Contents

Introduction

During the past decade, deep learning has taken the world by storm, with stunning applications to fields as diverse as computer vision, natural language processing, automatic speech recognition, reinforcement learning, and bioinformatics [1]. Recently, the public debate around the potentialities and risks of AI has exploded, inspired by a series of impressive breakthroughs in computer vision and NLP which were open-sourced or released to the public as an interactive demo [2] [3].
However, despite its spectacular successes, the theoretical understanding of deep learning is seriously lagging behind. In fact, most recent breakthroughs have been guided almost exclusively by intuition and experimentation, and they are often the result of clever engineering and scaling up, without being grounded in a solid theoretical understanding. One of the main reasons for the existing gap between theory and applications in deep learning is the overwhelming complexity of these devices, which makes it extremely hard to investigate their behaviour analytically. That being said, there do exist methods that, in largely simplified settings, allow to study these systems and gain insights that can turn out to be important for more complex architectures as well. In this direction, one of the most promising lines of research employs tools originally developed in the framework of statistical physics and complex systems to study the geometrical and statistical properties of the large-scale optimization problems involved in the training of neural networks. In turn, the understanding of such properties can promote algorithmic advances in the training of these devices.
In the first chapter of my thesis, I will start by providing a very brief overview of some of the main ideas from statistical physics that are useful for the analysis of neural networks. Then, I will present some results from the literature that used these techniques to find evidence of the existence of large and connected dense clusters of solutions in the binary perceptron learning problem, which are both rare and accessible to algorithms.
In the second chapter, I will show how insight from the analysis of the geometry of the landscape of the binary perceptron has been used to derive very general and effective algorithmic schemes, that turn out to be useful not only for shallow networks, but also for deep learning. These are called entropic algorithms.
After that, in the third chapter, I will present some analytical results of my own, obtained in the study of a recurrent network of neurons with random couplings. Using techniques from statistical physics and spin glass theory, I computed the number of fixed points of

such a network under a zero-temperature metropolis dynamics that includes in the energy a self-coupling term, and studied the dependence of the results on the strength of the latter.
Finally, in the fourth chapter, I will present the results of some numerical experiments that I carried out with shallow and deep autoencoders in the random feature model. The aim was twofold. First, I assessed the ability of such devices to estimate the latent dimensionality of data. Then, in this unsupervised setting, I compared the performance of an entropic algorithm called replicated Adam with that of standard gradient-based methods like vanilla Adam, considering flatness in weight space, ability to denoise corrupted patterns and reconstruction error on unseen patterns.

Chapter 1

The Local Entropy

1.1 Statistical Physics of Learning

As I mentioned in the introduction, one of the most promising lines of research that try to understand theoretically learning in neural networks is based on tools from statistical physics and the study of complex systems. The star of the game is the Boltzmann distribution, a description of the equilibrium properties of any system that is equipped with an energy function in terms of a probability measure over its possible states σ:

$$p(\sigma;\beta) = \frac{1}{Z(\beta)} e^{-\beta E(\sigma)} \tag{1.1}$$

Here, $E(\sigma)$ is the energy of state σ, β is called inverse temperature and it controls the shape of the distribution, while the normalizing constant $Z(\beta)$ is called partition function and can be used to derive all the thermodynamic properties (i.e, averages over states) of the system at equilibrium. In many cases, quantities of interest turn out to be self-averaging and hence well-described by their ensemble average in the limit of a large system.
Importantly, the Boltzmann distribution is that of maximal entropy among those with a given average energy (which in 1.1 is controlled by β). In the limit $\beta \to \infty$, the distribution in 1.1 becomes a uniform distribution concentrated on the ground states of the energy. For a more thorough introduction to the tools of statistical physics, the interested reader can refer to [4].
In many problems of interest, there is randomness in the specific form of the energy. This feature is called quenched disorder. Spin glass theory provides with tools to study the typical behaviour of such systems, by computing the average logarithm of the partition function (which is generally self-averaging, contrary to the partition function itself) with methods like the replica method [5].
But how does this help study properties of optimization problems? In that case, take for the energy the objective function to be minimized, shifted so that its minima have 0 energy, and consider the Boltzmann distribution associated to it. In the limit $\beta \to \infty$, the Boltzmann distribution concentrates on the global minima of the optimization problem,

which are what we care about! By studying the properties of this Boltzmann distribution, interesting properties of the optimization landscape can be deduced [6].
As an illustrative example of this point, these techniques were used to establish a link between the phase transitions of the random k-SAT problem and the algorithmic difficulty of finding satisfying assignments. In fact, it was shown that the random k-SAT problem exhibits different phases, characterized by the arrangement of the solutions into one, few or many connected clusters. Polynomial-time algorithms seem to exist only when the system presents so-called 'unfrozen' clusters: extensive and connected clusters of solutions in which at most a sub-linear (in the size of the system) number of variables have a fixed value [7] [8].
In the following sections, I will present the results of [9], in which the authors used statistical physics techniques to analyze the energy landscape of the binary perceptron learning problem, and found evidence of the existence, in certain regimes, of dense and connected clusters of solutions that are both rare and accessible to algorithms. This analysis seems to have important implications for the understanding of learning even in deeper networks, and I will later show how it can be used to derive general algorithmic schemes that explicitly target these rare regions.

1.2 Binary Perceptron Learning

1.2.1 Equilibrium Analysis

To set the notation, let me formally define the binary perceptron learning problem. A binary perceptron maps an N-dimensional input vector $\xi \in \{+1, -1\}^N$ to a label $\tau(W, \xi) = sgn(W \cdot \xi)$, where $W \in \{+1, -1\}^N$ is a vector of binary weights. Given αN random patterns ξ^μ and their corresponding labels σ^μ, the learning problem consists in finding a vector of weights W^* such that $\sigma^\mu = \tau(W^*, \xi)$ for every $\mu = 1, 2, ..., \alpha N$.
Because of their tractability, two scenarios are usually studied. In both, the components ξ_i^μ of the patterns are drawn iid from an unbiased distribution over ± 1. As for the labels, they can either be random from the same distribution (classification scenario), or they can be provided by another 'teacher' perceptron that acts on the patterns (generalization or teacher-student scenario).
Both cases are known to display a phase transition in the thermodynamic limit $N \to \infty$ as α crosses a critical threshold. In the classification scenario, for $\alpha < \alpha_c = 0.833$ the typical problem has solution with probability 1, while for $\alpha > \alpha_c$ the probability that this happens is 0 [10]. In the teacher-student case instead, for $\alpha < \alpha_{TS} = 1.245$ there are exponentially many solutions, while when $\alpha > 1.245$ the only solution to the typical problem is the teacher itself [11].
Define:

$$X_\xi(W) = \prod_\mu \mathbb{1}(\tau(W, \xi^\mu) = \sigma^\mu) \tag{1.2}$$

in words, X_ξ is an indicator of whether the vector of weights W is a solution to the learning problem with patterns ξ^μ and labels σ^μ. The standard equilibrium analysis of the problem consists of defining an energy $E(W)$ that counts the errors made by the perceptron W on the training set. In the zero-temperature limit, in which the Boltzmann distribution concentrates on the solutions, the partition function reduces to

$$Z_{eq} = \sum_W X_\xi(W) \tag{1.3}$$

The typical case is described by taking the quenched average $\langle \log Z_{eq} \rangle$ over all possible realizations of the patterns, which can be done using the replica method.
This equilibrium description reveals the existence, in the typical classification case, of an exponential number of local minima that can easily trap standard search algorithms based on energy minimization, such as Monte Carlo methods. Furthermore, the typical solutions are isolated (they have extensive mutual Hamming distance) and thus hard to access [10] [11] [12] [13] [14] [15]. A similar picture also holds for the teacher-student scenario [9].
With this glassy landscape in mind, it seems unreasonable to expect that efficient algorithms capable of solving the learning problem can exist. And yet, there are at least four local and distributed algorithms that are believed, based on numerical evidence, to be able to solve the classification problem with nonzero capacity (i.e., with $\alpha > 0$ in the large N limit) and in a sub-exponential running time. These are Reinforced Belief Propagation [16], Reinforced Max-Sum [17], SBPI [18] and CP+R [19], and they achieve capacities between 0.69 and 0.75. Also in the teacher-student case these algorithms perform well, except in a window $1.0 \leq \alpha \leq 1.5$ around α_{TS}.
On top of that, numerical experiments using random walks to explore the local geometry around the solutions found by algorithms provided evidence that these are not in general isolated, but rather they tend to belong to large connected clusters of solutions [9]. This is in clear contrast with what the equilibrium analysis suggests. In fact, the equilibrium description will turn out to be insufficient to capture the properties of the known efficient algorithms, which are attracted to sub-dominant and yet accessible dense clusters of solution. This will be the subject of the next section.

1.2.2 Large-deviation Analysis: the Local Entropy

Motivated by the discrepancy between the equilibrium description of the landscape of the binary perceptron and the behaviour of algorithms, the authors of [9] have studied a large-deviation free energy density that increases the importance of solutions surrounded by a large number of other solutions:

$$F(d, y) = -\frac{1}{Ny} \log \left(\sum_{\{\tilde{W}\}} X_\xi(\tilde{W}) N(\tilde{W}, d)^y \right), \tag{1.4}$$

Here, y is an inverse temperature, and $N(\tilde{W}, d)$ counts the number of solutions at normalized Hamming distance d from the reference $\tilde{W}$:

$$N(\tilde{W}, d) = \sum_{\{W\}} X_\xi(W)\delta(W \cdot \tilde{W}, N(1-2d)) \tag{1.5}$$

The system under study is therefore governed by a formal energy density given by:

$$\epsilon(\tilde{W}) = -\frac{1}{N}\log N(\tilde{W}, d) \tag{1.6}$$

Importantly, the presence of the factor $X_\xi(\tilde{W})$ in 1.4 is a constraint that leaves nonzero statistical weight only to solutions.
The formal energy associated to 1.6 is sometimes called in the literature 'local free entropy'. Another important quantity, $S_I(d, y) = \left\langle \epsilon(\tilde{W}) \right\rangle_{\xi, \tilde{W}}$, is often called 'local entropy', and it represents the average density of solutions surrounding a given solution, where the average is taken over the ensemble and the pattern realizations. I will note here that there is some confusion in the literature when it comes to the exact usage of the phrases 'local entropy' and 'local free entropy'. In fact, in many cases the shorter local entropy is used to refer to (some version of) the local free entropy (e.g, in the phrase 'local entropy landscape', common in the literature). Anyways, the intended usage is usually clear from the context. In some cases, especially when discussing algorithms, if that doesn't generate confusion I will do the same.
Because of the reweighting provided by the factor $N(\tilde{W}, d)$, in this large-deviation ensemble the statistical importance of isolated solutions is suppressed, in favor of solutions surrounded by a large number of other solutions (where the precise meaning of 'surrounded' is modulated by the choice of d). The authors of [9] have studied the system in the replica-symmetric (RS) ansatz. The results were found to be consistent up to a maximum inverse temperature $y^*(\alpha, d)$, beyond which the predictions become unphysical. Here I present the main results in schematic form (for the maximum consistent inverse temperature, hence dropping the y-dependence), which were found to be qualitatively very similar for the two scenarios below the respective critical values of α:

- For all values of α below the critical value, $S_I(d) > 0$ in a neighborhood of $d = 0$. This signals the existence of exponentially large clusters of solutions. Their size reduces as α grows, and vanishes at the critical value. Furthermore, close to $d = 0$, the curves $S_I(d)$ are all approximately equal to the one for $\alpha = 0$. Since all configurations are solutions when $\alpha = 0$, this means that these extensive clusters are extremely dense at their core.

- For d large enough, $S_I(d)$ becomes equal to the equilibrium entropy: for large distances, the typical solutions dominate the ensemble.

- There is a critical value α_U of α (0.77 and 1.1 in the classification and teacher-student cases, respectively) beyond which there exist regions of d where the RS

ansatz becomes incorrect. For $\alpha < \alpha_U$, instead, the RS ansatz is consistent and the curves $S_I(d)$ are increasing. These facts could be a signal of a change in the geometry of the solutions: for $\alpha < \alpha_U$ the dense cores would be immersed in a large connected structure, while for $\alpha > \alpha_U$ this structure would fracture into isolated substructures and the dense cores would become harder to find.

- in the teacher-student scenario, the generalization error of the ground states of the large-deviation ensemble is generally better than that of the typical solutions, and the theoretical predictions of the generalization error agree with the experimental findings using the SBPI algorithm.

