Continuous Hopfield Networks

Marcello Catacchini Stefano Magrini Alunno Dario Ferracci

November 18, 2024

Contents

Chapter 1

Introduction

Neural networks found their roots in the 1940s, when Norbert Wiener introduced the concept of cybernetics with his book *Cybernetics: Or Control and Communication in the Animal and the Machine*. The goal of cybernetics was to mimic natural systems to create mathematical models that could support the development of artificial intelligences. In particular, the attempt to replicate biological neural networks proved to be an attractive goal not only for mathematics, but also for many other disciplines. [1]

In the 1950s, Frank Rosenblatt defined the perceptron in his article *The perceptron: A probabilistic model for information storage and organization in the brain*, the first example of a neural network capable of autonomous learning to classify data. Despite its many limitations, the perceptron was an important starting point for understanding the direction of neural network research.

In the 1980s, John Hopfield in his article *Neural networks and physical systems with emergent collective computational abilities* proposed the first example of a neural network with an associative memory function. As time went on, researchers succeeded better and better in mimicking some of the basic functions of the human brain. Hopfield's networks were able to reconstruct or recall patterns without the use of indexical systems: the network stored information in a distributed manner and was able to retrieve it independently.

In the 1980s, the backpropagation algorithm was also developed, marking a breakthrough for training feedforward networks and opening the door to deep learning [2]. In the 1990s, convolutional networks (CNNs) and recurrent networks (RNNs), specific models that address practical application needs of neural networks, are introduced. Finally, thanks to technological advancement and the availability of more computational resources since 2010, it has become possible to effectively explore deep neural networks, revealing their extraordinary potential.

Hopfield networks introduced the possibility of storing patterns nonsequentially, making it possible to retrieve incomplete or noisy information. The network can recognize a pattern even if it is presented with a partial or distorted version, as human memory does when it recognizes a familiar face or object. This concept was novel compared to "addressed" memory systems, opening new avenues for implementing more flexible patterns.

Hopfield networks represented an example of distributed memory, in which every neuron in the network participates in the storage process. Unlike centralized memory systems, information in a Hopfield network is distributed among all synaptic connections, which makes it resistant to local errors and increases storage capacity.

Despite limitations in scalability, Hopfield networks have been successfully used to solve combinatorial optimization problems, such as the traveling salesman problem. The novel approach of Hopfield networks has inspired further developments in recurrent networks and associative memories, also influencing applications of modern recurrent neural networks.

Hopfield networks, while fundamental, had limitations such as low storage capacity and difficulty scaling to complex problems. However, they provided a strong impetus for subsequent research, leading

[1] For more about cybernetics, see https://en.wikipedia.org/wiki/Cybernetics:_Or_Control_and_Communication_in_the_Animal_and_the_Machine

[2] For more about backpropagation, see *Applications of advances in nonlinear sensitivity analysis* of Paul J. Werbos

to the development of more advanced architectures such as Boltzmann networks and deep neural networks, which built on their insight to solve increasingly complex problems.

In recent decades, the development of neural networks has increasingly turned toward feedforward models and deep neural networks. These models are based on a layered structure in which neurons, organized in different layers, process information sequentially. Unlike recurrent and associative networks, feedforward networks do not have internal loops, allowing information to be handled in a simple and linear manner, making them suitable for classification and regression tasks.
Deep networks make extensive use of logits, real numeric values that represent the activation of each neuron with respect to a certain input. These logits indicate how much a neuron "believes" the input to be similar to a specific class or feature, allowing the network to make decisions at different levels of abstraction.
This statistical approach, which relies on the activation of neurons through logits, is the basis of modern backpropagation and learning algorithms, making neural networks increasingly accurate and flexible in solving complex problems.

The analysis of continuous Hopfield networks proves fundamental to understanding the evolution of modern neural architectures. Unlike their discrete counterparts, these networks can directly incorporate the concept of backpropagation, allowing continuous optimization of weights through gradient activation functions.
Continuous Hopfield networks introduce smoother dynamics in pattern representation and information retrieval, improving learning ability compared to discrete models. This aspect is highlighted in the article **"Hopfield Networks Are All You Need"**, where the authors explore how these networks can support more complex and adaptive learning functions.
In addition, continuous Hopfield networks lay the foundation for the adoption of advanced techniques, such as deep neural networks, which make use of more sophisticated learning algorithms. Therefore, the evolution from discrete to continuous Hopfield networks offers important insights for research and application in the field of machine learning.

In this article, we will explore the applications of Hopfield networks in advanced contexts. We will begin by examining discrete Hopfield networks in detail, explaining their foundational mechanisms and functionality, as well as their connections to statistical mechanics. Additionally, we will focus on using Hopfield networks for data cleaning, demonstrating how effective results can be achieved with minimal parameterization, a concept that parallels techniques in convolutional neural networks (CNNs).
We will highlight how Hopfield networks can be employed for pattern management and reconstruction, allowing information retrieval even in the presence of noisy or incomplete data.
In addition, we will analyze continuous and stochastic Hopfield networks, studying their relationships with Variational Autoencoders (VAEs) and how these interactions can further expand their scope in contemporary machine learning.
Through these explorations, we intend to illustrate the potential of Hopfield networks not only as theoretical models, but as practical tools to address specific challenges in data preprocessing and performance improvement in learning models.

