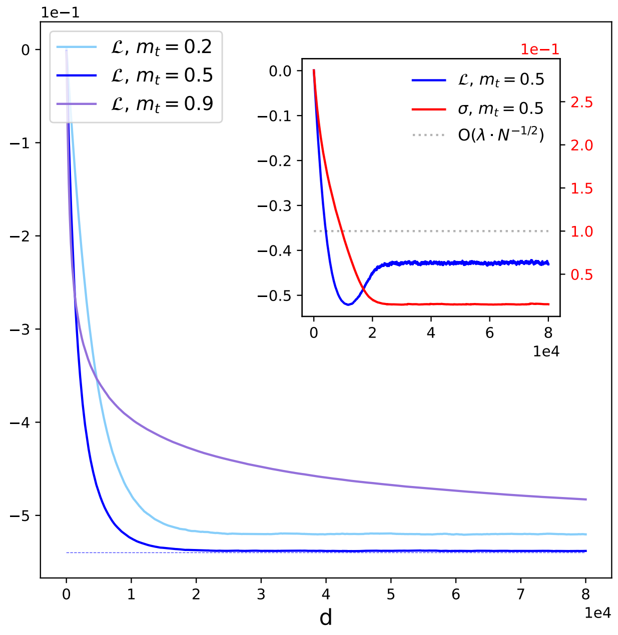


Figure 2.1. The lines in the main plot report the function $\mathcal{L}(m = 0.5, J(m_t))$ for different training overlaps as functions of the number of algorithm steps d. The *dotted* line represents the theoretical minimum value from [66]. The learning strength λ has been rescaled by the standard deviation of the couplings as described in the text. The subplot reports the case $m_t = 0.5$ when the learning strength is not rescaled: $\mathcal{L}$ is in *blue*, while a measure of the standard deviation of the couplings, defined as $\sigma = \frac{1}{N}\sum \sigma_i$, is reported in *red*. The value $\lambda \cdot N^{-1/2}$ of the standard deviation is also depicted in *light gray* to properly signal the moment when equation (2.9) loses its validity. All measures are averaged over 5 realizations of the couplings J. Choice of the parameters: $N = 100$, $\alpha = 0.3$, $\lambda = 1$, the initial couplings are Gaussian with unitary mean, zero variance and $J_{ii}^{(0)} = 0 \ \forall i$.

2.1.1 Perfect-retrieval

The computations contained in [66], and resumed in appendix C, are now used to calculate n_{SAT} as a function of m_t and α in the TWN problem. The probability distribution function of the stabilities in the trained network (see equation (C.2)) has always a tail in the negative values, implying that perfect-retrieval is never reached. The only exception to this statement is the trivial case of $m_t = 1^-$, where $n_{SAT} = 1$ for $\alpha \leq 2$. Nevertheless, the values of n_{SAT} remain close to unity for relatively high values of m_t and relatively low values of α (see fig. 2.2).

2.1.2 Robustness

The robustness properties of a network trained through TWN are now discussed. The color map in fig. 2.3 reports the estimate of the retrieval map $m_f(m_0)$ at $m_0 = 1$ in the limit $N \to \infty$, i.e. a measure of the distance between a given memory and the

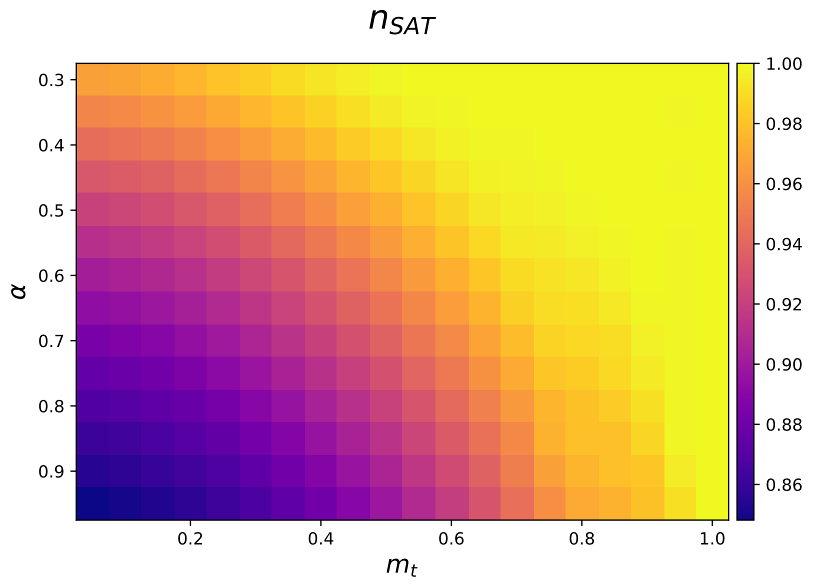


Figure 2.2. n_{SAT} as a function of m_t and α. Warmer shades of colour are associated to higher retrieval performances.

closest attractor. Notice the emergent separation between two regions: one where $m_f(1)$ is mostly smaller than 0.5, and memories are far from being at the bottom of the basin; another region where $m_f(1)$ is mostly close to unity, i.e. the memory is very close to the center of the basin. Such separation reminds the typical division between *retrieval* and *non retrieval* phases in fully connected neural networks [12], differently from sparse neural networks [66, 69] where the possible topologies of the basins result more various yet harder to get measured by experiments. In appendix D we propose an empirical criterion to separate these two regions and so limit ourselves to the retrieval one. Consider the retrieval map $m_f(m_0)$ measured with respect to the attractor of the basin to which the memory belongs and not to the memory itself. Then one must have $m_f(1) = 1$. Our criterion is based on assuming that $m_f(m_0)$ always develops a plateau starting in $m_0 = 1$ and ending in some $m_c < 1$ when $N \to \infty$. The behavior of the basin radius can be observed numerically as a function of m_t: when the plateau disappears (i.e. $m_c = 1$) then one can suppose that basins get shattered in the configurations space due to the interference with the other attractors. Given α, this occurs at some value of m_t. The empirical transition line is reported in a dashed style in fig. 2.3. Limiting ourselves to the retrieval region we employ a procedure also described in appendix D to compute the typical size of the basins of attraction. White dots in fig. 2.3 signal the combinations of (m_t, α)

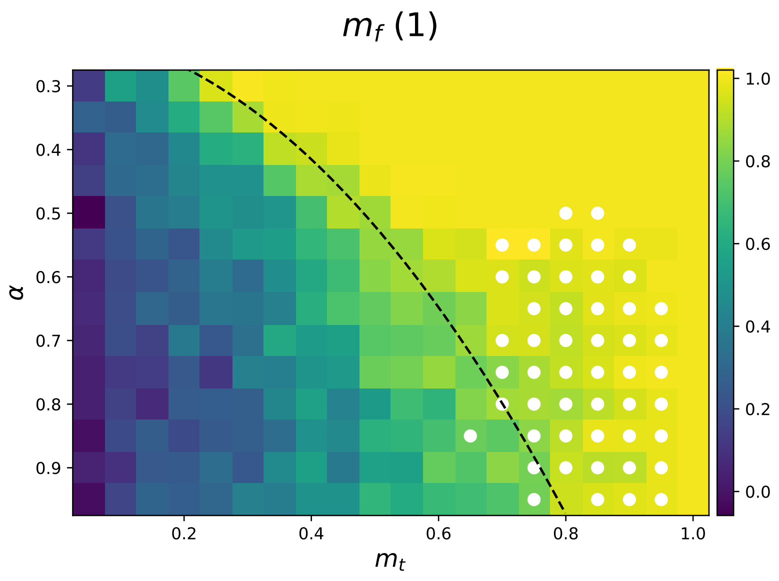


Figure 2.3. $m_f(1)$ as a function of m_t and α. Warmer shades of color are associated to higher retrieval performances. The *black dashed* line represents the boundary of the retrieval regime according to the criterion in appendix D, *white* dots signal the points where basins of attraction to which memories belong are larger than those obtained from a SVM at $N = 200$.

where the basins of attraction found by TWN algorithm resulted larger than the ones obtained by a SVM at the same value of α. We want to stress the importance of a comparison between the TWN and the corresponding SVM, since numerical investigations have shown the latter to achieve extremely large basins of attraction, presumably due to the maximization of the stabilities [44, 21]. One can conclude that for most of the retrieval region the robustness performance is worse than the SVM, which maintains larger basins of attraction; on the other hand, at higher values of α the trained-with-noise network sacrifices its perfect-retrieval property to achieve a basin that appears wider than the SVM one. In conclusion, the TWN algorithm never outperforms the relative SVM without reducing its retrieval capabilities.

Checkpoint

In this section we have seen that:

- The TWN algorithm can be interpreted as a linear perceptron that learns to align noisy versions of one memory to the memory itself, enhancing associativity. The control parameter of the algorithm is the training overlap m_t.

- The TWN algorithm trains a network that solves the optimization problem studied by Wong and Sherrington. Hence, by tuning the training overlap one can interpolate between the Hebbian model (i.e. $m_t = 0^+$) and a SVM (i.e. $m_t = 1^-$).

- The TWN never reaches perfect retrieval when $m_t < 1^-$. In all this region of the network configurations the TWN algorithm never outperforms the robustness performance of a SVM without moving each memory away from the closest attractor.

2.2 Optimal noisy data-points

As previously stated, SVMs are considered to be highly efficient associative memory models, due to their very good classification and generalization capabilities. Wong and Sherrington's analytical argument proves that minimization of $\mathcal{L}(m = 1^-, J(m_t))$ trains a SVM. The TWN procedure proposed by Gardner and collaborators, relying on a Bernoulli process to generate noise, accomplishes this task only when $m_t = 1^-$ (see section 2.1). Injecting a larger amount of noise during training would deteriorate the performance: specifically, $m_t = 0^+$ trains a Hebbian neural network. Nevertheless, training a network with examples that are nearly uncorrelated with the memories can significantly speed up the sampling process, since such states can be generated in an unsupervised fashion, without knowledge of the memories (as seen for HU in section 1.1.2). Such unsupervised processes are also considered more biologically plausible.
In this section, we show that it is possible to use maximally noisy configurations (i.e. $m_t = 0^+$) to train a network approaching the performance of a SVM, by means of the TWN algorithm. For this purpose, one must change the way noisy data are generated: they need to meet specific constraints which lead to internal dependencies among the features (i.e. a *structure*). We derive a theoretical condition characterizing the optimal structure of noise, and show that specific configurations in the Hebbian energy landscape, including local minima, match well the theoretical requirements.

It will be helpful for our purposes to implement a symmetric version of rule (2.1), i.e.

$$\delta J_{ij}^{(d)} = \frac{\lambda}{N}\left(\epsilon_i^{\mu_d}\xi_i^{\mu_d}S_j^{\mu_d} + \epsilon_j^{\mu_d}\xi_j^{\mu_d}S_i^{\mu_d}\right). \tag{2.10}$$

Equation (2.10) can be rewritten explicitly making use of (2.2), leading to

$$\begin{aligned}\delta J_{ij}^{(d)} =& \frac{\lambda}{2N}(\xi_i^{\mu_d}S_j^{\mu_d} + S_i^{\mu_d}\xi_j^{\mu_d}) + \\ &-\frac{\lambda}{2N}(S_i^{1,\mu_d}S_j^{\mu_d} + S_i^{\mu_d}S_j^{1,\mu_d})\end{aligned} \tag{2.11}$$

where $S_i^{1,\mu_d} = \text{sign}\left(\sum_{k=1}^N J_{ik}S_k^{\mu_d}\right)$. The total update to the coupling at time D can be decomposed as a sum of two contributions

$$\Delta J_{ij}(D) = \Delta J_{ij}^N(D) + \Delta J_{ij}^U(D). \tag{2.12}$$

The first term on right-hand side, which will be referred to as *noise* contribution, is expressed in terms of noise units as

$$\Delta J_{ij}^N(D) = \frac{\lambda}{2N}\sum_{d=1}^{D}\xi_i^{\mu_d}\xi_j^{\mu_d}\chi_j^{\mu_d} + \frac{\lambda}{2N}\sum_{d=1}^{D}\xi_j^{\mu_d}\xi_i^{\mu_d}\chi_i^{\mu_d}, \tag{2.13}$$

while the second term, which will be referred to as *unlearning* contribution, is given by

$$\Delta J_{ij}^U(D) = -\frac{\lambda}{2N}\sum_{d=1}^{D}\left(S_i^{1,\mu_d}S_j^{\mu_d} + S_i^{\mu_d}S_j^{1,\mu_d}\right). \tag{2.14}$$

In the maximal noise case $m_t = 0^+$, $\Delta J_{ij}^N(D)$ averages to zero over the process because its variance is λ/N when the number of steps D is proportional to N/λ, leading to

$$\Delta J_{ij}^N(D) = 0^+ + \mathcal{O}\left(\sqrt{\frac{\lambda}{N}}\right). \tag{2.15}$$

2.2.1 Characterizing the good training configurations

The variation of $\mathcal{L}(m, J)$ can be expressed (see Appendix B.2) as

$$\delta\mathcal{L} = \delta\mathcal{L}_N + \delta\mathcal{L}_U.$$

As detailed in Appendix B.2, in the case of maximal noise (i.e. $m_t = 0^+$) $\delta\mathcal{L}_N$ is negligible, and the only relevant contribution is

$$\delta\mathcal{L}_U \propto \frac{m}{\sqrt{2\pi(1-m^2)}} \sum_{i,\mu}^{N,p} \omega_i^\mu \exp\left(-\frac{m^2 {\Delta_i^\mu}^2}{2(1-m^2)}\right), \tag{2.16}$$

where ω_i^μ play the role of weights to the positive Gaussian terms and they are given by

$$\omega_i^\mu = \frac{1}{2\sigma_i}\left(m_\mu \chi_i^{1,\mu} + m_{1,\mu}\chi_i^\mu\right), \tag{2.17}$$

with

$$\chi_i^\mu = \xi_i^\mu S_i^{\mu_d} \qquad \chi_i^{1,\mu} = \xi_i^\mu S_i^{1,\mu_d}, \tag{2.18}$$

and

$$m_\mu = \frac{1}{N}\sum_{j=1}^N S_j^{\mu_d}\xi_j^\mu \qquad m_{1,\mu} = \frac{1}{N}\sum_{j=1}^N S_j^{1,\mu_d}\xi_j^\mu. \tag{2.19}$$

For the case of fixed points, we have $\omega_i^\mu = m_\mu \chi_i^\mu$. We know that minimization of $\mathcal{L}(m, J(m_t))$ trains a SVM when $m \to 1^-$. When $m \to 1^-$, the Gaussian terms contained in the sum become very peaked around 0. Since we want $\delta\mathcal{L}$ to be negative, we need, for most of the pairs i, μ,

$$\omega_i^\mu < 0 \text{ when } |\Delta_i^\mu| < 0^+. \tag{2.20}$$

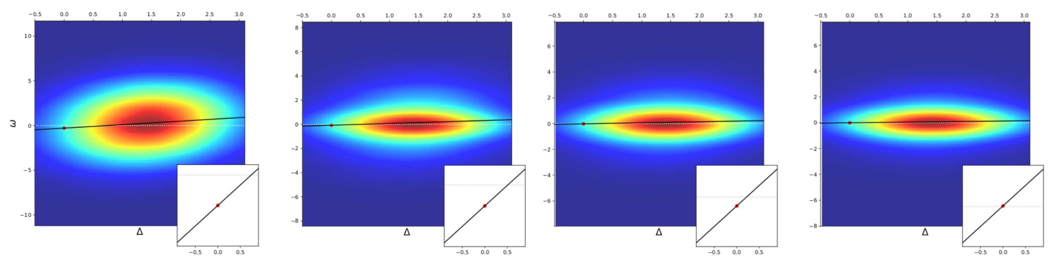
The more negative ω_i^μ is when $\Delta_i^\mu \sim 0$, the more powerful is its contribution to approach the SVM performances. Selecting training data that satisfy equation (2.20) amounts to imposing specific internal dependencies among the noise units $\vec{\chi}$, which are no more i.i.d. random variables, as it was in [68]. We refer to such dependencies as *structure* of the noise. One should also bear in mind that training is a dynamic process: to reduce $\mathcal{L}(m = 1^-, J(m_t = 0^+))$ condition (2.20) should hold during training.

2.2.2 Position of good training configurations in the energy landscape

The HU algorithm is based on choosing training configurations on a dynamical basis, namely as fixed points of zero temperature dynamics in the energy landscape dictated by Hebb's learning rule. On the other hand, traditional TWN relies on fully random states, i.e. very high states in the energy landscape. In this section, we generalize this dynamical approach by evaluating the performance of training configurations which lie at different altitudes in the energy landscape of a symmetric neural network, i.e. states that are not limited to be stable fixed points or random states. To do so, we sample training configurations by means of a Monte Carlo routine at temperature T. Temperature acts as a control parameter: when $T = 0$ training configurations are stable fixed points of eq. (1.1), as in standard HU. Higher values of T progressively reduce the structure of noise in training configurations, and in the limit $T \to \infty$, training configurations are the same as in the TWN algorithm. The Monte Carlo of our choice is of the Kawasaki kind [70], to ensure that all training configurations are at the prescribed overlap $m_t = 0^+$. We are going to use this technique to probe the states across two types of landscapes: the one resulting from a Hebbian initialization and the one resulting from a SK model [71].

Regarding the Hebbian initialization of the network, numerical results are reported in fig. 2.4 for four different temperatures. Each panel shows the distribution of $(\omega_i^\mu, \Delta_i^\mu)$. Data points are collected over fifteen realizations of the network, then plotted and smoothed to create a density map. We are interested in the *typical* behavior of ω_i^μ when $\Delta_i^\mu \sim 0$, which can be estimated by a linear fit of the data. We consider the intercept of the best fit line as an *indicator* $\omega_{emp}(0)$ of the typical value of ω_i^μ around $\Delta_i^\mu = 0$. We find that at lower temperatures the sampled configurations favor both perfect-retrieval and robustness because $\omega_{emp}(0)$ is more negative. As temperature becomes too high, $\omega_{emp}(0)$ gets closer to zero, suggesting low quality in terms of training performance.

One can also study how the distribution of $(\omega_i^\mu, \Delta_i^\mu)$ evolves during the training process. Fig. 2.5a shows the value of $\omega_{emp}(0)$ at different time steps of the training process, for different values of α, when configurations at $T = 0$ are given to the algorithm. We find that $\omega_{emp}(0) < 0$ for $\alpha \leq 0.6$. The progressive increase of $\omega_{emp}(0)$ means that the structure of the fixed points is more effective in the starting Hebbian landscape compared to intermediate stages of training. In the last part of training, points reacquire more negative values, but this is not a reliable indication of good performance: as shown in fig. 2.5c, in this part of the process the standard deviation of the couplings σ_i is comparable to $\mathcal{O}(\lambda \cdot N^{-1/2})$, and the expansion of the $\mathcal{L}$ in eq. (2.16) is not valid. The last part of the training, where $\sigma_i \simeq 0 \ \forall i$, has been neglected from the plot. The experiment is thus consistent with the characterization of the HU algorithm presented in [21], which showed decreasing perfect-retrieval and robustness when increasing α. This is confirmed by the study of the Pearson correlation coefficient between ω_i^μ and the associated stabilities Δ_i^μ (see fig. 2.5,b). High values of the Pearson coefficient show a strong dependence of the structure of noise on the relative stabilities. For all α, the Pearson coefficient is highest at $d = 0$, and progressively decreases during training, suggesting that the quality of



(a) $T = 0$ **(b)** $T = 0.5$ **(c)** $T = 1$ **(d)** $T = 8$

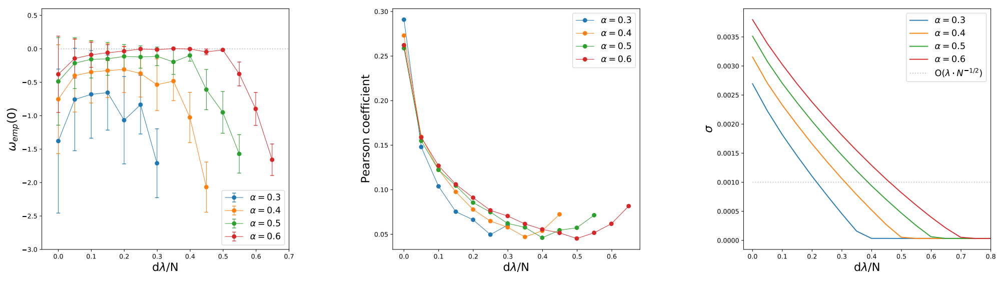
Figure 2.4. Distribution of ω_i^μ as a function of Δ_i^μ for training configurations sampled with a Monte Carlo at temperature $T = 0$ i.e. stable fixed points only (a), $T = 0.5$ (b), $T = 1$ (c), $T = 8$ (d), on a Hebbian network. Warmer colors represent denser region of data points. The *full black* line is the non-weighted best fit line for the points, the *dotted white* line represents $\omega = 0$, the *red dot* is the value of the best fit line associated with $\Delta = 0$. Sub-panels to each panel report a zoom of the line around $\Delta = 0$. Measures have been collected over 15 samples of the network. Choice of the parameters: $N = 500$, $\alpha = 0.5$.

the training configurations is deteriorating. The final increase in the coefficient is, again, due to the vanishing of the standard deviations σ_i of the couplings, and does not indicate good performance.

For a comparison, one can study the distribution of ω_i^μ in the case of a random initialization of the coupling matrix J. We chose the Sherrington-Kirkpatrick (SK) model [71] as a case of study. Panels (a) and (b) in fig. 2.6 report the smoothed distribution of $(\omega_i^\mu, \Delta_i^\mu)$ showing a different scenario with respect to the Hebbian one. The distribution looks anisotropic, as in the Hebbian case, yet the stabilities are centered Gaussians, so $\omega_{emp}(0)$ is positive. In particular, things appear to improve when T increases, in contrast with the previous case of study, though in accordance with the Hebbian limit of the TWN algorithm explained in section 2.1. Panel (c) displays more clearly the mutual dependence between ω_i^μ and stabilities Δ_i^μ by reporting the Pearson coefficient at the various T between these two quantities: both Hebb's and SK show some mutual dependence, but the distribution in the Hebbian landscape is way more skewed. Furthermore, panel (d) shows $\omega_{emp}(0)$ in both cases. This measure is consistent with the indication coming from the Pearson coefficient: while for the Hebb's initialization $\omega_{emp}(0)$ remains significantly negative and reaches the lowest values at low temperatures, the random case shows the opposite trend, with the estimated $\omega_{emp}(0)$ staying generally close to zero.

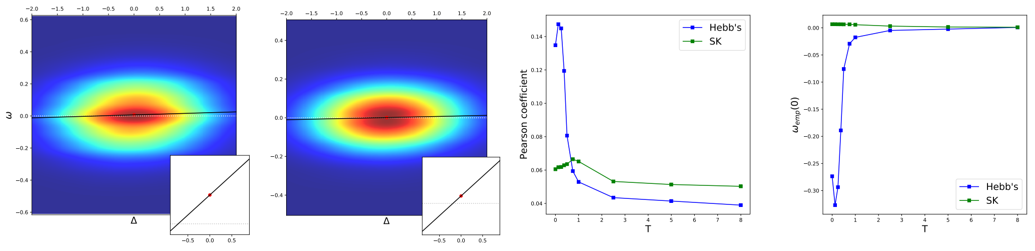
2.2.3 The role of saddles

Notice, from both panels (c), (d) of fig. 2.6, the existence of an optimum which does not coincide with the stable fixed points of the dynamics (i.e. $T = 0$). As noted in previous studies on spin glasses [72], one can associate configurations probed by a Monte Carlo at finite temperatures with configurations which are typically saddles in the energy landscape with a given *saddle index*. The *saddle index* is defined as the ratio between the number of unstable sites under the dynamics (see eq. (1.1)) and the total number of directions N. Stable fixed points have $f = 0$, while random



(a) **(b)** **(c)**

Figure 2.5. The TWN algorithm is implemented by sampling stable fixed points of the network dynamics with $m_t = 0^+$. (a) The empirical measure of ω_i^μ around $\Delta_i^\mu = 0$ for the case of stable fixed points as a function of the rescaled number of iterations of the learning algorithm. Errorbars are given by the standard deviations of the measures. (b) Pearson coefficient measured between ω_i^μ and Δ_i^μ. (c) The standard deviation of the couplings during learning, defined as $\sigma = \frac{1}{N}\sum \sigma_i$. Points are averaged over 50 samples and the choice of the parameters is: $N = 100$, $\lambda = 10^{-2}$.



(a) T = 0 **(b)** T = 8 **(c)** **(d)**

Figure 2.6. (a), (b): Distribution of ω_i^μ as a function of Δ_i^μ for training configurations sampled with a Monte Carlo at temperature $T = 0$ i.e. stable fixed points only (a), and $T = 8$ (b) on a SK model. Warmer colors represent denser region of data points. The *full black* line is the non-weighted best fit line for the points, the *dotted white* line represents $\omega = 0$, the *red spot* is the value of the best fit line associated with $\Delta = 0$. Sub-panels to each panel report a zoom of the line around $\Delta = 0$. (c), (d): Comparison between the Hebbian initialization and the Random one through evaluation of: the Pearson coefficient between ω_i^μ and Δ_i^μ (c) and the estimated value of $\omega_{emp}(0)$ from the dispersion plots (d). Measures have been collected over 15 samples of the network. Choice of the parameters: $N = 500$, $\alpha = 0.5$.

configurations are expected to have $f = 1/2$. In order to check whether a particular f is capturing relevant features of the virtuous training configurations, we sampled training data according to the requirement that their saddle fraction assumes a specific value f and $m_t = 0^+$. Saddles are then employed for training the network according to eq. (2.1). Sampling is performed by randomly initializing the network on a configuration having training overlap $m_t = 0^+$ with a reference memory, and performing a zero temperature dynamics on the landscape defined by the energy

$$E(\vec{S}|f,J) = \frac{1}{2}\left(\frac{1}{N}\sum_{i=1}^{N}\Theta(S_i\sum_{j=1}^{N}J_{ij}S_j) - f\right)^2 , \tag{2.21}$$

where $\Theta(x)$ is the Heaviside function. Yet again, the value of m_t was maintained constant during the descent. The left panel in fig. 2.7 shows how the minimum stability evolves during the training process while a symmetric TWN algorithm is initialized in the Hebbian matrix and learns saddles of different indices. For a network of $N = 100$ and $\alpha = 0.35$, we found that perfect-retrieval is reached until a certain value of f, suggesting that saddles belonging to this band are indeed good training data. The band of saddles that are suitable for learning is reduced when α increases until such states do not significantly satisfy eq. (2.20) anymore. Such limit capacity is located around the critical one for HU. It should be stressed that the precise performance as a function of f is quite sensitive to the sampling procedure. Simulated annealing routines [73] have also been employed to minimize (2.21), obtaining qualitatively similar results yet not coinciding with the ones reported in fig. 2.7. A qualitative study of the basins of attraction of the network has been performed and reported in the right panel in fig. 2.7. In particular, the retrieval map $m_f(m_0)$ has been measured relatively to the saddle indices f at the first time they reached perfect-retrieval, in analogy to what has been measured in [21]. The curves coincide quite well, suggesting that finite sized networks trained with different f assume similar volumes of the basins of attraction when they are measured at the very first instant they reach perfect-retrieval. The plot also shows that the robustness performance is comparable with the one of a SVM trained with the same choice of the control parameters.

2.2.4 Going unsupervised

It is essential to note that, since the training overlap under consideration is close to 0, stable fixed points and saddles in their proximity can be sampled in a full unsupervised fashion: when the dynamics (e.g. zero temperature Monte Carlo or gradient descent over (2.21)) is initialized at random, which implies having an overlap $m_t = 0^+$ with some memory, it will typically conserve a small overlap with the same memory even at convergence [29]. As a consequence, the argument presented above holds, and a supervised algorithm as TWN can be reduced to a more biologically plausible and faster unsupervised learning rule. This aspect will be deepened by the next section, where we show a particular scenario where TWN coincides with the HU rule, a fully unsupervised learning procedure [17].

Checkpoint

In this section we have seen that:

- While the standard TWN employed random noise to train a SVM by tuning $m_t = 1^-$, one can approach a SVM using structured noise with $m_t = 0^+$. At each step of the TWN algorithm, the best training data are the ones satisfying condition (2.20). Maximizing the noise is useful because it can be sampled

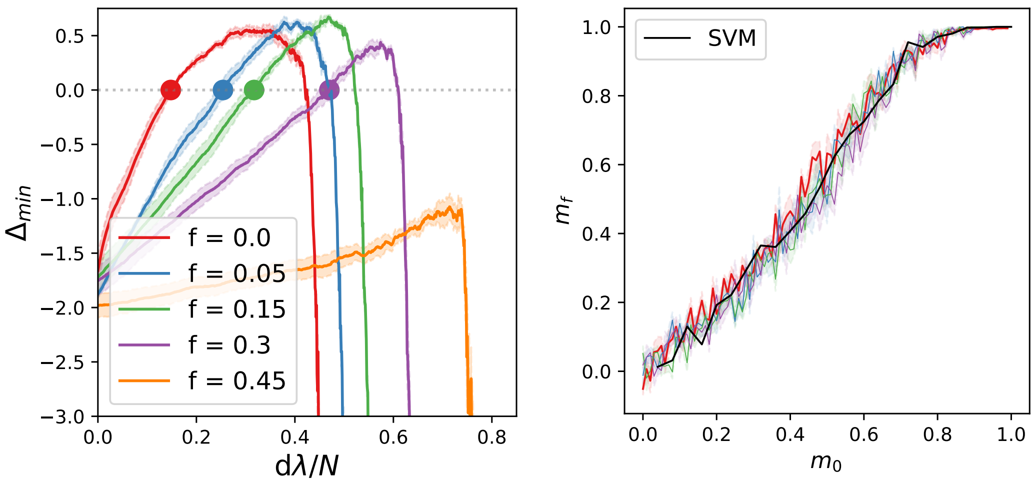


Figure 2.7. Left: Minimum stability Δ_{min} as a function of the algorithm steps on a network trained with the symmetric TWN routine that learns saddles of various indices f. The initial matrix is assembled according to the Hebb's rule. Full dots report the amount of iterations needed to accomplish perfect-retrieval. Right: the retrieval map $m_f(m_0)$ as measured on the positions of the colored dots from the right panel, with the same color code being used. A comparison with a SVM trained with the same choice of the parameters is also presented through the *dashed blue* line. All measures are averaged over 5 samples with the shaded region indicating the experimental errors. The choice of the parameters is: $N = 100$, $\alpha = 0.35$, $\lambda = 10^{-3}$.

in an unsupervised manner, i.e. by initializing the dynamics on a random network state and then descending the energy function.

- While the energy landscape of random networks contain few good training data, a Hebbian landscape contains many of these states in form of local minima and surrounding saddle points. A Hebbian initialization of the couplings is thus a good starting point for the TWN algorithm.

2.3 Quasi-optimality of the Unlearning algorithm

As shown in section 2.2, local minima of the Hebbian energy landscape are good training configurations, yet they are necessarily not the optimal ones (see fig. 2.6). In this section, we analyze the performance of TWN when training data are minima in the energy landscape characterized by an overlap m_t with the memories. Local minima in the energy landscape are convenient training data, since they can be easily and efficiently sampled by the asynchronous dynamics in eq. (1.1). After initializing the couplings according to the Hebbian rule, we will study two different scenarios: we firstly evaluate training in the maximal noise case $m_t = 0^+$, when stable fixed points can be sampled in a unsupervised fashion, i.e. by the network dynamics when started on a fully random state. In this setting, where the considerations of section 2.2.1 apply, we show that the TWN algorithm converges to the traditional HU rule in the small λ/N limit. Then, we examine in detail training with finite overlap (i.e. $m_t > 0^+$), providing an estimate of the critical capacity reached by the neural network, and showing that results from the previous section about the effectiveness of stable fixed points can be generalized to this case.