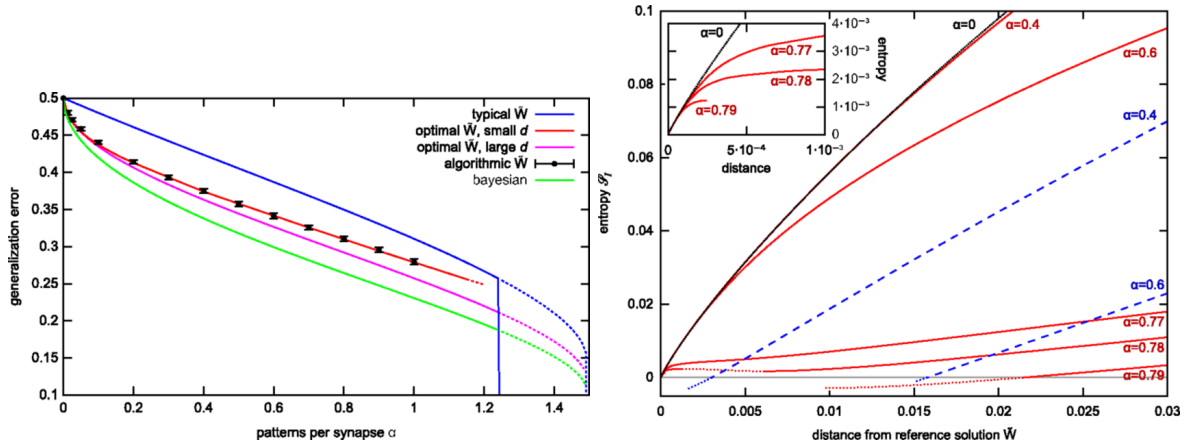


Figure 1.1: Reprinted from [9]. (left) Generalization error curves in the teacher student scenario. Algorithmic (SBPI) solutions have better generalization than typical solutions. (right) Large-deviation local entropy curves at varying distance d from the reference solution $\tilde{W}$ for various α in the classification scenario; blue dotted curves come from the equilibrium analysis.

The numerical results were reported to be in agreement with the theoretical predictions. Furthermore, while the results here presented hold for a shallow network with binary synapses, they will turn out to be important (and I'll comment on this in a later section, pointing to the relevant references in the literature) for deep networks with continuous weights as well.

1.3 The Local Entropy Landscape

Summing up, the results of [9] have shown that in the loss landscape of the binary perceptron there exist extensive and connected clusters of solutions, with an extremely dense core, that are rare and yet accessible to algorithms. Building on these results, the authors of [20] have extended the analysis of the previous section to the case in which the explicit constraint that the reference configuration must be a solution is removed in the definition

of the free energy density 1.4, i.e. not only solutions but all configurations have a nonzero statistical weight depending on the number of solutions by which they are surrounded.
In this case, the theoretical analysis is more challenging since the RS ansatz gives unphysical results; therefore, a 1-step of Replica Symmetry Breaking (RSB) was studied, in the limit $y \to \infty$. The qualitative picture that emerges from the analysis is essentially the same as in the constrained case. Furthermore, given a configuration $\tilde{W}$ that is a ground state of the unconstrained large-deviation ensemble, the probability that $\tilde{W}$ is a solution tends exponentially to 1 as $d \to 0$, despite the fact that the explicit constraint has been removed. This can be intuitively understood thinking that, for small d, $\tilde{W}$ will be roughly at the center of a dense cluster of solutions, and will therefore be itself a solution with high probability.
This fact is very important, since it suggests a way to translate the theoretical findings about the structure of the space of solutions into a general, practical solver that, instead of minimizing the energy directly, attempts to minimize the local free entropy 1.6. If possible, this will produce configurations immersed in a dense region of solutions, which are very likely to be themselves solutions at small d and are inherently robust. Furthermore, as it turns out, the local entropy landscape is smoother than the rough energy landscape, and hence easier to optimize.
Of course, this ignores an important challenge: the local entropy is not easily computed in general, as it involves an integral over the whole configuration space. In the next chapter, I will present a few ways in which this obstacle has been overcome and derive a general algorithmic scheme whose utility, according to numerical evidence, is not limited to the shallow networks with discrete synapses in which the theoretical analysis was performed, but extends to modern neural network architectures as well.

Chapter 2

Entropic Algorithms

2.1 Local Entropy, in general

The work reviewed in the last chapter has shown that, at least in shallow networks with discrete synapses, the behaviour of the known efficient algorithms is well-captured by studying a large-deviation measure that enhances the weight of configurations immersed in a region with a high density of solutions. In fact, to sum up, the work of [9] and [20] has shown that:

- known efficient algorithms find solutions belonging to regions with a high density of solutions, and fail when these do not exist;
- these solutions are rare, as they do not emerge from the equilibrium description that is dominated by numerous isolated solution;
- despite being sub-dominant, these solutions are accessible to algorithms;
- these solutions are inherently robust, being surrounded by numerous other good configurations. And in fact, they tend to have better generalization capabilities than the typical solutions.

In this chapter, I will show how their insights can be used to derive algorithms that explicitly target these dense regions, and can be successfully applied even beyond the simple setting where the theoretical analysis was carried out. To this end, let me first reformulate the local free entropy of 1.6 in a more general form:

$$\Phi(x, \beta, \gamma) = \log \int dx' e^{-\beta E(x') - \gamma d(x,x')} \tag{2.1}$$

Here, $d(\cdot, \cdot)$ is a strictly increasing function of the distance between configurations, whose definition depends on the particular model at hand. In the case of the binary perceptron, it was an affine transformation of the Hamming distance. β is an inverse temperature, and in the limit $\beta \to \infty$ the term $e^{-\beta E(x')}$ becomes an indicator of whether x' is a ground state

of the energy E, like in the case of the binary perceptron. Finally, the hard constraint on the distance through the choice of d has been substituted by a soft constraint embodied by the term $e^{-\gamma d(x,x')}$, which decays exponentially the importance of the configurations as their distance from the reference x increases.
The parameter γ modulates the decay of the weight with the distance, allowing to concentrate on wider or narrower regions around the reference configuration. Furthermore, in the thermodynamic limit the integral is dominated by configurations having a certain fixed distance $d^*(x, \beta, \gamma)$ from the reference x: γ plays the same role as d used to. However, while the hard constraint on the distance is better suited for the theoretical analysis, the soft constraint through γ is better suited for the implementation of algorithms, as we shall see.
The integral in 2.1 of course reduces to a sum when the state space is discrete. To $\Phi(x, \beta, \gamma)$, as in the case of the binary perceptron, we can associate a large-deviation measure given by:

$$p(x; \beta, y, \gamma) = \frac{1}{Z(\beta, y, \gamma)} e^{y\Phi(x,\beta,\gamma)}, \tag{2.2}$$

Like before, y is the inverse temperature. When $y \to \infty$, the distribution in 2.2 concentrates on the ground states of 2.1.
With this general framework in mind, we are ready to show how to design algorithms that try to minimize 2.1. First, I will present an algorithm called Entropy-driven Monte Carlo, which optimizes the local entropy landscape by directly obtaining estimates of the local entropy. Then, I will show how obtain a general algorithmic scheme that allows to explore the local entropy landscape through a number of interacting replicas of the system under consideration. Finally, I will present an algorithm called Entropy SGD, that attempts to optimize the local entropy by estimating its gradient.

2.2 Entropy-drive Monte Carlo

As an important proof of concept, the authors of [20] proposed to optimize the local free entropy 2.1 using a standard Monte Carlo simulation in which random local updates are accepted or rejected based on the Metropolis rule at fixed temperature y^{-1}. In their experiments, they considered the version of 2.1 with $\beta \to \infty$, in which the local entropy counts the number of ground states of the energy surrounding a reference configuration, with a weight that decreases exponentially with the distance from the reference.
In order to estimate the local entropy, which is of course frequently necessary to run a Monte Carlo simulation on its landscape, the authors proposed to resort to the Belief Propagation (BP) algorithm. This is a cavity method that allows to approximately compute the equilibrium properties at a given temperature for a system described as a factor graph. The validity of the approximation, called Bethe-Peierls approximation, must be assessed for each particular problem at hand [4].

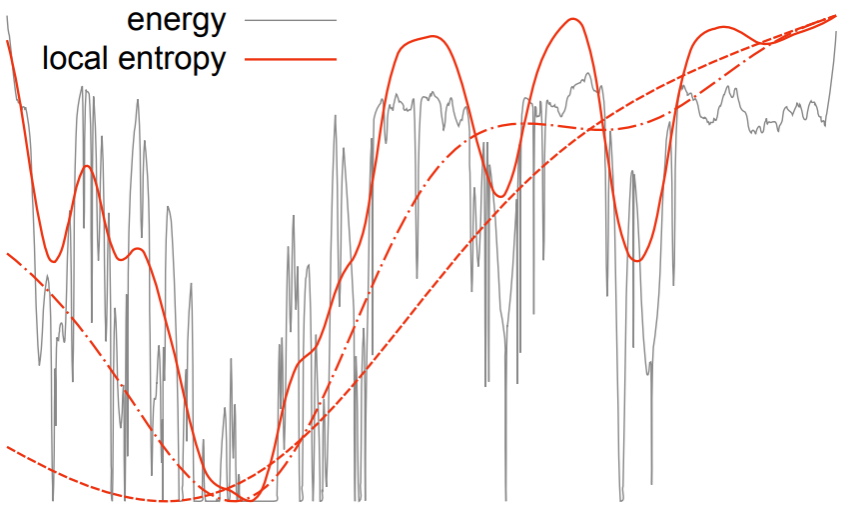


Figure 2.1: Reprinted from [21]. Energy landscape compared to local entropy landscape in an illustrative toy example. Finer-grain features of the energy landscape progressively appear as γ increases. For large γ, the global minimum is located inside a wide global optimum of the energy.

The resulting algorithm is called Entropy-driven Monte Carlo (EdMC). The authors of [20] tested EdMC against standard Simulated Annealing (SA) in two very different constraint satisfaction problems, the binary perceptron learning and the 4-SAT problem, with excellent results: even without cooling (i.e., using the simple greedy strategy of a zero-temperature Monte Carlo), EdMC is orders of magnitude faster than SA in terms of attempted moves, and does not suffer from trapping in local minima. Furthermore, EdMC can be made even more efficient through a simple heuristic called 'scoping': the γ parameter is progressively increased during training, making the local entropy computation focus on progressively narrower regions around the reference configuration. This strategy was reported to be far more important than cooling in practice, and proved to be crucial in some cases to produce a quick solution.
The scoping procedure will turn out to be important for other algorithms as well, so let me spend a few words about its interpretation. Consider the expression 2.1. When γ is very small, the local entropy gives non-negligible weight to an extremely large region around the reference configuration, resulting in a very smooth landscape. In the limit $\gamma \to 0$, it becomes completely flat. On the contrary, the larger *gamma*, the more the local entropy becomes blind to anything that happens outside a small region around the reference configuration, and the rougher the local entropy landscape becomes. In the limit $\gamma \to \infty$, the local entropy landscape becomes the same as the energy landscape. With this in mind, let's turn to scoping. In the first phases of training, when γ is small, the algorithm explores the energy landscape at a very low resolution, and it is quickly driven to regions populated by a large number of configurations with low energy; as time goes on, and γ increases, the energy landscape is explored on progressively finer scales, and

the algorithm gets to the minima of the energy within those densely populated regions. The success of EdMC depends on the possibility to estimate the local entropy using the BP algorithm, which won't be the case in all scenarios. However, the effectiveness of a simple learning protocol like EdMC in accessing the sub-dominant clusters that minimize the local entropy is an important proof of concept: it suggests that it could be possible to design simple dynamical processes that are capable of navigating the smooth local entropy landscape and find configurations with low energy that are inherently robust. In the next section, I will present one very general class of such algorithms.

2.3 The Robust Ensemble

Consider again the large-deviation measure in 2.2. Following [21], when y is a non-negative integer the partition function can be rewritten as follows:

$$Z(\beta, y, \gamma) = \int dx\ e^{y\Phi(x,\beta,\gamma)} = \int dx\ \int \prod_{a=1}^{y} dx^a e^{-\beta \sum_{a=1}^{y} E(x^a) - \gamma \sum_{a=1}^{y} d(x,x^a)} \tag{2.3}$$

In this form, the partition function can be interpreted as describing a system of $y+1$ interacting replicas of the original system. One of them, with configuration denoted x, acts as a reference towards which the other replicas are attracted through the terms $\gamma\ d(x, x^a)$ in the formal energy. The strength of the attraction is modulated by the parameter γ. The y identical replicas, but not the reference, are also subject to the energy $E(x^a)$ of the original system.
It is clear that studying the equilibrium statistics of the replicated system and tracing out the replicas is equivalent, by construction, to studying the large-deviation measure on the original system. This simple observation provides a general scheme to explore robust and accessible regions of the energy landscape: replicate the model y times, add an attractive interaction with a reference and let the replicas collectively explore the energy landscape, subject to the interaction with the reference. Ideally, we would like $y \to \infty$ to access the ground states of the local entropy. However, as it turns out, already a small number of replicas has significant effects.
It is also possible to trace out the reference configuration, leaving us with a system of y interacting replicas that is called by the authors of [21] 'Robust Ensemble' (RE):

$$Z(\beta, y, \gamma) = \int \prod_{a=1}^{y} dx^a e^{-\beta\left[\sum_{a=1}^{y} E(x^a) + A(\{x^a\},\beta,\gamma)\right]} \tag{2.4}$$

$$A(\{x^a\}, \beta, \gamma)) = -\frac{1}{\beta} \log \int dx\ e^{-\gamma \sum_{a=1}^{y} d(x,x^a)} \tag{2.5}$$

In the following, I will demonstrate how the replication procedure can be applied to augment a variety of different algorithms by describing two representative examples: Replicated Simulated Annealing and Replicated Stochastic Gradient Descent.

2.3.1 Replicated Simulated Annealing

A straightforward but illustrative application of the general scheme outlined above is Replicated Simulated Annealing (rSA). The idea is simple: we sample the space of configurations with a Monte Carlo Markov Chain that uses the objective function given either by 2.3 or by 2.4. I will comment explicitly only on the second case, since it is less immediate to understand.
The reference configuration which mediates the interaction among the replicas of the RE is traced out in this representation, therefore we want to sample from a probability distribution:

$$P(\{x^a\}) \propto \int dx \ \exp\left(-\beta \sum_{a=1}^{y} E(x^a) - \gamma \sum_{a=1}^{y} d(x, x^a)\right) \tag{2.6}$$

We explicitly maintain the configuration of the y replicas that are subject to the energy; the most probable configuration for the reference is instead obtained by computing the baricenter of the replicas with respect to the distance d in the state space.
To perform the sampling effectively, we employ both a cooling procedure on β and a scoping procedure on γ. At each step, we propose a random local update in the configuration of a random replica, and we accept it or reject it according to the standard Metropolis rule, using the formal energy function

$$\beta \sum_{a=1}^{y} E(x^a) + \gamma \sum_{a=1}^{y} d(\bar{x}, x^a) \tag{2.7}$$

where $\bar{x}$ is the baricenter of the configurations of the replicas.
The authors of [21] have compared the performance of rSA with that of standard SA in the case of shallow networks (a perceptron and a committee machine with discrete synapses). Their numerical results show that, with optimized annealing and scoping parameters, the minimum number of iterations required to find a solution scales exponentially with the size of the system with vanilla SA, while it only scales polynomially using rSA. Furthermore, the difference in performance was reported to widen greatly as the number of patterns per synapse increases. Remarkably, already $y = 3$ replicas were enough to obtain a significant improvement over the non-replicated version.
The scheme of rSA has strong similarities with the EdMC algorithm presented before. The main advantage of rSA over EdMC is that replicating the system avoids the need to use BP for the estimation of the local entropy, making the procedure simpler and much more widly applicable. On the other hand, with systems in which BP can provide a reasonable estimate of the local entropy, it can do so directly at a given temperature, and thus it eliminates the need to thermalize the replicas.

2.3.2 Replicated SGD

Monte Carlo methods can become computationally extremely expensive for very large systems, which require long times to thermalize. A popular alternative, especially in

the deep learning field, are gradient-based methods such as Stochastic Gradient Descent (SGD) and its variants. Despite little theoretical guarantees in non-convex scenarios (although some progress is slowly being made), these heuristic methods work extremely well in practice, and they are responsible for virtually every recent success in the field of deep learning.
Also in the case of SGD, we can consider its replicated version in the form of the Robust Ensemble. Like in the case of rSA, we replicate the system y times, and we optimize the loss given in 2.7. At each step, when a minibatch is presented, we first let each replica update its weights independently from the others, subject to the energy, using SGD, Nesterov momentum, Adam or any other gradient descent variant. Then, we have each replica make a step in the direction of the baricenter, with length proportional to γ and to the gradient of the distance.
We usually adopt a scoping procedure, progressively increasing γ to let the replicas explore the energy landscape on progressively finer scales, with the benefits already discussed in the paragraph about EdMC. It should be noted that, in practical implementations, the inverse temperature β is usually set to 1 for simplicity, since ultimately what is relevant is the relative importance of the energy and attractive terms, and tuning γ already allows control over that. This simplification gives what is sometimes called in the literature 'local entropy loss':

$$L_{LE}(\{x^a\}) = \sum_{a=1}^{y} E(x^a) + \gamma \sum_{a=1}^{y} d(\bar{x}, x^a) \tag{2.8}$$

The resulting algorithm is summarized as follows. The last line can be substituted by the update of any variant of gradient descent.

Algorithm 1 Replicated SGD

Input: $\{w^a\}_{a=1}^{y}$
  for $t = 1, 2, ...$ **do** :
    $\bar{w} \leftarrow \frac{1}{y} \sum_{a=1}^{y} w^a$
    for $a = 1, ..., y$ **do** :
      $\Xi \leftarrow$ sample minibatch
      $dw^a \leftarrow \nabla L(w^a; \Xi)$
      $dw^a \leftarrow dw^a + \gamma(w^a - \bar{w})$
      $w^a \leftarrow w^a - \eta \, dw^a$
    end for
  end for

For a binary committee machine, numerical experiments with $y = 7$ replicas in [21] showed that replicated SGD greatly improves the capacity of the network (from 0.3 to 0.6 patterns per synapse), decreases the error rate even when not all patterns are fit correctly, and requires less epochs to learn the dataset compared to standard SGD. Furthermore, in the same setting, already with $y = 3$ the authors reported significant improvements with respect to the non-replicated case.

When it comes to modern, deep architectures, the authors of [22] have performed extensive numerical experiments with image classification tasks (CIFAR-10, CIFAR-100, Tiny Imagenet) using deep feed-forward architectures (ResNet, PyramidNet + ShakeDrop, DenseNet to name a few). They reported that replicated SGD with $y = 7$ replicas achieves consistently lower generalization error compared to standard SGD across the benchmarks and networks that they experimented with. Furthermore, it was observed that the configurations sampled by rSGD are generally located in flatter regions of the energy landscape. These results suggest that the geometrical properties studied in [9] in the context of shallow networks could be relevant for deep architectures as well. This is in fact the case, and I will now point to a few relevant sources in the literature that prove this.

To begin with, analytical results in simplified settings [23] [24], which adapt the large-deviation approach presented in the previous chapter to the continuous case, show that, also in the loss landscape of two-layer neural networks with continuous weights, a few wide flat minima (WFM), that are robust to perturbation of weights and of inputs, coexist with numerous narrow minima and critical points. Furthermore, extensive numerical testing with deep networks on an array of architectures and tasks in [25] have shown that the flatness of the minima (more precisely, the local energy, which I will come back to in the last chapter) is one of the most consistent predictors of generalization performance.

Finally, empirical evidence in [26], obtained analyzing the spectrum of the Hessian of the energy at configurations found by vanilla SGD, shows that, for a wide variety of datasets and of network architectures and sizes, the optima found by SGD are mostly flat, with few directions with large positive curvature and few with almost null negative curvature: similarly to the binary perceptron case [9], solutions found by efficient algorithms tend to lie in "wide valleys" of the energy landscape rather than in sharp, isolated minima, despite the latter being more abundant.

These results, together with the effectiveness of replication with deep architectures, establish that the conclusions of the study of the binary perceptron have relevance well beyond the simple scenario in which they were first drawn, and the RE provides a simple yet effective and very general scheme to algorithmically exploit this fact.

With this in mind, in the next section I will present yet a different approach to optimize the local entropy, which is based on the repeated estimation of its gradient within a gradient descent scheme.

2.4 Entropy SGD

The Robust Ensemble presented in the previous section provides an extremely general and simple to implement algorithmic scheme that allows to access dense and robust regions of the energy landscape. This is achieved through (approximately, at finite y) optimizing the local entropy by allowing replicas of the system to collectively explore the energy landscape on progressively finer scales, thanks to a time-dependent interaction that pulls the replicas closer and closer to each other.

In this section, I want to present a different approach to maximizing the local free entropy of 2.1 that was proposed in [26]. For the same reasons remarked in the case of rSGD, we omit the inverse temperature parameter and aim to minimize over x the loss

$$L(x,\gamma) = -\log \int dx' \exp\left(-E(x') - \gamma d(x,x')\right) \tag{2.9}$$

In order to do this, we want to use (any variant of) the SGD algorithm. This requires estimating the gradient of $L(x)$. A simple computation shows that the gradient of the loss function can be written as:

$$\nabla_x L(x,\gamma) = -\gamma(x - \langle x' \rangle) \tag{2.10}$$

where the expectation of x' is taken with respect to the modified Gibbs measure

$$p(x';x,\gamma) \propto \exp\left(-E(x') - \gamma d(x,x')\right) \tag{2.11}$$

When using a stochastic optimization method based on mini-batches such as SGD, the same computation shows that to estimate the gradient we substitute the energy of x' in 2.11 with the batch average of the energy. Concretely, given a randomly sampled batch $\xi^1, ..., \xi^m$ of m samples, we want to take the expectation in 2.10 with respect to:

$$p(x';x,\gamma,\xi) \propto \exp\left(-\frac{1}{m}\sum_{\mu=1}^{m} E(x';\xi^\mu) - \gamma d(x,x')\right) \tag{2.12}$$

The expectation is hard to compute in general. The authors of [26] proposed to approximate it using a Monte Carlo method, and in particular they suggest stochastic gradient Langevin dynamics (SGLD). This is a MCMC algorithm designed to draw samples from a Bayesian posterior, and it scales well to large datasets because it employs mini-batch updates [27].
The resulting algorithm, called Entropy SGD, has a strong flavor of 'SGD inside SGD', to use the authors' words: there is an inner loop that runs SGLD to obtain estimates of the gradient of L, while an outer loop running some variant of SGD uses the computed gradients to navigate the local entropy landscape. As usual, a scoping procedure on γ can be employed. Making everything explicit, the algorithm is summarized in Algorithm 2.

Under some assumptions on the eigenvalues of the Hessian of the energy, the authors of [26] proved rigorously that the objective optimized by Entropy SGD is smoother than the energy, and that entropy SGD generalizes better than vanilla SGD (precise statements and proofs can be found in the referenced paper). Furthermore, extensive experiments with deep architectures in [22] (for a description of the setting, see the paragraph on rSGD) have shown numerically that solutions found by entropy SGD generalize better than those obtained with vanilla SGD, supporting the view outlined at the end of the previous section about wide flat minima and their importance for learning in deep networks.

Algorithm 2 Entropy SGD

Input: w
  for $t = 1, 2, ...$ **do** :
    $w', \mu \leftarrow w$
    for $l = 1, ..., L$ **do** :
      $\Xi \leftarrow$ sample minibatch
      $dw' \leftarrow \nabla L(w'; \Xi) + \gamma(w' - w)$
      $w' \leftarrow w' - \eta' dw' + \sqrt{\eta'}\epsilon N(0, I)$
      $\mu \leftarrow \alpha\mu + (1 - \alpha)w'$
    end for
    $w \leftarrow w - \eta(w - \mu)$
  end for

Chapter 3

The asymmetric Hopfield model

3.1 The Hopfield model

The Hopfield model [28] [29] is a model of an associative memory consisting of a recurrent network of binary neurons that can store a set of patterns by adjusting the strength of the interactions between pairs of neurons in the network. Denote with $\sigma_1, ..., \sigma_N$ the neurons with possible states ± 1 and with $\xi^1, ..., \xi^P$ a set of N-dimensional binary patterns to be stored. The dynamics of the network is governed by the energy function:

$$E(\{\sigma_i\}) = -\frac{1}{2} \sum_{i<j} J_{ij} \sigma_i \sigma_j \tag{3.1}$$

where the coupling J_{ij} is a measure of the correlation in the activities of neurons i and j throughout the pattern set, a choice loosely inspired by neuroscience and called in the literature 'Hebb rule':

$$J_{ij} = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu \tag{3.2}$$

Biologically, there are at least two reasons of concern with 3.2. First, the interaction between neurons i and j is reinforced when both neurons are silent, which is not usually observed in biological networks. Second, the interaction is symmetric, while the synapses in the brain are often found to be one-way, with neurons obeying Dale's law, according to which a given neuron tends to only form excitatory of inhibitory synapses, and hence are necessarily asymmetric. However, the simplicity of this model makes it possible to study it analytically, and to gain important insights into the emergent collective properties of large computational systems.
In the limit of large N it can be shown that with the learning rule of 3.2, when $\alpha = P/N$ is not too big, the network will act as an associative memory: when presented with a new pattern, letting the dynamics evolve will retrieve the closest pattern among those presented during training. Stored patterns will in fact be fixed points of the dynamics,

satisfying:

$$\xi_i^\mu = sgn\left(\sum_{j=1}^{N} J_{ij}\xi_j^\mu\right) \tag{3.3}$$

However, they will not be the only fixed points; there also exist so-called spurious stable states. Some of them are a linear superposition of an odd number of stored patterns, but there are others which are not correlated with any finite number of the original patterns and are sometimes called spin-glass-like. Stored patterns (as well as spurious states) have a certain 'basin of attraction' around them: initializing the network in a state sufficiently close to a stored pattern will make the dynamics correct the mistakes and converge to it. These properties make the Hopfield model a simple yet powerful framework for understanding how neural networks can store and retrieve information. A full account of the basic theory and properties of the model can be found in [30].