As mentioned in sec. 2.2, in this case the only relevant contribution to the update rule is

$$\Delta J^U_{ij}(D) = -\frac{\lambda}{N}\sum_{d=1}^{T} S_i^{\mu_d} S_j^{\mu_d}, \tag{2.22}$$

which is the classic HU update rule. As a result, when $\lambda/N \to 0$ the TWN algorithm and the HU algorithm will converge to the same updating rule for the couplings when stable fixed points of the dynamics are used in the training. The same argument can be applied to the original asymmetric rule (2.1), however asymmetric networks may have no stable fixed points of (1.1) that can be easily reached and employed in the learning.

We now perform a numerical test of the argument above, in the case of a symmetric connectivity matrix. At each step of the algorithm, the network is initialized with an initial overlap contained in $(0, N^{-1/2})$ with one memory $\vec{\xi}^{\mu_d}$. Then, asynchronous dynamics (1.1) is run until convergence, and the final overlap m_t is measured. If $m_t \in (0, N^{-1/2})$, we use the sampled configuration for training, otherwise the process is repeated. Typically, an initial overlap equal to 0^+ implies a similar order of magnitude for the final overlap, hence no reiteration is usually needed.

The algorithm (2.10) is repeated for $D = \mathcal{O}(N/\lambda)$ steps. The order of magnitude of $\Delta J^U_{ij}(D)$ is supposed to be the same of $J^{(0)}_{ij}$, in order to see significant modifications to the initial connectivity matrix. The network is initialized according to the Hebb's rule (1.8), i.e. $J^{(0)}_{ij} = \mathcal{O}(N^{-1/2})$ which implies $\Delta J^U_{ij}(D) = \mathcal{O}(N^{-1/2})$ at leading order. The contributions U and N are compared by computing the norm of the relative ΔJ matrix and evaluating the ratio $|\Delta J^U|/|\Delta J^N|$. From our previous considerations we expect $|\Delta J^U|/|\Delta J^N|$ to be linear in $\lambda^{-1/2}$ when corrections vanish. Results are reported in fig. 2.8: $|\Delta J^U|/|\Delta J^N|$ grows when N increases and λ decreases, according to the scaling relation predicted by our argument. In addition to this, curves are collapsing on the expected line when $\lambda \to 0$ and $N \to \infty$.

We also measured $\Delta_{\min}$ at its maximum over the course of the algorithm (as described in 2.1.1). Results are reported in fig. 2.9. $\Delta_{\min}$ produced by TWN and HU are found to coincide when λ is sufficiently small. Moreover, the number of steps necessary to reach the maximum are the same for both algorithms, confirming that couplings are transforming at the same way. This last aspect is corroborated by the subplot in fig. 2.9, representing the set of J_{ij} obtained with the traditional HU algorithm as a function of the one resulting from the TWN algorithm, for one realization of the network. The strong correlation is evident, as predicted from our pseudo-analytical arguments.

During the rest of the section we will drop the noisy part of the synaptic update and use the traditional rule

$$\delta J_{ij} = -\frac{\epsilon}{N} S_i^* S_j^*$$

where we renamed λ with ϵ not to confuse it with the learning rate used for the linear perceptron. Specifically the symmetric perceptron (SP) rule with a tunable margin k will be applied (see eq. (1.15), eq. (1.16)). We also substituted the index μ_d with a star since, for the rest of this section, the HU procedure will be implemented in a fully unsupervised fashion.

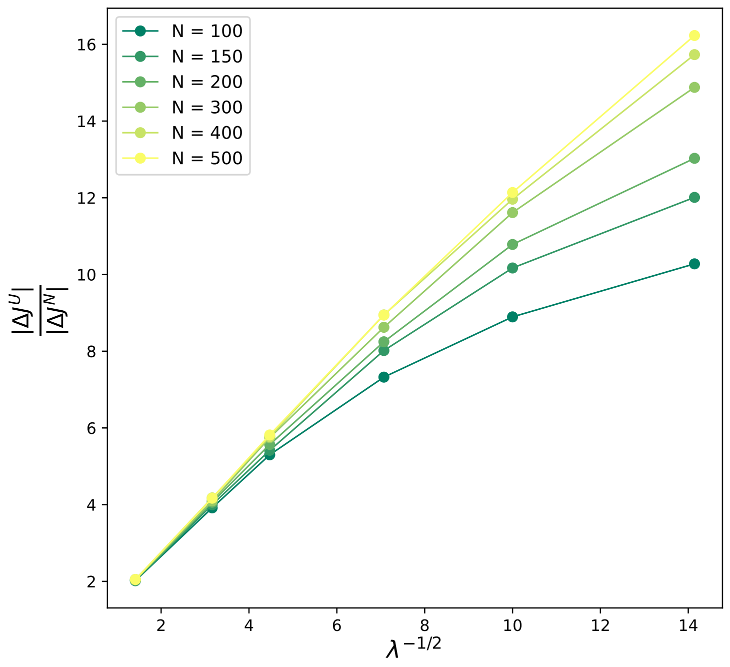


Figure 2.8. Estimates of the ratio $|\Delta J^U|/|\Delta J^N|$ as a function of $\lambda^{-\frac{1}{2}}$ and N for $\alpha = 0.5$. Measures are averaged over 5 samples. Error bars are not indicated because smaller than the symbols.

2.3.1 Robustness performance and basins of attraction

We have compared the performance of SP and HU by measuring the shape of the basins of attraction around each memory. This is done by measuring the retrieval map

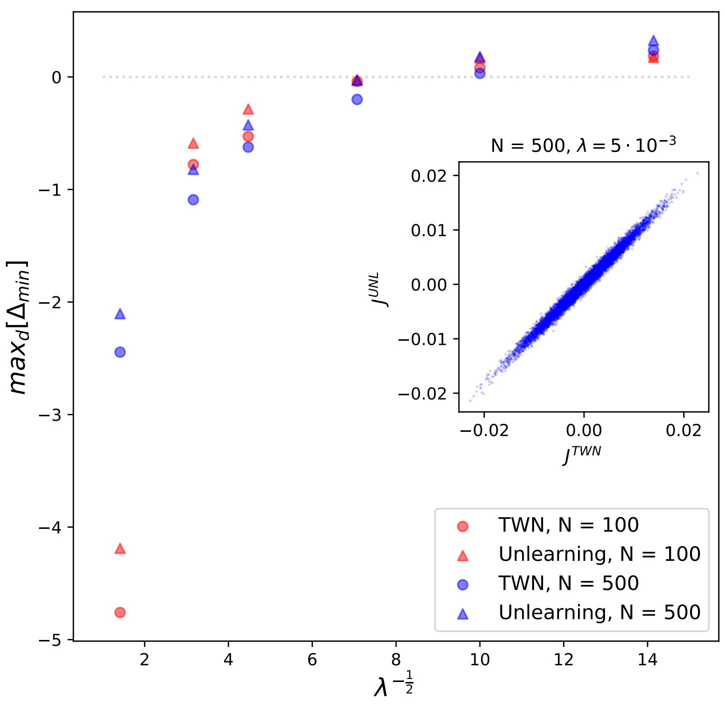


Figure 2.9. The quantity $\max_d(\Delta_{\min})$ as a function of $\lambda^{-1/2}$. Colors are: *red* for $N = 100$ and *blue* for $N = 500$. The *gray* line represents the null value for the stability. Symbols are: *circles* for the TWN rule, *triangles* for the HU rule. In the subplot on the center right, the couplings obtained through the HU algorithm are plotted as a function of the ones resulting from the TWN, at the same amount of iterations, for one sample at $N = 500$ and $\lambda = 5 \cdot 10^{-3}$. Measures are averaged over 50 samples. The choice of the parameters is: $\alpha = 0.5$, $m_t = 0^+$ for TWN.

$m_f(m_0)$. In fig. 2.10 we plot m_f as a function of m_0. Colored dashed curves refer to SP for different values of k, up to the highest k that allows the algorithm to converge in $\mathcal{O}(10^3)$ iterations. This slight underestimations of the real $k_{max}(\alpha)$ bares very little consequences to our results. Related to this, we underline the importance of the choice of λ at a given value of N. This is another crucial topic that is rarely discussed in the literature. Higher values of λ imply larger learning steps, while smaller values are associated to a finer exploration of space of coupling matrices during training. It is observed that the algorithm, operating at $\lambda = \{1, 10^{-1}, 10^{-2}\}$, converges to almost identical matrices already when k is equal to the maximal stability for diluted networks [37] that, according to fig. 1.3, is slightly lower than the actual k_{max}. This suggests that the final state lies very closely to the unique optimal solution even when we are not exactly at k_{max}. Hence, no significant changes are expected in our numerical results when k is pushed further towards its maximal value. On the other hand, when λ assumes smaller values, i.e. $\lambda = \{10^{-3}, 10^{-4}, 10^{-5}\}$, basins are observed to be smaller in size and the volume of solutions is larger, indicating that the final state remains further from the maximal performance. In order to recover the numerical results obtained at a larger λ, one needs to progressively increase k to values that are difficult to reach numerically. As a result, the choice of $\lambda = 1$ in

this section seems to us well justified to reproduce the optimal performance of the symmetric perceptron at $k \simeq k_{max}$.
Consistently with the literature, we find that increasing the stability leads to an increase of m_f at fixed m_0 [44]. In particular, when the stability is equal to zero, the memories, albeit being fixed point of the dynamics, have zero basin of attraction, as indicated by the very low values of m_f for $m_0 \neq 1$. The gray dashed line at the bottom of fig. 2.10 refers to the Hopfield model without unlearning: since $\alpha > \alpha_c^H$ the model does not learn. The colored continuous lines refer to HU, for different amounts of unlearning. More specifically, we measured the performance of the model for the three values $d = D_{in}$, D_{top}, and D_{fin} defined in section 1.1.2. It is clear how Unlearning improves the performance of the network, which is not maximized at $d = D_{top}$ as one could expect [35], but at $d = D_{in}$, where the requirement for perfect-retrieval of the memories is satisfied with zero margin.

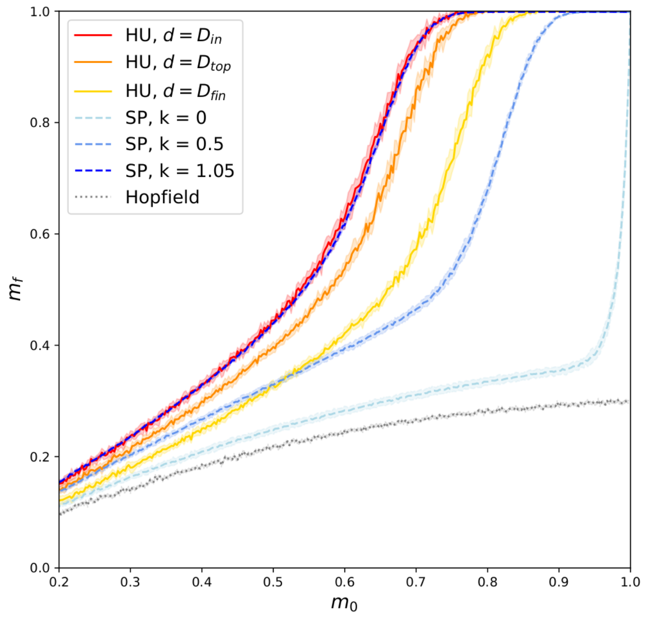


Figure 2.10. The average size of basins of attraction at $N = 800$, $\alpha = 0.4$ for both symmetric perceptron (SP) and Hebbian Unlearning (HU), averaged over several realizations of the disorder. The colored area around each curve represents the statistical errors. Continuous lines are for basins at D_{in}, D_{top} and D_{fin} for HU with $\epsilon = 10^{-2}$. Dashed lines are for three values of k, including the $k \simeq k_{max}$, used for SP with $\lambda = 1$. The gray dotted line represents the performance of the Hopfield model at the same value of α. Notice that attraction basins of the SP and HU almost coincide when the two algorithms operate in their optimal regime, namely $d = D_{in}$ and $k \simeq k_{max}$.

We also found that the performance of HU at $d = D_{in}$ and the one of the SP at $k \simeq k_{max}$ are indistinguishable within our numerical resolution. This is a remarkable fact, since the two algorithms have a radically different structure: the SP algorithm is supervised, i.e. it needs to have access at every step to all the memories that the network needs to memorize, while the HU is not, and only exploits the topology of the spurious states generated by Hebb's prescription. These findings are robust to change in the load α and to finite size effects, as illustrated in fig. 2.11. The mean basin radius at finite N is defined as $1 - m_0$, selecting the value of m_0 below which more than 30% of the memories are reconstructed with more then 5% error. The dots represent our extrapolation of this quantity to the limit $N \to \infty$, for different values of α. The lower dots relative to the SP correspond to $k < k_{max}$, and the value of the mean basin radius gets higher as k is increased up to $k \simeq k_{max}$. Again, one can see that even in the thermodynamic limit, our simulations suggest that in their optimal regime the two algorithms perform essentially in the same way.

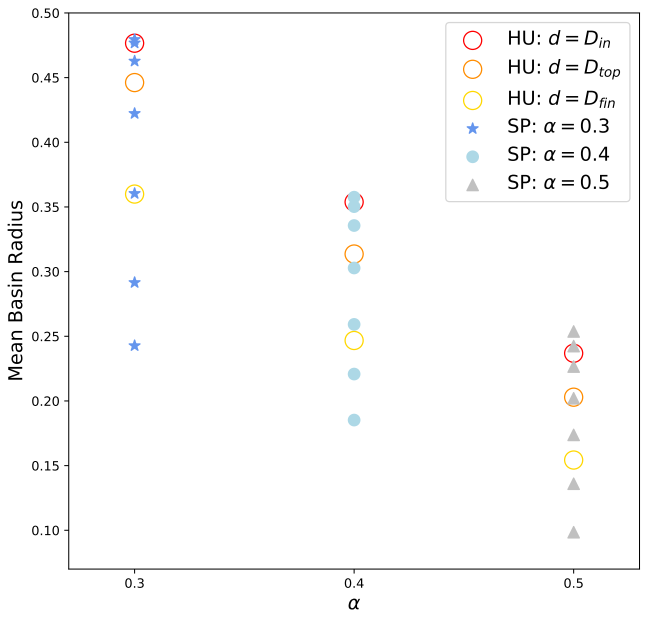


Figure 2.11. Mean attraction basin radius for symmetric perceptron (SP) and Hebbian Unlearning (HU) measured as in [44] and extrapolated to $N \to \infty$, for $\alpha = 0.3, 0.4, 0.5$ and $\lambda = 1$. Points for the SP correspond to the following values of k: $\alpha = 0.3 \to k \in \{0.4, 0.5, 0.7, 0.9, 1.1, 1.296, 1.32\}$; $\alpha = 0.4 \to k \in \{0.4, 0.5, 0.6, 0.75, 0.9, 0.988, 1.05\}$; $\alpha = 0.5 \to k \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.768, 0.85\}$. Error bars are smaller than the symbols size.

2.3.2 Learning paths in the space of the interactions

One way to visualize the solutions of the optimization problem, and the way these solutions are reached by means of the algorithm, is to exploit the space of interactions as conceived by Gardner [15]. Consider a spherical surface in $N(N-1)/2 - 1$ dimensions where each point $\vec{J}$ is a vector composed by the off-diagonal elements $J_{j>i}$ of the connectivity matrix normalized by their standard deviation. These position vectors hence will be

$$\vec{r} = \vec{J}/\sigma_J \ , \tag{2.23}$$

with

$$\sigma_J = \sqrt{\frac{2}{N(N-1)} \sum_{i<j}^{1,N} J_{ij}^2} \ . \tag{2.24}$$

For what concerns the SP, after fixing the value of α and a set of memories, one can imagine the sphere as composed by an UNSAT and a SAT region. These regions are connected sub-spaces of the original sphere, so that one can go from a matrix to another one in a continuous fashion. The SAT region contains the point relative to the unique solution at $k = k_{max}(\alpha)$.

We now define an *overlap* parameter quantifying the covariance of two generic symmetric matrices J_{ij} and U_{ij}

$$q = \frac{2}{N(N-1)} \sum_{i<j}^{1,N} \overline{\frac{J_{ij}U_{ij}}{\sigma_J \sigma_U}} \ , \tag{2.25}$$

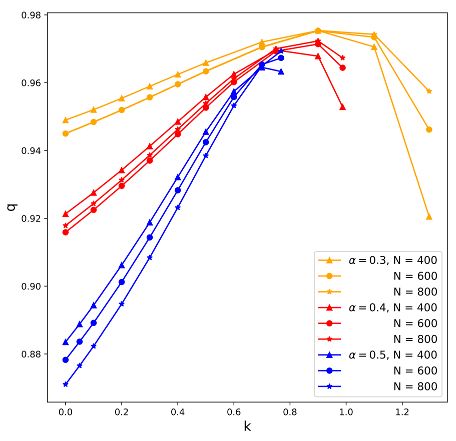
where $\overline{\cdot}$ is the average over the disorder.

We first evaluate the final points where the two algorithms converge in the space of interactions. HU is stopped at $d = D_{in}$ as that is the relevant amount of iterations identified in section 1.1.2. The SP is run at $\lambda = 1$. Fig. 2.12a, displays the overlap between the resulting matrices when the SP is performed at different values of k before reaching $k_{max}(\alpha)$. The plot shows that q increases with k, suggesting that HU pushes the system to the same region of solutions where the SP converges when k is close to k_{max}. Finite size effects evidently appear near the abrupt transition from SAT to UNSAT, but the increase of q with the size of the network suggests that the maximum overlap might be associated to the maximal stabilities when N becomes large enough.

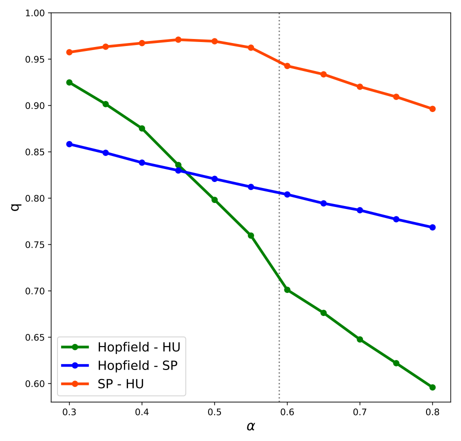
The plot of q as a function of α, see fig. 2.12b, shows how the distance between the final points and the initial Hebbian matrix increases when the number of memories becomes larger, while the distance between the two final points remains small and stable for $\alpha < \alpha_c^{HU}$. By comparing the final states of convergence we conclude that two networks, starting from the same initial matrix, end up in very similar configurations of the couplings J_{ij}. Now we analyze the whole trajectory traced by the two algorithms in the space of interactions.

We set $\alpha = 0.55$, so that the overlap between the initial and the final state is small enough, i.e. they are distant on the sphere, $N = 800$, $\lambda = 10^{-4}$ and k close to k_{max} in one single sample. The choice of a small value of the learning rate λ allows to trace a continuous path in the space of the interactions. HU is run choosing $d = D_{in}$ for 10 samples in total. Fig. 2.13a reports the projection of the resulting trajectories in the space of J along three randomly chosen directions. The plot shows that the two algorithms explore the same region of the space of interactions, proceeding along a similar direction. We also observe that the convergence velocities of the two algorithms are very different. Indicating with t the time steps for both processes, fig. 2.13b shows the logarithm of the absolute value of the *variation* of vector $\vec{J}$, defined as

$$\Delta\vec{J}^{(t)} = \vec{J}^{(t+1)}/\sigma_J^{(t+1)} - \vec{J}^{(t)}/\sigma_J^{(t)} \ . \tag{2.26}$$



(a)



(b)

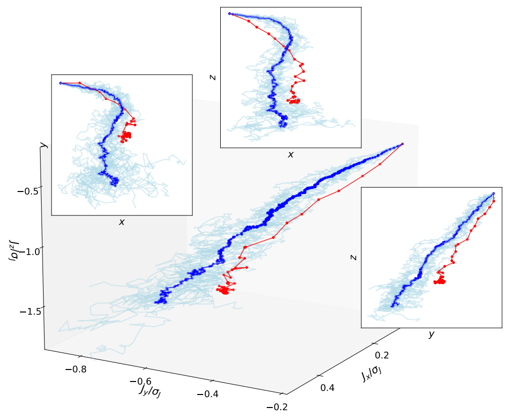
Figure 2.12. (a) Overlap q between the final states of Hebbian Unlearning (HU) with $\epsilon = 10^{-2}$ at $d = D_{in}$ and symmetric perceptron (SP) with $\lambda = 1$ having reached a stability k. Measures are for different values of N and α and points represent the mean computed over 5 realizations of the disorder. Error bars are smaller than the data symbol. Values of k range from 0 to slightly below $k_{max}(\alpha)$. For each α, q peaks around $k_{max}(\alpha)$, indicating that the two algorithms converge to coupling matrices which are closest near to the value $k = k_{max}(\alpha)$. (b) Overlap q as a function of α at $N = 800$. Points represent the mean of 10 realizations of the disorder and error bars are smaller than the data symbol. The orange symbols correspond to the overlap between the final states of SP with $\lambda = 1$, $k \simeq k_{max}(\alpha)$ and HU with $\epsilon = 10^{-2}$. For HU we chose $d = D_{in}$ for $\alpha < \alpha_c$, while for $\alpha > \alpha_c$ we chose $d = D_{top}$ (α_c is represented by the gray dotted line). The overlap between the initial Hebbian matrix and the final state of the HU (green) or SP (blue) with the same choice of the parameters is also shown. While in both algorithms the distance between initial and final matrix increases as α is increased, the distance between the final points remains small up to α_c.

The direction of this vector coincides with the one of the gradient followed by the algorithm in the space of interactions at a given time step.

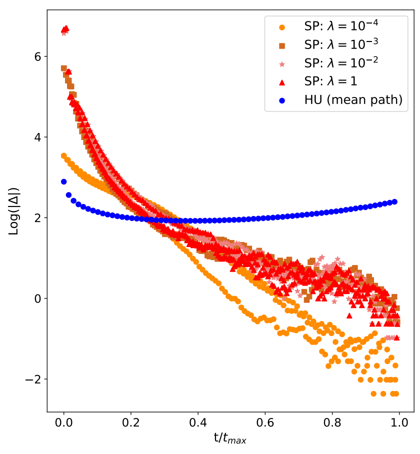
While the convergence speed of the HU does not significantly vary, the SP shows, at any scale of λ, an acceleration in time that resembles an exponential law. In other words, while HU explores the space of interactions nearly uniformly in speed, the SP takes about $15 \div 20$ time steps to reach a smaller condensed region where it gets confined until convergence.

The different speeds of the algorithms imply an inherent difficulty in comparing the trajectories *point-by-point*. Our analysis will thus rely on defining a particular direction $\hat{v}$ in the space of interactions that we will use to compare the two trajectories and their gradients. Such a direction is defined by the line that connects the initial Hebbian matrix with the point of convergence of the SP,

$$\hat{v} = \frac{\vec{J}_{SP}^{(t_{max})}/\sigma_{SP}^{(t_{max})} - \vec{J}^{(0)}/\sigma^{(0)}}{|\vec{J}_{SP}^{(t_{max})}/\sigma_{SP}^{(t_{max})} - \vec{J}^{(0)}/\sigma^{(0)}|} \,. \tag{2.27}$$

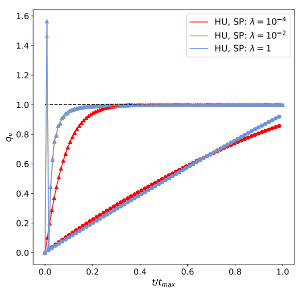


(a)

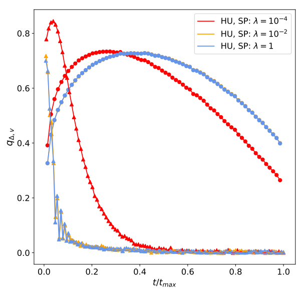


(b)

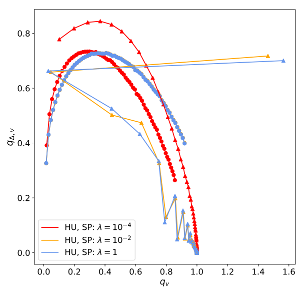
Figure 2.13. (a) 3-dimensional projection of the trajectories followed by the system in the space of interactions during the dynamics of Hebbian Unlearning (HU) and symmetric perceptron (SP). Numerical measurements have been taken for one sample at $N = 800$ and $\alpha = 0.55$, $\epsilon = 10^{-2}$, $\lambda = 10^{-4}$. 10 trajectories of the HU are drawn in light blue, while the average Unlearning path is in blue. The path followed by the SP is depicted in red. Points represent different steps of the algorithms. HU has been resampled at regular intervals along the trajectory for simplicity of the data analysis. (b) Absolute value of the variation $\Delta\vec{J}$ in logarithmic scale as a function of the normalized time scale t/t_{max} where t_{max} is the maximum number of steps reached by the algorithm in a given sample. Numerical measurements are for one sample at $N = 800$ and $\alpha = 0.55$, $\epsilon = 10^{-2}$, $\lambda = 1, 10^{-2}, 10^{-4}$. Three samples were simulated for the SP and one sample for the HU



(a)



(b)



(c)

Figure 2.14. (a) q_v, angular distance of the trajectory from the reference direction $\hat{v}$, as a function of time. (b) $q_{\Delta,v}$, projection of the variation along the direction $\hat{v}$, as a function of time. (c) $q_{\Delta,v}$ as a function of q_v. Numerical measurements are collected from one single sample at $N = 800$ and $\alpha = 0.55$ at $\epsilon = 10^{-2}$ for the Hebbian Unlearning (HU) (*circles*) and different values of λ for the symmetric perceptron (SP) (*triangles*).

We can now define two time-dependent observables that can help us in the analysis of the trajectories:

$$q_v(t) = \frac{\vec{J}^{(t)}/\sigma^{(t)} - \vec{J}^{(0)}/\sigma^{(0)}}{|\vec{J}^{(t_{max})}/\sigma^{(t_{max})} - \vec{J}^{(0)}/\sigma^{(0)}|} \cdot \hat{v} \ , \tag{2.28}$$

which is a measure of the angular distance of any point of the trajectory from the line traced by the direction $\hat{v}$ at time t. The smaller this quantity is, the more evidently the trajectory is diverging from $\hat{v}$. One can also introduce

$$q_{\Delta,v}(t) = \frac{\vec{J}^{(t+1)}/\sigma^{(t+1)} - \vec{J}^{(t)}/\sigma^{(t)}}{|\vec{J}^{(t+1)}/\sigma^{(t+1)} - \vec{J}^{(t)}/\sigma^{(t)}|} \cdot \hat{v} \ , \tag{2.29}$$

that is, the projection of the variation of $\vec{J}$ at the step t along the direction $\hat{v}$. The larger this quantity is, the more aligned to $\hat{v}$ the trajectory is.

Fig. 2.14a represents the values of q_v during the same trajectory that is depicted in fig. 2.13a. One can see that $q_v(0)$ is small for both the HU and the SP. This means that they both start in the wrong direction: this is particularly reasonable for the HU, which involves a random picking of the initialization state. However, while the SP rapidly reaches $q_v = 1$ because of its high initial acceleration at all the considered values of λ, in the HU algorithm q_v is increasing at lower rate. An initial overshooting of the SP at high values of λ is signaled by an anomalously high value of q_v at the second step of the process.

The directions followed by the two algorithms with respect to $\hat{v}$ are shown in fig. 2.14b. The SP at $\lambda = 10^{-4}$ has a peak in $q_{\Delta,v}$ in the first part of the trajectory, signaling a high degree of alignment between the gradient and $\hat{v}$. Later on, the SP rapidly converges towards the condensed region, where gradients lose their polarization with $\hat{v}$, but the convergence point has already been reached. When higher values of λ are used, no relevant polarization is measured. The HU shows a similar behavior: the trajectory starts along a direction that is barely aligned with $\hat{v}$ but a consistently high degree of alignment is obtained after more or less half of the iterations. Eventually, the HU also converges towards the final state losing the alignment with $\hat{v}$. Fig. 2.14c displays the direction of the variation as a function of the distance from $\hat{v}$. Three different behaviors of the trajectory can be thus recognized for both algorithms. First the trajectory moves away from $\hat{v}$ after a bad start, in a second phase it aligns to $\hat{v}$, and in a third phase the matrix plunges towards the convergence state.

2.3.3 A geometric interpretation of the effective weights

Considering our analysis of the optimal structure of noise and the robustness properties of a network trained via HU, we provide for a physical interpretation of the effective weights ω_i^μ that were so fundamental for the TWN algorithm (and so to HU when fixed points are employed) to reproduce the SVM performance.
Let us introduce the vector $\vec{J_i}$ as the collection of the elements contained in the i^{th} row of the connectivity matrix, $\vec{\eta_i}^\mu$ as the *memory pattern* defined as

$$\vec{\eta_i}^\mu = \xi_i^\mu \vec{\xi^\mu}, \tag{2.30}$$

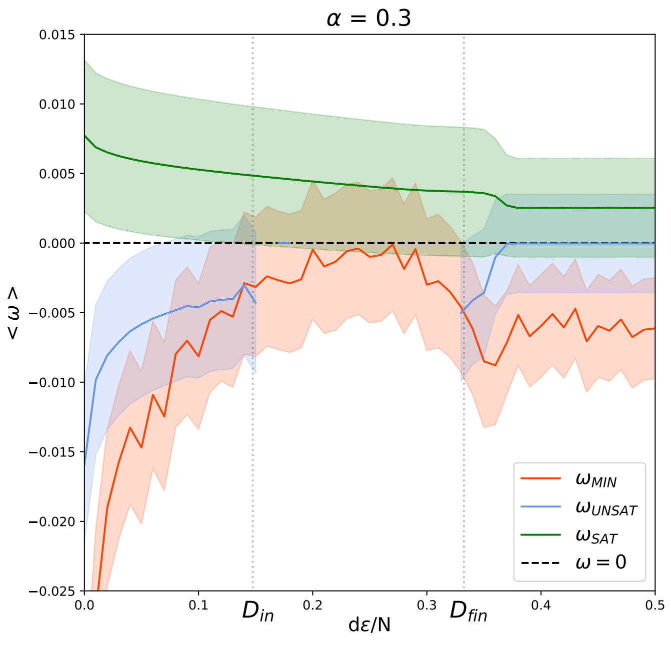


Figure 2.15. Mean values assumed by the perceptron overlap ω for SAT, UNSAT and *min* patterns during the Hebbian Unlearning process. Measurements are performed at $N = 800$, $\alpha = 0.3$, $\epsilon = 10^{-2}$ over 200 spurious states and several realizations of the disorder. Error bars are indicated by the shaded region.

and eventually the *glassy pattern*, defined in analogy with the memory patterns as

$$\vec{\eta_i}^* = S_i^* \vec{S}^*. \tag{2.31}$$

Let us now rewrite a *positive* perceptron update, i.e. one where the mask ϵ_{ij}^{μ} assumes a non-zero value, as

$$\delta \vec{J}_i^{P,d} = +\frac{\lambda}{N}\vec{\eta_i}^{\mu}. \tag{2.32}$$

We can also rewrite the HU update in the same vectorial fashion, i.e.

$$\delta \vec{J_i}^{HU,d} = -\frac{\epsilon}{N}\vec{\eta_i}^* , \tag{2.33}$$

Hence, the effective weights ω_i^{μ}, for the case of data-points being fixed points of the dynamics, are

$$\omega_i^{\mu} = \frac{1}{N}\vec{\eta_i}^* \cdot \vec{\eta_i}^{\mu} . \tag{2.34}$$

It comes natural to reinterpret ω_i^{μ} as the projection of the synaptic update for HU on the SP positive one, i.e.

$$\omega_i^{\mu} \propto -\delta \vec{J_i}^{SP} \cdot \delta \vec{J_i}^{HU} \tag{2.35}$$

at each iteration d. This implies that the more negative ω_i^{μ} the higher is its contribution to align HU with SP learning. We know that each pair (i, μ) is related to a given constraint of the associated optimization problem, with $\Delta_i^\mu \geq 0$ for SAT constraints, and $\Delta_i^\mu < 0$ for UNSAT ones. Fig. 2.15 shows $\overline{\langle \omega_i^\mu \rangle}$, with the brackets indicating the average over the spurious states in a given realization of the disorder, at $\alpha = 0.3$ and $N = 800$ for the three types of constraints: SAT, UNSAT and *minimally satisfied*, i.e. the SAT constraints with the lowest measured stability. The fact that the perceptron overlap is negative for both UNSAT and *min* constraints, but positive for SAT constraints, suggests that the distribution of the $\vec{\eta}_i^\mu$ looks anisotropic from the reference frame of the glassy patterns. This is certainly induced by the fact that glassy patterns $\vec{\eta}_i^*$ are SAT by definition, so they are more likely to be contained in the same half of hyperspace, defined by the orthogonal plane to $\vec{J}_i$, that contains SAT memory patterns. Consequently, since there is a minus sign on r.h.s. of eq. (2.33), HU is performing the same geometric transformation of the perceptron in order to align the $\vec{J}_i$ vectors to the memory patterns $\vec{\eta}_i{}^\mu$. By only exploiting the rugged shape of the Hebbian landscape out of the retrieval regime, the HU algorithm manages to accomplish this task in an quasi-optimal way.