3.2 A random asymmetric Hopfield-type model

The symmetry of the synapses of the Hopfield model was crucial to study its behaviour using equilibrium statistical physics techniques [31] [32] [33] [34]. However, in addition to being biologically more plausible, the more difficult asymmetric case has also be shown to possess interesting properties. In fact, the results of [35] [36] [37] [38] suggest that the introduction of random asymmetry in the synaptic connections has the effect of destabilizing the sping-glass-like spurious attractors which characterize the symmetric Hopfield model, while retrieval states remain stable with moderate amounts of asymmetry. Furthemore, [39] argued that asymmetry could allow to make trajectories leading to spin-glass-like fixed points chaotic, allowing to discriminate between convergence to stored patterns and convergence to these spurious attractors based on the temporal evolution of the network. In many of these studies, a random non-symmetric component is added to the symmetric synaptic connection matrix obtained by the Hebb rule 3.2. This can be thought to reflect a tabula-non-rasa scenario [40], in which Hebbian learning takes place on top of an existing random initial state. For example, in [41] the authors study the number of fixed points of such a model, and find that there is a critical level of asymmetry, dependent on the ratio of patterns to neurons, after which the expected number of fixed points is 0.
It could be interesting, especially through the lens of the tabula-non-rasa interpretation, to take this one step further and investigate the properties of a fully random asymmetric connectivity matrix. If successful, this could provide insight into what geometrical properties of the state space (when it comes to the dynamics of the network, i.e. fixed points, attractors, chaos, etc.) are available for a potential learning algorithm to work with from the get-go when confronted with a typical random network. Furthermore, this could have implications for the understanding of memory and learning in biological networks as well. As an important first step in this direction, in this chapter I will use statistical physics techniques to study exactly the number of fixed points of a model of this kind, under a slightly modified dynamics that includes a self-coupling term, and study the dependence

on its intensity. To do so, I will use techniques from statistical physics and spin glass theory. First, let me formally define the model.
I consider a recurrent network of N binary neurons $s_1, ..., s_N$ with states ± 1. There is a random coupling J_{ij}, drawn iid and unbiased from ± 1, for every ordered pair (i, j) of neurons. The dynamics is modified to include a self-coupling term, whose intensity is treated as an external parameter D and is the same for all neurons:

$$s_i \to Ds_i + \frac{1}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_j \tag{3.4}$$

To study this problem, I will use the general approach described in the section 'statistical physics of learning' of the first chapter. I will start by defining an energy function,

$$E = \frac{1}{2} \sum_{i=1}^{N} \left[1 - s_i sgn \left(Ds_i + \frac{1}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_j \right) \right] \tag{3.5}$$

that counts the number of spins (I will use interchangeably 'neuron' and 'spin') that would flip under the dynamics 3.4. The fixed points I'm interested in are the ground states of the energy 3.5. I will then study the Boltzmann distribution with inverse temperature β associated to the energy, and take the $\beta \to \infty$ limit so that the Boltzmann measure concentrates uniformly on the ground states; By computing the entropy, which is easily derived from the partition function, in this limit, I will be able to compute the number of ground states (i.e., fixed points) of the typical network.
There is one difficulty: the average of the partition function is not informative, in general; to have a self-averaging quantity, the logarithm of the partition function must usually be considered (see the discussion in the 'statistical physics of learning' section). Its average, however, is harder to compute. In order to do it without having to resort to the replica method, I will use the following approach.
Let Z_J be the partition function of the system when the stochasticity due to the disorder in the couplings has realized; we are interested in computing:

$$\langle \log Z_J \rangle_J \tag{3.6}$$

By Jensen's inequality, we have the inequality

$$\langle \log Z_J \rangle_J \leq \log \langle Z_J \rangle_J \tag{3.7}$$

So if we can compute $\langle Z_J \rangle_J$, which is in general easier to do, we obtain an upper bound to the value that we care about. However, after doing this, I will show that in this particular case the upper bound is tight, and the results obtained using $\log \langle Z_J \rangle_J$ are exact. To show this, I will compute the second moment of the partition function, $\langle Z_J^2 \rangle_J$, and prove that

$$\lim_{N \to \infty} \frac{\langle Z_J^2 \rangle_J - \langle Z_J \rangle_J^2}{\langle Z_J \rangle_J^2} = 0 \tag{3.8}$$

If this is the case, the variance-to-mean ratio for the partition function is 0 in the thermodynamic limit: the partition function becomes extremely concentrated around its mean value, and the inequality in 3.7 becomes an equality. Then, we can use the expression for $\log \langle Z_J \rangle_J$ to compute the entropy and, in the $\beta \to \infty$ limit, we will obtain the exact number of fixed points of the system, in the limit of a large system.
This is what I set out to do in the next sections of this chapter.

3.3 First moment

I will now compute the first moment of the partition function. To ease the notation, I will denote the average of the partition function over coupling realizations with $\bar{Z}$. I want to compute, in the limit $N \to \infty$,

$$\bar{Z} = \sum_J p(J)\ Z_J = e^{-\frac{\beta}{2}N} \sum_{s,J} p(J)\ \exp\left(\frac{\beta}{2} \sum_i sgn\left(D + \frac{1}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_i s_j\right)\right) \tag{3.9}$$

where the outer summation is over all values of s and of J, while the indices i and j in the inner summations run over all integers $1, 2, ..., N$.
For this computation, as is customary in the statistical physics literature, I will use well known properties of the delta function without resorting to the abstract formalism of tempered distributions. A rigorous treatment of the properties of the Dirac delta requires some fairly sophisticated mathematical machinery, and can be found for example in [42].
I want to use the integral representation of the Dirac delta function:

$$\delta(x) = \int_{-i\infty}^{i\infty} \frac{d\hat{x}}{2\pi i} e^{-x\hat{x}} \tag{3.10}$$

I introduce auxiliary integration variables h_i , where $i = 1, ..., N$, and enforce the equalities $h_i = \frac{1}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_i s_j$ through convolution with delta functions, represented as integrals over the variables $\hat{h}_i$ according to 3.10. Doing this I obtain:

$$\bar{Z} = e^{-\frac{\beta}{2}N} \sum_{s,J} p(J) \int \prod_i \frac{dh_i d\hat{h}_i}{2\pi i} \\ exp\left(\frac{\beta}{2} \sum_i sgn\left(D + h_i\right) - \sum_i h_i \hat{h}_i + \sum_i \frac{\hat{h}_i}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_i s_j\right) \tag{3.11}$$

Here, variables $\hat{h}_i$ are integrated between $-i\infty$ and $i\infty$, while variables h_i run between $-\infty$ and ∞.
Now, the variables J_{ij} are decoupled and they can be averaged out. In fact,

$$\sum_J p(J)\ \exp\left(\sum_i \sum_{j \neq i} \frac{\hat{h}_i}{\sqrt{N}} J_{ij} s_i s_j\right) = \prod_i \prod_{j \neq i} \cosh\left(\frac{\hat{h}_i}{\sqrt{N}} s_i s_j\right) = \prod_i \prod_{j \neq i} \cosh\left(\frac{\hat{h}_i}{\sqrt{N}}\right)$$

where in the last step I used that the hyperbolic cosine is even to get rid of the product $s_i s_j$, which can only contribute a sign to the argument and is thus irrelevant. Substituting back in 3.11, we get:

$$\bar{Z} = e^{-\frac{\beta}{2}N} \sum_s \int \prod_i \frac{dh_i d\hat{h}_i}{2\pi i} exp\left(\frac{\beta}{2}\sum_i sgn\left(D+h_i\right) - \sum_i h_i\hat{h}_i + \sum_i \sum_{j\neq i} \log\cosh\left(\frac{\hat{h}_i}{\sqrt{N}}\right)\right)$$

Importantly, because of the parity of cosh, also the spin variables have disappeared. The sum over their configurations can now be carried out trivially and contributes a factor 2^N. After rearranging, we are left with:

$$\bar{Z} = e^{-\frac{\beta}{2}N + N\log 2}\left[\int \frac{dhd\hat{h}}{2\pi i} exp\left(\frac{\beta}{2} sgn\left(D+h\right) - h\hat{h} + (N-1)\log\cosh\left(\frac{\hat{h}}{\sqrt{N}}\right)\right)\right]^N$$

If we can evaluate the integral, we are done.
In the large N limit, we can use the Taylor expansion $\log\cosh x \sim \frac{1}{2}x^2$, valid for $x << 1$, to rewrite the integral we need to compute as:

$$\int_{-\infty}^{\infty} dh \int_{-i\infty}^{i\infty} \frac{d\hat{h}}{2\pi i} \exp\left(\frac{\beta}{2} sgn\left(D+h\right) - h\hat{h} + \frac{1}{2}\hat{h}^2\right) := I \tag{3.12}$$

where I made explicit the integration limits. This rewriting becomes exact in the thermodynamic limit. With this simplification, the integral in $d\hat{h}$ is Gaussian and can be computed applying standard identities. Carrying out the computations, we get:

$$\begin{aligned}
I &= \int_{-\infty}^{\infty} dh \int_{-i\infty}^{i\infty} \frac{d\hat{h}}{2\pi i} \exp\left(\frac{\beta}{2} sgn\left(D+h\right) - h\hat{h} + \frac{1}{2}\hat{h}^2\right) \\
&= \int_{-\infty}^{\infty} \frac{dh}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}h^2 + \frac{\beta}{2} sgn(D+h)\right) \\
&= e^{-\frac{\beta}{2}} \int_{-\infty}^{-D} \frac{dh}{\sqrt{2\pi}}\ \exp\left(-\frac{1}{2}h^2\right) + e^{\frac{\beta}{2}} \int_{-D}^{\infty} \frac{dh}{\sqrt{2\pi}}\ \exp\left(-\frac{1}{2}h^2\right) \\
&= \frac{1}{2} e^{-\frac{\beta}{2}}\ erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2} e^{\frac{\beta}{2}} \left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)
\end{aligned}$$

Putting everything together, we conclude (in the large N limit):

$$\bar{Z} = e^{-\frac{\beta}{2}N + N\log 2} \exp\left\{N \log\left[\frac{1}{2} e^{-\frac{\beta}{2}}\ erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2} e^{\frac{\beta}{2}} \left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)\right]\right\} \tag{3.13}$$

I have succeded in computing the first moment of the partition function, in the thermodynamic limit. In the next section, I will compute its second moment. This will be more challenging, because the spin variables will not disappear as they did here, but still it will turn out to be computable with similar methods.

3.4 Second moment

I want to compute, in the thermodynamic limit, the second moment $\bar{Z}^2$ of the partition function. This requires averaging over the random coupling realizations the product of two copies of the partition function:

$$\bar{Z}^2 = \sum_J p(J) Z_J^2 = e^{-\beta N} \sum_{s',s'',J} p(J) exp\left(\frac{\beta}{2} \sum_{i,n} sgn\left(D + \frac{1}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_i^n s_j^n\right)\right) \quad (3.14)$$

Here, I denote with s_i' the i-th spin of the first copy of Z_J, and with s_i'' the i-th spin of the second copy of Z_J. The apex n in the summations always ranges over $'$ and $''$. Using again 3.10, introduce auxiliary integration variables h_i^n , where $i = 1, ..., N$ and $n \in \{', ''\}$, and enforce $h_i^n = \frac{1}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_i^n s_j^n$ using delta functions, represented as integrals over the variables $\hat{h}_i^n$. This allows to rewrite the summand as:

$$exp\left(\frac{\beta}{2} \sum_{i,n} sgn\left(D + \frac{1}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_i^n s_j^n\right)\right) =$$
$$\int \prod_{i,n} \frac{dh_i^n d\hat{h}_i^n}{2\pi i} \, exp\left(\frac{\beta}{2} \sum_{i,n} sgn\left(D + h_i^n\right) - \sum_{i,n} h_i^n \hat{h}_i^n + \sum_{i,n} \frac{\hat{h}_i^n}{\sqrt{N}} \sum_{j \neq i} J_{ij} s_i^n s_j^n\right)$$

where variables $\hat{h}_i^n$ are integrated between $-i\infty$ and $i\infty$, and variables h_i^n between $-\infty$ and ∞. The couplings J_{ij} are now decoupled and can be averaged out:

$$\sum_J p(J) \exp\left(\sum_i \sum_{j \neq i} \sum_n \frac{\hat{h}_i^n}{\sqrt{N}} J_{ij} s_i^n s_j^n\right) = \prod_i \prod_{j \neq i} \cosh\left(\sum_n \frac{\hat{h}_i^n}{\sqrt{N}} s_i^n s_j^n\right)$$

Substituting back in 3.14, we obtain:

$$\bar{Z}^2 = e^{-\beta N} \sum_{s',s''} \int \prod_{i,n} \frac{dh_i^n d\hat{h}_i^n}{2\pi i}$$
$$exp\left(\frac{\beta}{2} \sum_{i,n} sgn\left(D + h_i^n\right) - \sum_{i,n} h_i^n \hat{h}_i^n + \sum_{i \neq j} \log\cosh\left(\sum_n \frac{\hat{h}_i^n}{\sqrt{N}} s_i^n s_j^n\right)\right)$$

Importantly, the spin variables do not disappear this time, because the argument of *cosh* is a sum of two terms. We must find a way to average them out. Focus on:

$$\sum_{s',s''} exp\left(\sum_{i \neq j} log\ cosh\left(\sum_n \frac{\hat{h}_i^n}{\sqrt{N}} s_i^n s_j^n\right)\right) \quad (3.15)$$

Since we are interested in the limit $N \to \infty$, we can use the expansion $log\ cosh(x) \sim \frac{x^2}{2}$, valid for $x << 1$, and rewrite 3.15 as:

$$\sum_{s',s''} exp\left(\sum_{i\neq j}\left[\frac{(\hat{h}_i')^2}{2N} + \frac{(\hat{h}_i'')^2}{2N} + \frac{1}{N}\hat{h}_i'\hat{h}_i''s_i's_j's_i''s_j''\right]\right) \tag{3.16}$$

This becomes exact in the thermodynamic limit. Notice that we can rewrite:

$$\sum_{s',s''} exp\left(\sum_{i\neq j}\frac{1}{N}\hat{h}_i'\hat{h}_i''s_i's_j's_i''s_j''\right) = \sum_{s',s''} exp\left(\sum_{i,j}\frac{1}{N}\hat{h}_i'\hat{h}_i''s_i's_j's_i''s_j'' - \sum_i \frac{1}{N}\hat{h}_i'\hat{h}_i''\right) \tag{3.17}$$

Further restrict attention to the term:

$$\sum_{s',s''} exp\left(\sum_{i,j}\frac{1}{N}\hat{h}_i'\hat{h}_i''s_i's_j's_i''s_j''\right) \tag{3.18}$$

Once more, introduce an auxiliary integration variable k, and enforce $k = \sum_i \frac{1}{\sqrt{N}}\hat{h}_i'\hat{h}_i''s_i's_i''$ using a delta function, represented as an integral over a variable $\hat{k}$. 3.18 becomes:

$$\int \frac{dkd\hat{k}}{2\pi i}\sum_{s',s''} exp\left(\sum_j \frac{k}{\sqrt{N}}s_j's_j'' - k\hat{k} + \sum_i \frac{\hat{k}}{\sqrt{N}}\hat{h}_i'\hat{h}_i''s_i's_i''\right) \tag{3.19}$$

Now, the spins are decoupled and they can be averaged out. In fact:

$$\sum_{s',s''}\exp\left(\sum_i \frac{k}{\sqrt{N}}s_i's_i'' + \sum_i \frac{\hat{k}}{\sqrt{N}}\hat{h}_i'\hat{h}_i''s_i's_i''\right) = \sum_{s''}\prod_i 2\cosh\left(\frac{k}{\sqrt{N}}s_i'' + \frac{\hat{k}}{\sqrt{N}}\hat{h}_i'\hat{h}_i''s_i''\right) =$$
$$4^N\prod_i \cosh\left(\frac{k}{\sqrt{N}} + \frac{\hat{k}}{\sqrt{N}}\hat{h}_i'\hat{h}_i''\right) \tag{3.20}$$

Putting everything together, and using that $\frac{N-1}{N} \approx 1$ in the limit $N \to \infty$, 3.15 becomes:

$$e^{Nlog4}\int \frac{dkd\hat{k}}{2\pi i} exp\left(\sum_i log\ cosh\left(\frac{k}{\sqrt{N}} + \frac{\hat{k}}{\sqrt{N}}\hat{h}_i'\hat{h}_i''\right) - k\hat{k} - \sum_i \frac{1}{N}\hat{h}_i'\hat{h}_i'' + \frac{1}{2}\sum_i\left[(\hat{h}_i')^2 + (\hat{h}_i'')^2\right]\right) \tag{3.21}$$

We have managed to average out spins and couplings. Now we need to evaluate the integral. To begin, expand $log\ cosh(x) \sim \frac{x^2}{2}$, which is valid in the limit $N \to \infty$. We get:

$$\sum_i log\ cosh\left(\frac{k}{\sqrt{N}} + \frac{\hat{k}}{\sqrt{N}}\hat{h}_i'\hat{h}_i''\right) \sim \frac{k^2}{2} + \frac{1}{N}k\hat{k}\sum_i \hat{h}_i'\hat{h}_i'' + \frac{(\hat{k})^2}{2N}\sum_i(\hat{h}_i'\hat{h}_i'')^2 \tag{3.22}$$

With this expansion, the integral in $d\hat{k}$ is Gaussian and can be evaluated as:

$$\int_{-i\infty}^{i\infty} \frac{d\hat{k}}{2\pi i} exp\left(\frac{1}{2}a\hat{k}^2 + bk\hat{k}\right) = \sqrt{\frac{1}{2\pi a}} exp\left(-\frac{b^2k^2}{2a}\right) \tag{3.23}$$

with the shorthand $a = \frac{1}{N}\sum_i (\hat{h}_i'\hat{h}_i'')^2$ and $b = \frac{1}{N}\sum_i \hat{h}_i'\hat{h}_i'' - 1$. Now integrate in dk to get:

$$\int_{-\infty}^{\infty} dk \sqrt{\frac{1}{2\pi a}} exp\left(-\frac{1}{2}\left[\frac{b^2}{a} - 2\right]k^2\right) = \sqrt{\frac{1}{b^2 - 2a}} \tag{3.24}$$

We are left with:

$$\bar{Z}^2 = e^{-\beta N + log4N} \int \prod_{i,n} \frac{dh_i^n d\hat{h}_i^n}{2\pi i}$$
$$exp\left(\sum_{i,n}\left[\frac{\beta}{2}sgn\left(D + h_i^n\right) - h_i^n\hat{h}_i^n + (\hat{h}_i^n)^2\right] - \sum_i \frac{2}{N}\hat{h}_i'\hat{h}_i'' - \frac{1}{2}log(b^2 - 2a)\right)$$

Write explicitly $b^2 - 2a$:

$$b^2 - 2a = 1 - \frac{2}{N}\sum_i \hat{h}_i'\hat{h}_i'' - \frac{2}{N}\sum_i (\hat{h}_i'\hat{h}_i'')^2 + \frac{1}{N^2}\left(\sum_i \hat{h}_i'\hat{h}_i''\right)^2 \tag{3.25}$$

In the limit $N \to \infty$, we see that the terms $-\sum_i \frac{1}{N}\hat{h}_i'\hat{h}_i'' - \frac{1}{2}log(b^2 - 2a)$ become negligible with respect to $\sum_{i,n}\left[\frac{\beta}{2}sgn\left(D + h_i^n\right) - h_i^n\hat{h}_i^n + \frac{1}{2}(\hat{h}_i^n)^2\right]$, and can therefore be ignored. To conclude, we need to evaluate:

$$\bar{Z}^2 = e^{-\beta N + log4N} \int \prod_{i,n} \frac{dh_i^n d\hat{h}_i^n}{2\pi i} exp\left(\sum_{i,n}\left[\frac{\beta}{2}sgn\left(D + h_i^n\right) - h_i^n\hat{h}_i^n + \frac{1}{2}(\hat{h}_i^n)^2\right]\right) =$$
$$e^{-\beta N + log4N}\left[\int_{-\infty}^{\infty}\int_{-i\infty}^{i\infty} \frac{dhd\hat{h}}{2\pi i} exp\left(\frac{\beta}{2}sgn\left(D + h\right) - h\hat{h} + \frac{1}{2}(\hat{h})^2\right)\right]^{2N} =$$
$$e^{-\beta N + log4N}\left[\int_{-\infty}^{\infty} \frac{dh}{\sqrt{2\pi}} exp\left(\frac{\beta}{2}sgn\left(D + h\right) - \frac{1}{2}h^2\right)\right]^{2N}$$

Where I integrated in $d\hat{h}$ at the last equality. But

$$\int_{-\infty}^{\infty} \frac{dh}{\sqrt{2\pi}}\, exp\left(\frac{\beta}{2}sgn\left(D + h\right) - \frac{1}{2}h^2\right) = e^{-\frac{\beta}{2}} \int_{-\infty}^{-D} \frac{dh}{\sqrt{2\pi}}\, exp\left(-\frac{1}{2}h^2\right) +$$
$$e^{\frac{\beta}{2}} \int_{-D}^{\infty} \frac{dh}{\sqrt{2\pi}}\, exp\left(-\frac{1}{2}h^2\right) = \frac{1}{2}e^{-\frac{\beta}{2}}\, erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2}e^{\frac{\beta}{2}}\left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)$$

So we obtain the final result, valid in the limit $N \to \infty$:

$$\bar{Z}^2 = e^{-\beta N + log 4N} exp\left(2N\ log\left[\frac{1}{2}e^{-\frac{\beta}{2}}\ erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2}e^{\frac{\beta}{2}}\ \left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)\right]\right) \quad (3.26)$$

3.5 The zero-temperature limit

Comparing the results of 3.13 and 3.26, we see that, in the thermodynamic limit, the second moment of the partition function becomes equal to the square of the first moment, and hence 3.8 holds. Because of this, to deduce the properties of the typical network in the thermodynamic limit we can use the so-called annealed free energy density:

$$\begin{aligned} f &= -\frac{1}{\beta N}\log \bar{Z} \\ &= \frac{1}{2} - \frac{\log 2}{\beta} - \frac{1}{\beta}\log\left[\frac{1}{2}e^{-\frac{\beta}{2}}\ erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2}e^{\frac{\beta}{2}}\left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)\right] \end{aligned} \quad (3.27)$$

in fact, because of 3.8, the annealed free energy density becomes equal to the true one in the thermodynamic limit.
In the thermodynamic limit, macroscopic quantities such as the energy and the average entropy have thermal fluctuations of order $\sqrt{N}$ and magnitude of order N, hence they become extremely concentrated around their mean. Furthermore, in the limit $\beta \to \infty$, the entropy becomes the logarithm of the number of absolute minima of the energy function, i.e. in this case the logarithm of the number of fixed points (see for example [6]).
Then, we can use well known identities involving the derivatives of the free energy density to compute the average energy and entropy (both normalized by N), and we can take the limit $\beta \to \infty$ to obtain the number of fixed points of the typical network.
First, let me verify that the average energy density becomes 0 in the zero-temperature limit, signaling that the Boltzmann measure concentrates on fixed points (i.e. ground states of our energy), as it should:

$$E = \frac{\partial}{\partial\beta}(\beta f) = \frac{1}{2} - \frac{-\frac{1}{4}e^{-\frac{\beta}{2}}erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{4}e^{\frac{\beta}{2}}\left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)}{\frac{1}{2}e^{-\frac{\beta}{2}}\ erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2}e^{\frac{\beta}{2}}\left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)} \quad (3.28)$$

and the expression converges to 0 as $\beta \to \infty$.
Now, I will compute the average entropy of the system:

$$\begin{aligned} S = \beta^2\frac{\partial}{\partial\beta}f = \log 2 + \log\left[\frac{1}{2}e^{-\frac{\beta}{2}}\ erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2}e^{\frac{\beta}{2}}\left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)\right] + \\ - \beta\frac{-\frac{1}{4}e^{-\frac{\beta}{2}}erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{4}e^{\frac{\beta}{2}}\left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)}{\frac{1}{2}e^{-\frac{\beta}{2}}\ erfc\left(\frac{D}{\sqrt{2}}\right) + \frac{1}{2}e^{\frac{\beta}{2}}\left(1 + erf\left(\frac{D}{\sqrt{2}}\right)\right)} \end{aligned} \quad (3.29)$$

in the limit $\beta \to \infty$, we have:

$$S \to \log\left[1 + erf\left(\frac{D}{\sqrt{2}}\right)\right] \tag{3.30}$$

Spelling everything out, this limit is the logarithm of the number of fixed points of the typical network, normalized by the size of the system N, and it is exactly what I set out to compute. Notice that this result is exact, in the thermodynamic limit, thanks to 3.8. We can visualize the dependence of the number of fixed points on the intensity of the self-coupling D:

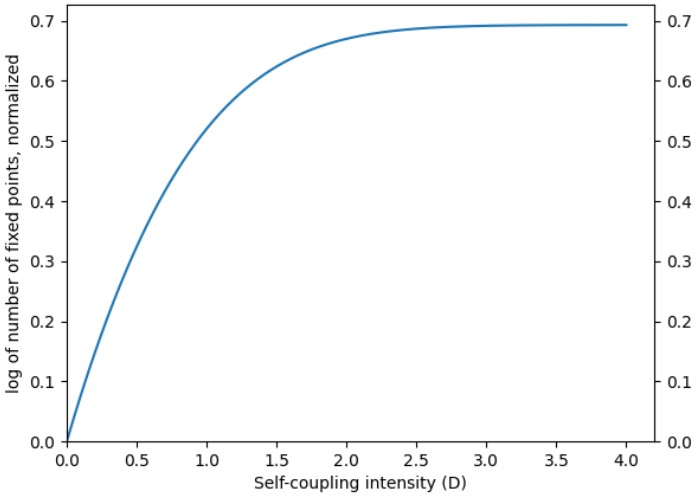


Figure 3.1: Behaviour of the normalized logarithm of the number of fixed points of the random asymmetric Hopfield model as the intensity of the self-couplings increases.