2.3.4 Evolution of the Unlearning algorithm

When training configurations are fixed points of the dynamics with $m_t > 0^+$, the considerations of the previous section no longer apply. Such configurations can be generated by initializing the network at an overlap $m > 0^+$ with a memory, and let it evolve according to (1.1). The connectivity matrix is initialized according to Hebb's learning rule (1.8). In this setting, if at some point during training m happens to enter the basin of attraction of the memories, the fixed point reached by the dynamics will be the memory itself, and the noise contribution will cancel exactly the learning contribution, giving $\delta J = 0$. On the other hand, for sufficiently high values of the load α, at the start of the training procedure memories will have zero size basins of attraction and trajectories will drift away from the memories following the dynamics. In this scenario, the two contributions δJ^N and δJ^U decorrelate, and δJ^N will take again a similar role to what described in the previous section. The result is an algorithm which interpolates between HU when d is small and a supervised algorithm when basins increase to a size close to $(1 - m)$. In this regime, δJ^N acts as a breaking term, preventing the algorithm to further modify the coupling matrix J. A similar mechanism has been studied in [74] where a supervised term was added to the standard HU update rule, leading to $\delta J_{ij} \propto -S_i^{\mu_d} S_j^{\mu_d} + \xi_i^{\mu_d} \xi_j^{\mu_d}$. Notice that the term $\xi_i^{\mu_d} \xi_j^{\mu_d}$ is deterministically reproducing Hebb's learning rule, while in our study the HU is modified by a stochastic term, whose nature we have already discussed. Given a sufficiently small learning rate λ, there will exist a characteristic number of steps of the algorithm over which the coupling matrix does not change significantly. We will refer to this timescale as *epoch*. Averaging the effect of training steps over an epoch, we get a snapshot of how the algorithm is affecting the couplings at a given point during training. This can be used to study the relation between δJ^N and δJ^U, quantified by the connected correlation coefficient

$$\mathrm{Cov}_{\text{N-U}} := \frac{2}{N(N-1)} \sum_{i,j>i} \left(\langle \delta J_{ij}^N \delta J_{ij}^U \rangle_{epoch} - \langle \delta J_{ij}^N \rangle_{epoch} \langle \delta J_{ij}^U \rangle_{epoch} \right), \tag{2.36}$$

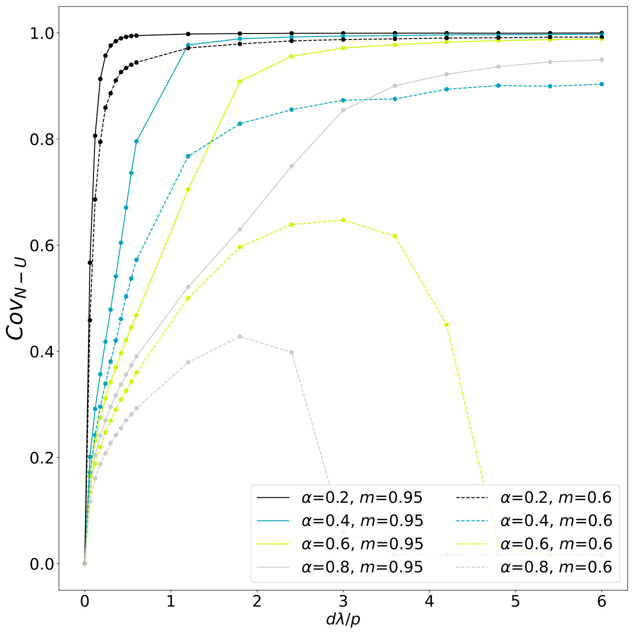


Figure 2.16. Correlation between *noise* and *unlearning* contributions to δJ as a function of the rescaled number of training steps $\frac{d\lambda}{p}$, for two values of the training parameter m, and different values of α. When $\mathrm{Cov}_{\text{N-U}} = 1$, the algorithm stops modifying the coupling matrix. Results are averaged over 100 samples. Choice of the parameters: $N = 400$, $\lambda = 10^{-2}$.

where the average is computed over an epoch. When this quantity equals one, there is no effective update of the couplings over an epoch. Results are presented in fig. 2.16 for different values of m and of α, as a function of the number of training steps d. The number of iterations has been rescaled by a factor p/λ for clarity of the plot. At any given α, training with higher m results in a faster increase of $\mathrm{Cov}_{\text{N-U}}$, i.e. a faster convergence of the algorithm. If the value of m is too low, the algorithm never manages to build a big enough basin of attraction, and never stops. As α is increases, higher and higher values of m are required for the algorithm to stop, since the typical size of the basins of attraction shrinks.

The network performance can be benchmarked by tracking the evolution of the lowest stability Δ_{min} throughout the training procedure. Results are presented in fig. 2.17 for different values of α and m. For sufficiently low values of α and sufficiently high values of m, Δ_{min} surpasses the zero. Once this condition is met, the value Δ_{min} becomes essentially constant, even if $\mathrm{Cov}_{\text{N-U}} < 1$, signaling that the update of the coupling matrix is still in progress. The result is a curve $\Delta_{\text{min}}(d)$ which barely surpasses zero. For each value of m there exists a critical value $\alpha_c(m)$ beyond which no amount of steps is able to produce $\Delta_{\text{min}} > 0$. Extrapolating empirical results to the $N \to \infty$ limit, one finds

$$\alpha_c(m) = A \cdot (m)^B + C,$$

where

$$A = 0.35 \pm 0.01, \quad B = 6.9 \pm 0.5, \quad C = 0.58 \pm 0.01$$

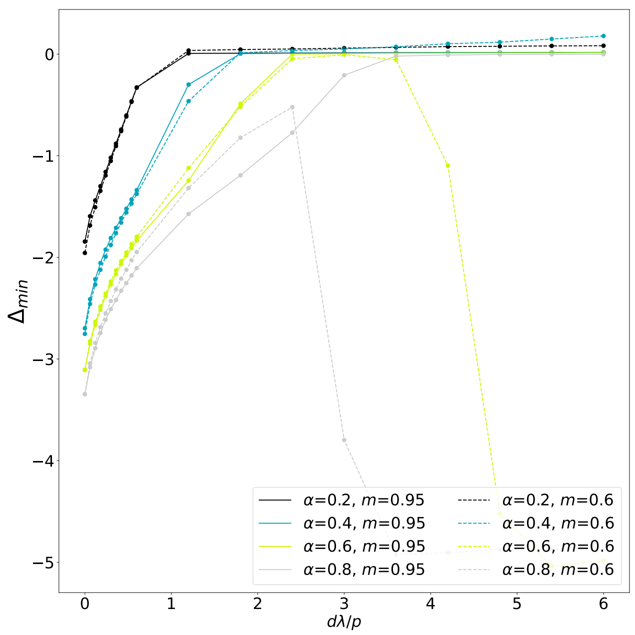


Figure 2.17. Minimum stability as a function of the rescaled number of training steps $\frac{d\lambda}{p}$, for two values of the training parameter m, and different values of α. When $\Delta_{\min} \geq 0$, each memory is a stable fixed point of the dynamics. Results are averaged over 100 samples. Choice of the parameters: $N = 400$, $\lambda = 10^{-2}$.

Consistently with what presented in the previous section, in the limit $m \to 0^+$ one finds the critical capacity of the Unlearning algorithm [31, 21]. The critical capacity increases up to a value $\alpha_c(1) = 0.93 \pm 0.01$ when m reaches unity.

Regardless of whether $\Delta_{\min} > 0$ is obeyed, one can monitor the network performance as an associative memory device by measuring the retrieval map $m_f(m_0)$. We find that the best performance always corresponds to the number of training step maximizing $\Delta_{\min}$, hence the curves $m_f(m_0)$ are all relative to this number of steps. Results are presented in fig. 2.18. When perfect-retrieval is achieved (i.e. $\alpha < \alpha_c(m)$), lower values of m increase the degree of robustness of the network (i.e. enlarge the basins of attraction), at the cost of a lower critical capacity.

Checkpoint

In this section we have seen that:

- The TWN algorithm formally converges to HU when training data are stable fixed points having an overlap $m_t = 0^+$ with the memories.

- Consistently with the theory, the HU algorithm approaches the performance of a SVM (both in terms of perfect-retrieval and robustness) in a full unsupervised way: it is thus faster and more biologically relatable. This behaviour is conserved until a critical capacity $\alpha_c^{HU} \simeq 0.6$.

- An explanation for the quality of the stable fixed points that can be sampled from a landscape initialized with the Hebbian rule can be found in the dispo-

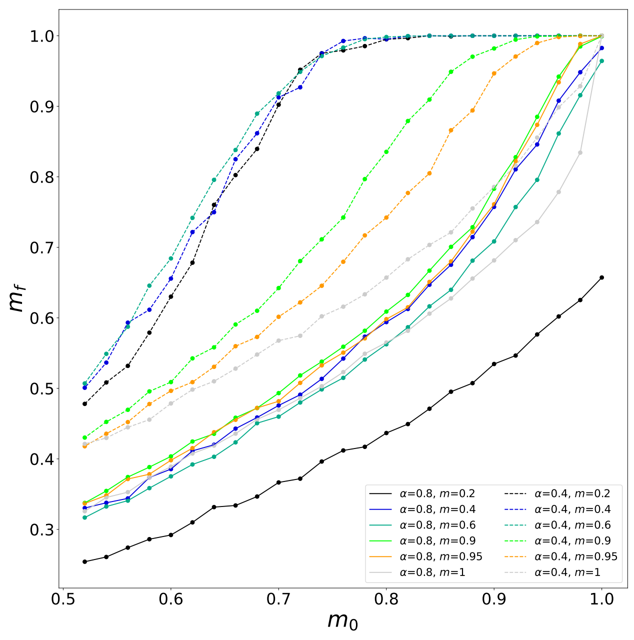


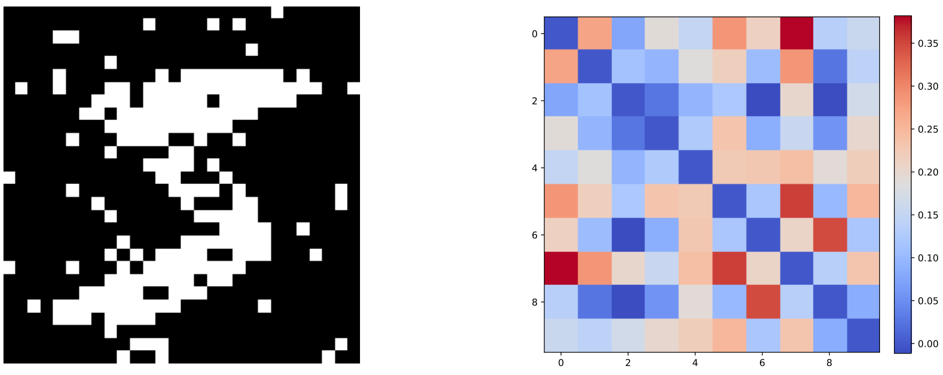
Figure 2.18. Retrieval map $m_f(m_0)$ for two values of α and different values of the training parameter m. At $\alpha = 0.4$, every m leads to stable memories, i.e. $m_f(1) = 1$. At $\alpha = 0.8$, only the highest values of m lead to stable memories, while for low values of m one has $m_f(1) < 1$. Results are averaged over 100 samples. Choice of the parameters: $N = 400$, $\lambda = 10^{-2}$.

sition of this states in the configuration space. When applying a particular transformation of the space of the network configurations (see eq. (2.31)), stable fixed points appear to be distributed anisotropically around the memories.

- The TWN algorithm has be exploited to propose a supervised version of the HU algorithm that is fast (because it keeps sampling the stable fixed points of the landscape) and reaches a higher critical capacity, i.e. $\alpha_c \simeq 0.93$.

2.4 The case of correlated memories

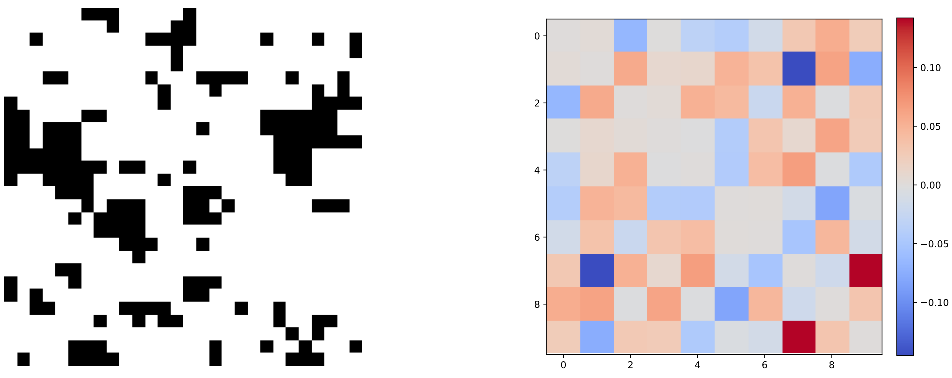
We now consider two sets of memories $\{\vec{\xi^{\mu}}\}_{\mu=1}^{\alpha N}$ that contain different types of internal correlations and evaluate whether the new regularization technique we found in terms of noise structure can be applied in the case of correlated memories.
One set is composed by MNIST images representing handwritten digits, conveniently reshaped into one-dimensional arrays. Each image is a square matrix of $N = 28^2 = 784$ pixels assuming values between 0 and 255. Each pixel ξ_i^{μ} is then transformed into -1 if its original value is 0 and $+1$ if its original value is larger than 0. In addition to this we flip the entries of the vector with a probability $p = 0.1$ to inject some disorder in the data-point. This operation is necessary to smooth the very strong biases that might cancel σ_i, standard deviation of the coupling matrix along line i, inducing some relevant quantities to diverge.
The other set of memories under consideration is composed by configurations sampled from a 2-dimensional Ising model in the paramagnetic phase (i.e. with at different temperatures larger than the critical one). Each configuration is also presented as a $N = 784$ sized array with $\xi_i \in \{-1, +1\}$. Figures 2.19a and 2.20a show examples of training configurations from these collections.



(a) Data-point **(b)** Correlation matrix

Figure 2.19. Representation of the disordered MNIST set. The data-point is represented in the matrix shape, while the correlation matrix is computed with 10 examples and the unitary diagonal entries is set to 0 to enhance the off-diagonal terms.

Both types of memories have internal correlations among their elements. MNIST data are very regular: the digit is written in the middle of the image and each digit is sufficiently symmetric, under the geometric point of view. On the other hand, Ising configurations have thermal correlations. Moreover, different memories can be mutually correlated across the set. In particular, MNIST digits are repeated across the set, and they might be generally similar with each other (e.g. a ’four’ does not look very different from a ’one’, as well as a ’eight’ might look similar to a ’zero’). On the contrary, Ising configurations have low mutual correlations, because they are sampled in the paramagnetic phase, where there are no biases due to the non-zero



(a) Data-point **(b)** Correlation matrix

Figure 2.20. Representation of the Ising set. The data-point is represented in the matrix shape, while the correlation matrix is computed with 10 examples and the unitary diagonal entries is set to 0 to enhance the off-diagonal terms.

magnetization. These aspects appear evident from the correlation matrices reported in figures 2.19b, 2.20b.

2.4.1 Numerical analysis

In order to characterize the quality of the noise contained into the training data, we generate Hebbian couplings J_{ij}^H from the memories and perform a Monte Carlo routine at different temperatures T, to probe the energy landscape at different levels, as previously done for the set of random memories. As we know from the theory in appendix B.2 the gradient of the Loss function of a SVM is given by

$$\delta\mathcal{L} \propto \lim_{m\to 1^-} \sum_{i,\mu}^{N,p} \frac{1}{2\sigma_i}\left[(m_\mu \chi_i^{1,\mu} + m_{1,\mu}\chi_i^\mu) - (m_\mu \xi_i^\mu \xi_i^{\mu_d} + M_\mu^{\mu_d}\chi_i^\mu)\right] \exp\left(-\frac{m^2 \Delta_i^{\mu^2}}{2(1-m^2)}\right) \tag{2.37}$$

that can be rewritten in a more compact fashion as

$$\delta\mathcal{L} = \delta\mathcal{L}_N + \delta\mathcal{L}_U. \tag{2.38}$$

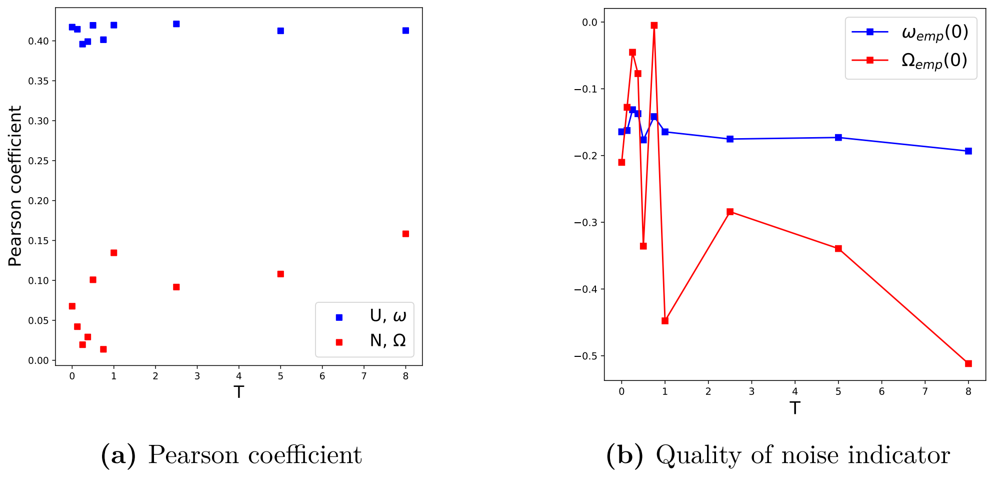
The quantity indicated as $M_\mu^{\mu_d}$ is an overlap between two memories, defined as $M_\mu^{\mu_d} = \frac{1}{N}\sum_{i=1}^N \xi_i^\mu \xi_i^{\mu_d}$, while $m_{1,\mu}$ is the overlap between the one-step evolution of the picked data-point and the μth memory, namely $m_{1,\mu} = \frac{1}{N}\sum_{j=1}^N S_j^{1,\mu_d}\xi_j^\mu$ (see appendix B.2 for further details). The main contribution to the gradient are then given by the two following quantities

$$\omega_i^\mu = \frac{1}{2\sigma_i}\left(m_\mu \chi_i^{1,\mu} + m_{1,\mu}\chi_i^\mu\right), \tag{2.39}$$

and

$$\Omega_i^\mu = \frac{1}{2\sigma_i}\left(m_\mu \xi_i^\mu \xi_i^{\mu_d} + M_\mu^{\mu_d}\chi_i^\mu\right). \tag{2.40}$$

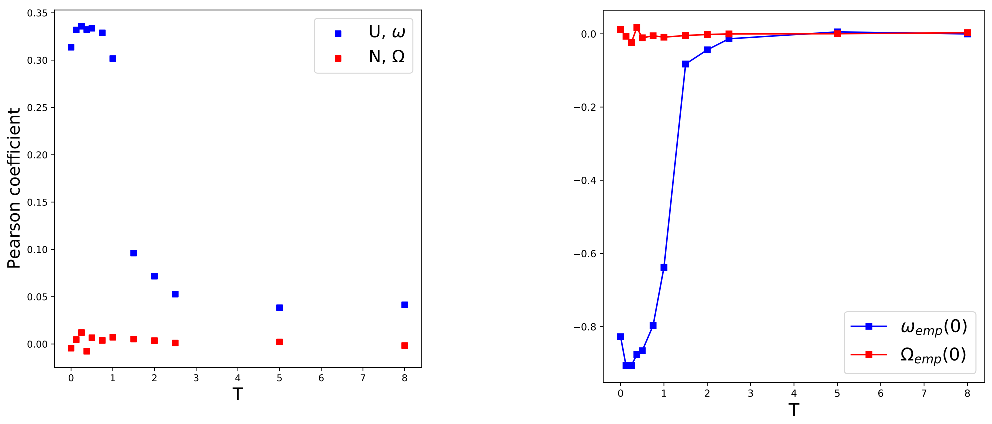
One is specifically interested in two features: the order of magnitude of ω_i^μ and Ω_i^μ in (B.4) and their values when $|\Delta_i^\mu| < 0^+$. In order for the optimal noise condition to apply, ω_i^μ should be dominant with respect to Ω_i^μ and negative in the relevant interval of values. Hence we measure the Pearson coefficient between ω_i^μ, Ω_i^μ and the stabilities Δ_i^μ across different levels of the Hebbian landscape as well as the representative value for the same two quantities around $\Delta_i^\mu = 0$, i.e. $\omega_{emp}(0)$ and $\Omega_{emp}(0)$. We perform such measures for both the two sets of memories, across the Hebbian landscape.



(a) Pearson coefficient **(b)** Quality of noise indicator

Figure 2.21. Numerical analysis of the noise contained in the configurations sampled from the Hebbian landscape for the MNIST data-set. The panel on the left reports a measure of the dependence of the relevant observables on the stabilities. The panel on the right reports an interpolation of the relevant observable for very small stabilities, indicating whether the optimal noise condition is well satisfied or not. Measures are averaged over 50 landscapes. Choice of the parameters: $N = 784$, $\alpha = 0.5$.

From the study of the disordered MNIST set shows we can conclude that such configurations would not lead to a good minimization of the SVM Loss at altitude in the Hebbian landscape. This is implied by two observations: the noise contribution to the gradient is not negligible due to a rather significant dependence on the stabilities (see fig. 2.21a), and the values assumed by the same contribution relatively to small stabilities are large and negative (see fig. 2.21b).
On the other hand, Ising configurations are good training configurations at low temperatures T, showing a similar scenario to the one encountered with random configurations: the noise contribution does not depend on stabilities, suggesting it is going to cancel step by step in learning (see fig. 2.22a), and $\omega_{emp}(0)$ is negative and dominant with respect to $\Omega_{emp}(0)$ in the sensible range of Δ_i^μ (see fig. 2.22b).
The interpretation of the results is straight-forward. In contrast with the Unlearning contribution, the Noise contribution to the gradient of the Loss depends on the mutual correlations among memories in the set. Since digits in the MNIST set are similar with each other, such points result difficult to be learned by the TWN algorithm. On the contrary, Ising configuration are sufficiently separated with each



(a) Pearson coefficient **(b)** Quality of noise indicator

Figure 2.22. Numerical analysis of the noise contained in the configurations sampled from the Hebbian landscape for the Ising data-set. The panel on the left reports a measure of the dependence of the relevant observables on the stabilities. The panel on the right reports an interpolation of the relevant observable for very small stabilities, indicating whether the optimal noise condition is well satisfied or not. Measures are averaged over 10 landscapes. Choice of the parameters: $N = 784$, $\alpha = 0.5$.

other, implying the optimal noise condition to be satisfied for some regions of the landscape. We can also conclude that internal dependencies in the memories, which were present in both the MNIST and Ising data, are not relevant in the process, since they do not enter in both ω and Ω.

Checkpoint

In this section we have seen that:

- The paramagnetic bi-dimensional Ising configurations are correctly memorized by the TWN algorithm with structured noise with $m_t = 0^+$ because, similarly to the random data scenario, the Hebbian landscape generated from these data contain good training configurations; the MNIST data, on the other hand, fail at reaching a good memory performance because the relative Hebbian landscape is poor of good training configurations.

- One can approach a SVM that solidly stores a set of memories by means of the TWN algorithm as far as these memories are well separated from each other, i.e. they overlap weakly. Internal correlations inside the features of the memories are allowed as far as memories are mutually distant in the space of network configurations.

2.5 Unlearning on Continuous Attractor Neural Networks

A Continuous Attractor Neural Network (CANN) represents a successful model that reproduces spatial associative memory [38, 75, 76, 77, 78, 79], i.e. the ability to retrieve real environments embedded in a Euclidean space.
This time we consider a neural network of N binary units $S_i \in \{0,1\}$. In analogy with the biology of the hippocampus, units are associated to *place cells* i.e. special neurons associated to particular positions in space. We can generate such positions at random for each one of the L environments that can be memorized by the neural network. We will employ the so called *remapping* ansatz, i.e.

$$\vec{r}_i^l \in [-1/2, +1/2]^D, \qquad i = 1,..,N \quad l = 1,..,L.$$

Each place cell monitors a D-dimensional sphere of volume w, called *place field*, also indicating the average fraction of the whole amount of place fields *seen* by that particular neuron. The radius of such a sphere is d_c. The radius for $D = 1,2$ measures

$$d_c = \frac{1}{2w}, \quad D = 1 \qquad\qquad d_c = \sqrt{\frac{w}{\pi}}, \quad D = 2. \tag{2.41}$$

Imagining to pin one position in the real space $\vec{x}$ in one of the environments l, the relative neural activity configuration $\vec{S}$ will be given by the following rule

$$\begin{cases} S_i = 1 & \text{if} \quad |\vec{r}_i^{\,l} - \vec{x}| \le d_c \\ S_i = 0 & \text{otherwise} \end{cases} \tag{2.42}$$

where $|\cdot|$ is the Euclidean norm operator. The spatial maps to be learnt by the network are assembled in the following manner: consider a number L of physical environments in a dimension D; such environments have a cubic shape of unitary sides and periodic boundary conditions; then p positions are generated in the environment, i.e.

$$\vec{R}^{l,\mu} \in [-1/2, +1/2]^D, \qquad l = 1,..,L \quad \mu = 1,..,p.$$

The real space maps to be stored can be transferred in the activity configuration space in the shape of memories by applying the usual rule

$$\begin{cases} \xi_i^{l,\mu} = 1 & \text{if} \quad |\vec{r}_i^{\,l} - \vec{R}^{l,\mu}| \le d_c \\ \xi_i^{l,\mu} = 0 & \text{otherwise} \end{cases} \tag{2.43}$$

2.5.1 Hebbian Learning

Once there is a set of memories the Hebbian framework can be generalized to CANNs to build the couplings J_{ij} among neurons as

$$J_{ij} = \frac{1}{pL}\sum_{l,\mu}^{L,p}(\xi_i^{l,\mu} - w)(\xi_j^{l,\mu} - w) \qquad J_{ii} = 0 \quad \forall i \tag{2.44}$$

Also in this case we must define a particular dynamic rule to let the system evolve in time. The most reliable choice is the Glauber dynamics at temperature T [70].

According to this rule we start from an initial configuration of the units and we update them asynchronously according to the following prescription

$$P(S_i^{(t+1)} = x) = \frac{1}{1 + e^{-2\beta h_i^{(t)} \cdot (2x-1)}}, \tag{2.45}$$

with $x \in \{0, 1\}$, $\beta = 1/T$ and $h_i^{(t)} = \sum_{j=1}^{N} J_{ij} S_j^{(t)}$ being the *local field.* The correspondent dynamic rule at $T = 0$ is then

$$S_i^{(t+1)} = \frac{1}{2}\left[1 + \text{sign}\left(h_i^{(t)}\right)\right] \tag{2.46}$$

According to the rule (2.44), for α smaller than a critical value and $T > 0$, the system explores the real space as a moving bump of activity, when Glauber dynamics is implemented [77, 75]. Notice that the average activity contained in the moving bump is, by construction

$$w = \frac{1}{N}\sum_{i=1}^{N} S_i \tag{2.47}$$

When $T = 0$, in the same region of the phase diagram, memories are closed to be fixed points of the dynamics. At larger values of a certain threshold $\alpha_c(T)$ the system falls in a glassy phase where maps are not correctly recalled [76, 75].

2.5.2 Support Vector Machines Vs. Hebbian Rule at $T = 0$

We are now interested in the numerical study of the CANN at $T = 0$. Let us pick a random position $\vec{x}$ in one of the environments and build an activity configuration from that, i.e. by doing

$$\begin{cases} S_i^{(0)} = 1 & \text{if} \quad |\vec{x} - \vec{r}_i^{\,l}| \leq d_c \\ S_i^{(0)} = 0 & \text{otherwise} \end{cases} \tag{2.48}$$

The *spatial error* ϵ is defined as the average distance between the center of mass of the activity in space at the fixed point reached by implementation of rule (2.46) by initializing the network in $\vec{S}^{(0)}$ and the center of mass at the activity at initialization. The position of the center of mass of the bump in a given map l at time t of the dynamics is computed in the following way

$$\vec{r}^{\,(l)}_{c.o.m.}(t) = \frac{\sum_{i=1}^{N} S_i^{(t)} \vec{r}_i^{\,(l)}}{\sum_{i=1}^{N} S_i^{(t)}}. \tag{2.49}$$

Since both initial positions and maps are generated at random one expects that a good memory retrieves the stored positions in the environment by attracting the dynamics in their basins of attraction. In this case the maximal extension of the basins of attraction is expected to equal the average distance among the random stored positions, that is $\epsilon \sim p^{-\frac{1}{D}}$, in order for them to span the map uniformly.
It has been showed by [38] that this scenario is not reached by the more canonical CANNs by making use of a Hebbian-like criterion but rather by an optimized neural network such as a Support Vector Machine (SVM), where all the memories are

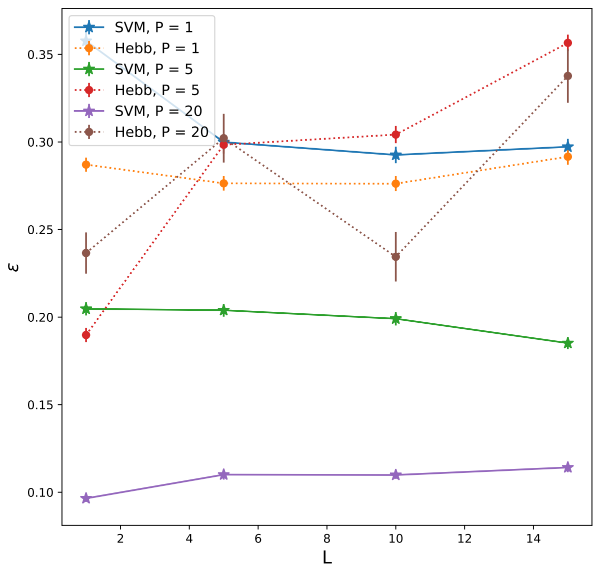


Figure 2.23. Spatial error ϵ measured as a function of L and p for both the SVM and Hebbian prescriptions. Measures have been averaged over 10 samples for $p = 1, 5$ and 1 sample for $p = 20$ for SVM, 10 samples at all p in the Hebbian case. Choice of the parameters: $N = 10^3$, $w = 0.3$, $D = 2$.

perfectly retrieved by (2.46) and the stability of the fixed points is maximized (i.e. local fields in the memory configurations are maximal in absolute value).
These results are reproduced in fig. 2.23 for $D = 2$ and $w = 0.3$. Notice that $\epsilon(L, p)$ is an increasing quantity in L that does not depend on p in the Hebbian network, while the *optimal packing* scenario is obtained in the SVM case.

2.5.3 Hebbian Unlearning

We now generalize the HU algorithm [17] for the case of a CANN. $J^{(0)}$ is chosen to be the Hebbian connectivity matrix according to rule (2.44).
Then the following three steps are iterated:

1. Random shooting: a random initial configuration of the neurons is generated.

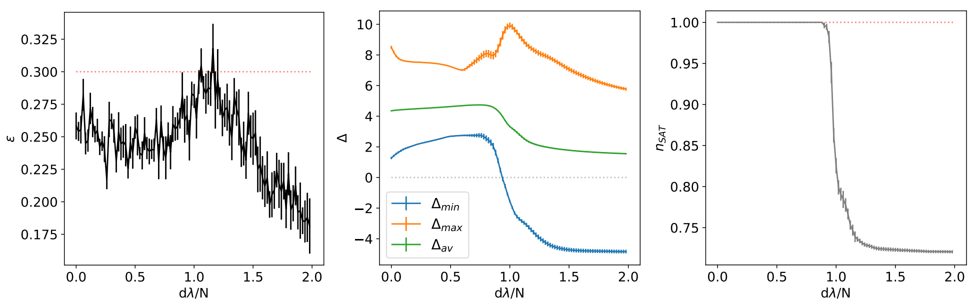
2. The initial configuration evolves until convergence into a stable state $\vec{S}^*$ by implementation of the dynamics (2.46).

3. The connectivity matrix is updated as

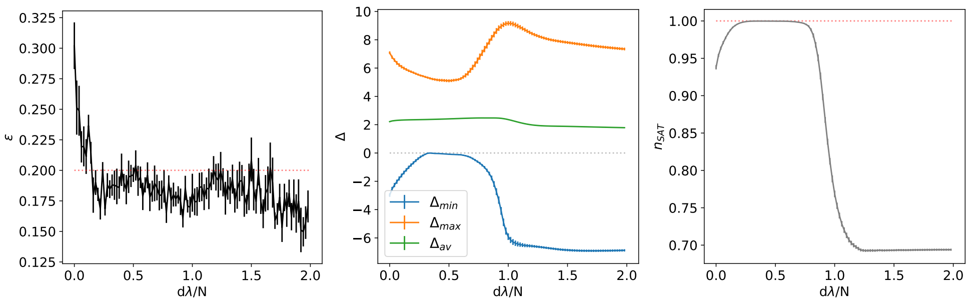
$$\delta J_{ij}^{(d)} = -\frac{\lambda}{N}(S_i^* - w)(S_j^* - w) \qquad J_{ii}^{(d)} = 0 \quad \forall i \tag{2.50}$$

   where λ is a small learning rate and d is the number of the algorithm iteration.

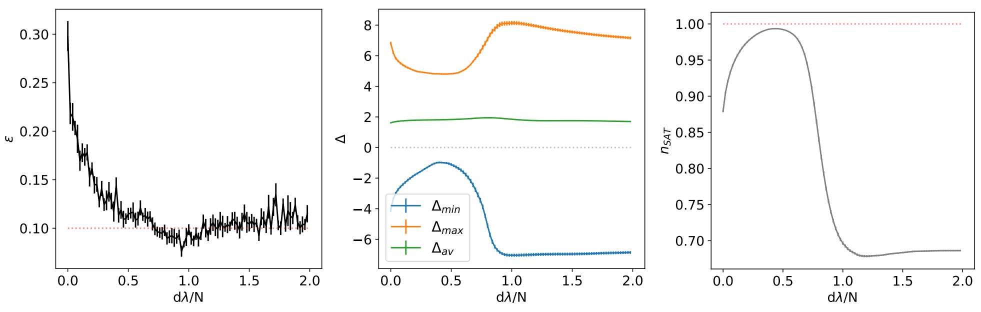
We can now characterize the Unlearning network by measuring three relevant quantities, i.e.



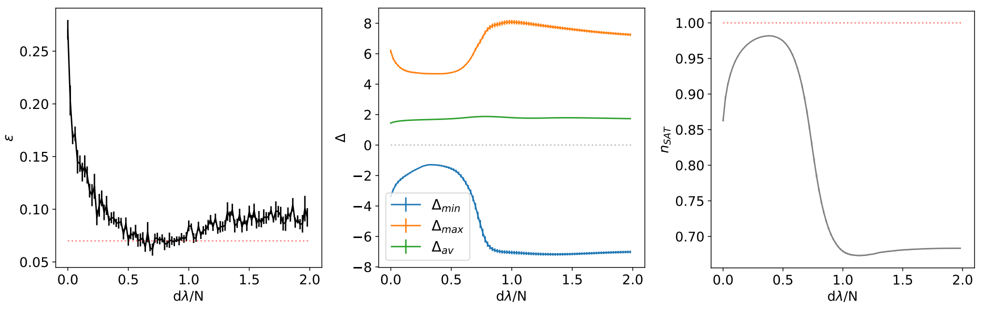
(a) L = 5, p = 1



(b) L = 5, p = 5



(c) L = 5, p = 20



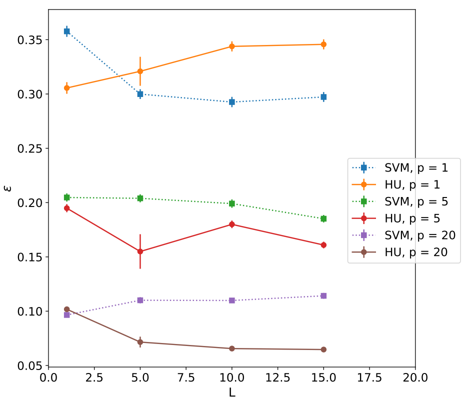
(d) L = 5, p = 50

Figure 2.24. Left: Spatial error ϵ as a function of the Unlearning steps. Center: average stabilities Δ as a function of the Unlearning steps. Right: fraction of satisfied constraints n_{SAT} as a function of the Unlearning steps. Measures averaged over 10 samples. Choice of the parameters: $D = 2$, $w = 0.3$, $\lambda = 10^{-2}$.

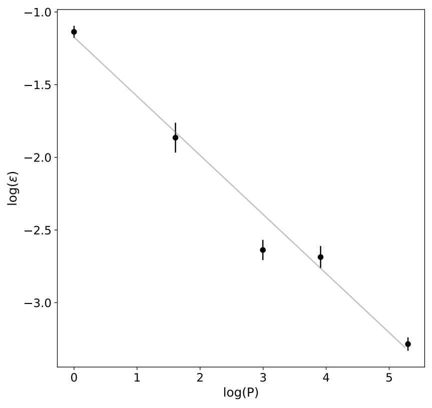
- The spatial error $\epsilon(L,p)$ as a function of the Unlearning steps.
- The average stabilities of the memories defined as

$$\Delta_i^{l,\mu} = (2\xi_i^{l,\mu} - 1)\sum_{j=1}^{N} \frac{J_{ij}}{\sqrt{\sum_k J_{i,k}^2}}\xi_j^{l,\mu} \tag{2.51}$$

  and we are specifically interested in $\Delta_{min} = \min_{(i,l,\mu)}(\Delta_i^{l,\mu})$.
- The fraction of satisfied constraints in the relative percepton problem n_{SAT}: a constraint labelled by (i,l,μ) is SAT when $\Delta_i^{l,\mu} \geq 0$, UNSAT otherwise.



(a)



(b)

Figure 2.25. (a): Spatial error ϵ measured as a function of L and p for both the SVM and after Hebbian Unlearning (HU). Measures have been averaged over 10 samples for $p = 1, 5$ and 1 sample for $p = 20$ for SVM, with $N = 10^3$, 10 samples at all p in the Unlearning case, with $N = 500$. (b): log-log plot of ϵ versus p at $L = 5$ for HU, best fit line in grey. Choice of the parameters: $w = 0.3$, $D = 2$, $\lambda = 10^{-2}$.

The behaviour of these three quantities is represented in fig. 2.24 for $L = 5$ and $p = 1, 5, 20, 50$. As one can see, the minimum value of the spatial error is reached in correspondence of the peak of Δ_{min} and n_{SAT}. However, perfect-retrieval is reached by the system only for small values of p, signaling the presence of a critical capacity for the HU procedure, as it was valid for random memories studied in [21, 31].
At this point we can experimentally extrapolate the number of iterations at which ϵ reaches its minimum value and we measure such value at different values of L to compare it with the results from the SVM. It should be noticed that the spatial error ϵ does not depend on L and N but only on p, as found for SVMs in [38]: this result is displayed in fig. 2.25a. Numerical values are also comparable between the SVM and the Unlearning network. Figure 2.25b reports the expected behaviour of $\epsilon(L,p)$. The opposite of the coefficient of the best fit line is found to be consistent in a 3σ confidence interval with the real dimension of space D. In fact, we have

$$D_{exp}^{-1} = 0.41 \pm 0.03$$

Another experiment is performed in order to show the effectiveness of HU in approaching the CANN to the optimal SVM scenario. A single 2-dimensional environment with $p = 20$ points disposed on a regular grid and learned by the network according to the Hebbian learning SVM and HU. Then the map is scanned in space and neurons are initialized according to the usual criterion (see eq. (2.43)): $T = 0$ dynamics is iterated until convergence and the node of the grid that is closest to the final position of the center of mass of the neural activity labels the starting point. Figure 2.26 reports the results. Same colours are associated to same basins of attraction of the stored points on the grid. Notice that Hebbian learning (fig. 2.26a) completely destroys the regularity of the grid, while HU and SVM (fig. 2.26b, fig. 2.26c) center basins of attraction, that are squared in shape with each side $\sim (1/p)^{\frac{1}{2}}$, around the grid points, maximizing their size. SVM performs this job optimally, as it could be predicted from [38] but HU seems to approach that limit very well.

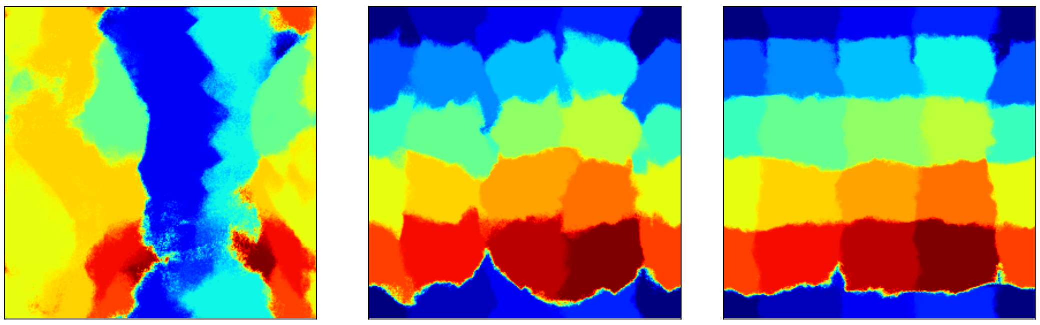

(a) Hebbian Learning **(b)** Hebbian Unlearning **(c)** SVM Learning

Figure 2.26. Representation of the basins of attraction in the real space after the three types of learning. Stored positions are disposed on a regular grid. Choice of the parameters: $D = 2$, $w = 0.3$, $L = 1$, $p = 20$, $N = 500$, $\lambda = 10^{-2}$.

2.5.4 Diffusion dynamics at $T > 0$

We now switch on the temperature $T > 0$ and study the dynamics of the activity bump in the space map as done in [76, 75]. Since we are interested in the diffusion in space of the bump only one environment will be stored in the network ($\alpha = 0$ case) in order to impede the activity to change map [78]. Figure 2.27 represents diffusion when p is low, i.e. $p = 20$, for all the three considered learning rules: the position of the center of mass of the neuronal activity is plotted as a function of the time steps. Before choosing the temperature it has been checked whether the network is in the *clump* phase [76, 77], i.e. neuronal activity is well clustered on the map. Dynamics is initialized in one given stored position on the map. Disorder appears to be very strong as the system is stuck for a very long time in one basin of attraction separated by the others by high barriers in free energy. The SVM explores a region that is well centered around the stored position, that is reasonable since all memories are stable fixed points of the $T = 0$ dynamics. On the other hand, when Hebbian Learning and HU are implemented the clump descends a bit in the Lyapunov function before finding a metastable minimum to probe. Another experiment is performed

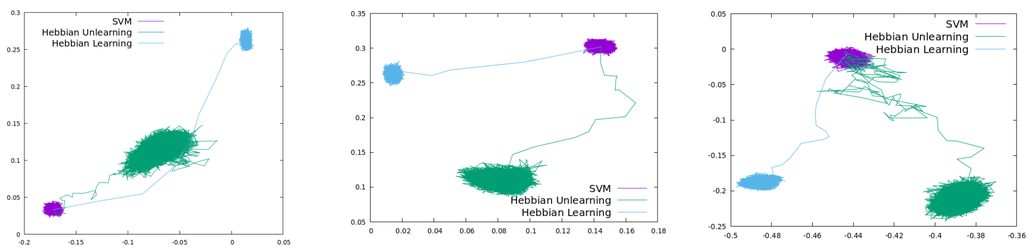


Figure 2.27. Diffusive trajectories of the moving bump in the landscape resulting from the Hebbian Learning, Hebbian Unlearning and SVM. Each time step is chosen to be one sweep of the network, i.e. N spins being evaluted by the the MCMC algorithm, and simulations have been run over $5 \cdot 10^4$ sweeps in total. Choice of the parameters: $D = 2$, $w = 0.3$, $\lambda = 10^{-2}$, $N = 500$, $L = 1$, $p = 20$, $T = 1.0$.

by increasing the number of stored positions to $p = 200$. Just HU and Hebbian Learning are evaluated this time, because of the high computational cost needed to find the connectivity matrix of the SVM.
Figure 2.28 displays the results for one realization of the network and one common initial condition. We notice that diffusion of the Hebbian network is still ineffective since it gets easily stuck in one metastable state. On the other hand diffusion of the system after the performance of HU is more efficient since it explores a wider portion of space. The 2-dimensional track of the center of mass is represented in fig. 2.28a while the motion in the two separated dimensions are reported in fig. 2.28b.

A diffusion coefficient $\mathcal{D}$ can be measured by simulations as done in [77]. In particular, for $D = 2$, one has

$$\mathcal{D} = \frac{1}{4N_{\text{sweeps}}} \sum_{t=1}^{N_{\text{sweeps}}-1} \left(\delta x_t^2 + \delta y_t^2\right) \tag{2.52}$$

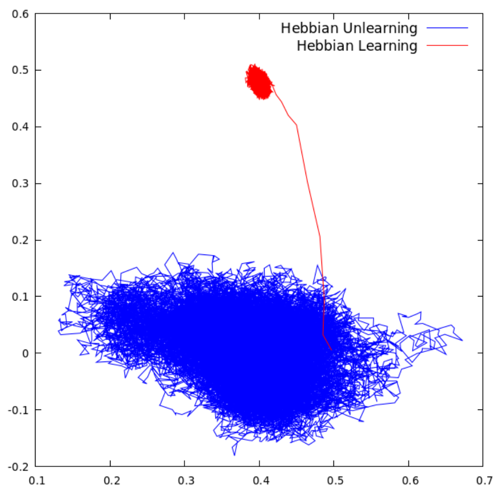
where $\delta x_t = x(t+1) - x(t)$ and $\delta y_t = y(t+1) - y(t)$.
We thus analyse the case of $N = 500$, $D = 2$, $w = 0.3$, $L = 1$, $p = 200$ at a temperature $T = 1$ for both the learning procedures. Ten samples of the system are evaluated. We obtain

$$\mathcal{D}_{\text{Hebbs}} = (7.0 \pm 2.0) \cdot 10^{-5}, \qquad \mathcal{D}_{\text{HU}} = (3.23 \pm 0.51) \cdot 10^{-3}, \tag{2.53}$$

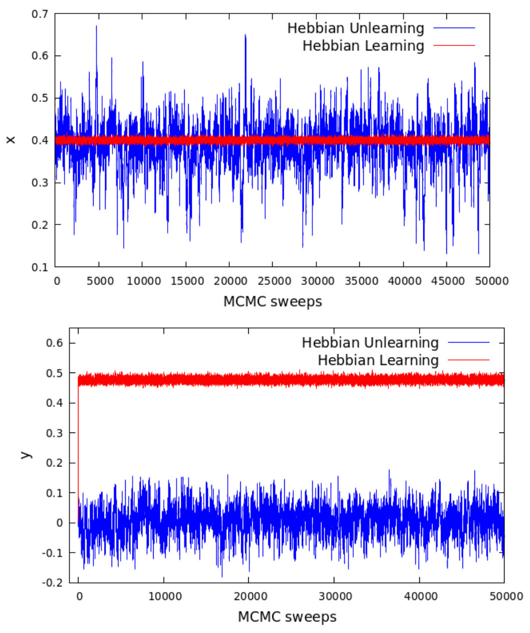
where errors are the standard deviation of the mean over the samples. To prove that we can effectively refer to the motion as a pure diffusive process, the mean square displacement of the bump after HU is reported in fig. 2.29 as a function of time: notice the good agreement of data with the Einstein relation for Brownian motion. One concludes that diffusion with Unlearning is $\sim 10^2$ times more effective than in the standard Hebbian scenario, given this choice of the parameters. The supplemental material in [38] suggests that this is the case for SVM too. In conclusion, HU reproduces, yet again, the optimal conditions for diffusion in one single environment.

Checkpoint

In this section we have seen that:



(a)



(b)

Figure 2.28. Hebbian Unlearning (blue) and Hebbian Learning (red). Each time step is chosen to be one sweep of the network, i.e. N spins being evaluated by the the MCMC algorithm, and simulations have been run over $5 \cdot 10^4$ sweeps in total. Choice of the parameters: $D = 2$, $w = 0.3$, $\lambda = 10^{-2}$, $N = 500$, $L = 1$, $p = 200$, $T = 1$.

- HU approaches the performance of a SVM even when implemented on spatially correlated memories. This result is consistent with the analysis advanced in section 2.2.
- When spatial maps are learned through SVM, or nearly equivalently HU, a Monte Carlo samples the real Euclidean space very effectively, without getting trapped in a limited region.

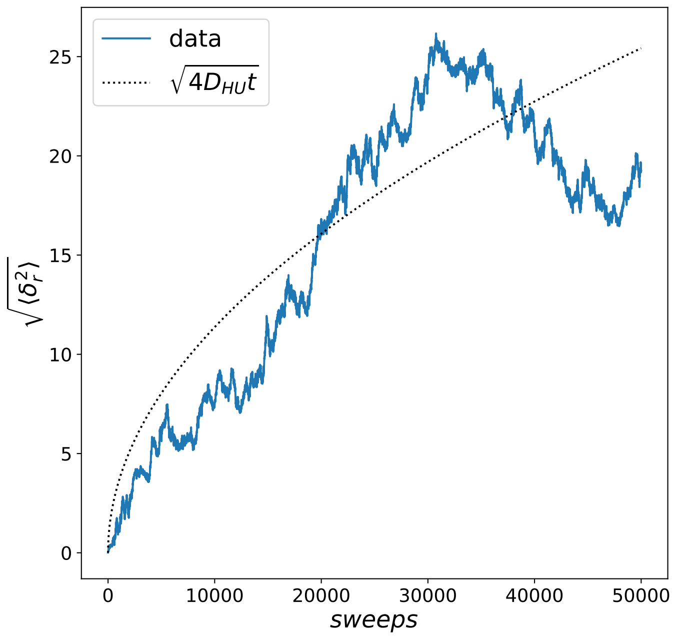


Figure 2.29. Mean square displacement of the bump after the optimal amount of Unlearning. The diffusion coefficient $\mathcal{D}_{HU}$ has been fitted from the data. Data have been averaged over 10 realizations. Choice of the parameters: $D = 2$, $w = 0.3$, $\lambda = 10^{-2}$, $N = 500$, $L = 1$, $p = 200$, $T = 1$.

2.6 Sampling the optimal noise

Selecting training data that satisfy equation (2.20) amounts to imposing specific internal dependencies among the noise units $\vec{\chi}$, which are no more i.i.d. random variables, as it was in [68]. We refer to such dependencies an *structure* of noise, and this is what apparently characterizes particular states in the Hebbian landscape of attractors, such as stable fixed points and surrounding saddle points.
Insights from the previous sections on the structure of well performing training data can be used to sample good training configurations according to other strategies. Namely, one can use a supervised Monte Carlo routine that searches for maximally noisy configurations (i.e. $m_t = 0^+$) satisfying condition (2.20). The coupling matrix is initialized according to Hebb's rule (1.8), and updated recursively according to either TWN (see eq. (2.1)) or HU (see eq. (1.10)). We first introduce the sampling algorithm and then report some numerical results regarding both the TWN and HU routines. We sample maximally noisy training configurations $m_t = 0^+$ such that $E(\vec{\chi}|m, J) < 0$, where

$$E(\vec{\chi}|m, J) := \frac{m}{\sqrt{2\pi(1-m^2)}} \sum_{i,\mu}^{N,p} \omega_i^\mu \exp\left(-\frac{m^2 {\Delta_i^\mu}^2}{2(1-m^2)}\right), \tag{2.54}$$

that is a function of the noisy variables χ_i^μ, conditioned on a reference overlap m and the couplings J. Sampling is done through the following procedure:

1. The network is initialized in a random configuration and the asynchronous dynamics in eq. (1.1) is run until convergence on a fixed point. The final state

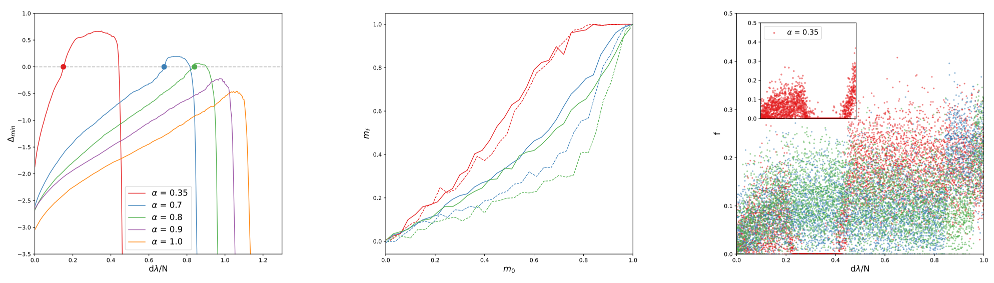
$\vec{S}^{\mu_d}$ must have an overlap m_t in the interval $(0, 1/\sqrt{N})$ with one memory μ_d, otherwise the procedure is repeated.

2. A $T = 0$ temperature dynamics in the landscape of $E(\vec{\chi}|m, J)$ is performed until $E(\vec{\chi}|m, J) < 0$. We use a Kawasaki kind of dynamics over the noisy variables $\vec{\chi}$ to make sure that m_t maintains the prescribed value.

Since $E(\vec{\chi}|m, J)$ is proportional to $\delta\mathcal{L}_U$ (see eq. (2.16)), the procedure will lead to a reduction in the Loss in eq. (2.7). In this setting, the perfect-retrieval and robustness properties can be tuned by the parameter m, while the training configurations always have a fixed value of $m_t = 0^+$. In particular, to require a performance that is most similar to the one of a SVM, we will set $m \to 1^-$.
The sampling procedure starts from fixed points because we know, from the previous sections, that they are close to be the most effective configurations. Interestingly, the described procedure results significantly more effective than a standard minimization of $\mathcal{L}(m = 1^-, J)$: in the latter case training stops when $\mathcal{L} = -1$, while the former technique apparently pushes the stabilities further in the positive values.

2.6.1 Algorithm performance

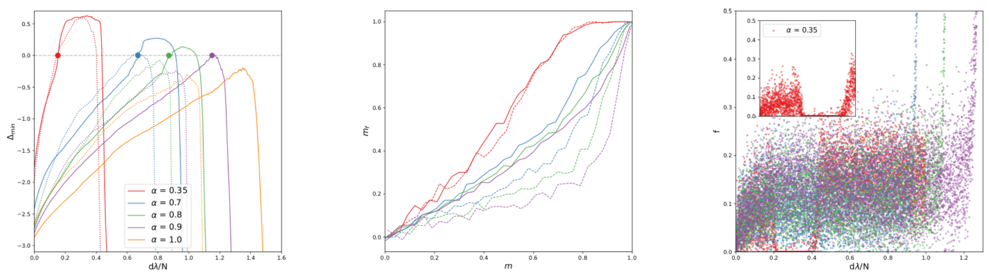
The sampling procedure results in a better performance for both TWN and the HU update rules. Results for TWN are reported in fig. 2.30. Panel (a) shows that



(a) **(b)** **(c)**

Figure 2.30. Performance of the TWN algorithm taught with noisy configurations sampled according to section 2.6 regarding four progressing values of the load α. (a) Minimum stability as a function of the algorithm time: *full* line is the HU with sampling, *dotted* line is the traditional HU. (b) Retrieval map $m_f(m_0)$, relatively to the *circles* in panel (a) for $\alpha \in [0.35, 0.7, 0.8]$: *full* line is the HU with sampling, *dashed* line is a SVM trained with no symmetry constraints with the same control parameters. (c) Saddle index f as a function of the algorithm steps for $\alpha \in [0.35, 0.7, 0.8]$. The sub-panel zooms over $\alpha = 0.35$ alone. All data points have been averaged over 20 samples in (a),(b) and 5 samples in (c). Errors are neglected for clarity of the image. The choice of the parameters: $N = 100$, $\lambda = 10^{-3}$, $m = 0.9999$.

perfect-retrieval is reached up to $\alpha \simeq 0.8$ for a network of size $N = 100$. Panel (b) shows the retrieval map $m_f(m_0)$, for different values of α and for the lowest number of algorithm iterations leading to perfect-retrieval. The high values of m_f for $m_0 \sim 1$ indicates that the network, while achieving perfect-retrieval, maintains



(a) (b) (c)

Figure 2.31. Performance of the HU algorithm trained with noisy configurations sampled according to section 2.6 regarding four progressing values of the load α. (a) Minimum stability as a function of the algorithm time: *full* line is HU with sampling, *dotted* line is the traditional HU. (b) Retrieval overlap $m_f(m_0)$, relatively to the *circles* in panel (a) for $\alpha \in [0.35, 0.7, 0.8, 0.9]$: *full* line is HU with sampling, *dashed* line is a SVM trained with no symmetry constraints with the same control parameters. (c) Saddle index f as a function of the algorithm steps for $\alpha \in [0.35, 0.7, 0.8, 0.9]$. The sub-panel zooms over $\alpha = 0.35$ alone. All data points have been averaged over 20 samples in (a),(b) and 5 samples in (c). Errors are neglected for clarity of the image. The choice of the parameters: $N = 100$, $\lambda = 10^{-3}$, $m = 0.9999$.

good robustness properties. Results for a SVM with the same control parameters are also shown, for comparison. The curves obtained from TWN are higher than ones relative to the SVM, signaling a better recalling performance, even though quantifying this effect will require a more detailed study of finite size effects. Fig. 2.31 shows analogous plots for the HU update rule. Again, the network compares favorably with traditional HU in terms of memory capacity, that is increased up to $\alpha \simeq 0.9$ (see panel (a)) with respect to the maximum capacity of $\alpha \simeq 0.7$ obtained with fixed points only. Panel (b) represents an indication for the sizes of the basins of attraction at the lowest number of algorithm iterations leading to perfect-retrieval. Yet again basins appear to be larger than the ones from a SVM trained with the same control parameters.

Panel (c) in fig. 2.30 and fig. 2.31 report the behavior of the saddle index f of the sampled configurations as a function of the number of steps of the algorithm, relatively to TWN and HU respectively. Most of the sampled training configurations are low index saddles. When α is low, there exists an interval in the training process when stabilities are all far above zero. During this interval, the exponential factors in eq. (2.54) are all extremely small, resulting in negligible improvement of the Loss function regardless of the training configuration chosen by the sampler. Correspondingly, the sampler contents itself with fixed points of the dynamics, resulting in the gap in the saddle bands visible in the insets of fig. 2.30 and fig. 2.31 (c) for $\alpha = 0.35$. For higher values of α, i.e. when the training task is more challenging, the algorithm never succeeds to increase Δ_{min} to the point of making sampling trivial, and there is no gap in the saddle band.

Checkpoint

In this section we have seen that:

- Instead of sampling specific states in the energy landscape of the model, i.e. saddles of a given index or local minima, one can directly sample the optimal noisy training configurations, i.e. the best configurations satisfying eq. (2.20) for each step of the training algorithm.
- The network obtained by sampling directly the best noisy training data outperforms both the TWN algorithm and HU in terms of maximum retrieval capacity and robustness.

2.7 Summary & Conclusions

In this chapter we have seen that:

- The training-with-noise algorithm [68] converges to a network that is fully described by the statistical mechanics developed by Wong and Sherrington in [65, 66]. For any value of α, when the training overlap is $m_t = 0^+$ the algorithm trains a Hebbian network; when $m_t = 1^-$ the same procedure trains a Support Vector Machine.

- The symmetric version of the training-with-noise algorithm can modified to learn configurations having $m_t = 0^+$ and satisfying the condition (2.20): while Gardner's standard algorithm would have trained a Hebbian network, now the resulting network approaches a Support Vector Machine.

- The initial conditions for the algorithm are important. When the model is initialized in a Hebbian network, stable fixed points and low saddles in the energy landscape having $m_t = 0^+$ with respect to the memories abundantly satisfy the optimal noise condition in eq. (2.20), favouring the training of a Support Vector Machine. Since these states are very far from the memories the procedure can go fully unsupervised (by means of a relaxation dynamics or a Monte Carlo sampling at low temperatures), increasing in speed and biological plausibility.

- The traditional Hebbian Unlearning algorithm is contained into the training-with-noise algorithm, when $m_t = 0^+$ and the sampled states are stable fixed points of the dynamics. Since such states are good noisy configurations, according to the criterion in eq. (2.20), the Unlearning routine must train a network that resembles a Support Vector Machine. This statement is supported by detailed numerical evidence.

- Memories containing an internal structure in the features (e.g. memories that can be mapped into positions in a Euclidean space, hence having strong spatial correlations), yet being well separated in the configuration space, can be learned by the modified training-with-noise algorithm and Unlearning. Memories that have both an internal structure and mutual correlations among each other fail to be learned by the same methods.

For the purpose of statistical mechanics, our focus in this study is on disordered systems, specifically in the context of associative memory, which is a prominent area of research. The work by Abbott and Kepler [80] investigates the learning of memory collections as a flux in Gardner's space of interactions [15]. This flux exhibits three distinct fixed points, or universality classes, representing *unique* (or *saturated*) solutions to the perfect-retrieval problem: the Hebbian matrix [30, 11, 3], the Pseudo-inverse matrix [81, 82], and the Support Vector Machine (SVM) [38, 6]. While we understand the retrieval properties of Hebbian networks, which do not achieve proper perfect-retrieval, it is important to note that Pseudo-inverse matrices achieve perfect-retrieval up to α being unity. However, they also represent overfitting systems, as memories have null basins of attraction when $\alpha > 1/2$. In this context, the training-with-noise algorithm serves as an interesting interpolation between two of these fixed points: the Hebbian point (corresponding to $m_t = 0^+$) and the SVM (corresponding to $m_t = 1^-$). By considering the noise injection of the training-with-noise algorithm as a regularization in the perceptron learning process, we introduce another degree of freedom, which is the *noise structure*. When the noise is i.i.d., an SVM can be trained with $m_t \to 1^-$. However, if the training data satisfy the condition for optimal noise, we only require $m_t = 0^+$ to approach the SVM fixed point, at least for α values that are not too high. Our work demonstrates that Hebbian Unlearning represents this type of flux in the space of interactions, challenging the prevalent belief in the statistical mechanics community that Unlearning leads the connectivity matrix to resemble a Pseudo-inverse matrix rather than a perceptron, particularly the most stable one. This supports the idea that Unlearning serves as an effective and unsupervised (hence more biologically relatable) training procedure for associative memory models.