We see that, as D increases, an exponential number of fixed points appears, with the growth saturating around $D = 2$. Importantly, to put this graph in perspective, notice that because of the factor $\frac{1}{\sqrt{N}}$ in 3.5 typical values of a coupling J_{ij} will be of order $O(\frac{1}{\sqrt{N}})$; on the contrary, here D is varying on a scale that is $O(1)$.
These results show that, when the self-coupling is strong enough, an exponential (in the size of the system) number of fixed points appear in the typical network. However, these results alone tell nothing about the stability of these fixed points, nor about their geometry. These questions are natural next steps to tackle in future work.

Chapter 4

Autoencoders in the Random Feature model

In this chapter, I will present the results of numerical experiments with neural networks that I carried out in the unsupervised setting, using data generated from a random feature model. The research questions that this empirical study aims to investigate are mainly two. First, can autoencoders be used to estimate the latent dimensionality of their input data through training with different bottleneck sizes? Second, in this unsupervised setting, does training with entropic optimization algorithms lead to solutions lying in wide and flat regions of the energy landscape, and what are the properties of these solutions in terms of generalization?
In the next section, I will carefully describe the generating model for the data, the networks that I trained, the optimization algorithms that I used and the metrics that I considered. Then, in subsequent sections, I will describe the experiments that I performed and the main results that I obtained.

4.1 Model and methods

4.1.1 Data generating model

Real data is widely believed to lie on low-dimensional manifolds in feature space. For instance, in computer vision, drawing pixels independently at random from some distribution will produce images that look nothing like natural images, and would most likely appropriately be called 'noise'. This is because those images which we recognize as natural images have strong correlations among their pixels, and can be thought to lie on a twisted and highly nonlinear submanifold of the feature space. In general, studying analytically the learning problem with data that exhibits strong internal correlations of this kind is not easy. Progress in this direction has been made in [43], where the authors have studied the dynamics of online supervised learning when the data exhibits this kind of structure. In order to capture this property of real data, and also to have full control on the proper-

ties and structure of the dataset, I have used a generating model similar to that studied in [43], which is often called random feature model. What it does is to generate patterns that differ from one another through the presence or absence of certain features. More concretely, a D x N feature matrix F collects D N-dimensional features as its rows. To generate a pattern, a random D-dimensional vector of zeros and ones is generated and its transpose (i.e., a row vector) is left multiplied with F. The result is an N-dimensional vector which is a superposition of a random subset of the features encoded by F. This vector is finally passed through a component-wise nonlinearity to obtain a random pattern ready for training. Often, before applying the nonlinearity, the vector is normalized by the square root of D to ensure that entries have magnitude of order 1.
This model allows to generate an arbitrary number P of patterns which live in a N-dimensional feature space, but actually lie on a D-dimensional manifold that is characterized by the choice of F and of the nonlinearity. Increasing D while leaving the rest untouched makes the data more complex and less redundant, reducing the degree of internal correlation of patterns and increasing the dimensionality of the latent manifold.

4.1.2 Networks and training algorithms

In the existing literature, entropic algorithms have mainly been studied in the supervised setting. In my empirical work, I therefore decided to study neural networks trained with an unsupervised principle. The simplest type of such networks is called autoencoder [44], and it is commonly employed for dimensionality reduction.
An autoencoder is a feed-forward architecture composed of an encoder and a decoder. The encoder takes an N-dimensional input and compresses it into a B-dimensional representation (B stands for 'bottleneck': usually, $B < N$). Then, the decoder takes this representation as input and outputs back an N-dimensional vector. The implementation of encoder and decoder can be done in many different ways, depending on the problem at hand. In my experiments, I used a simple multi-layer perceptron architecture for both.
The autoencoder is trained to minimize the 'reconstruction error' (i.e. some notion of distance) between its inputs and outputs. In a probabilistic framework, this can be interpreted as maximizing a lower bound to the mutual information between the inputs and their hidden representations in the bottleneck. In this view, the outputs of the autoencoder are interpreted as the parameters of a probability distribution on the input space, and the specific form of this distribution is linked to the choice of the reconstruction error to minimize [45]. This means that, when $B < N$, the network is forced to learn properties of the internal structure of the inputs that allow to compress them while retaining as much information as possible.
As for any other multi-layer perceptron, architectural choices include the number and width of hidden layers, the activation functions, weight initialization, and more. I will detail these design choices and the reasons for them when describing my experiments.
As an optimizer, I used an adaptive gradient-based method called Adam [46]. Adam is a variant of Stochastic Gradient Descent that adapts the learning rate independently for

each optimization parameter during training. This is achieved by maintaining exponential moving averages of the first and second moments of the partial derivatives with respect to each weight. The second moment provides information on the typical magnitude of the gradients, while the first moment is sensitive to oscillations in weight space thanks to cancellation. These estimates are used to adjust the learning rate of each parameter in such a way that the step size is larger along flatter directions in the loss landscape. In practice, Adam is usually faster than SGD, even using momentum, although the training is arguably more noisy in the last stages.
In some experiments, I used the replicated version of Adam, as described in the Robust Ensemble section of the second chapter: y replicas of the autoencoder interact with each other and with their baricenter to approximately explore the local entropy landscape associated to the loss. For the individual updates of each of the replicas, Adam is used. A scoping procedure on the interaction parameter γ is employed to allow the replicas to explore the loss landscape on progressively finer scales; following [22], I used an exponential updating procedure for γ, which consists in starting with a small value γ_0 at the beginning of training and then, at the end of each epoch, updating γ through multiplication by a fixed factor $1 + \gamma_1$. The resulting optimization scheme is quite sensitive to the choice of the scoping schedule, characterized by γ_0 and γ_1, which must therefore be carefully tuned.

4.1.3 Metrics

During the training of the networks, I kept track of a few informative metrics about the state of the model and of training. First, I monitored the train loss, as well as the test loss on a held-out dataset generated from the same random feature model as the training set. I also saved, at regular intervals during training, histograms of weight magnitude and of gradient magnitude for every layer separately. This was important to monitor the behaviour of the optimizer and detect potential issues such as vanishing or exploding gradients, as well as to better inform the choice of training hyperparameters. Furthermore, in addition to these standard metrics, I have also estimated, at regular intervals during training, the local energy profile as well as the ability of the model to denoise corrupted inputs. I will now explain exactly how I did this.
The local energy is a computationally cheap measure of flatness that was shown semi-analytically and numerically in [22] to correlate strongly with local entropy. Furthermore, extensive numerical experiments with deep architectures in [25] have shown that local energy is among the best predictors of generalization performance. The way it's computed is fairly simple. Starting from the initial network configuration, perturb each weight w_i according to $w_i \to (1 + \sigma_i)w_i$, where σ_i is a random number drawn iid from an unbiased Gaussian whose standard deviation characterizes the intensity of the injected noise. Then, using the perturbed network, compute the reconstruction error on the training set.
By repeating this procedure many times independently and averaging the results, we obtain a good aggregate measure of the loss at a given distance from the initial configura-

tion in weight space. Then, if we run this estimate for a range of noise intensities, we can cheaply explore the loss landscape at increasing distance from the reference configuration, obtainin a 'local energy profile' that traces the behaviour of the loss as the distance increases.
Notice that we do not employ additive noise, but rather we multiply each weight by a random number close to 1. This is important for at least two reasons. Firts, the magnitude of the weights may vary greatly throughout the network, and a change in one of the weights could have very different consequences compared to a change of the same magnitude in another weight. Employing multiplicative noise makes the procedure aware of this. Second, the effects of multiplicative noise are not changed by a certain type of reparametrizations that, with positive homogeneous activations (like the relus that I used), leave unaffected the behaviour of the network (multiply incoming weights to a neuron by a positive constant, and divide outgoing weights by the same constant).
To estimate the denoising capabilities of a network, instead, I perturb the patterns of the training set using additive Gaussian noise with zero mean, I feed the noisy patterns to the autoencoder and I compute the reconstruction error between the outputs and the original inputs. I average the results over many independent trials, and I repeat the procedure for different standard deviations of the iid Gaussian variables (i.e., different noise intensities). By adding random noise, the patterns will most likely move away from the data manifold on which they lie. This metric is then meant to quantify the ability of the network to filter out the noise and recover the original pattern, even though it was not explicitly trained to do so: it is measuring some form of generalization performance. Results of [23] suggest that, at least in the supervised setting, there is a positive correlation between the tolerance of a network to perturbation of its inputs and of its weights, meaning that solutions lying in wide and flat regions of the loss landscape tend to handle noisy inputs better. It will be interesting to see whether autoencoders lying in flat minima are better at denoising corrupted patterns, and how this relates to the reconstruction error on unseen patterns.

4.2 Experiments and Results

In this section, I will describe the numerical experiments that I carried out and illustrate their main results, both in words and showing some of the most significant plots.

4.2.1 Latent dimension estimation

As I mentioned in the introduction to this chapter, with the first set of experiments I wanted to investigate the following research question: is it possible to obtain a good estimate of the latent dimensionality of data by observing the reconstruction error achieved by autoencoders with increasing bottleneck width?
To answer this question, I generated a training set from a random feature model with parameters D, P, N as described in the methods section, and I trained a number of

autoencoders with the same architecture and training hyperparameters, with the only difference being the width of their bottleneck. What I expected to see is that the autoencoders achieve a high loss when the bottleneck width B is much smaller than the latent dimensionality D, and a small loss when B is much larger than D, with some transition between the two regimes when B is comparable to D. This reflects the intuitive fact that one is bound to lose some information if asked to compress some genuinely D-dimensional patterns using less than D independent numbers, while it should in principle be possible to compress the patterns using more than D independent numbers, if the effectiveness of the training and the expressiveness of the network are good enough (it surely should be possible using more than N numbers). Now, the question is what kind of transition should we expect between these two regime? Is it going to be a smooth, gradual decrease in loss, or will it be a more abrupt and localized change? If the second scenario was the case, and if the change was somewhere in the proximity of D, we could derive an estimate of the latent dimensionality of the data by inspecting a plot of the reconstruction error vs bottleneck width, and localizing the elbow.
Of course, It is possible that the qualitative behaviour of the autoencoders depends on the choice of the parameters D, P, N characterizing the structure of the data (and in particular, on the ratios D/N and P/N), as well as on the network architecture. In order to explore this possibility, I investigated a variety of significant regimes characterized by different choices for these hyperparameters.
The experiment can then be described as follows. I generate P N-dimensional random patterns from a random feature model with latent dimension D. I fix an autoencoder architecture, and I choose a single set of training hyperparameters that allow networks with this architecture and varying bottleneck widths to learn well. This is important to make the comparison of the reconstruction errors meaningful. In practice, I chose a conservative set of hyperparameters that worked well in all cases, at the cost of training speed (e.g., I chose a learning rate small enough to effectively navigate even the rougher landscapes, at the cost of slowing down training in the smoother landscapes where a larger learning rate could have been sustained). Importantly, the use of an adaptive method like Adam enormously mitigated this problem, which is the main reason why I adopted it instead of for example momentum SGD.
Once the hyperparameters are set, I select a range of bottleneck widths and for each of them I train an autoencoder until convergence of the loss, averaging the relevant metrics over a number of independent trials as is customary. Then, I can plot the reconstruction error on the training set against the bottleneck width, and inspect the results.
In all experiments, I used as nonlinearity in the random feature model a sigmoidal function with coefficient 5. This was chosen in such a way that the sigmoid operates in its nonlinear regime, but without making the pattern components binary. The degree of nonlinearity was estimated by considering the effectiveness of Principal Component Analysis at dimensionality reduction when the sigmoid parameter changes, while to avoid making the data binary direct inspection of the data histogram was sufficient. Since the data is between 0 and 1, I used a sigmoid activation in the last layer of the decoder. In

non-bottleneck hidden layers, when present, I used a relu activation function and I used a number of hidden units equal to N (the qualitative picture was not found to be influenced by the choice of the hidden layer width, when not too small).
I performed extensive numerical experiments using two different values of N, four different values of D, four different values of P and two different network architectures (in one case, encoder and decoder do not have any hidden layers, meaning that the full network has a single hidden layer. In the other case, both encoder and decoder have a single hidden layer, hence the full network has 3). This allowed me to check the robustness of my observations, as well as to investigate the effect of varying data and architectural hyperparameters on the behaviour of the networks. Then, for each individual case, I plotted the reconstruction error on the training set versus the bottleneck width. In figure 4.1 I show the experimental curves for a variety of significant regimes. As expected, the reconstruction error is high when the bottleneck is too small. Then, it decreases very rapidly in the region around $B = D$, with the curves showing an evident 'elbow' that allows to identify with good accuracy the value of the latent dimensionality. After the elbow, two behaviours are possible, depending on the depth of the network. In the case of networks with hidden layers, the loss stabilizes around the value it has at the elbow, meaning that increasing the bottleneck width further does not significantly improve the ability of the autoencoder to fit the training data, once $B \geq D$. In the case of shallow networks, instead, I generally observed that after the elbow, despite a significant slowdown in the decrease of the reconstruction error, which allows to identify the latent dimensionality, the loss does not saturate, but rather keeps slowly improving with the bottleneck width. In this sense, deeper networks tend to show a greater sensibility than shallower networks to the threshold $B = D$.
Data hyperparameters, on the other hand, did not show to have any influence on the qualitative shape of the curves; rather, they only determined its quantitative features. More specifically, increasing the ratio D/N, which has the effect of decreasing the level of internal correlation of the patterns, makes the learning task more difficult and results in a shift of the curve towards worse values of the loss. Similarly, increasing the ratio P/N, which can be thought to increase the number of 'constraints' of the learning problem, also results in a similar shift. These phenomena are clearly visible in Figure 4.1, which shows a comparison of the results with different data hyperparameters, holding the other parameters fixed.