Moving on to the potential implications of our work in biology and artificial intelligence, empirical observations indicate that the mammal brain is a highly efficient learner. It requires only a few examples to memorize concepts and generalize them [83, 84]. Some brain regions seem to function similarly to perceptrons [39, 40], even though natural learning is unlikely to be supervised. Neuroscientists increasingly emphasize the significance of sleep for memory consolidation [85, 86], and intriguing connections emerge between unsupervised training algorithms in machines and the necessity of specific synaptic plasticity processes occurring separately from daily experiences [19, 87, 88, 20, 89, 90]. Hoel [91] speculates on the importance of dream-sleep to avoid overfitting in the brain by injecting noise through hallucinoid contents, thereby enhancing robustness. Based on these works, a local Hebbian-like action on synapses can ensure decorrelation of stored memories and prevent confusion. Contrary to past literature repeatedly stating Hebbian Unlearning as a form of *reverse* learning, our research reveals that it is responsible for learning coherent noisy versions of memories, helping minimize overfitting.

It is essential to emphasize that the theoretical neuroscientists [92, 93, 94] simultaneously, yet independently, proposed the importance of an anti-Hebbian rule in learning. This rule emerged as a useful mechanism for decorrelating information in the system and maximizing the quality of the retrieval process. Scientists also conjectured the potential homeostatic role of such inverse processes, observed indirectly in real networks of neurons, indicating a tendency for the total synaptic volume to be conserved in the brain [95, 96]. The fact that this inverse rule spontaneously arises from noisy perceptron learning (see eq. (1.15)), encourages the possibility of observing it in future neurophysiology experiments.

In light of these findings and the work presented in this article, one could conceive natural learning as a *two-phase* process, making use of a single synaptic modification rule. In the first *online* phase, external stimuli are processed by the network using the standard training-with-noise algorithm. These stimuli can be imagined as maximally noisy versions of unknown *archetypes* embedded in the environment. Consequently, the training shapes a pure Hebbian landscape of attraction outside the retrieval regime. In the second *offline* phase, the early-formed network samples structured noisy neural configurations, weakly correlated with the archetypes, from the landscape of attractors. These states could be lower saddles or stable fixed points of the neural dynamics. Subsequently, when these neural configurations undergo the same kind of training-with-noise algorithm, memory is consolidated by centering the unconscious archetypes in the middle of large basins of attraction.

In summary, our work makes progress on three fronts. Firstly, it advances a new type of regularization for associative memory models that strongly relies on the topology of the landscape of attractors in spin-glass-like neural networks. Future developments of these studies might focus on characterizing the structure of noise more rigorously, possibly computing a proper density distribution for the effective weights in eq. (2.20).
Secondly, it sheds light on the specific structure of noise that is optimal for learning in neural networks, which may contribute to develop a more refined theory underlying the empirical techniques of noise injection used in training deep networks [62, 63, 97, 98]. Finally, it establishes a connection between unsupervised learning processes, which are more biologically relevant, and the supervised ones that underpin most modern neural network theory. This, in turn, encourages a deeper investigation into the noise and its structure present in the stimuli used by the brain to effectively shape associative memory.

Chapter 3

The inferential power of Unlearning

We are now going to deal with Boltzmann Machines (BMs), a class of neural networks introduced in section 1.3.1. BMs accomplish a generative task, i.e. the typical neural states are meant to be indistinguishable from the data that were used to train the network. One can thus sample new data from the probability distribution of the model. For what concerns this kind of systems it is useful to define the concepts of *overfitting* and *generalization*.
In the context of BMs, three probability distributions play a role: the inferred model P_{mod}, the empirical distribution of the training data P_{data} and P_{true}, i.e. the real hidden distribution from which data are sampled. We say that the inferred model is *overfitting* (the data) when P_{mod} resembles P_{data} more than P_{true}, i.e. the model fits the training data better than it does with unseen examples generated from the same distribution. This translates into an over-specialization of the model, namely it becomes too focused on the details of the specific training-set, instead of understanding the broader structure highlighted by the data. Conversely, when the performance of the model does not change significantly when it is trained with a new unseen set of data generated from the same source, we will say that it *generalizes* well.
There are many aspects that impede neural networks to generalize well. For instance, BMs always tends to overfit the training data, by construction. Though, if we assume the data to satisfy the law of large number, and the size of the training-set is sufficiently large, one can approach the true hidden distribution. In principle, if all the possible data are used for training, all the realizable testing data will be included in the statistics. However, when N is large enough, the number of available data might not be enough to reach a good degree of generalization. A general idea from statistics wants overfitting to be related to an over-parametrization of the model: there are too many degrees of freedom that the model can employ to reproduce the training data in detail. This argument holds for BMs, that fit the first two moments of a probability distribution by minimizing a Loss is a simple way. Nevertheless, this idea seems not to be valid in deep neural networks, which instead benefit from over-parametrization [99, 100] . In this case the gradient of the Loss must be performed by computing quantities over the several layers: this makes the

process complex and barely controllable. Since real data are not generated by a Boltzmann distribution, the real statistics of the data cannot be fully known, i.e. the loss of the BM learning problem cannot be fully minimized. Nevertheless one can employ regularization methods to minimize the distance between the the model and the real statistics of the data while, at the meantime, selecting specific models in this region of comparable loss. This provides different inferred systems with different properties, depending on additional requirements imposed on the inferential problem. For instance, some of these models can be sparse or even contain null parameters (e.g. L_0, L_1-norm and information based regularization schemes [101]), others are fully connected (e.g. L_2-norm method). Past literature [102, 103, 104] highlighted that critical models, i.e. inferred systems being highly susceptible to small changes in the parameters, can be attractive for BM learning. In fact, if we consider training as an homogeneous sampling of the models that minimize $\mathcal{L}$, critical models have a large basin of attraction and are thus sampled most often [104]. Criticality can be problematic for data generation: it implies a long correlation time in the Monte Carlo dynamics, slowing down the sampling process; it makes the model susceptible under rescaling of the parameters, reducing its predictivity in real data applications. Hence, avoiding criticality might help increasing generalization. To summarize, we define the generalization performance of the model according to two properties: *accuracy*, i.e. the capability of the model to be as similar as possible to the distribution that generated the data; *robustness*, i.e. the tendency of the model not to change its typical configurations when slightly changing (e.g., rescaling by a common temperature factor) the parameters. The goal of this chapter is to introduce a new type of regularization in order to find a better compromise between these two requirements. This new tool, that we named Unlearning regularization, appears to be more effective than other techniques in approaching a higher generalization performance of the neural network. The performance of the Unlearning regularization is evaluated from artificial data generated by both a Curie-Weiss (CW) and a Sherrington-Kirkpatrick (SK) model with a small number of neurons. In this way the original parameters are known, the fixed points of the learning equations are reached precisely and the useful quantities can be computed exactly.
A particular limit of this type of regularization coincides with a thermal version of the traditional Hebbian Unlearning (HU) algorithm studied in chapter 2. We hence evaluate this particular case of study and conclude that HU is able to infer the original data distribution with a very good degree of generalization. The performance shows an optimum in time that scales with the control parameters, as it was for the associative memory task. We conclude that HU can be interpreted as a two-steps BM learning procedure. Finally, our last contribution is a study of the under-sampling regime of BMs in the case of random data, which provides for a new linking trait between generative models and the best performing associative memory models.

The structure of this chapter is the following. Some extensively employed regularization techniques (i.e. $L1$ and $L2$) are firstly introduced in section 3.1. Then the new Unlearning regularization is defined in section 3.2 and its generalization performance is analyzed for two different data sources: a CW model (section 3.2.1) and a SK model (3.2.2). The study of a particular limit of the regularization technique leading to Unlearning follows in 3.3, with an explanation for its inferential behaviour. Eventually, in section 3.4, we identify a common trait between SVMs and BMs in their under-sampling regime, i.e. when the number of training data is small.

3.1 Regularization in Boltzmann Machines

L_p regularizations are certainly the most famous regularization methods, also due to their intuitive interpretation. This approach penalises high values of the parameters by adding a L_p norm of the same variables in the expression of the Loss function. As a consequence, the sparsity of the graph is incentivized: the redundant parameters are weakened with respect to the relevant ones. The expression of the Loss of a BM with L_1 and L_2 regularizations is given by the following equations

$$\mathcal{L}_1 = \mathcal{L}^{BM}(J, \vec{h}) + \gamma_1 \sum_{i,j>i} |J_{ij}| + \gamma_1 \sum_i |h_i|, \tag{3.1}$$

$$\mathcal{L}_2 = \mathcal{L}^{BM}(J, \vec{h}) + \frac{\gamma_2}{2} \sum_{i,j>i} J_{ij}^2 + \frac{\gamma_2}{2} \sum_i h_i^2, \tag{3.2}$$

where $\mathcal{L}^{BM}$ is defined in eq. (1.25) and γ_1, γ_2 two small regularization rates. This translates into new updating rules for the parameters, i.e.

$$\delta J_{ij}^{(1)} = \delta J_{ij}^{BM} - \lambda \gamma_1 \text{sign}(J_{ij}) \qquad \delta h_i^{(1)} = \delta h_i^{BM} - \lambda \gamma_1 \text{sign}(h_i), \tag{3.3}$$

$$\delta J_{ij}^{(2)} = \delta J_{ij}^{BM} - \lambda \gamma_2 J_{ij} \qquad \delta h_i^{(2)} = \delta h_i^{BM} - \lambda \gamma_2 h_i. \tag{3.4}$$

We are now interested in a criterion that describes the generalization performance of a BM. As mentioned before, we divide generalization into two features: accuracy, i.e. the similarity of P_{mod} to P_{true}, in terms of a measure in the space of the probability distributions; robustness, i.e. capability of the model to change slightly under variations of the parameters.
The first aspect can be quantified by the Kullback-Leibler divergence $D_{KL}(true|mod)$ between the inferred model and the original one, defined as

$$D_{KL}(true|mod) = \sum_{\vec{S}} P_{true}(\vec{S}) \log \left(\frac{P_{true}(\vec{S})}{P_{mod}(\vec{S})} \right). \tag{3.5}$$

This quantity is not symmetric under the exchange of the two distributions, which can be inconvenient to define a distance. We can then adopt a symmetric version of the divergence as

$$sD_{KL}(true, mod) = D_{KL}(true|mod) + D_{KL}(mod|true). \tag{3.6}$$

Even if numerical results do not show a significant difference between these two quantities, we will mainly adopt sD_{KL}.
Regarding the second trait, it is known that Ising-like models inferred via BM learning are close to be critical [102, 103, 104]. Criticality is associated with a susceptibility of the system to small variations of the temperature and can be measured through the *specific heat* [105]. This observable is defined as

$$C_v(\beta) = -\beta \frac{\partial S_\beta}{\partial \beta} = \beta^2 \left(\langle E^2 \rangle_\beta - \langle E \rangle_\beta^2 \right), \tag{3.7}$$

where S_β is the entropy of the model at a given inverse temperature β. Notice that eq. (3.7) defines a susceptibility with respect to the energy E, which also uniquely determines the model with its parameters. The fact that inferred neural networks typically display a peak in C_v around the value of β used for training implies the approaching of a critical behaviour at that same temperature (the finite size of the system impedes C_v to properly diverge and to show a real criticality). Therefore, since all parameters naturally scale with β in eq. (1.23), C_v is a measure of the sensibility of the model to a small perturbation of the parameters: high values reached by C_v suggest strong variations of the inferred statistics to small variations of the parameters. From now on we will consider the quantity $C_v(\beta)/\beta$ for $\beta \neq 1$ or, equivalently, $C_v(1)$ since they properly represent the variation of a thermodynamic quantity, i.e. the entropy of the model, with respect to the inverse temperature β.

Checkpoint

In this section we have seen that:

- L_p regularizations are used in BMs to weaken the redundant parameters with respect to the relevant ones.
- The generalization capability of a BM is declined in two different properties: the similarity between P_{mod} and P_{true}; the robustness of P_{mod} under perturbations of the parameters.
- The distance between the inferred model and the true distribution of data can be measured through the Kullback-Leibler divergence; the robustness of the model under a perturbation of the parameters is signaled by a flattening of the rescaled specific heat $C_v(\beta)/\beta$, or equivalently $C_v(1)$ when the model is inferred at $\beta = 1$.

3.2 Unlearning regularization

We now propose a new type of regularization that has, as a goal, to impose the maximum robustness under rescaling of the parameters, i.e.

$$J \longrightarrow aJ \qquad \vec{h} \longrightarrow a\vec{h} \qquad \text{with} \quad a \geq 0. \tag{3.8}$$

We can interpret this operation as a redefinition of the inverse temperature β in the model. We will evaluate the performance of this method on different types of data-sets, then we will compare it to L_p regularizations showing a gain in the generalization capability of the network.

For the model to be robust under a redefinition of the parameters, we can add a regularization term to the traditional BM loss function that shifts the peak of the specific heat (i.e. the critical temperature) away from $\beta = 1$, where the data are generated. Hence, let us define the following loss function

$$\mathcal{L}(J, h|a) = D_{KL}(data|1) + \left(\frac{a-1}{a}\right) D_{KL}(data|a), \tag{3.9}$$

with

$$D_{KL}(data|\beta) = \sum_{\vec{S}} P_{data}(\vec{S}) \log\left(\frac{P_{data}(\vec{S})}{P_\beta(\vec{S})}\right). \tag{3.10}$$

Notice that the chosen distance between the two distributions is asymmetric, but the reason for this choice will be evident in a few lines. The gradient descent equations for this Loss function become

$$\delta J_{ij} = \lambda\left[a\left(\langle S_i S_j\rangle_{data} - \langle S_i S_j\rangle_a\right) - \left(\langle S_i S_j\rangle_1 - \langle S_i S_j\rangle_a\right)\right], \tag{3.11}$$

$$\delta h_i = \lambda\left[a\left(\langle S_i\rangle_{data} - \langle S_i\rangle_a\right) - \left(\langle S_i\rangle_1 - \langle S_i\rangle_a\right)\right]. \tag{3.12}$$

When $a = 1$ equations (3.11), (3.12) coincide with the original BM learning. In the limit $a \to 0$ one has $\langle S_i S_j\rangle_{a=0} \to 0$ when $N \gg 1$ and eq. (3.11) tends to a thermal version of HU (see eq. (1.10)) performed at $\beta = 1$, i.e.

$$\delta J_{ij} = -\lambda\langle S_i S_j\rangle_1,$$

which is similar to previous attempts from the associative memory literature [106]. On the other hand, when $a \to \infty$, and the learning rate is redefined as $\lambda = \mathcal{O}(a^{-1})$, the algorithm becomes

$$\delta J_{ij} = \lambda\left(\langle S_i S_j\rangle_{data} - \langle S_i S_j\rangle_\infty\right)$$

which also resembles the Unlearning procedure. As a difference with the standard routine, there is a Hebbian *input* term $\langle S_i S_j\rangle_{data}$ which impedes the couplings to vanish at convergence.

At this point, the choice of the expression for $\mathcal{L}$ should appear more clear: we search for an algorithm that interpolates between the BM learning algorithm and HU. In this way HU emerges as a particular limit of a regularization method on BMs.

For the rest of our study we will deal with a small sized network, i.e. $N \sim 18 \div 20$, which will allow to reach the fixed points of equations (3.11) and (3.12) precisely, whether such points are admitted, with no errors due to the finite sampling. We can then can compute the observables C_v/β or sD_{KL} more easily, in order to determine the generalization performance of the system. For simplicity of notation we will rename

$$\langle S_i S_j \rangle_{data} = c^d_{ij} \qquad\qquad \langle S_i S_j \rangle_{\beta} = c^{\beta}_{ij},$$

where we will mainly deal with $\beta = a$ and $\beta = 1$. The inverse temperature of the generating model is $\beta = \beta_d$. The generating models for the data will be of two kinds: a mean field fully connected Ising network (i.e. Curie-Weiss [107]) and a fully disordered network (i.e. Sherrington-Kirkpatrick [71]) which are both critical in the thermodynamic limit, i.e. they admit a divergence of the specific heat. The divergence appears as a peak in the finite size case. Solving the BM learning algorithm equations would lead to the exact parameters of the generating model, that approaches a criticality at finite N. Nevertheless, solving equations (3.11) and (3.12) leads to a non-trivial network with its own generalization capability.

3.2.1 Data generated from a Curie-Weiss model

Let us consider the case of inferring a Curie-Weiss model (CW), that is defined by the following energy function

$$E_{CW}[\vec{S}|J] = -\sum_{i,j>i} S_i J_{ij} S_j \qquad\qquad J_{ij} = \frac{J}{N} \qquad \forall i,j. \tag{3.13}$$

This model can be treated fully analytically in the finite size case, and the solution procedure for the main quantities is here reported. We will make use of the algorithm presented in eq. (3.11) and eq. (3.12). In this case correlations can be obtained exactly even in the finite N case. Since fields are zero, by construction of the model, we are interested in the evolution equation for the couplings, i.e.

$$\dot{J} = \lambda \left[a \left(c_d - c_a(t) \right) - \left(c_1(t) - c_a(t) \right) \right], \tag{3.14}$$

where we used the fact that both couplings and correlation functions assume the same values $\forall i, j$. Notice that all elements of the matrices are identical, because each node receives the same fields from its neighbours. The fixed point equation for the correlation functions from eq. (3.14) is

$$c^*_a = \frac{c^*_1}{1-a} - \frac{a}{1-a} c_d, \tag{3.15}$$

where the star indicates the value of the correlations at convergence. Moreover we know that

$$Z_{\beta J} = \sum_m \binom{N}{\frac{N}{2}(1+m)} \exp\left(\frac{\beta J}{2} (N m^2 - 1) \right), \tag{3.16}$$

where $m \in [-1, -1 + \frac{2}{N}, .., 1 - \frac{2}{N}, +1]$ and we used

$$E[\vec{S}] = -J \left(N \left(\frac{1}{N} \sum_i S_i \right)^2 - 1 \right) = -J \left(N m^2 - 1 \right). \tag{3.17}$$

Then one has that the second moment of the magnetizations $\langle m^2 \rangle_\beta$ with respect to the Gibbs-Boltzmann measure can be written as

$$I_N(\beta J) = \frac{\sum_m m^2 \binom{N}{\frac{N}{2}(1+m)} \exp \frac{\beta}{2} J(Nm^2 - 1)}{\sum_m \binom{N}{\frac{N}{2}(1+m)} \exp \frac{\beta}{2} J(Nm^2 - 1)} = \frac{1 + (N-1)c(\beta J)}{N}. \tag{3.18}$$

Thus

$$c(\beta J) = \frac{N I_N(\beta J) - 1}{N - 1} \tag{3.19}$$

The fourth moment of the magnetization $\langle m^4 \rangle_\beta$ can be written as

$$I_N^{(4)}(\beta J) = \frac{\sum_m m^4 \binom{N}{\frac{N}{2}(1+m)} \exp \frac{\beta}{2} J(Nm^2 - 1)}{\sum_m \binom{N}{\frac{N}{2}(1+m)} \exp \frac{\beta}{2} J(Nm^2 - 1)} = \frac{1 + (N^3-1)k(\beta J)}{N^3}, \tag{3.20}$$

where $k(\beta J)$ is the fourth-order correlation among spins. Eq. (3.15) becomes

$$c(aJ) = \frac{c(J)}{1-a} - \frac{a}{1-a} c(\beta_d J), \tag{3.21}$$

or equivalently

$$I_N(aJ) = \frac{I_N(J)}{1-a} - \frac{a}{1-a} I_N(\beta_d J), \tag{3.22}$$

that has be to solved for J.
The specific heat of the inferred model can be computed as

$$C_v(\beta) = \left(\frac{\beta N J(a)}{2}\right)^2 \left(I_N^{(4)}(\beta J(a)) - {I_N^{(2)}}^2(\beta J(a))\right). \tag{3.23}$$

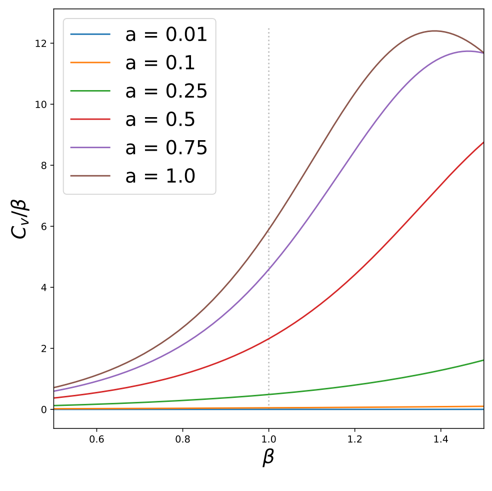
Equation (3.22) is solved for a network of $N = 20$ neurons and results are reported in fig. 3.1. As showed in fig. 3.1a the specific heat has a peak slightly after $\beta = 1$ that flattens progressively when a is decreased. Note that, for each value of a the couplings $J = J(a)$ are obtained from the calculations, but quantities are computed from the Gibbs-Boltzmann distribution with the rescaled parameters $\beta \cdot J(a)$. Moreover, in fig. 3.1b $J(a)$ is computed and the quantity $aJ(a)/\beta_d$ is plotted as a function of a for different choices of β_d. As one can notice, the lines approach a linear regime before reaching $J(1)/\beta_d = 1$. This behaviour signals that $J(a)$ is nearly constant in a, hence the inferred model changes little passing from $\beta = 1$ to $\beta = a$, as required by the regularization.

3.2.2 Data generated from a Sherrington-Kirkpatrick model

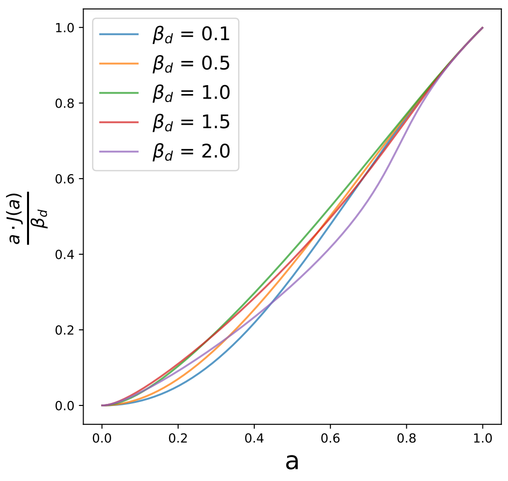
Let us consider the case of inferring a Sherrington-Kirkpatrick model (SK), i.e.

$$E_{SK}[\vec{S}|J] = -\sum_{i,j>i} S_i J_{ij} S_j \qquad J_{ij} \sim \mathcal{N}(0, N^{-1/2}) \tag{3.24}$$

For clarity of the results, we will use only one realisation of the parameters for the SK, having a peak in the specific heat represented in fig. 3.2. A network of



(a)



(b)

Figure 3.1. Inferred couplings and specific heat for a small network trained with a CW model with $N = 20$ spins. (a) The specific heat is reported in a window centered around $\beta = 1$ at different values of the parameter a. (b) The inferred couplings multiplied by a/β_d as a function of a at different values of β_d: the linear regime displays the robustness the parameters under rescaling.

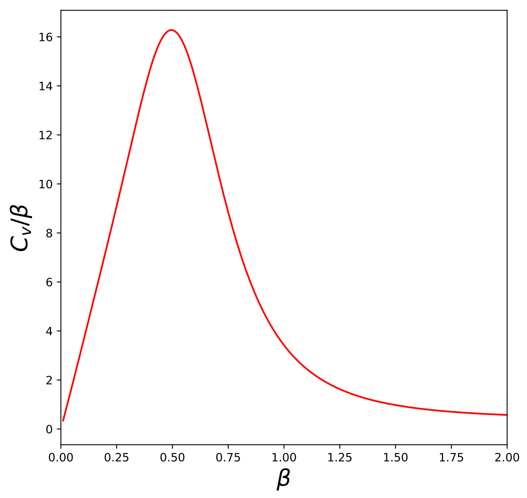


Figure 3.2. Specific heat C_v/β as a function of the inverse temperature β for the single SK spin-glass used in our study. The network contains $N = 18$ neurons.

$N = 18$ neurons is trained with a SK model at $\beta_d = 0.4$ with the new BM learning algorithm. The parameters are initialised as $J^{(0)} = c_d$ and $\vec{h}^{(0)} = 0$. Figure 3.3 plots the Pearson coefficient ρ for the 2-point correlation matrices and the sD_{KL} between the inferred model with $\beta = a$ and $\beta = 1$ and the original SK, both as functions of the algorithm steps and the parameter a. As one can conclude from fig. 3.3a and fig. 3.3b, the quality of the correlation between c_d and c_β is better for $\beta = 1$ with respect to $\beta = a$. This is probably due to the fact that the gradient deriving from $D_{KL}(data|a)$ in eq. (3.11) is weighted by the factor $(a-1)/a$, that

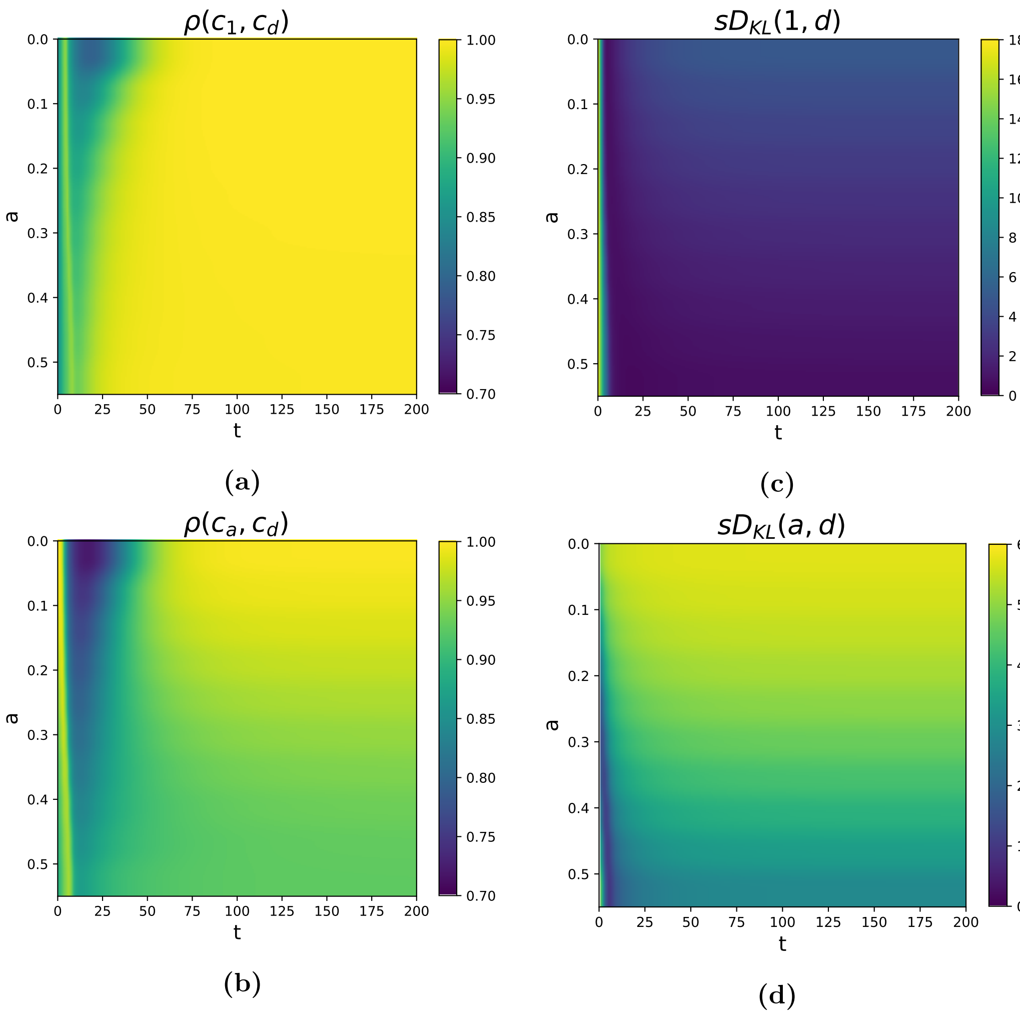


Figure 3.3. Relevant observables for a small network trained from a SK model as functions of the parameter a and the algorithm time t. The generating model for the data has a inverse temperature $\beta_d = 0.4$. Parameters are initialized in the standard way, i.e. $J^{(0)} = c_d$ and $\vec{h}(0) = 0$. $\rho(A, B)$ is the Pearson coefficient between the elements of the matrices A and B, while $sD_{KL}(\beta, d)$ is the symmetrized Kullback-Leibler divergence betwen the Gibbs-Boltzmann distribution at β and the original one that generated the data. Choice of the parameters: $N = 18$, $\lambda = 0.06$.

in our experiment is smaller than unity. The same trend is observed with sD_{KL} in fig. 3.3c and fig. 3.3d: the model with $\beta = 1$ approaches the original system way more closely than the one with $\beta = a$ at any value of a. Another important aspect is that both ρ and sD_{KL} show a transient regime in the first few training steps. In this regime ρ oscillates while the sD_{KL} reaches its minimum value, pushing the model to its best generalizing configuration of the parameters. This holds for both the models with $\beta = a$ and $\beta = 1$. Fig. 3.4 compares the standard Hebbian initialization of the couplings with other kinds of initializations, when $a = 0.3$: $J_{ij}^{(0)} = 0$ and $J_{ij} \sim \mathcal{N}(0, N^{-1/2})$ $\forall i, j$. The study is repeated for both $\beta = 1$ and $\beta = a$, showing a general better performance for $\beta = 1$, as was observed from the previous colour plots. Since the transient regime does not appear in the other two initializations, we conclude that the initialization of couplings plays a fundamental role in the training of a BM.

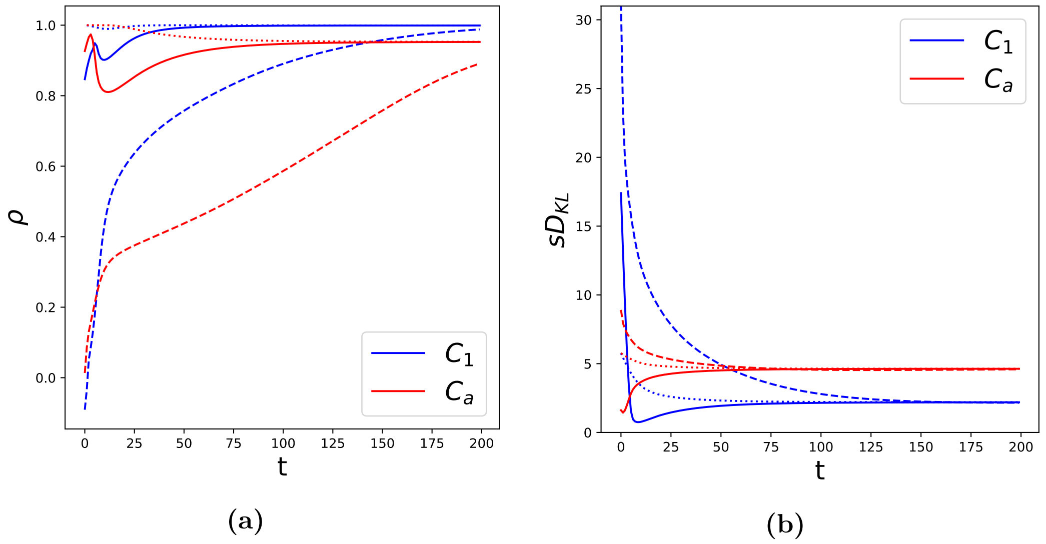


Figure 3.4. (a) The Pearson coefficient ρ between the correlation matrix c_β with $\beta = a$ and $\beta = 1$ and the data correlation matrix c_d. (b) The symmetric Kullback-Leibler divergence between the Gibbs-Boltzmann distribution at $\beta = a$ and $\beta = 1$ and the one that has generated the data, i.e. with $\beta_d = 0.4$. The *full* line reports the case of Hebbian initialization of the couplings; the *dashed* line corresponds to a random initialization; the *dotted* line represents the initialization to $J^{(0)} = 0$. Choice of the parameters: $N = 18$, $a = 0.3$, $\lambda = 0.06$.