4.2.2 Effects of replication

The second research question which I set out to investigate is the following: does training with entropic optimization algorithms lead to solutions lying in wide and flat regions of the energy landscape, and what are the properties of these solutions in terms of generalization properties?
To investigate this, I carried out two experiments in which I compared the performance

of the Adam optimization algorithm against that of Replicated Adam, a description of which can be found in the first section of this chapter. I trained the networks for a fixed number of epochs, chosen to be long enough that there is time for the loss to converge. To estimate the flatness around a minimum I computed the local energy profile according to the procedure described in the methods section, varying the noise intensity in a range and performing 20 independent perturbations of the network weights every time. As for the generalization properties, I considered two metrics. First, the reconstruction error on a held-out test set that was generated from the same random feature distribution as the training set. This represents the ability of the network to effectively compress unseen data. Second, I considered the ability of the network to denoise corrupted patterns from the training data, according to the procedure described in the methods section. Also in this case, I used a range of noise intensities, performing 20 independent perturbations of the training patterns every time. This can be considered as some sort of generalization metric because the autoencoders where not explicitly trained to denoise corrupted patterns, and one would expect a good autoencoder to partition the input space in such a way that errors in the patterns are, to some extent, corrected when compressed through the bottleneck.

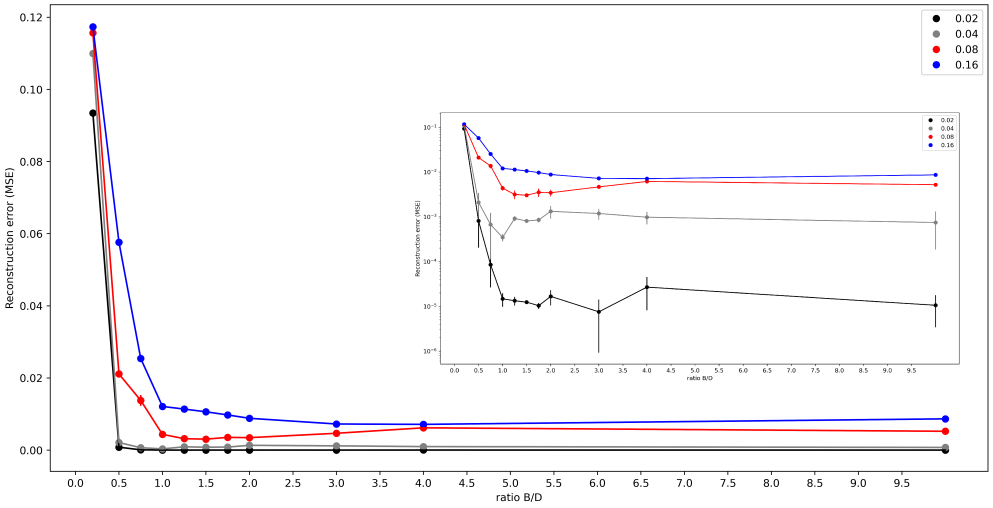
In the first experiment, I fixed the data hyperparameters and I considered two different network depths (same as last experiment) and a range of bottleneck widths. Since without any nonlinearity in the bottleneck the loss landscape of the shallow networks risks being too simple to display interesting phenomena, for these experiments I also included relu activations in the bottleneck. The training hyperparameters of Adam are the same in the replicated and non replicated case. In the replicated case, I adopted an exponential scoping procedure on γ, as described in the methods section, and I used $y = 5$ replicas. The other design choices are the same as last experiment.

In all cases, a comparison of the local energy profiles reveals that replicated Adam finds solutions lying in flatter regions of the loss landscape compared to Adam, both for shallow and deep autoencoders. This can be seen in the top portion of Figure 4.2, which shows a significant difference in reconstruction error when weights are perturbed between the replicated and non replicated case. As the bottleneck width increases, I observed that the local energy profile of all networks, including the non replicated ones, tends to become flatter, signalling that the landscape is becoming smoother and more flat directions are appearing.

When it comes to denoising capababilities, networks trained with replicated Adam are better at filtering noise on the training set compared to vanilla Adam. This can be seen in the bottom portion of Figure 4.2. The difference is particularly pronounced for deep networks, while for shallow networks it only becomes significant with a big enough bottleneck.

Finally, deep networks trained with replicated Adam achieved a better generalization error compared to vanilla Adam. This can be seen in Figure 4.3. Like with denoising capabilities, when training shallow networks replicated Adam did not improve significantly over Adam in terms of generalization error, with small differences appearing only with

large bottleneck widths.
These results show, also in this unsupervised setting, a positive correlation between flatness of the minima (here measured by the local entropy) and generalization properties (here measured by test error and denoising error). Furthermore, they show that with complex enough networks (deep autoencoders or shallow autoencoders with large enough bottleneck, i.e., with sufficiently many parameters) optimizing the local entropy landscape through interacting replicas allows to find solutions lying in flatter regions of the loss landscape, which enjoy better generalization properties.
In the second experiment, I tried to investigate the dependence of the generalization error on the number of training patterns, and also see how this affects the difference in performance between replicated and non replicated Adam. To this end, I used a similar experimental design as I described for the previous experiment, but this time I fixed the bottleneck width and I let the number of patterns vary in a range. As expected, I found that the generalization error improves when more patterns are used for training, contrary to the training error which becomes worse due to the need to fit a larger number of 'constraints'. Furthermore, the difference in generalization error between Adam and replicated Adam also increased with the number of training patterns, as the plot in Figure 4.4 shows. Coherently with the results of the previous experiment, the curves in the shallow case where found to be almost indistinguishable, with a small gap appearing for sufficiently many training patterns (i.e., the regime of the previous experiment).



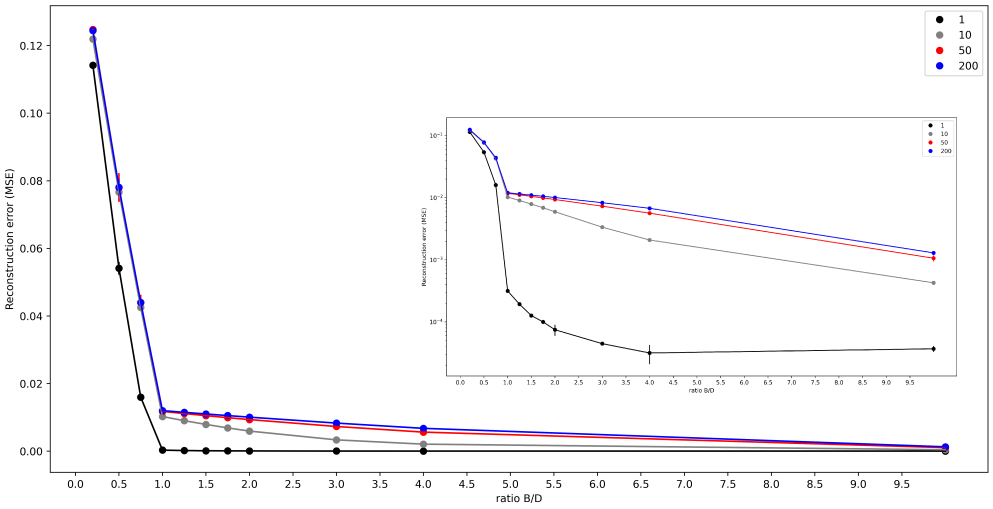


Figure 4.1: Reconstruction error vs bottleneck width curves; error bars show the standard deviation of the loss over 5 independent trials. The smaller window shows the same graph, but with the y axis in log-scale. N is fixed to 800 in these experiments. (top) Autoencoders with hidden layers, and P/N fixed to 50. Different curves correspond to different ratios D/N. (bottom) Autoencoders without hidden layers. This time, D/N is fixed to 0.08 and different curves correspond to different ratios P/N.

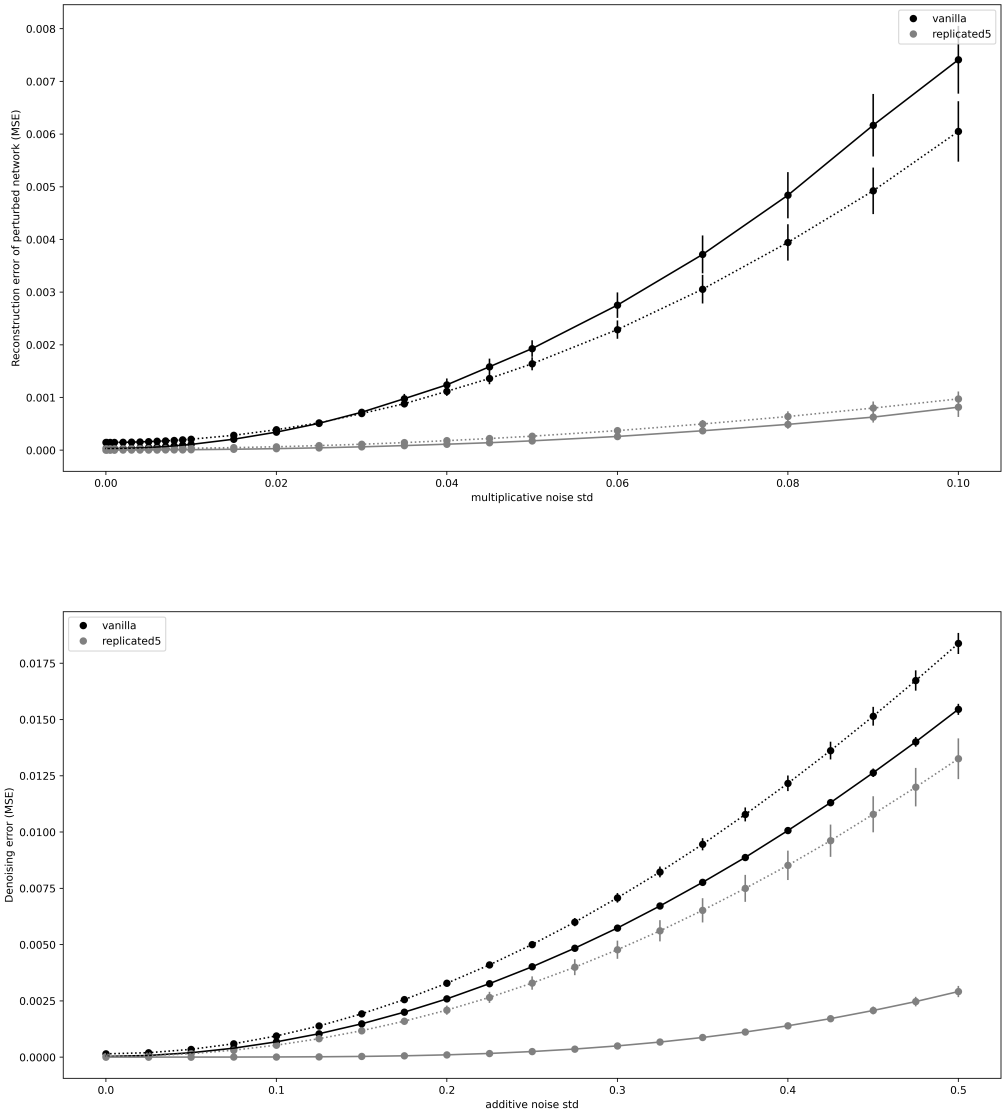


Figure 4.2: Comparison of Adam and replicated Adam with D = 16, P = 8000, B = 160 with deep (continuous) and shallow (dotted) autoencoders. Results are averaged over 5 independent trials. (top) local energy profile; that of replicated Adam is significantly flatter in both cases. (bottom) denoising error for varying noise intensities; there is a large difference with deep autoencoders, less so with shallow networks.