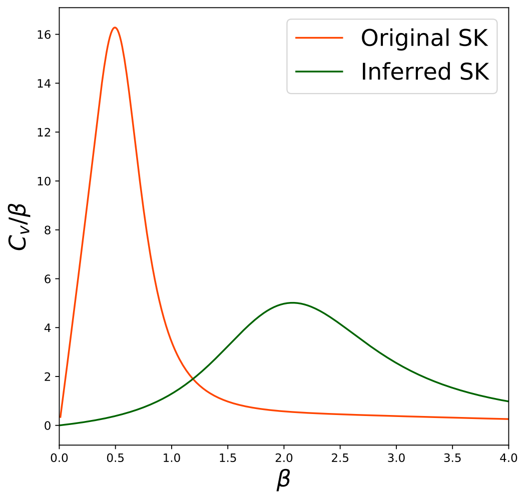


Figure 3.5. Specific heat C_v/β as a function of the inverse temperature β for the original SK spin-glass and the inferred one via the Unlearning regularization. Choice of the parameters: $N = 18$, $a = 0.3$, $\beta_d = 0.4$, $\lambda = 0.06$.

In conclusion, we compare the specific heat C_v/β of the original model with the curve obtained through the Unlearning regularization, at convergence of the update equation (1.28), always for $a = 0.3$. We want to stress that the experiments permits to compute the matrix J associated to the particular choice of a, but the specific heat at each β is relative to a Gibbs-Boltzmann distribution with rescaled parameters βJ. The results displayed in fig. 3.5 show that the inferred system has a lower peak, that is also shifted towards higher values of β, as we had observed in the CW case.
We conclude that the regularization has both reproduced the statistics of the original spin system at $\beta = \beta_d$ and reached a higher robustness under variation of the parameters, constructing a good generative model.

3.2.3 Comparing different regularization techniques

It is now necessary to compare the generalization performance of the Unlearning method with other kinds of regularization. We will consider, in particular the L_1 and L_2 types described in section 3.1. Data will be generated from a SK model with $\beta_d = 0.4$, as in section 3.2.2.

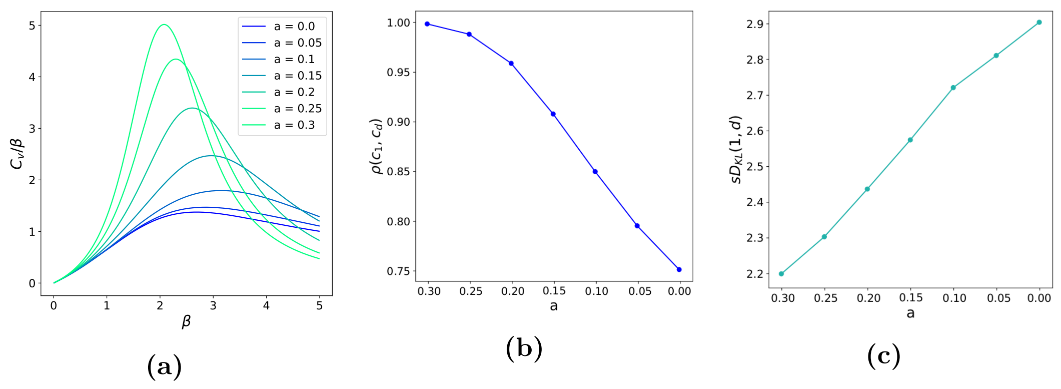


Figure 3.6. Results relative to a Boltzmann-Machine trained with the Unlearning regularization. (a) Specific heat C_v/β of the model as a function of the inverse temperature β. (b) Pearson coefficient between the correlation matrix of the model at $\beta = 1$ and the data correlation matrix c_d. (c) Symmetric Kullback-Leibler divergence between the model at $\beta = 1$ and the generating model. Choice of the parameters: $N = 18$, $\lambda = 0.06$.

To confront the networks obtained by the different methods we require the standard deviations of the coupling matrices to be comparable among each other. The network is initialized in the Hebbian fashion, i.e. $J^{(0)} = c_d$. The value of $a = 0.3$ is thus chosen and the couplings are optimized through eq. (3.11) until convergence. The standard deviation of the couplings is measured obtaining $\sigma_J \simeq 0.17$. Consequently the BM learning is repeated by adding the $L1$ and $L2$ regularizations. The rates γ_1 and γ_2 are chosen in order to obtain the same standard deviation σ_J at convergence of the algorithm. At this point the specific heat C_v/β, the Pearson $\rho(c_1, c_d)$ and the symmetric divergence $sD_{KL}(1, d)$ are measured across the three different techniques. The same procedure is reiterated at different values of a, γ_1, γ_2 that lead to the same value of σ_J: the only difference with the original experiment

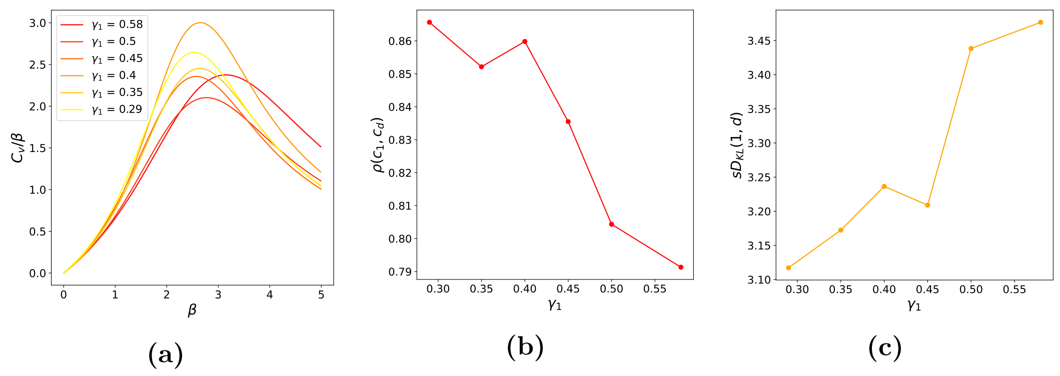


Figure 3.7. Results relative to a Boltzmann-Machine trained with the L_1 regularization. (a) Specific heat C_v/β of the model as a function of the inverse temperature β. (b) Pearson coefficient between the correlation matrix of the model at $\beta = 1$ and the data correlation matrix c_d. (c) Symmetric Kullback-Leibler divergence between the model at $\beta = 1$ and the generating model. Choice of the parameters: $N = 18$, $\lambda = 0.06$.

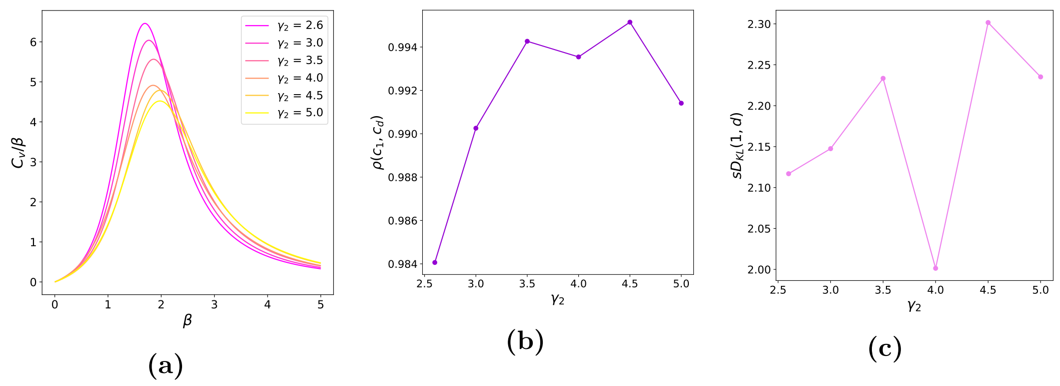


Figure 3.8. Results relative to a Boltzmann-Machine trained with the L_2 regularization. (a) Specific heat C_v/β of the model as a function of the inverse temperature β. (b) Pearson coefficient between the correlation matrix of the model at $\beta = 1$ and the data correlation matrix c_d. (c) Symmetric Kullback-Leibler divergence between the model at $\beta = 1$ and the generating model. Choice of the parameters: $N = 18$, $\lambda = 0.06$.

is that now eq. (3.11) is not iterated until convergence to the fixed point, but it is instead early-stopped as soon as $\sigma_J \simeq 0.17$ is reached by the system.
The results of the experiment are reported in figures 3.6, 3.7 and 3.8 for the three methods. The general trend shows that one cannot obtain a minimum specific heat (let us here consider $C_v(1)$ as the most representative observable to measure) without a Loss in the distance sD_{KL} between the inferred and the original model. Regarding the Pearson coefficient ρ, its behaviour can slightly change across the regularizations, but it is in general anticorrelated with sD_{KL}. Stronger fluctuations in the measures of ρ and sD_{KL} for the L_1 and L_2 regularizations are due to a the choice of λ that needs to be improved for further experiments.
The same results are resumed by the scatter plot in fig. 3.9. The generalization performance of each run is represented as a point with coordinates $(sD_{KL}(1, d), C_v(1))$:

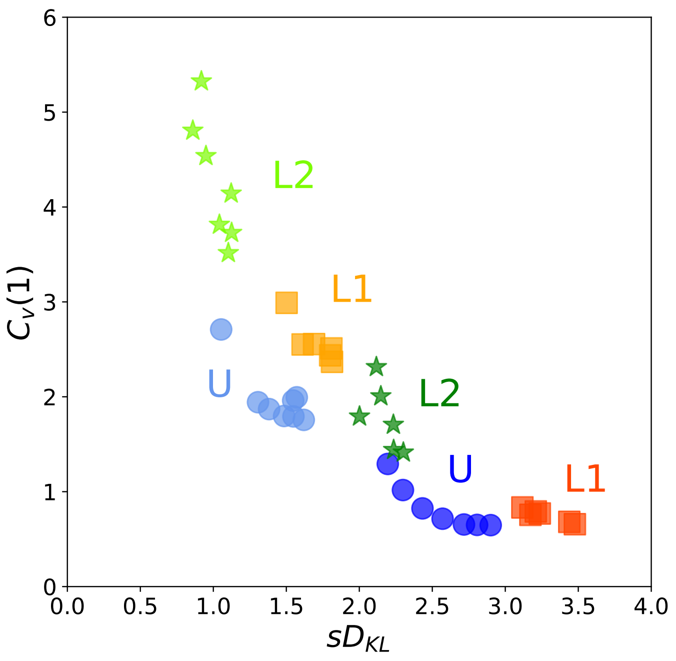


Figure 3.9. Graphical representation of the comparison among the three regularization techniques: U, L_1 and L_2. The performance of the network is benchmarked through measuring the specific heat at $\beta = 1$ versus the symmetric Kullback-Leibler divergence between the original model and the inferred one at $\beta = 1$. The point $(0, 0)$ represents the best generalization performance of the generative model. The resulting networks from the three regularizations have the same standard deviation σ_J of the couplings, specifically darker colours are relative to $\sigma_J \simeq 0.17$ while lighter colours are associated to $\sigma_J \simeq 0.26$. For each cluster of symbols, i.e. one regularization at a given σ_J, only one point is related to the fixed point of the BM learning equations: the rest of the points are obtained by early-stopping the algorithm at different values of a (U), γ_1 (L_1) and γ_2 (L_2) when σ_J matches the prescribed value.

each regularization has a different symbol and colour, darker colours represent the previous experiment with $\sigma_J \simeq 0.17$ while lighter ones refer to the same experimental procedure with $\sigma_J \simeq 0.26$. In both the experiments we can evidently see the Unlearning method to be closest to the origin of the plane, i.e. the best, yet unreachable, generalization power. The L_2 method generally reaches a lower sD_{KL} while keeping the specific heat high. Conversely, the L_1 technique flattens the specific heat significantly, while gaining in distance between the original and the inferred models. Unlearning, on the other hand, places itself in between the two methods, appearing as the best choice to increase the way the network predicts the original model.

Checkpoint

In this section we have seen that:

- The Unlearning regularization method requires the inferred model to be more robust under variation of the temperature (i.e. a total rescaling of the parameters) and at the same time interpolates between two different training algorithms: the standard BM learning (i.e. $a = 1$) and HU (i.e. $a = 0$).

- Both the cases of data sampled from a Curie-Weiss and a Sherrington-Kirkpatrick model show a good generalization performance obtained through the Unlearning regularization method.
- The Unlearning regularization outperforms L_1 and L_2 methods in generalization.

3.3 The Hebbian Unlearning limit

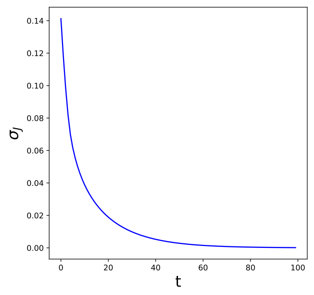
We will now study the specific limit of the Unlearning regularization leading to an algorithm which strongly resembles the HU routine. The inferential performance of the learning procedure is examined by measuring the distance between the inferred model and the original one at different training steps. Results show, for the first time, that HU can be employed as an inferential tool, due to the beneficial effect of the Hebbian initialization of the couplings. We advance an explanation for its performance that is supported by further numerical evidence.

In the limit $a \to 0$ the updating rules (3.11) and (3.12) become

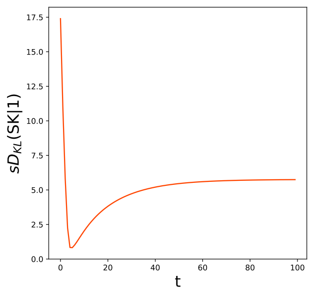
$$\delta J_{ij} = -\lambda \langle S_i S_j \rangle_1, \tag{3.25}$$

$$\delta h_i = -\lambda \langle S_i \rangle_1. \tag{3.26}$$

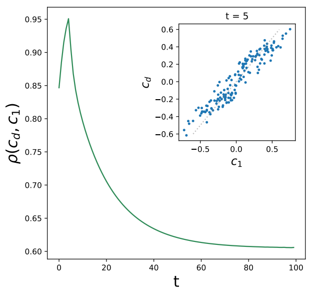
Eq. (3.25) strongly resembles the traditional HU algorithm in eq. (1.10). By contrast with the original rule, the new regularization samples configurations at $\beta = 1$ instead of stable fixed points of the neural dynamics, and makes use of a thermal average, rather than summing each contribution at each time step. We will keep dealing with small networks so that, given the initial conditions for the parameters, the Loss function of the problem can be minimized exactly. As a numerical experiment, we train a BM with $N = 18$ neurons to learn a SK model at $\beta_d = 0.4$. The Unlearning regularization is performed with $a = 0$ and the usual initial conditions for the parameters, i.e. $J^{(0)} = c_d$ and $h_i^{(0)} = 0 \; \forall i$.



(a)



(b)



(c)

Figure 3.10. Three relevant observables to benchmark the inferential performance of the Unlearning regularization with $a = 0$: the standard deviation of the couplings σ_J (a), the Kullback-Leibler divergence between the original SK model and the inferred one at $\beta = 1$ (b), the Pearson coefficient between the data correlation matrix c_d and c_1, i.e. the correlation matrix for the model at $\beta = 1$ (c). Choice of the parameters: $N = 18$, $\lambda = 0.06$, $\beta_d = 0.4$.

Even if training is performed over both the couplings and the fields, according to the rules (3.25) and (3.26), the fields $\vec{h}$ do not contribute to the energy of the original model, hence we will focus exclusively on the evolution of the interactions J. Fig. 3.10a displays the standard deviation of the couplings: as we can see the general

trend shows an exponential decay of the interactions, as one typically observes in the HU algorithm [31]. The decay to zero is implied by the fact that the fixed point of the gradient descent equations for the Boltzmann Machine must be found when $c_1 \equiv 0$. Fig. 3.10b depicts the symmetric Kullback-Leibler divergence between the Gibbs-Boltzmann distribution of the original SK model, from which data were sampled, and the inferred model with $\beta = 1$. The sD_{KL} is high at the beginning, it decreases until reaching a global minimum signaling an optimum in the inferential performance, then it increases again and stabilizes on a plateau. The minimum is sufficiently low to indicate a good statistical consistency between the model and data. The Pearson coefficient between the correlation matrix of the data c_d and the one of the inferred model at unitary temperature c_1 is measured step-by-step in the learning process and reported in fig. 3.10c: as one can notice, there is a global maximum near the position of the global minimum of the sD_{KL} that we previously identified. A sub-plot displays the good agreement between the two matrices, the inferred c_1 and c_d, at the position of the peak.
This analysis suggests that the Unlearning regularization with $a = 0$ displays two working regimes: one transient regime where the model at $\beta = 1$ shows a good statistical agreement with the generating model; another regime where $c_1 \to 0$, and the total performance deteriorates. The first transient regime is the most important one, because it suggests that HU can be read as inferential tool, aside of its associative memory use. We know from section 1.3.1 that in BM learning data are shown to the model each time-step of the algorithm and this imposes the moment matching. The co-existence of a positive Hebbian term and a negative Unlearning one in the traditional BM learning has already been pointed out by different works in the literature [20, 88]. In fact, the variation of each pair of couplings J_{ij} is given by

$$\delta J_{ij} = \delta^H J_{ij} + \delta^U J_{ij}, \tag{3.27}$$

where we neglected the dependence on time, and

$$\delta^H J_{ij} = \langle S_i S_j \rangle_{data}, \tag{3.28}$$

$$\delta^U J_{ij} = -\langle S_i S_j \rangle_{mod}. \tag{3.29}$$

We can interpret each step in the minimization of $\mathcal{L}$ as the result of two contributions, one constant quantity deriving from the data, and another one deriving from the evolving model. While a standard BM can start wherever in the space of the parameters and ends up at the fixed point of eq. (3.27), in the HU case the update is performed by using only the negative unlearning contribution. Nevertheless, the data have been seen by the model, specifically through the Hebbian choice of the initial conditions. This observation suggests that HU approaches the minimum of the loss function in two temporally separated steps: by first descending along the *data* direction and then, progressively, along the *model* one. As a consequence, we can assume that such a two-steps minimization equals the standard BM learning only while

$$|\langle S_i S_j \rangle_{mod}| \gg |\langle S_i S_j \rangle_{data}| \tag{3.30}$$

for most of the pairs i, j.
To test whether this condition holds while training a BM regularized with $a = 0$, and

initialized with $J^{(0)} = c_d$, we measure the standard deviation of ratio between the elements of c_1 and c_d as a function of time. Results are presented in fig. 3.11a, where

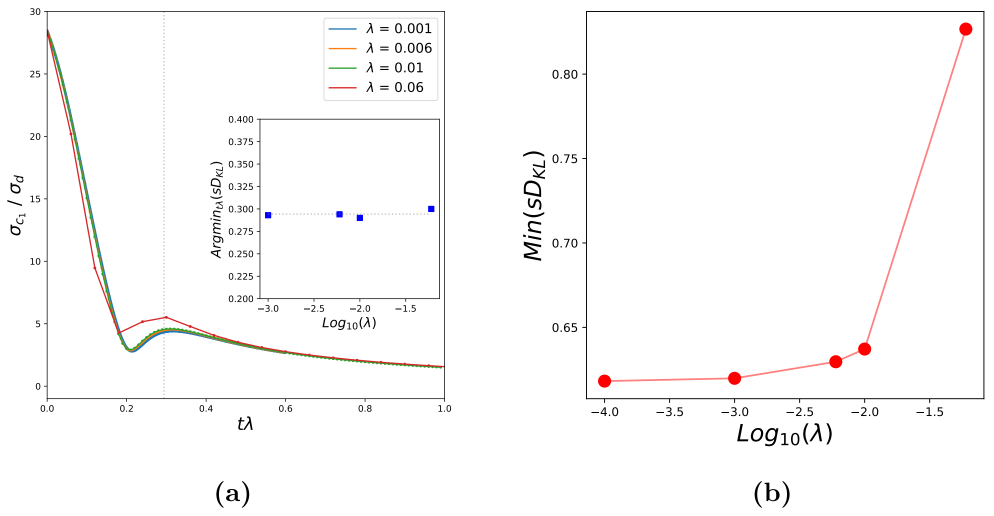


Figure 3.11. (a) Standard deviation of the ratio between the elements of c_1 and c_d as a function of the normalized time $t\lambda$; the subplot reports the values of $t\lambda$ associated to global minima of the symmetric Kullback-Leibler divergence (sD_{KL}) between the generating model and the inferred one for different choices of λ; both the vertical and horizontal gray dotted lines report the mean of the points in subplot. (b) Global minimum of the symmetric Kullback-Leibler divergence between the generating model and the inferred one as a function of λ in log-scale. Choice of the parameters: $N = 18$, $a = 0$, $\beta_d = 0.4$.

the standard deviation of the matrix obtained from the ratio of the elements of c_1 over c_d is plotted as a function of $t\lambda$ for different choices of λ. The curves collapse well from λ smaller than $\mathcal{O}(10^{-2})$. The system starts with c_1 generally being much larger than c_d, which satisfies the condition (3.30). Around $t \sim 0.2\lambda^{-1}$ the curves reach a local minimum, then increases until a local maximum at $t \sim 0.29\lambda^{-1}$, to start a slow decay right after. Both these two stationary points are located where c_1 and c_d share the same order of magnitude, i.e. where the condition for the two-steps minimization of the Loss ceases to hold. Specifically, the second stationary point is associated to the minimum of the sD_{KL} between the original SK and the model, as reported in the subplot contained in fig. 3.11a. Moreover, fig. 3.11b reports the values assumed by sD_{KL} at its global minimum for various choices of λ. As one can notice, the generalization of the model increases when λ is small: we can imagine that the smaller is λ the stronger is the effect of the initial overshooting at minimizing $\mathcal{L}_{data}$. We have thus showed that, when the network is initialized in the Hebbian fashion, the dominant contribution to the BM learning is the Unlearning one, and this is the reason for the HU algorithm to be a good inferential device.

Checkpoint

In this section we have seen that:

- The HU algorithm can be implemented as an inferential tool, i.e. to learn the joint probability distribution of the data, instead of storing a small subset of data-points.
- The quality of the inferential performance is due to the initial conditions for the parameters. When couplings are initialized in the Hebbian way a BM can significantly reduce the Loss function, and thus obtain a good neural network in two separate steps, and this two-steps procedure is the HU algorithm.
- Similarly to the associative memory scenario, the best inferential performance of HU is reached after an optimal amount of iterations of the training routine.

3.4 The analogy between a SVM and a BM in the under-sampling regime.

The aim of this section is to show that symmetric SVMs tend to satisfy the moment matching condition $c_d = c_\beta$ when β is sufficiently large (i.e. $\beta > 1$ in our case of study) and α is small, where we recall α to control the number of stored memories $p = \alpha N$. Since the moment matching condition must hold in BMs, the emergence of this property in SVMs must underline an important similarity between the memory performance of maximally stable linear perceptrons and networks trained through BM learning. The analysis of the landscape of attractors of a SVM can be a hard problem to tackle [29, 108]. However, the discovery of this new signature of the model can shed light on new dynamic properties of this type of networks. A physical interpretation of this observation, also given in the course of this section, relates this property of SVMs to the depth reached by the memories in the energy landscape, a topological trait that is, in turn, linked to the notable size of the basins of attraction. Eventually, by reason of the detailed comparison between SVMs and the HU algorithm advanced in chapter 2, we will conclude that the main learning algorithm examined in this work, i.e. HU, SVMs and BMs, all tend to train the same types of neural networks when the number of memories is small. This result is the very conclusive point of this thesis and it will be reached at the end of this section.

As usual in the associative memory framework we consider a number $p = \alpha N$ of N-dimensional memories that are generated as Rademacher variables $\{\vec{\xi}^\mu\}_{\mu=1}^p$ with

$$P(\xi_i^\mu = +1) = P(\xi_i^\mu = -1) = 1/2 \quad \forall i, \mu.$$

These memories are thus memorized by a symmetric linear perceptron with tunable margin k, by means of the algorithm (1.16). Let us define

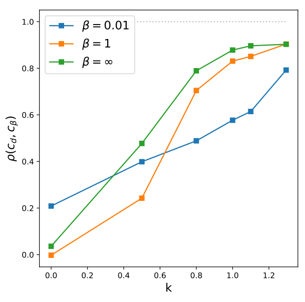
$$c_{ij}^d = \frac{1}{p}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu, \tag{3.31}$$

that is the Hebbian matrix of the memories. We also introduce

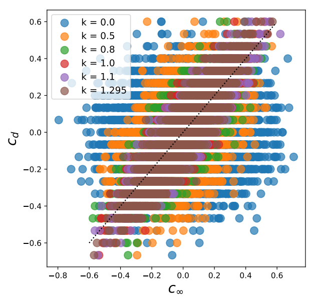
$$c_{ij}^\beta = \langle S_i S_j \rangle_\beta, \tag{3.32}$$

that is the thermal 2-point correlation matrix of the system, where $\langle \cdot \rangle_\beta$ is the probability measure according a Gibbs-Boltzmann distribution of the states at a temperature β^{-1} (see eq. (1.23)). This quantity can be practically estimated through a Monte Carlo routine.

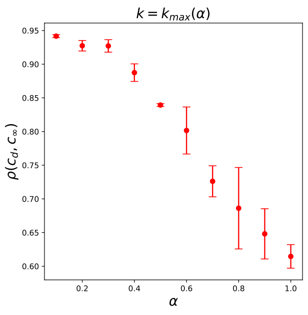
We now provide for some numerical evidence that SVMs approach a good matching between c_d and c_β for high values of β and lower values of α. We first evaluate the correlation between c_d and c_β in the linear perceptron trained at different values of k with the same set of memories. For this purpose, we measure the Pearson coefficient between the elements of the two matrices (see eq. (1.32)). Fig. 3.12a shows an increasing trend of ρ with respect to k. The maximum value is reached when $k = k_{max}(\alpha)$. Moreover, the correlation is higher for smaller temperatures. The same behaviour is displayed in fig. 3.12b for $\beta = 1$: c_d is not related to c_1 at all when $k = 0$, but the two quantities correlate very well when we tune the stability to its maximal value k_{max}. We have also measured how the quality of the moment matching approached when $k = k_{max}(\alpha)$ changes with α. Results are reported in fig. 3.12c and display a deterioration of this property when α progressively increases. As a conclusion, we expect the moment matching condition to be approached for sufficiently low temperatures and lower values of α.



(a)



(b)



(c)

Figure 3.12. Numerical measures relative to a symmetric linear perceptron trained through (1.16). The maximum margin is estimated empirically for each realization of the memories and random initial conditions of J. (a) Pearson coefficient between c_d and c_β as a function of the margin k, $\alpha = 0.3$. The plots for different values of β are displayed. (b) Correlation matrix c_d plotted as a function of c_∞, $\alpha = 0.3$. The black dotted line reports the line $c_d = c_\beta$. All points are relative to one realization of the network. (c) Pearson coefficient between the elements of c_d and c_β in a SVM (i.e. $k = k_{max}(\alpha)$) as a function of α. Points are averaged over five realizations of both memories and random initial conditions for J, the errors are the standard deviations of the measures. Choice of the parameters: $N = 100$, $\lambda = 0.01$.

We also propose another equivalent experimental study, that compares the empirical spectrum of the eigenvalues of the matrices c_d and c_1 for a symmetric SVM with the theoretical expected distribution. The network is chosen to have $N = 500$ and $\alpha = 0.3$. Since c_d is a Wishart matrix the density function of its eigenvalues if given by the Marchenko-Pastur distribution

$$MP(e|\alpha) = (1-\alpha)\,\theta(1-\alpha)\delta(e) + \frac{1}{2\pi}\frac{\sqrt{(e_+ - e)(e - e_-)}}{\alpha^{-1}e}, \tag{3.33}$$

with

$$e_\pm = (1 - \alpha^{-1/2})^2, \tag{3.34}$$

and $\theta(x)$ being the Heaviside function. This suggests that, for an associative memory model that stores a number of memories in a very robust way, i.e. resembling a SVM, one can compute the thermal correlations at $\beta \geq 1$ and measure the spectrum of the eigenvalues of the covariance matrix. Whether this fits a MP distribution with a parameter α^*, then one can infer the number of dominant basins of attraction. Of course this is valid only if such robust memories, that are hidden in the landscape of attractors, are treatable as random binary vectors, i.e. they do not contain a particular internal structure or mutual dependencies.

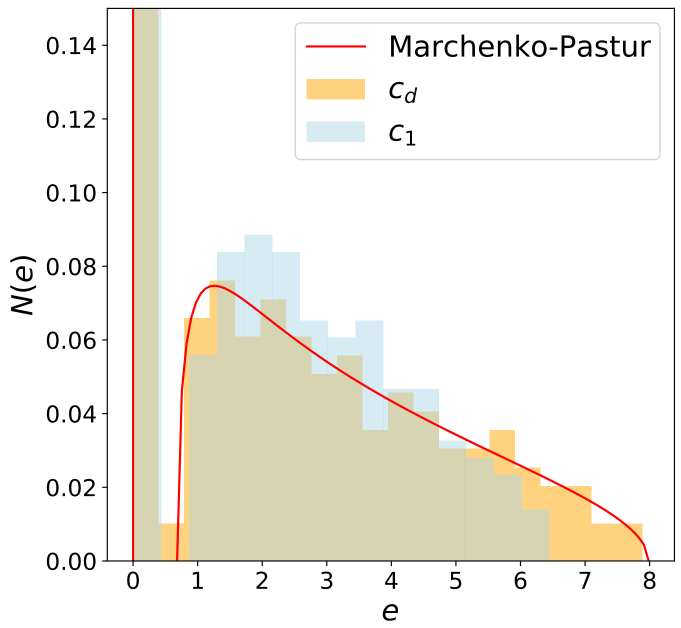


Figure 3.13. Comparison between the theoretical Marchenko-Pastur distribution of the eigenvalues of a Wishart matrix with a parameter $\alpha = 0.3$, and the empirical histogram of the eigenvalues of c_1 and c_d for a SVM with $N = 500$.

A physical interpretation for the moment matching condition, and in particular why $c_d \simeq c_\beta$ in the SVM is now proposed. The thermal correlation c_β can rewritten explicetely as

$$c_{ij}^{\beta} = \frac{\sum_{\vec{S}} S_i S_j \exp\left(-\beta E[\vec{S}]\right)}{\sum_{\vec{S}} \exp\left(-\beta E[\vec{S}]\right)}, \tag{3.35}$$

that one can rewrite in terms of the typical states at that temperature

$$c_{ij}^{\beta} = \frac{\sum_{\alpha} S_i^{\alpha} S_j^{\alpha} \exp\left(-\beta E_{\alpha}\right)}{\sum_{\alpha} \exp\left(-\beta E_{\alpha}\right)}. \tag{3.36}$$

When $\beta \to \infty$ the exponential terms in eq. (3.36) are peaked over the ground state of the energy of the model $E_{GS} < 0$. Such state can be single or degenerate with multiplicity p. Hence, the index α is substituted with $\mu = 1, .., p$, the degenerate states are named $\vec{\xi}^{\mu}$ obtaining

$$c_{ij}^{\infty} = \lim_{\beta \to \infty} \frac{\sum_{\mu=1}^{p} \xi_i^{\mu} \xi_j^{\mu} \exp\left(-\beta E_{GS}\right)}{p \exp\left(-\beta E_{GS}\right)} = \frac{1}{p} \sum_{\mu=1}^{p} \xi_i^{\mu} \xi_j^{\mu}, \tag{3.37}$$

that corresponds to

$$c_{\infty} = c_d, \tag{3.38}$$

if $\{\vec{\xi}^{\mu}\}_{\mu}^{p}$ are also the binary memories. As a result, approaching the condition (3.38) implies that, when $k \to k_{max}(\alpha)$, memories become close to be degenerate ground states of the energy landscape. This condition is rare in recurrent neural networks.

In a Hebbian network, for instance, eq. (3.38) holds exclusively for $\alpha = 0$ in the thermodynamic limit, and $\alpha \sim 0$ when N is finite. In the Pseudo-inverse model [81, 82], that is represented by the following connectivity matrix

$$J_{ij} = \frac{1}{N}\sum_{\mu,\nu} \xi_i^\mu C^{-1}_{\mu,\nu}\xi_j^\nu \qquad C_{\mu,\nu} = \sum_{k=1}^{N} \xi_k^\mu \xi_k^\nu, \tag{3.39}$$

perfect-retrieval is admitted up to $\alpha = 1$, with memories lying at a fixed energy $E = -N$ independently of α; moreover, the fact that the basins of attraction are absent when $\alpha > 1/2$ implies that memories cannot be ground states of the landscape, since one single step of the dynamics started in the proximity of each memory would decrease the energy. A similar behaviour would be observed in a symmetric linear perceptron with $k = 0$, where memories lack of basins of attraction.
The relation between the size of the basins of attraction and the depth of memories in the energy landscape is still an open question in the spin-glass community. Previous studies [21, 44] relate the typical size of the basins of attraction to the margin k of a linear perceptron. Hence, the moment matching condition reached at certain values of α by SVMs must be connected to the size of the basins of attraction.

As we have seen from the previous sections, BMs satisfy a moment matching condition. As a consequence, one can imagine to train a BM in a under-sampling regime, i.e. when the number of data-points is small and P_{data} must differ from P_{true} (see the discussion in section 1.3). To be more specific, we are considering the typical associative memory scenario, where data-points are also binary memories $\{\vec{\xi}^\mu\}_{\mu=1}^p$ being randomly generated.
Since symmetric SVMs approach the moment matching condition for certain values of α, we should expect that the same network can be found by training a BM with the same control parameters. As a numerical experiment, we trained a BM with $\beta = \infty$ and a SVM with the same set of memories, for different values of α. Even if the couplings were initialized completely at random at each training for each model, both the BMs and the SVM reached a mutual overlap of the final J being superior to 90%. Figure 3.14 shows the retrieval map (see eq. (1.6)) for the three final models with $N = 500$ neurons at $\alpha = \{0.3, 0.5, 0.7\}$. As one can notice, they all reach perfect-retrieval with comparable sizes of the basins of attraction. To be more precise, BMs display basins that are slightly larger than SVMs, yet more detailed numerical studies should be performed to exclude eventual finite size effects. The memory performance of BMs with $\beta = 1$ appears to be experimentally consistent with the one at $\beta = \infty$ for each choice of α.
Hence, numerical evidence suggests that BMs being trained on a small number of random memories can reproduce the SVM performance pretty well, up to consistent values of α. Since we proved in chapter 2 that HU tend to the SVM, and thus to the same moment matching condition, for $\alpha < \alpha_c$, one can conclude that the three learning rules in question, i.e. HU, SVM and BM, must approach the same spot in the space of the couplings.

Checkpoint

In this section we have seen that:

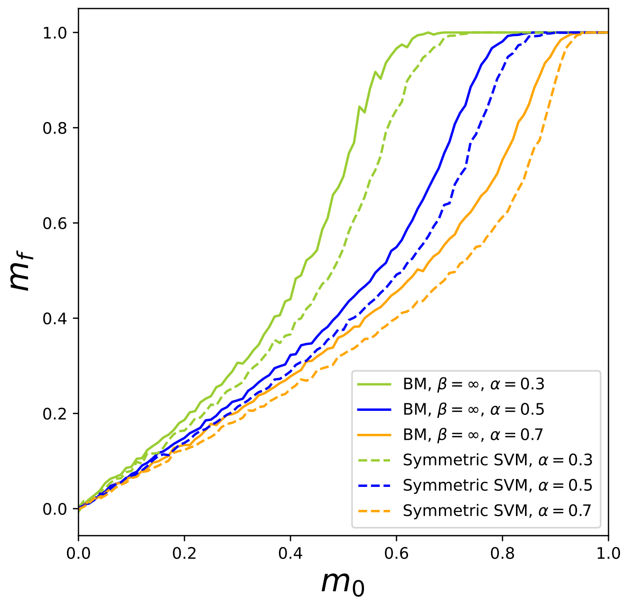


Figure 3.14. Retrieval map $m_f(m_0)$ for a symmetric SVM and BM with $\beta = \infty$ at different values of α. BMs are reported with a *full* line while SVMs are represented by the *dashed* line. Measures are averaged over 5 samples. Errors are neglected because small. Choice of the parameters: $N = 500$, $\lambda = 1$.

- SVMs tend to match the 2-points correlation matrix measured at temperature β (i.e. $\langle S_i S_j \rangle_\beta$, with $\langle \cdot \rangle_\beta$ being the Gibbs-Boltzmann measure) and the Hebbian matrix of the memories. Specifically, this condition, named moment matching, holds for high values of β and low values of α.

- The moment matching condition when $\beta \to \infty$, with the 2-points correlations being measured according by the Gibbs-Boltzmann distribution of the model, means that memories are degenerate ground states of the energy. This condition is peculiar of SVMs, with respect of other neural network models.

- Since moment matching is intrisically satisfied by BMs, we conclude that, given the same choice of the control parameters, BM learning and SVMs train similar recurrent neural networks. Moreover, because chapter 2 has proved that HU approaches a SVM when $\alpha < \alpha_c^{HU}$, we conclude that, given the same choice of the control parameters, BM learning, SVM and HU train very similar recurrent neural networks.

3.5 Summary & Conclusions

In this chapter we have seen that:

- In the context of Boltzmann Machines, and specifically for what concerns the generalization performance of the inferred network, the standard L_1 and L_2 regularization techniques are outperformed by the Unlearning regularization.
- A particular case of the Unlearning regularization reproduces a thermally averaged Hebbian Unlearning rule, that shows good inferential capabilities when initialized in the standard Hebbian fashion. Hebbian Unlearning can be interpreted as a two-steps Boltzmann-Machine learning.
- In the framework of associative memory models, that learn how to retrieve an extensive number of randomly generated memory vectors, there is a strong analogy between Support Vector Machines, Boltzmann Machines and the Unlearning algorithm. Such analogy lies into a moment matching condition which is common to all these three learning routines. This property, and so the similarity among the models, is stronger for smaller values of α, i.e. $\alpha < 0.6$.

The goal of this chapter is twofold:

1. to show the effectiveness of Unlearning as a form of regularization in Boltzmann-Machines;
2. to establish a connection between three learning rules: Support Vector Machines, Boltzmann Machines, Unlearning.

For what concerns the first goal, we have used small networks that allowed to compute all quantities in closed form. Precisely, the ground-truth distribution belonged to two models, such as the Curie-Weiss and then Sherrington-Kirkpatrick, known to approach criticality even for finite N. As a result, we could not only test the proximity of the inferred model to the original one, but also the susceptibility of the network under variations of the parameters, by comparing the original true trend of the specific heat with respect to the inverse temperature, with the trend obtained after training. We could conclude that the two main components that define the generalization of the model, i.e. its consistency with the ground-truth distribution and its stability under rescaling of the parameters, are enhanced by the Unlearning regularization.
Moreover we analyzed the particular limit of the regularization that gives as learning rules the ones expressed in equations (3.25) and (3.26). This procedure, which provides that a thermal 2-point correlation is computed at $\beta = 1$, strongly resembles the traditional Hebbian Unlearning routine: previous works in literature [106] support the fact that the associative memory performance of a *thermal* or *paramagnetic* Unlearning does not deviate much from the original rule (see eq. (1.10)). As a new contribution to the Unlearning literature we tested its inferential power, i.e. the capability of reproducing the original model by evaluating the entire set of data (which remains feasible due to the reduced dimension of the network). So far the Unlearning algorithm was exclusively tested in an associative memory framework, where data are actually memories, that are sub-dominant in number with respect

to the entire set of possibile configurations. Results show a good generalization performance of the algorithm. The explanation for this success relies on the choice of the initialization of the parameters. An initial Hebbian network (where the Hebbian matrix is constructed over the entire set of data), by contrast with other types of initialization, implies that the contribution to the gradient descent of the Loss (see **??**) deriving from the data (i.e. the positive *Hebbian* contribution in the Boltzmann Machine learning) is much smaller than the contribution deriving from the cross-entropy of the model given the data (i.e. the negative *Unlearning* contribution in the Boltzmann Machine learning). This condition holds until an optimal amount of iterations that can be estimated from the numerical simulations. We conclude that Boltzmann Machine learning can be performed in two steps: an initial abrupt descent over the Loss (i.e. *overshooting*) along the direction of the data and then a gradual adjustment along the direction of the model with increments being updated at each step of the algorithm. The effectiveness of a two-step training for a Boltzmann Machine encourages a biological interpretation of this kind of training, as many members in the artificial intelligence and computational neuroscience community seem convinced that optimization of synapses in the brain must occur through the alternation of the daily online experience and offline sleep [109, 87]. This point, together with the importance of the Hebbian initialization of the network, appears to be consistent with our previous conclusions regarding Unlearning from the point of view of the associative memory modeling, treated in chapter 2.
At last, we have showed a new property of Support Vector Machines. These types of models satisfy the same moment matching condition that Boltzmann Machines meet. This implies that using the Boltzmann Machine learning algorithm or training a Support Vector Machine should lead to similar outcomes in terms of an associative memory model at least when data are randomly generated, they are few in number, and training is performed at sufficiently low temeperature. Moment matching for high β and low α is related to stable and deep memories, with the depth directly associated to large basins of attraction. This aspect also gives interesting insights regarding the under-sampling regime of a Boltzmann Machine, where the model is overfitting the data while surrounding them with wide basins of attraction. An interesting approach to the overfitting regime of generative recurrent neural network models has already been tackled in [110] for a controlled set-up where data were learned in a Hebbian fashion: in this case overfitting did not imply perfect-retrieval and robustness as it actually does for a proper Boltzmann Machine.
Eventually, since we have evidence that Boltzmann Machine learning in the under-sampling regime can be effectively trained in two steps and thus reproduced by the Hebbian Unlearning algorithm at the same temperature, the similarity between Support Vector machines and Boltzmann Machines implies that these two methods, along with Unlearning, belong to the same kind of training procedures. This is the most important contribution of this thesis work.

Chapter 4

Discussion & Future perspectives

We will now discuss the future perspectives for this research, dedicated to the study of the Unlearning rule across different learning frameworks.

Concerning the study of the optimal noise structure to train linear perceptrons, contained in chapter 2, the future will certainly see a simplification of the model possibly aimed at obtaining an analytical phase diagram for the network performance. One possible way would be the generalization of the calculations contained in appendix C with some structured data that can be treated rigorously. One might start by trying different types of structure and see how the phase diagram changes. Two possible candidates are memories containing combinatorial disorder [111] or memories embedded in a manifold [112, 113]. These types of data-set would even satisfy the requirement for the functioning of the training-with-noise algorithm in presence of structured noise: the memories can contain internal correlations among their features, but they should be well separated from each other in the space of configurations. This aspect has been clearly pointed out in section 2.4.
Another important point stressed by the analysis in chapter 2 is the importance of the Hebbian initialization of the network, due to its topological properties. The disposition of the glassy states and the surrounding saddles in a Hebbian network out of the retrieval regime appears non-trivial and extremely important in the perspective of finding beneficial noisy configurations to employ in the training. Specifically, the distribution of the observables ω_i^μ with respect to the stabilities Δ_i^μ (see section 2.2) was crucial to determine whether the class of configurations under consideration decreased the Loss of a maximally stable perceptron problem or not. The physical interpretation of this quantity emerges more clearly in the case of fixed points, where $\omega_i^\mu = \xi_i^\mu S_i(\vec{\xi}^\mu \cdot \vec{S})/N$: while a normal projection of the memories over the glassy states would show an isotropic scenario in the landscape, this new scalar product shows a peculiar anisotropy that is responsible for the functioning of the Unlearning algorithm and its generalizations. Understanding the reason for this particular topological property would be fundamental to close the Unlearning problem.

In chapter 3 we have showed that the *negative learning* principle behind Unlearning exceeds the associative memory application and it can be applied to inferential problems, where the goal is to use a model to reproduce the real distribution of a data-set. Even though this idea was already suggested by past works by Hinton [20, 88], nobody in literature had isolated and singularly evaluated the Unlearning contribution to Boltzmann Machine learning. A follow-up of our study will be testing the Unlearning regularization in the case of real data-sets, and comparing it to the usual regularization techniques. This would imply to deal with the empirical sampling errors and the intrinsic noise contained into data.
Moreover, the moment-matching property of Support Vector Machines needs further investigations. From the point of view of statistical mechanics, calculations are hard to perform when symmetry is constrained: Gardner solved the issue by limiting the problem to a diluted network [37]. Even though it has been showed that this regime of connectivity affects the perfect-retrieval behaviour of the Support Vector Machine [21], things in practice do not qualitatively change. One should be now interested in computing the distribution of the energy for the memories as a function of the load parameter α, and compare it with the one of the ground state. In light of our results, we would expect the memories to be close to the ground state for small values of α and then to be progressively lifted up when α increases. For what concerns the inferential problem instead, and the reason why Boltzmann Machines in the under-sampling regime can resemble maximally stable perceptrons, the attention should be focused on the topological properties of the Hebbian energy landscape. Since this type of model can be seen as an interpolation (controlled by α) between the a Curie-Weiss and a Sherrington-Kirkpatrick model, it might be that fixed points sampled by the dynamics at $\beta = \infty$ are still representative of the ground states of the energy. As a consequence Hebbian landscapes, especially at lower values of α can be employed to train a Support Vector Machine in a faster unsupervised manner, by means of a Boltzmann Machine algorithm or, equivalently, the Hebbian Unlearning routine.

Ideally, this research line would lead to the construction of a model that is able to perform both the associative memory and the generative task at the same time: one single training algorithm assembling a network that both retrieves a number of *important* concepts, and generates new examples that are consistent with the stimuli received from the environment. This idea is close to what specialists in statistical learning call *benign overfitting* [114, 115]: the model can both remember the data and correctly fit new unseen examples. As a further speculation, we might conjecture that, by implementing algorithms that reach deeper states in the Hebbian energy landscape (e.g. [73, 116]) at higher values of α, we might use such states to train a Boltzmann Machine. At this point, if moment matching is satisfied (i.e. eq. (3.38)) we might reach a condition of benign overfitting: a robust associative memory coexisting with an inferential device. This objective might be gained through the choice of continuous neural representations, as it is done for the rate models used in theoretical neuroscience [26, 117], that would make the dynamics smoother in the landscape. On the other hand, the maximum storage limit of the symmetric Support Vector Machine, i.e. $\alpha_c \simeq 1.3$ as obtained by [37] for diluted symmetric perceptrons, would be overcome by imposing sparser representation of the memories [118].

Finally, another aim of this thesis is to incentive a stronger experimental effort for the investigation of anti-Hebbian synaptic mechanisms in the brain. We believe that, in light of the analysis exposed in this work, and consistently with the intuition of a series of scientists from past to recent times [92, 93, 19, 109, 87, 119], this counter-intuitive and apparently not biological mechanism could, instead, be at the basis of the formation and the consolidation of memory in the brain.

Appendix A

Analytical expression of the one-step retrieval map

As done in [29, 16], one can define the following quantity

$$h_i^\mu = \frac{\xi_i^\mu}{\sqrt{N}} \sum_{j=1}^{N} J_{ij} S_j, \tag{A.1}$$

that we will call *cross-stability* of the configuration $\vec{S}$. Configurations $\vec{S}$ have an overlap m_0 with the memory $\vec{\xi}^\mu$ and they are generated by the probability distribution in eq. (2.3). As a consequence, h_i^μ is a random variable distributed according to

$$P(h_i^\mu|m_0, J) = \frac{\sum_{\vec{S}} \left[\delta\left(m_0 - \frac{1}{N}\sum_{k=1}^{N} S_k \xi_k^\mu\right) \delta\left(h_i^\mu - \frac{\xi_i^\mu}{\sqrt{N}} \sum_{k\neq i} J_{ik} S_k\right)\right]}{\sum_{\vec{S}} \left[\delta\left(m_0 - \frac{1}{N}\sum_{k=1}^{N} S_k \xi_k^\mu\right)\right]}, \tag{A.2}$$

which can computed by using the Fourier transform of the Dirac-delta function and summing over $\{-1,+1\}^N$. This leads to the following probability density function

$$P(h_i^\mu|m_0, J) = \frac{1}{\sqrt{2\pi(1-m_0^2)}} \exp\left(-\frac{(h_i^\mu - m_0 \Delta_i^\mu(J))^2}{2(1-m_0^2)}\right), \tag{A.3}$$

where the neuron index i was also dropped for simplicity, and the stabilities depend on the realization of the couplings. The one-step retrieval map averaged over all the possible $\vec{S}$ configurations is a function of m_0 and J, and it is given by

$$m_1(m_0, J) = \frac{1}{\alpha N^2} \sum_{i,\mu}^{N,p} \int dh_i^\mu P(h_i^\mu|m_0, J) \mathrm{sign}\,(h_i^\mu) = \frac{1}{\alpha N^2} \sum_{i,\mu}^{N,p} \mathrm{erf}\left(\frac{m_0 \Delta_i^\mu}{\sqrt{2(1-m_0^2)}}\right), \tag{A.4}$$

and this result is fully general and it is relative to one realization of the network. If we assume that stabilities are self-averaging quantities and they satisfy the probability distribution function $\rho(\Delta)$, then we can compute the one-step overlap as averaged over many realizations of the network, i.e. for many training processes. One obtains

$$m_1(m_0) = \lim_{N\to\infty} \frac{1}{\alpha N^2} \sum_{i,\mu}^{N,p} \mathrm{erf}\left(\frac{m_0 \Delta_i^\mu}{\sqrt{2(1-m_0^2)}}\right) = \int_{-\infty}^{+\infty} d\Delta \rho(\Delta) \mathrm{erf}\left(\frac{m_0 \Delta}{\sqrt{2(1-m_0^2)}}\right), \tag{A.5}$$

which is the quantity defined in eq. (1.7). This observable can provide with some rough intuition about the shape of the basins of attraction in any fully connected neural network, where no exact calculation exists to estimate their typical radius. It will also be crucial for the definition of a Loss function for the *training-with-noise* algorithm [68] presented in section 2.1.

Appendix B

Gradient of the Loss function of the Training-with-noise algorithm

B.1 Training with noise

At each step of the algorithm a memory label μ_d is sampled at random and the update (2.1) is performed over the couplings. The new value of the $\mathcal{L}$ function (2.7) is

$$\mathcal{L}' = -\frac{1}{\alpha N^2}\sum_{i,\mu}^{N,p} \operatorname{erf}\left(\frac{m\Delta_i^\mu}{\sqrt{2(1-m^2)}} + \frac{\lambda m}{N\sigma_i\sqrt{2N(1-m^2)}}\epsilon_i^{\mu_d}\xi_i^\mu\xi_i^{\mu_d}\sum_{j\neq i} S_j^{\mu_d}\xi_j^\mu\right). \quad \text{(B.1)}$$

Since $\delta\sigma_i \propto \frac{\lambda}{N}\frac{J_i}{\sigma_i} + \mathcal{O}\left(\frac{\lambda}{N}\right)^3$ and the mean J_i of the couplings along line i equals zero by initialization and it is naturally maintained null during the algorithm, we have considered $\sigma_i' \simeq \sigma_i$. Then $\mathcal{L}'$ can be rewritten as

$$\begin{aligned}\mathcal{L}' = &-\frac{1}{\alpha N^2}\sum_{i,\mu\neq\mu_d}^{N,p} \operatorname{erf}\left(\frac{m\Delta_i^\mu}{\sqrt{2(1-m^2)}} + \mathcal{O}\left(\frac{1}{N}\right)\right) - \\ &-\frac{1}{\alpha N^2}\sum_{i}^{N} \operatorname{erf}\left(\frac{m\Delta_i^{\mu_d}}{\sqrt{2(1-m^2)}} + \frac{\lambda m\cdot m_t}{\sigma_i\sqrt{2N(1-m^2)}}\epsilon_i^{\mu_d} + \mathcal{O}\left(\frac{1}{N}\right)\right), \quad \text{(B.2)}\end{aligned}$$

where we have used that $\frac{1}{N}\sum_{j\neq i}\xi_j^\mu S_j^{\mu_d} = \mathcal{O}(N^{-1/2})$ when $\mu\neq\mu_d$ and $m_t = \mathcal{O}(1)$. We thus expand the error-function at the first order in $\mathcal{O}(N^{-1/2})$ obtaining the variations to $\mathcal{L}$ in equation (2.9).

B.2 Training with structured noise

The new value of $\mathcal{L}$ is derived by using equation (2.10) to evaluate the variation of stabilities

$$\mathcal{L}' = -\frac{1}{\alpha N^2}\sum_{i,\mu}^{N,p}\mathrm{erf}\Bigg(\frac{m\Delta_i^\mu}{\sqrt{2(1-m^2)}} + \frac{\lambda m \xi_i^\mu \xi_i^{\mu_d}}{2N\sigma_i\sqrt{2N(1-m^2)}}\sum_{j=1}^N S_j^{\mu_d}\xi_j^\mu + \\ + \frac{\lambda m \xi_i^\mu S_i^{\mu_d}}{2N\sigma_i\sqrt{2N(1-m^2)}}\sum_{j=1}^N \xi_j^{\mu_d}\xi_j^\mu - \\ - \frac{\lambda m \xi_i^\mu S_i^{1,\mu_d}}{2N\sigma_i\sqrt{2N(1-m^2)}}\sum_{j=1}^N S_j^{\mu_d}\xi_j^\mu - \frac{\lambda m \xi_i^\mu S_i^{\mu_d}}{2N\sigma_i\sqrt{2N(1-m^2)}}\sum_{j=1}^N S_j^{1,\mu_d}\xi_j^\mu\Bigg), \quad \text{(B.3)}$$

where $\sigma_i' \simeq \sigma_i$ as in the previous paragraph. We now recall $\chi_i^\mu = \xi_i^\mu S_i^{\mu_d}$, $\chi_i^{1,\mu} = \xi_i^\mu S_i^{1,\mu_d}$, $m_\mu = \frac{1}{N}\sum_{j=1}^N S_j^{\mu_d}\xi_j^\mu$ and $m_{1,\mu} = \frac{1}{N}\sum_{j=1}^N S_j^{1,\mu_d}\xi_j^\mu$ and expand the error-function at the first order in $\mathcal{O}(N^{-1/2})$ obtaining

$$\delta\mathcal{L} = \frac{m\lambda}{\sqrt{2\pi\alpha^2 N^5(1-m^2)}}\sum_{i,\mu}^{N,p}\frac{1}{2\sigma_i}[(m_\mu\chi_i^{1,\mu} + m_{1,\mu}\chi_i^\mu) - \\ - (m_\mu\xi_i^\mu\xi_i^{\mu_d} + M_\mu^{\mu_d}\chi_i^\mu)]\exp\left(-\frac{m^2{\Delta_i^\mu}^2}{2(1-m^2)}\right) \quad \text{(B.4)}$$

where $M_\mu^{\mu_d} = \frac{1}{N}\sum_{i=1}^N \xi_i^\mu\xi_i^{\mu_d}$. Equation (B.4) can be decomposed in

$$\delta\mathcal{L} = \delta\mathcal{L}_N + \delta\mathcal{L}_U \quad \text{(B.5)}$$

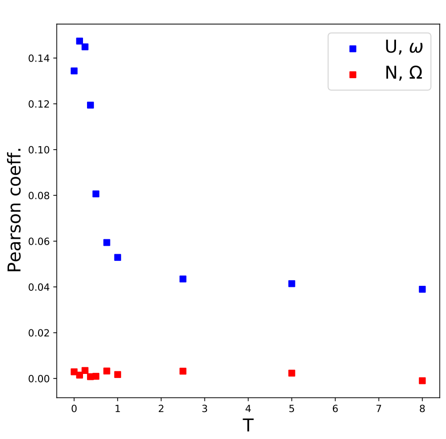
where $\delta\mathcal{L}_U$ contains the weight

$$\omega_i^\mu = \frac{1}{2\sigma_i}\left(m_\mu\chi_i^{1,\mu} + m_{1,\mu}\chi_i^\mu\right), \quad \text{(B.6)}$$

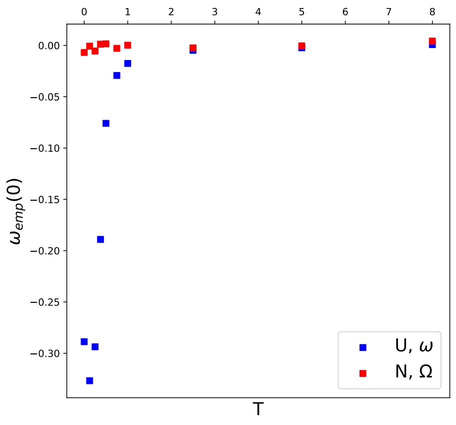
while $\delta\mathcal{L}_N$ contains

$$\Omega_i^\mu = \frac{1}{2\sigma_i}\left(m_\mu\xi_i^\mu\xi_i^{\mu_d} + M_\mu^{\mu_d}\chi_i^\mu\right). \quad \text{(B.7)}$$

We study the two contributions numerically, on a Hebbian network, i.e. with no learning going on, for the case of $m_t = 0^+$. The Pearson coefficient is measured between the vector of the stabilities Δ_i^μ and the weights ω_i^μ as well as with Ω_i^μ separately. This quantity should underline an eventual reciprocal dependence between $\omega_i^\mu, \Omega_i^\mu$ and Δ_i^μ. The test is repeated over states sampled by a Monte Carlo at different temperatures T. Results are reported in fig. B.1a where it is evident that Ω_i^μ does not have any correlation with Δ_i^μ, while the dependence of ω_i^μ on the stabilities is evident. Moreover, we measured the indicator $\omega_{emp}(0)$, signaling the typical values of the weights ω_i^μ and Ω_i^μ when $\Delta_i^\mu \sim 0$ (see section 2.2.1 for further details), as reported in fig. B.1b. The plot clearly shows that Ω_i^μ is small and generally fluctuating around zero. These aspects hold during the training procedure also, as it can be observed by performing the same measure at different step of the TWN procedure over states with $m_t = 0^+$. We will thus refer to $\delta\mathcal{L}_U$ as the relevant contribution to the variation of the function $\mathcal{L}$.



(a)



(b)

Figure B.1. (a) Pearson coefficient between ω_i^μ, Ω_i^μ and the stabilities Δ_i^μ, (b) $\omega_{emp}(0)$ for the case of a Hebbian network at different temperatures T. Configurations at a given temperature T have been sampled by a Monte Carlo of the Kawasaki kind, in order to choose only maximally noisy states ($m_t = 0^+$). Points are collected from 15 samples of the network. Choice of the parameters: $N = 500$, $\alpha = 0.5$.

Appendix C

Replica symmetric analysis of the Training-with-noise algorithm

In this Appendix we resume the replica calculations in the replica-symmetric ansatz [9] performed by [65, 66] showing their agreement with the numerical results obtained through Gardner's TWN algorithm [68].