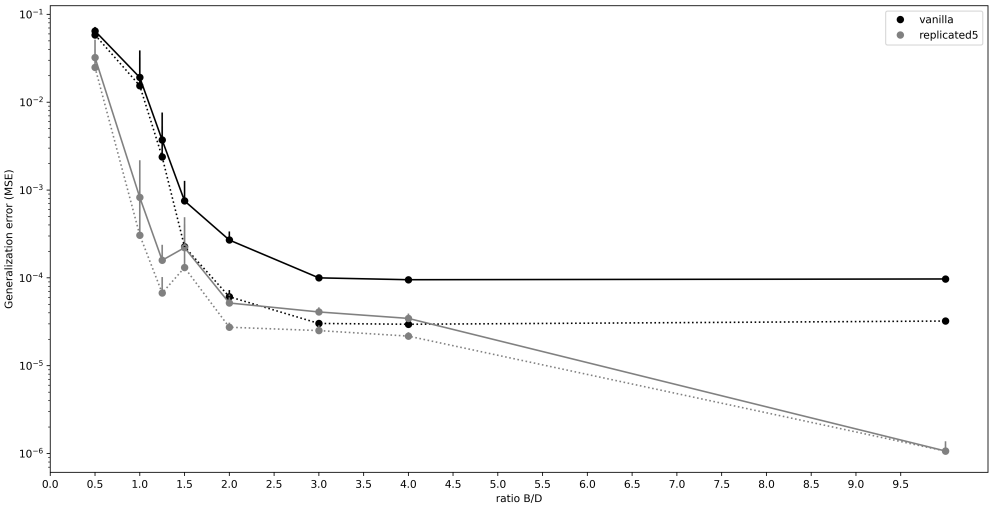


Figure 4.3: Comparison of test error (continuous) and training error (dotted) of Adam and replicated Adam across a range of bottleneck widths. D = 16, P = 8000, deep autoencoders, 5 independent trials.

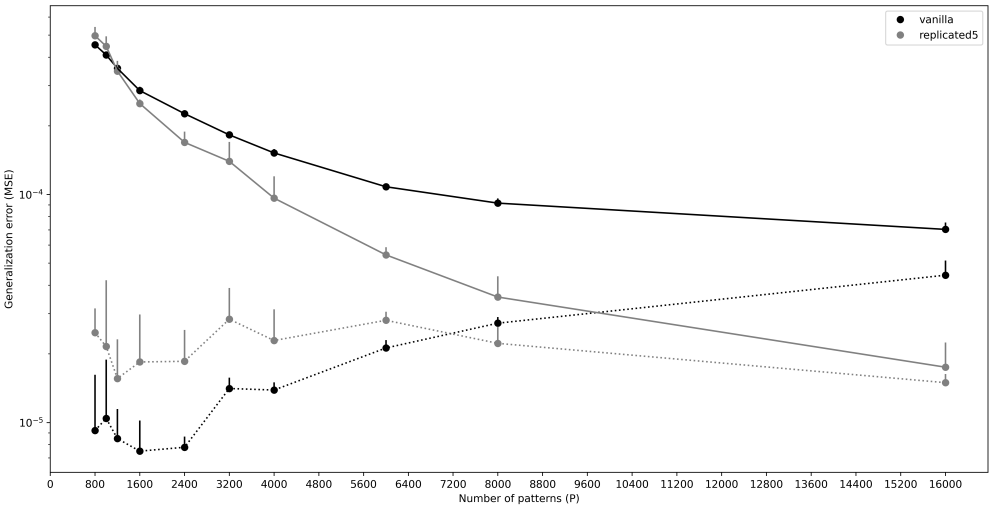


Figure 4.4: Comparison of test error (continuous) and training error (dotted) of Adam and replicated Adam as the number of training patterns increases. D = 16, B = 64, deep autoencoders, 5 independent trials.

Conclusions and Future Work

After a review of some ideas and results from the existing literature, concerning wide flat minima and entropic algorithms in artificial neural networks, I went on to present some analytical and numerical results of my own.
First, I used the tools of statistical physics to study a Hopfield-type random recurrent network of neurons with asymmetric couplings. I computed exactly, in the limit of a large system, the number of fixed points of such a network and found its dependency on the self-coupling strength. When the self-couplings are strong enough, an exponential number of fixed points appear. A natural follow-up question would be to study the stability of such abundant fixed points, to understand the dynamics of the network and whether it could be useful as a computational device. Furthermore, it could be interesting to study the geometry of the fixed points, and in particular their connectivity structure, which could be important to determine their accessibility by algorithms. These questions require extensive further analysis, and they are currently under active investigation by the researchers at the Bocconi Artificial Intelligence Lab.
Second, I carried out numerical experiments with shallow and deep autoencoders in the random feature model. I found that both are able to estimate the latent dimensionality of data with good accuracy through training with varying bottleneck and visual inspection of the loss vs bottleneck curve, which reports a visible elbow close to the true dimensionality. Furthermore, while shallow networks keep benefitting from increasing the bottleneck well after the elbow, the loss of deep networks tends to saturate earlier.
Finally, I studied the performance of an entropic optimization algorithm, replicated Adam, in the same unsupervised setting, and I compared it with vanilla Adam. I found that replicated Adam is consistently able to find solutions lying in flatter regions of the loss ladnscape, as measured by the local energy. Furthermore, I found that with complex enough networks (deep autoencoders, or shallow autoencoders with sufficiently large bottleneck) the solutions found by replicated Adam enjoy better generalization properties, in terms of reconstruction error on a held-out test set as well as ability to denoise corrupted patterns. These numerical results suggest that the conclusions about flat minima and their role in learning and generalization should extend beyond supervised learning. This is especially relevant considering the tremendous importance of self-supervised learning techniques in many of the most recent breakthroughs in fields like NLP and computer vision. Using the empirical evidence as a starting point, it could be interesting to try

and use the statistical physics formalism to study learning in this unsupervised setting, and the effects of replication on generalization. However, a big challenge in this direction is that the effects of replication were most evident either using deep networks, or using shallow autoencoders with many hidden units in the bottleneck, and none of these two regimes are easy to study analytically. Further empirical investigation would be necessary to find a suitable model that is both analytically tractable and keeps displaying the phenomena of interest (for instance, freezing some of the degrees of freedom could be an option to consider).

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". en. In: *Nature* 521.7553 (2015), pp. 436–444.

[2] OpenAI. "GPT-4 Technical Report". In: (2023). eprint: 2303.08774.

[3] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022, pp. 10674–10685.

[4] Marc Mezard and Andrea Montanari. *Information, Physics, and Computation*. en. London, England: Oxford University Press, 2009.

[5] M Mezard, G Parisi, and M Virasoro. *Spin glass theory and beyond: An introduction to the replica method and its applications*. WORLD SCIENTIFIC, 1986.

[6] Olivier C Martin, Rémi Monasson, and Riccardo Zecchina. "Statistical mechanics methods and phase transitions in optimization problems". en. In: *Theor. Comput. Sci.* 265.1-2 (2001), pp. 3–67.

[7] M Mézard, G Parisi, and R Zecchina. "Analytic and algorithmic solution of random satisfiability problems". en. In: *Science* 297.5582 (2002), pp. 812–815.

[8] Florent Krzakała et al. "Gibbs states and the set of solutions of random constraint satisfaction problems". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 104.25 (2007), pp. 10318–10323.

[9] Carlo Baldassi et al. "Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses". en. In: *Phys. Rev. Lett.* 115.12 (2015), p. 128101.

[10] Werner Krauth and Marc Mézard. "Storage capacity of memory networks with binary couplings". en. In: *Journal Phys.* 50.20 (1989), pp. 3057–3066.

[11] H Sompolinsky, N Tishby, and H S Seung. "Learning from examples in large neural networks". en. In: *Phys. Rev. Lett.* 65.13 (1990), pp. 1683–1686.

[12] Haiping Huang and Yoshiyuki Kabashima. "Origin of the computational hardness for learning with binary synapses". en. In: *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 90.5-1 (2014), p. 052813.

[13] Tomoyuki Obuchi and Yoshiyuki Kabashima. "Weight space structure and analysis using a finite replica number in the Ising perceptron". In: *J. Stat. Mech.* 2009.12 (2009), P12014.

[14] Heinz Horner. "Dynamics of learning for the binary perceptron problem". en. In: *Z. Physik B - Condensed Matter* 86.2 (1992), pp. 291–308.

[15] Haiping Huang, K Y Michael Wong, and Yoshiyuki Kabashima. "Entropy landscape of solutions in the binary perceptron problem". In: *J. Phys. A Math. Theor.* 46.37 (2013), p. 375002.

[16] Alfredo Braunstein and Riccardo Zecchina. "Learning by message passing in networks of discrete synapses". en. In: *Phys. Rev. Lett.* 96.3 (2006), p. 030201.

[17] Carlo Baldassi and Alfredo Braunstein. "A Max-Sum algorithm for training discrete neural networks". In: *J. Stat. Mech.* 2015.8 (2015), P08008.

[18] Carlo Baldassi et al. "Efficient supervised learning in networks with binary synapses". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 104.26 (2007), pp. 11079–11084.

[19] Carlo Baldassi. "Generalization learning in a perceptron with binary synapses". en. In: *J. Stat. Phys.* 136.5 (2009), pp. 902–916.

[20] Carlo Baldassi et al. "Local entropy as a measure for sampling solutions in constraint satisfaction problems". In: *J. Stat. Mech.* 2016.2 (2016), p. 023301.

[21] Carlo Baldassi et al. "Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 113.48 (2016), E7655–E7662.

[22] Fabrizio Pittorino et al. "Entropic gradient descent algorithms and wide flat minima". In: *J. Stat. Mech.* 2021.12 (2021), p. 124015.

[23] Carlo Baldassi, Enrico M Malatesta, and Riccardo Zecchina. "Properties of the geometry of solutions and capacity of multilayer neural networks with rectified linear unit activations". en. In: *Phys. Rev. Lett.* 123.17 (2019), p. 170602.

[24] Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. "Shaping the learning landscape in neural networks around wide flat minima". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 117.1 (2020), pp. 161–170.

[25] Yiding Jiang et al. "Fantastic generalization measures and where to find them". In: (2019). eprint: `1912.02178`.

[26] Pratik Chaudhari et al. "Entropy-SGD: biasing gradient descent into wide valleys". In: *J. Stat. Mech.* 2019.12 (2019), p. 124018.

[27] Max Welling and Yee Whye Teh. "Bayesian learning via stochastic gradient langevin dynamics". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Madison, WI, USA: Omnipress, 2011, pp. 681–688.

[28] J J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 79.8 (1982), pp. 2554–2558.

[29] J J Hopfield. "Neurons with graded response have collective computational properties like those of two-state neurons". en. In: *Proc. Natl. Acad. Sci. U. S. A.* 81.10 (1984), pp. 3088–3092.

[30] John Hertz, Anders Krogh, and Richard G Palmer. *Introduction to the theory of neural computation*. 1st Edition. Boca Raton, FL: CRC Press, 2018.

[31] D J Amit, H Gutfreund, and H Sompolinsky. "Spin-glass models of neural networks". en. In: *Phys. Rev. A Gen. Phys.* 32.2 (1985), pp. 1007–1018.

[32] D J Amit, H Gutfreund, and H Sompolinsky. "Storing infinite numbers of patterns in a spin-glass model of neural networks". en. In: *Phys. Rev. Lett.* 55.14 (1985), pp. 1530–1533.

[33] Daniel J Amit, Hanoch Gutfreund, and H Sompolinsky. "Statistical mechanics of neural networks near saturation". en. In: *Ann. Phys. (N. Y.)* 173.1 (1987), pp. 30–67.

[34] Daniel J Amit. *Modeling brain function: The world of attractor neural networks*. en. Cambridge, England: Cambridge University Press, 1989.

[35] J L van Hemmen and I Morgenstern, eds. *Heidelberg colloquium on glassy dynamics: Proceedings of a colloquium on spin glasses, optimization and neural networks held at the university of Heidelberg June 9-13, 1986*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987.

[36] A Crisanti and H Sompolinsky. “Dynamics of spin systems with randomly asymmetric bonds: Langevin dynamics and a spherical model”. en. In: *Phys. Rev. A Gen. Phys.* 36.10 (1987), pp. 4922–4939.

[37] M V Feigelman and L B Ioffe. “The statistical properties of the Hopfield model of memory”. In: *EPL* 1.4 (1986), pp. 197–201.

[38] M V Feigelman and L B Ioffe. “The augmented models of associative memory asymmetric interaction and hierarchy of patterns”. en. In: *Int. J. Mod. Phys. B* 01.01 (1987), pp. 51–68.

[39] G Parisi. “Asymmetric neural networks and the process of learning”. In: *J. Phys. A Math. Gen.* 19.11 (1986), pp. L675–L680.

[40] G Toulouse, S Dehaene, and J P Changeux. “Spin glass model of learning by selection”. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 83.6 (1986), pp. 1695–1698.

[41] M P Singh, Z Chengxiang, and C Dasgupta. “Fixed points in a Hopfield model with random asymmetric interactions”. en. In: *Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics* 52.5 (1995), pp. 5261–5272.

[42] Walter Rudin. *Functional Analysis*. en. 2nd ed. Maidenhead, England: McGraw Hill Higher Education, 1990.

[43] Sebastian Goldt et al. “Modeling the influence of data structure on learning in neural networks: The hidden manifold model”. en. In: *Phys. Rev. X.* 10.4 (2020).

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[45] Pascal Vincent et al. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”. In: *J. Mach. Learn. Res.* 11 (Dec. 2010), pp. 3371–3408. ISSN: 1532-4435.

[46] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]`.