C.1 Analytics

The partition function of the model is given by

$$Z = \int \prod_{i,j} dJ_{ij} \delta \left(\sum_{j \neq i} J_{ij}^2 - N \right) \exp \left(\beta \sum_i \sum_\mu \operatorname{erf} \left(\frac{m \xi_i^\mu \sum_j J_{ij} \xi_j^\mu}{\sqrt{2N(1-m^2)}} \right) \right), \tag{C.1}$$

while the distribution of the stabilities is

$$\rho(\Delta) = \overline{\frac{1}{Z} \int \prod_j dJ_{ij} \delta \left(\sum_j J_{ij}^2 - N \right) \exp \left(\beta \sum_\mu \operatorname{erf} \left(\frac{m \xi_i^\mu \sum_j J_{ij} \xi_j^\mu}{\sqrt{2N(1-m^2)}} \right) \right) \delta \left(\xi_i^1 \sum_j \frac{J_{ij}}{\sqrt{N}} \xi_j^1 - \Delta \right)}, \tag{C.2}$$

where we have neglected the factorization over i, since we treat the optimization process as independent along the lines of the J_{ij} matrix. Index i will be nevertheless indicated for the sake of completeness.
Replicas can be used to evaluate the normalisation, i.e.

$$1/Z = \lim_{n \to 0} Z^{n-1} \tag{C.3}$$

Consequently, equation (C.2) can be rewritten as

$$\rho(\Delta) = \lim_{n \to 0} \overline{\int \prod_a^n \prod_j^N dJ_{ij}^a \delta \left(\sum_j (J_{ij}^a)^2 - N \right) \exp \left(\beta \sum_{a,\mu} \operatorname{erf} \left(\frac{m \xi_i^\mu \sum_j J_{ij}^a \xi_j^\mu}{\sqrt{2N(1-m^2)}} \right) \right) \delta \left(\xi_i^1 \sum_j \frac{J_{ij}^1}{\sqrt{N}} \xi_j^1 - \Delta \right)} \tag{C.4}$$

that becomes

$$\rho(\Delta) = \lim_{n\to 0} \int \prod_a^n \prod_j^N dJ_{ij}^a \delta\left(\sum_j (J_{ij}^a)^2 - N\right) \prod_{\mu,a} \int_{-\infty}^{+\infty} dx_\mu^a \exp\left(\beta \sum_{\mu,a} \operatorname{erf}\left(\frac{m\cdot x_i^\mu}{\sqrt{2(1-m^2)}}\right)\right) \cdot$$

$$\overline{\cdot \prod_{\mu,a} \delta\left(\xi_i^\mu \sum_j \frac{J_{ij}^a}{\sqrt{N}} \xi_j^\mu - x_\mu^a\right) \delta\left(\xi_i^1 \sum_j \frac{J_{ij}^1}{\sqrt{N}} \xi_j^1 - \Delta\right)} \tag{C.5}$$

by rewriting the Dirac delta in its integral representation we get

$$\rho(\Delta) = \lim_{n\to 0} \int \prod_a^n \prod_j^N dJ_{ij}^a \delta\left(\sum_j (J_{ij}^a)^2 - N\right) \cdot$$

$$\cdot \prod_{\mu,a} \int_{-\infty}^{+\infty} dx_\mu^a \int_{-\infty}^{+\infty} \frac{dy_\mu^a}{2\pi} \int_{-\infty}^{+\infty} dz \exp\left[\beta \sum_{\mu,a} \operatorname{erf}\left(\frac{m\cdot x_i^\mu}{\sqrt{2(1-m^2)}}\right) - \sum_{\mu,a} x_\mu^a y_\mu^a - iz\Delta\right] \cdot$$

$$\cdot \overline{\prod_{\mu\neq 1} \prod_j \exp\left(i \sum_a \frac{J_{ij}^a}{\sqrt{N}} \xi_i^\mu \xi_j^\mu y_\mu^a\right)} \cdot \overline{\prod_j \exp\left(+i \frac{y_1^a}{\sqrt{N}} \sum_a J_{ij}^a \xi_i^1 \xi_j^1 + i \frac{z}{\sqrt{N}} J_{ij}^1 \xi_i^1 \xi_j^1\right)} \tag{C.6}$$

We will now compute the averages over the disorder, i.e. the random memories. The first one, relative to $\mu \neq 1$ variables, becomes

$$\overline{\prod_{\mu\neq 1} \prod_j \exp\left(i \sum_a \frac{J_{ij}^a}{\sqrt{N}} \xi_i^\mu \xi_j^\mu y_\mu^a\right)} = \exp\left(-\frac{1}{2} \sum_{\mu\neq 1} \sum_{a,b} Q_{ab} y_\mu^a y_\mu^b\right) \tag{C.7}$$

where we have Taylor expanded at the second order around the $\mathcal{O}(1/\sqrt{N})$ term and we have introduced the overlap matrix Q_{ab} defined as

$$Q_{ab} = \frac{1}{N} \sum_{j=1}^N J_{ij}^a J_{ij}^b \tag{C.8}$$

The second averaged expression becomes

$$\overline{\prod_j \exp\left(+i \frac{y_1^a}{\sqrt{N}} \sum_a J_{ij}^a \xi_i^1 \xi_j^1 + i \frac{z}{\sqrt{N}} J_{ij}^1 \xi_i^1 \xi_j^1\right)} = \tag{C.9}$$

$$= \overline{\prod_j \exp\left(i \sum_a \frac{J_{ij}^a}{\sqrt{N}} \xi_i^1 \xi_j^1 y_1^a + i \frac{J_{ij}^1}{\sqrt{N}} (y_1^1 + z) \xi_i^1 \xi_j^1\right)} =$$

$$= \prod_j \cos\left(\sum_{a\neq 1} \frac{J_{ij}^a}{\sqrt{N}} y_1^a + \frac{J_{ij}^a}{\sqrt{N}} (y_1^1 + z)\right) =$$

$$= \exp\left(-\frac{1}{2} \sum_{a,b\neq 1} Q_{ab} y_1^a y_1^b - \sum_{a\neq 1} Q_{a1} y_1^a (y_1^1 + z) - \frac{1}{2} (y_1^1 + z)^2\right) =$$

where, once again, a Taylor expansion at the second order in $\mathcal{O}(1/\sqrt{N})$ has been performed. We now make the following change of variables

$$\tilde{y}_1^1 = y_1^1 + z \qquad d\tilde{y}_1^1 = dy_1^1 \tag{C.10}$$

Hence equation (C.9) becomes

$$= \exp\left(-\frac{1}{2}\sum_{ab} Q_{ab} y_1^a y_1^b\right) \tag{C.11}$$

where, instead of y_1^1 one considers the new one given by (C.10).
The averaged expression for the distribution of the stabilities is now given by

$$\begin{aligned}\rho(\Delta) = \lim_{n\to 0}\int DQ \det Q^{\frac{N}{2}}[\prod_a \int_{-\infty}^{+\infty}\frac{dx^a}{2\pi}dy^a \exp(\beta\sum_a \operatorname{erf}\left(\frac{mx^a}{\sqrt{2(1-m^2)}}\right) - i\sum_a x^a y^a -\\ -\frac{1}{2}\sum_{a,b} Q_{ab}y^a y^b)]^{\alpha N-1}\cdot \prod_{a\neq 1}\int_{-\infty}^{+\infty}\frac{dx^a}{2\pi}dy^a dx^1 dy^1 d\tilde{y}^1 \exp(\beta\sum_a \operatorname{erf}\left(\frac{mx^a}{\sqrt{2(1-m^2)}}\right) -\\ -i\sum_a x^a y^a - i(\tilde{y}^1 - y^1)\Delta - \frac{1}{2}\sum_{a,b} Q_{ab} y^a y^b) =\end{aligned} \tag{C.12}$$

with

$$DQ = dQ\prod_a \delta\left(Q_{aa}-1\right)$$

We now assume the replica symmetry of Q_{ab}, i.e.

$$Q_{ab} = \delta_{ab} + (1-\delta_{ab})q \tag{C.13}$$

Hence the computation follows

$$\begin{aligned}= \lim_{n\to 0}\int Dq \exp\left(N g_n(q)\right)\int D_q w\left[\int\frac{dx}{2\pi}\int dy \exp\left(-\beta\mathcal{L}_m(x) - i(x-w)y - \frac{(1-q)}{2}y^2\right]^{n-1}\cdot\\ \cdot\int\frac{dx}{2\pi}dy d\tilde{y}\exp\left(-\beta\mathcal{L}_m(x) - i(x-\Delta)y - i(\Delta-w)\tilde{y} - \frac{(1-q)}{2}\tilde{y}^2\right)\end{aligned} \tag{C.14}$$

where

$$D_q w = \frac{1}{\sqrt{2\pi q}}e^{-\frac{w^2}{2q}} \tag{C.15}$$

and $g_n(q)$ is the *Cramer* function, given by

$$\begin{aligned}g_n(q) = \frac{1}{2}\log(1+(n-1)q) + \frac{n-1}{2}\log\left(1-q\right)+\\ +\alpha\log\left[\int D_q w\left(\int\frac{dx}{\sqrt{2\pi(1-q)}}\exp\left(-\beta\mathcal{L}_m(x) - \frac{(x-w)^2}{2(1-q)}\right)\right)^n\right]\end{aligned} \tag{C.16}$$

The limit for $n\to 0$ of $g_n(q)$ gives

$$\begin{aligned}\lim_{n\to 0} g_n(q) = n\cdot g_{\beta,m}(q) =\\ = n\left[\frac{1}{2}\log\left(1-q\right) + \frac{q}{2(1-q)} + \alpha\int D_q w \log\left[\int_{-\infty}^{+\infty}\frac{dx}{\sqrt{2\pi(1-q)}}\exp\left(-\beta\mathcal{L}_m(x) - \frac{(x-w)^2}{2(1-q)}\right)\right]\right]\end{aligned} \tag{C.17}$$

Consequently, the distribution of the stabilities becomes

$$\rho(\Delta) = \int_{-\infty}^{+\infty} D_{q^*}w \frac{\exp\left(-\frac{(\Delta-w)^2}{2(1-q^*)} + \beta\text{erf}\left(\frac{m\Delta}{\sqrt{2(1-m^2)}}\right)\right)}{\left[\int_{-\infty}^{+\infty} dx \exp\left(\beta\text{erf}\left(\frac{m\Delta}{\sqrt{2(1-m^2)}}\right) - \frac{(x-w)^2}{2(1-q^*)}\right)\right]} \tag{C.18}$$

where q^* is the particular value of the overlap s.t.

$$\frac{dg_{\beta,m}}{dq}|_{q=q^*} = 0$$

Following the same procedure one is able to compute the replica symmetric free-energy of the model, being

$$f_{RS,\beta}(m,\alpha) = -\frac{1}{N\beta}\overline{\log\left(Z_\beta\right)} = -\lim_{n\to 0}\frac{1}{nN\beta}(\overline{Z^n}-1) = -\frac{g_{\beta,m}(q^*)}{\beta} \tag{C.19}$$

We are now interested in the $\beta\to\infty$ limit with $q = 1$. Such a limit reproduces the ground truth of the problem, i.e. the very minimum of the Loss function.
One can thus couple q with the annealing temperature $T = 1/\beta$ by introducing an interaction susceptibility χ, such that

$$\chi = \frac{1}{T}\left[\langle J^2\rangle_T - \langle J\rangle_T^2\right] = \frac{1}{T}(1-q) \tag{C.20}$$

where $\langle\,\cdot\,\rangle_T$ is the thermal average and the last passage is justified by the spherical constraint over the couplings. One can now substitute the recurrent term $(1-q)$ in both the g function and the ρ with χT. The first one becomes

$$g_{T,m}(\chi) = \frac{1}{2}\log\left(\chi T\right) + \frac{1-\chi T}{2\chi T} + \\ + \alpha\int D_{1-\chi T}w\log\left[\int_{-\infty}^{+\infty}\frac{dx}{\sqrt{2\pi\chi T}}\exp\left(-\frac{1}{T}\left(-\text{erf}\left(\frac{mx}{\sqrt{1-m^2}}\right) + \frac{(x-w)^2}{2\chi}\right)\right)\right] \tag{C.21}$$

One can now substitute g into the expression for the free energy given by (C.19) and perform the $T\to 0$ limit which simultaneously pushes $q\to 1$ according to (C.20), since $\chi\geq 0$ by definition. We get

$$f_m(\alpha,\chi) = -\frac{1}{2\chi} - \alpha\int Dw\left[\text{erf}\left(\frac{mx^*}{\sqrt{1-m^2}}\right) - \frac{(x^*-w)^2}{2\chi}\right] \tag{C.22}$$

where x^* is obtained by a saddle point equation and χ should be found by partial derivation of the free energy. By identifying the exponent in the gaussian integral of the energetic contribution to the free energy with

$$e_m(\chi,x) = -\text{erf}\left(\frac{mx}{\sqrt{1-m^2}}\right) + \frac{(x-w)^2}{2\chi} \tag{C.23}$$

we have

$$\frac{\partial e_m(\chi,x)}{\partial x} = 0 \Longrightarrow w = x - \frac{\sqrt{2}m\chi}{\sqrt{\pi(1-m^2)}}\exp\left(-\frac{m^2x^2}{2(1-m^2)}\right) \tag{C.24}$$

from which one gets x^* as a function of w and χ. On the other hand, one must differentiate the entire free energy in χ to obtain χ^*. Hence we get

$$\frac{\partial f_m(\chi, x^*)}{\partial \chi} = 0 \Longrightarrow \int Dw \left(x^*(w,\chi) - w\right)^2 = \alpha^{-1} \tag{C.25}$$

Equation (C.24) leads to the graphic intersection problem depicted in figure (C.1). Given w and χ, one finds x^*. Then one uses such a x^* to solve equation (C.25), which permits to find χ^*. Once one has χ and $x^*(w,\chi)$, the free energy in (C.22) can be computed. Nevertheless, the support of the function in x in figure (C.1) is not monotonic, and so not invertible, which makes the solution of the second saddle equation (C.25) not feasible. Hence one can exploit the Maxwell construction of the thermodynamics, as represented by the dashed curve in figure (C.1), while solving the problem. It can be verified easily that the new expression for the density

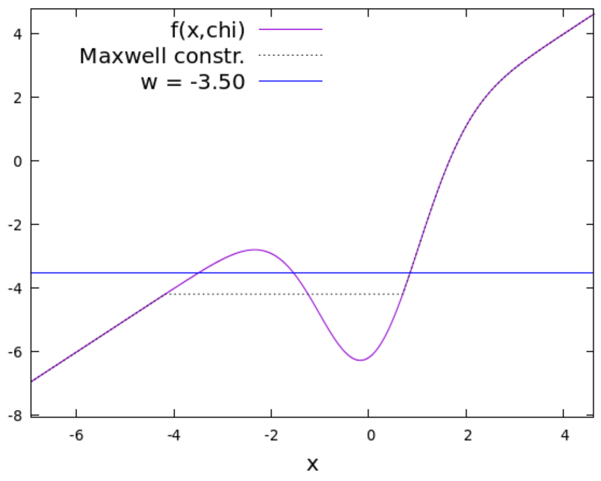


Figure C.1. Graphic representation of the saddle point equation (C.24). The choice of the parameters is: $\alpha = 0.3$, $m = 0.7$, $\chi = 7.89$.

function of Δ_i^μ when $T \to 0$ and $q \to 1$ is given by

$$\rho(\Delta) = \int Dw\delta\left(\Delta - x^*(w,\chi^*)\right) =$$
$$= \frac{1}{\sqrt{2\pi}}\left(1 + \sqrt{\frac{2}{\pi}}\frac{m^3\chi}{(1-m^2)^{3/2}}\Delta \exp\left(-\frac{m^2\Delta}{2(1-m^2)}\right)\right)\exp - \frac{w(\Delta)^2}{2} \tag{C.26}$$

where we have made use of the invertibility of $w(x,\chi)$ and also the following property of the Dirac delta

$$\delta\left(\Delta - x^*(w,\chi)\right) = \delta(F(w)) = \delta(w - w(x^*,\chi^*))\frac{dw(x^*,\chi^*)}{dx^*}|_{x^*=\Delta} \tag{C.27}$$

C.2 Numerics

The numerical procedure has consisted of a random initialization of the couplings J_{ij} followed by the implementation of the rule described in section 2.1. The values of the parameters are $N = 100$, $\alpha = 0.3$, $m = 0.7$, $\lambda = 10^{-2}$ and $t_{max} = 5 \cdot 10^5$. Data have been collected over five repetitions of the experiments. The histogram of stabilities (1.3) is empirically measured and compared to equation (C.26), as reported in figure (C.2). There is a good accordance between theory and experiment. Fig. C.3a shows

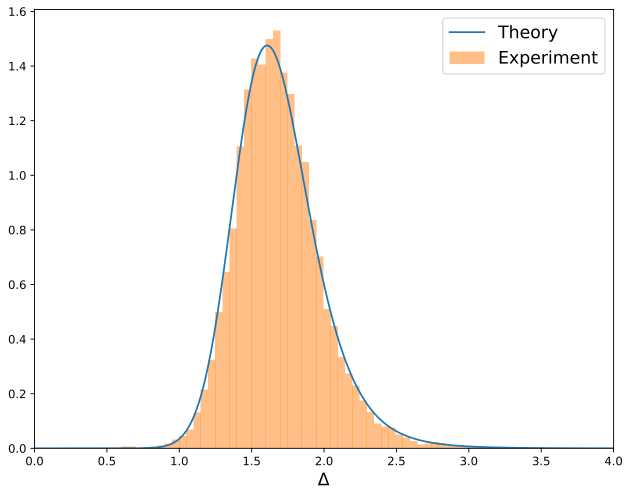
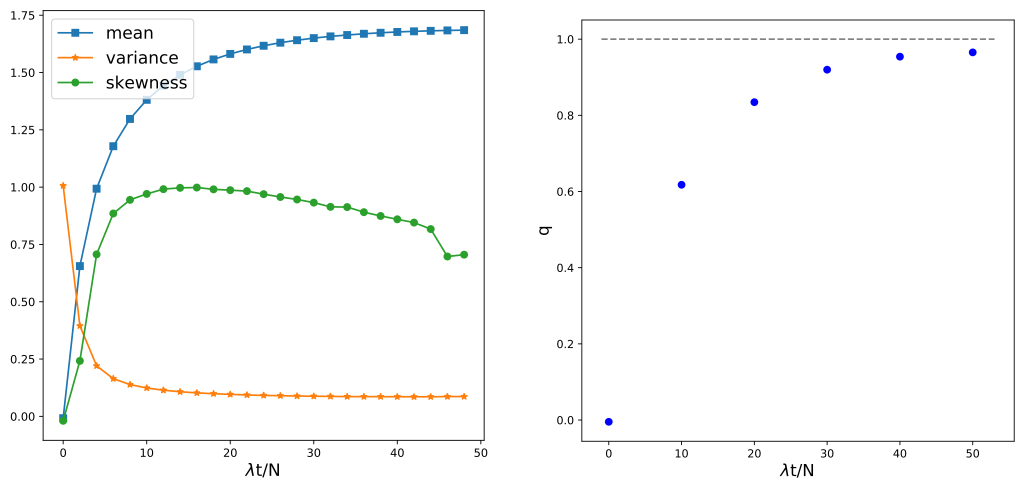


Figure C.2. Normalised histogram of the stabilities measured from the experiment compared with Wong and Sherrington's prevision. The choice of the parameters is: $N = 100$, $\alpha = 0.3$, $m = 0.7$, $\lambda = 10^{-2}$.

the evolution of the mean, variance and skewness of the *pdf* of the stabilities as a function of the algorithm steps. Fig. C.3b, instead, depicts the evolution of the overlap q measured between couples of connectivity matrices obtained from the same realisation of the memories, as a function of the algorithm steps: q correctly converges to unity while approaching the global minimum of the Loss.



(a) **(b)**

Figure C.3. Moments of the empirical distribution of the stabilities (a) and overlap q (b) as a function of the normalised time $\lambda t/N$ while implementing the TWN algorithm. Errorbars are not indicated because smaller than the symbols. The choice of the parameters is: $N = 100$, $\alpha = 0.3$, $m = 0.7$, $\lambda = 10^{-2}$.

Appendix D

Measure of the basins of attraction in a recurrent neural network

We report here a general experimental procedure to measure the average size of the basins of attraction of a fully connected neural network of finite size N and a given choice of the control parameters.
The network is firstly trained according to an algorithm of our choice. Once the couplings have been found, the asynchronous version of dynamics (1.1) is initialized in one of the memories. The dynamics is run until convergence onto the attractor associated to the basin of belonging of the memory. Now the retrieval map $m_f(m_0)$ is measured with respect to that particular attractor and the procedure is repeated over different memories and realizations of the network. The average radius of the basin of attraction is then measured as the value of $1 - m_0$ where $m_f(m_0)$ equals a reference value. In our case such value is $m_f = 0.98$.
We have applied this procedure on networks trained either as SVMs and with the TWN algorithm. In the former case a convex algorithm contained in the *cvxpy* Python domain [120] is implemented to train the network. To be more specific, N independent machines are trained to correctly classify $p = \alpha N$ binary memories of the kind of $\vec{\xi}^{\mu} \in \{-1, +1\}^N$ having as labels ξ_i^{μ} with $i \in [1, .., N]$.
Regarding the dynamics, the stability of fixed points is in general implied by some properties of the couplings, mainly their degree of symmetry. For the case of the TWN algorithm we start from a random symmetric matrix, as done in [29]: the update of the couplings will only perturb the initial symmetry yet allowing the measures to be still consistent with the theory. On the other hand, numerics show that SVMs are sufficiently symmetric to let the asynchronous dynamics converge.
The comparison between the retrieval maps obtained for the two algorithms with $\alpha = 0.45$ and $N = 200$ is reported in fig. D.1. The curve relative to the SVM is fixed, while the one associated with the TWN is changing with respect to the training overlap m_t.
Even if the SVM always reaches perfect-retrieval of the memories for $\alpha < 2$ [15], a network trained with noise might show two different behaviors: one associated to *retrieval* where each memory is close to an attractor, and one related to *non-retrieval*

where the memory is far from its attractor and the basins of attraction might contain orthogonal configurations with respect to the central attractor. In particular network models where couplings are assembled according to particular rules (e.g. [11], [82]) the transition between these two regimes can be computed analytically. In the case of TWN this cannot be done. It is then important to find an empirical criterion to divide the two behaviors as a function of (α, m_t).
Let us assume that when $N \gg 1$ the retrieval map $m_f(m_0)$ develops a plateau starting from $m_0 = 1$ and ending in some limit value $m_c < 1$ such that $m_f = 1$ along all this interval. Hence one can associate the formation of such a plateau with the existence of a cohesive basin of attraction, where close configurations in hamming distance to the attractor converge to the attractor. One then wants to measure the value of m_t at which such property of the basin disappears. As a possible estimate, it is convenient to consider the m_t^* such that

$$\frac{dm_f}{dm_0}(m_t^*)\Big|_{m_0=1} = 1. \tag{D.1}$$

The numerical extrapolation of the overlap in fig. 2.3 at different values of N shows a good agreement between the approximate separation between the two regions showed in fig. 2.3 and the line estimated by condition (D.1).

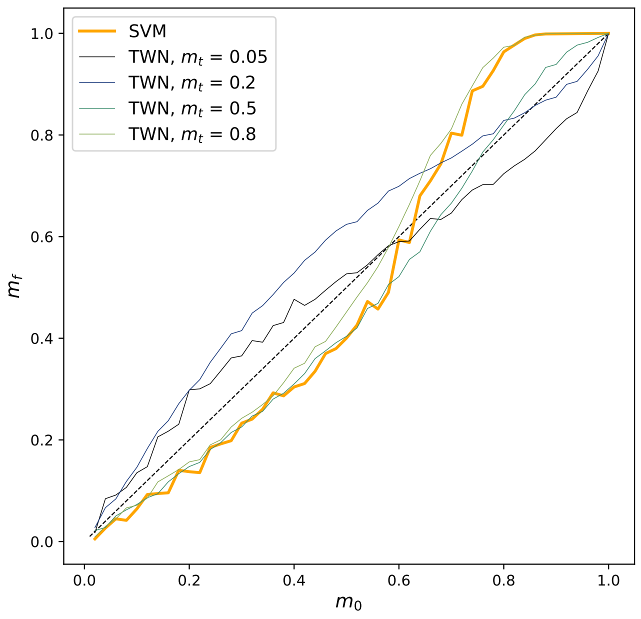


Figure D.1. Retrieval map $m_f(m_0)$ in the case of networks trained through SVM and TWN algorithms. Curves are shown as a function of m_t and compared with the bisector, indicated with a *dashed black* line. Points are averaged over 10 samples. Choice of the parameters: $N = 200$, $\alpha = 0.45$.

Ringraziamenti

Questo capitolo della tesi è in realtà il più importante.

La Fisica è meravigliosa, ne sono innamorato: anche se ho imparato a concepirla come un lavoro, sarà sempre la mia più grande passione. Però nulla di nulla avrebbe senso, nemmeno la Fisica, senza le persone che rendono il percorso così ricco, e significativo per me. Nella speranza che anche io sia stato significativo per questi miei compagni di viaggio, è giunto il momento di esprimere, per iscritto, l'immensa gratitudine che nutro nei loro confronti.

Ringrazio innanzitutto la mia famiglia. Mamma e Papà per avermi dato tutto, e Maria, perché qualunque cosa accada mi prendi sempre per mano. Grazie ai Nonni, per il loro amore e la loro complicità, grazie alle Zie, a Zio, ma soprattutto a Francesco e Giorgia, due cugini fraterni.

Ricordo coloro che mi hanno allevato nella Scienza. Ringrazio prima di tutto Francesco e Giancarlo, per essere stati così umani e comprensivi, oltre che due mentori. Grazie inoltre a Simona, Rémi, Enzo e Marco, da cui ho imparato tantissimo.

Grazie ai numerosi specialisti che si sono presi cura della mia salute fisica e mentale in questi anni. Sono molto riconoscente a Barbara, e a tutta la squadra di fisioterapisti tra Roma e Parigi. La fisioterapia e la psicoterapia, oltre ad essere simili, sono materie molto più complicate della Fisica. Vorrei infine ricordare la flotta di Kugoo Kirin S3, audaci monopattini da combattimento, specialmente Kugoo n.1 e Kugoo n.2, caduti sotto le mani degli scassinatori: mi avete sempre riportato a casa.

Il passaggio dagli studi al mestiere di ricercatore è stato accompagnato da un mio importante, seppur lento e graduale, mutamento interiore, che ha spostato l'attenzione dalla ricerca di me in me stesso alla ricerca di me negli altri. Lasciate dunque che mi rivolga a quelle persone per me così fondamentali, alle quali vorrei dedicare questo manoscritto: i miei Amici.
Voglio ringraziare i miei compagni fraterni: Cami, la mia migliore amica, e poi Carlo, Fede, Jambo, Filo, Gianmarco, Marika, Manu, Ale, da quel turbinio inarrestabile che chiameremo Malatesta Nightloop. Grazie a Ianno ed il Leo, sempre disponibili per un supplì, Luca e Lucia per la loro creatività, e l'inarrestabile Ago. Grazie alle anime della Garbatella, Morlu, Tere, Francesca, Vanessa, per le serate passate a Biffi o sul tetto.
Grazie a quei merenderos de La Sapienza, i.e. Simone, Anto, Stefano, Giovanni,

Caja e i Chimerini, Mattia, Edoardo e Giorgio dell'IIT.
Ed ora ricordo chi mi ha accompagnato nei due anni passati a Parigi, una città alla quale sarò eternamente legato. Ringrazio Gianloca che nun se ferma n'attimo, Gianmarco, Antonio e Maria, Matte, Giulia e Marghe, Alberto e Federico, Sam, Niccolò, Leah, Aram: vi ricorderò stesi e ammiccanti sulla sponda della Senna altezza Tino Rossi, con camembert e vino rosso. Grazie ai miei colleghi ed amici all'ENS, in particolare a Felix, Giampaolo, Silvia, Simone, Laura e Ada, Flavio, Jaron, Joseph, Jeanne, Sabrina, Jules, i ragazzi del gruppo Cocco-Monasson e Walzckak-Mora, e poi la Gardner Squad, ovvero gli irreplicabili Mauro e Fabian.
Arrivo adesso ad un luogo speciale, così bello non perché una pagota in mezzo all'Île-de-France, ma per via delle splendide persone che lo abitano: la Maison Du Japon. Ripenso ai bei momenti spesi con Juan, Matheus, Martina, Mohak, Lore, Gianluca, Jaz e Paola, Igor, Anouk, Taka, Leire, Francesco, alle serate passate nella cucina del secondo piano oppure in giardino a cantare. Ricordo con affetto Shashwath e Kai del terzo piano, e poi Paula, Nara e Monica, e i coniugi Nishiumi, che ci hanno custoditi nel loro nido. E infine ringrazio Ester, una stella: con te ho imparato qualcosa che va ben oltre le pagine di una tesi di dottorato.

Vi porto sempre con me.

Enrico.

Bibliography

[1] E.R. Kendal. *Principles of Neural Science.* McGraw Hill Professional, 5 edition, 2013.

[2] J.G. Nicholls, A. Martin, D. Brown, M. Diamond, D. Weisblat, and P. Fuchs. *From neuron to brain.* Oxford University Press, 2011.

[3] D.J. Amit. *Modeling Brain Function.* Cambridge University Press, 1989.

[4] S. Legg and M. Hutter. A collection of definitions of intelligence. *arXiv:0706.3639*, 2007.

[5] F. Rosemblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[6] M. Minsky and P. Seymour. *Perceptrons: an introduction to computational geometry.* MIT Press, 1969.

[7] M. Stampanoni Bassi, E. Iezzi, L. Gilio, D. Centonze, and F. Buttari. Synaptic plasticity shapes brain connectivity: Implications for network topology. *International Journal of Molecular Science*, 20(24):6193, 2019.

[8] W. A. Little and G. L. Shaw. Analytic study of the memory storage capacity of a neural network. *Mathamatical Biosciences*, 39(3):281, 1978.

[9] M. Mezard, G. Parisi, and M. Virasoro. *Spin Glass Theory and Beyond: An Introduction to the Replica Method and Its Applications*, volume 9 of *World Scientific Lecture Notes in Physics.* World Scientific, 1986.

[10] P. W. Anderson. More is different. *Science*, 177(4043):393, 1972.

[11] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554, 1982.

[12] D. J. Amit, H. Gutfreund, and H. Sompolinsky. Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks. *Physical Review Letters*, 55(14):1530, 1985.

[13] G. Hinton and T. Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1983.

[14] G. Hinton and T. Sejnowski. Analyzing cooperative computation. In *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*, 1983.

[15] E. Gardner. The space of interactions in neural network models. *Journal of Physics A: Mathematical and General*, 21(1):257, 1988.

[16] E. Gardner. Optimal basins of attraction in randomly sparse neural network models. *Journal of Physics A: Mathematical and General*, 22(12):1969, 1989.

[17] J.J. Hopfield, D.I. Feinstein, and R.G. Palmer. 'Unlearning' has a stabilizing effect in collective memories. *Nature*, 304(5922):158, 1983.

[18] E. Gardner. Structure of metastable states in the Hopfield model. *Journal of Physics A: Mathematical and General*, 19(16):L1047, 1986.

[19] F. Crick and G. Mitchison. The function of dream sleep. *Nature*, 304(5922):111, 1983.

[20] G. Hinton and T.J. Sejnowski. *Unsupervised Learning: Foundations of Neural Computation.* The MIT Press, 1999.

[21] M. Benedetti, E. Ventura, E. Marinari, G. Ruocco, and F. Zamponi. Supervised perceptron learning vs unsupervised Hebbian unlearning: Approaching optimal memory retrieval in Hopfield-like networks. *The Journal of Chemical Physics*, 156(10):104107, 2022.

[22] M. Benedetti and E. Ventura. Training neural networks with structured noise improves classification and generalization. *arXiv.2302.13417, submitted to Journal of Physics A, under review*, 2023.

[23] E. Ventura, S. Cocco, R. Monasson, and F. Zamponi. Unlearning regularization for boltzmann machines. *arXiv:submit/5237914, submitted to Machine Learning: Science and Technology*, 2023.

[24] E. Ventura. *In preparation*, 2023.

[25] J. A. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Longman Publishing Co., Inc., 1991.

[26] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition.* Cambridge University Press, 1 edition, 2014.

[27] P. Peretto. Collective properties of neural networks: A statistical physics approach. *Biological Cybernetics*, 50:51–62, 1984.

[28] F. Altarelli, R. Monasson, G. Semerjian, and F. Zamponi. Connections to statistical physics. In *Handbook of Satisfiability 2nd Edition, Chapter 22*, pages 859–901. IOS, 2021.

[29] T.B. Kepler and L.F. Abbott. Domains of attraction in neural networks. *Journal de Physique*, 49(10):1657, 1988.

[30] D.O. Hebb. *The Organization of Behavior : A Neuropsychological Theory*. John Wiley and Sons, 1949.

[31] J.L. Van Hemmen, L.B. Ioffe, R. Kühn, and M. Vaas. Increasing the efficiency of a neural network through unlearning. *Physica A: Statistical Mechanics and its Applications*, 163(1):386, 1990.

[32] J.L. Van Hemmen and N. Klemmer. Unlearning and Its Relevance to REM Sleep: Decorrelating Correlated Data. In J.G. Taylor, C.L.T. Mannion, J.G. Taylor, E.R. Caianiello, R.M.J. Cotterill, and J.W. Clark, editors, *Neural Network Dynamics*, page 30. Springer London, London, 1992.

[33] S. Wimbauer, N. Klemmer, and J.L. van Hemmen. Universality of unlearning. *Neural Networks*, 7(2):261, 1994.

[34] D. Kleinfield and D. Pendergraft. Unlearning increases the storage capacity of content addressable memories. *Biophysical Journal*, 51:47, 1987.

[35] J. Horas and P. Pasinetti. On the unlearning procedure yielding a high-performance associative memory neural network. *Journal of Physics A: Mathematical and General*, 31:L463, 1998.

[36] B. Efron and C. Stein. The jackknife estimate of variance. *Annals of Statistics*, 9(3):586, 1981.

[37] E. Gardner, H. Gutfreund, and I. Yekutieli. The phase space of interactions in neural networks with definite symmetry. *Journal of Physics A: Mathematical and General*, 22(12):1995, 1989.

[38] A. Battista and R. Monasson. Capacity-resolution trade-off in the optimal learning of multiple low-dimensional manifolds by attractor neural networks. *Physical Review Letters*, 124(4):048302, 2020.

[39] N. Brunel, V. Hakim, P. Isope, J.P. Nadal, and B. Barbour. Optimal Information Storage and the Distribution of Synaptic Weights: Perceptron versus Purkinje Cell. *Neuron*, 43(5):745–757, 2004.

[40] N. Brunel. Is cortical connectivity optimized for storing information? *Nature Neuroscience*, 19(5):749–755, 2016.

[41] M.N. Barber. Finite-size scaling. In *Phase transitions and critical phenomena*, volume 8, pages 145–266. Academic Press, London, 1983.

[42] A. Theumann. Space of interactions with definite symmetry in neural networks with biased patterns as a spin-glass problem. *Physical Review E*, 53(6):6361, 1996.

[43] F. Aguirre-Lopez, M. Pastore, and S. Franz. Satisfiability transition in asymmetric neural networks. *Journal of Physics A: Mathematical and General*, 55(30):305001, 2022.