Chapter 2

Discrete Hopfield Networks

In this chapter we are going to resume the notes of the course of Mathematical Models for Neural Networks of Agliari [1]. In particular, we are going to talk about the derivation of Hopfield neural networks and their properties. We are going to start from the derivation as a neural network with discrete states and finally we are going to use the perspective of statistical mechanics to analyze these neural networks.

2.1 Derivation

It's possible to make a mathematical model in order to analyze a natural neural network. Each neuron is a node of a graph and each synapse is an edge. A neural network is thus a graph.
Let two neurons, A and B, be linked by a synapse. With respect to neuron A, we say that A is a post-synaptic neuron and B is a pre-synaptic neuron. A post-synaptic neuron receives information from all possible pre-synaptic neurons. This information consists of positive or negative ions, or even hormones that stimulate or alter the behavior of the neuron. In particular, these neurotransmitters either increase or decrease the potential of the neuron. If this potential exceeds a threshold, the neuron sends a message to the post-synaptic neurons and resets the potential. In fig. 2.1, we show a simple diagram of the interaction between pre-synaptic and post-synaptic neurons.

Notation. In this article we denote with:

- U_i^t The potential of i^{th} neuron at time t.
- S_i^t The signal of i^{th} neuron at time t from post-synaptic neurons.
- $U_i^\star$ the threshold of i^{th} neuron.
- J The matrix of weights, where J_{ij} is the weight over the synapse from i to j.
- θ The activation function (0 if the argument is not positive, 1 otherwise).

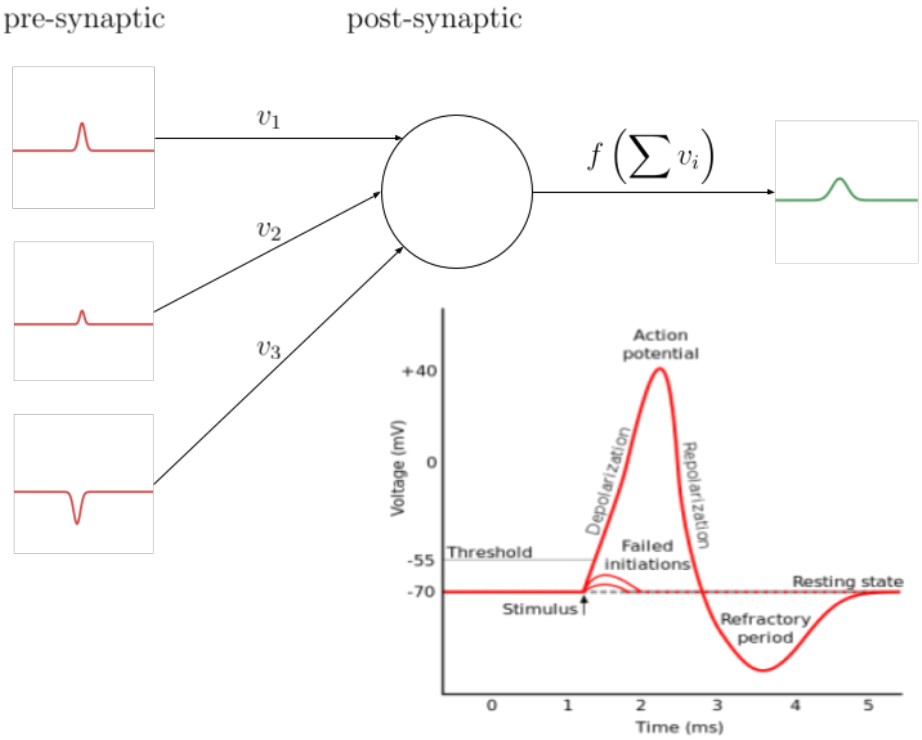


Figure 2.1: Pre-synaptic neurons send a signal to the post-synaptic neuron as a change in potential. The post-synaptic neuron accumulates all signals and generates a new signal if its potential exceeds a threshold.

Definition 2.1.1 (interaction)**.** We define the interaction rule with following formula, using the **McCulloch-Pitts** model:

$$U_i^t = \sum_j J_{ij} S_j^t \in \left[0, \sum_j J_{ij}\right] \tag{2.1}$$

$$S_i^{t+1} = \theta\left[U_i^t - U_i^\star\right] \in \{0,1\} \tag{2.2}$$

It's possible to get a different interpretation using only signals:

$$S_i^{t+1} = \theta\left[\sum_j J_{ij} S_j^t - U_i^\star\right]$$

then, we define **ising** model, where the signal is $\xi_i^t = 2S_i^t - 1 \in \{-1, 1\}$ and the threshold is $h_i = 2U_i^\star - \sum_j J_{ij}$:

$$\xi_i^{t+1} = \operatorname{sgn}\left[\sum_j J_{ij}\xi_j^t + h_i\right] \tag{2.3}$$

We define the interaction, but not the update of the entire neural network. In this article, we use both parallel and sequential dynamics, which are defined as follows:

Definition 2.1.2 (updating)**.** We use matrix notation as follows:
Parallel dynamics:

$$\xi^{t+1} = \operatorname{sgn}\left[J\xi^t + h\right]$$

All neurons are updated in parallel after one iteration.
Sequential dynamics:
Choose i randomly:

$$\xi_i^{t+1} = \operatorname{sgn}\left[\sum_j J_{ij}\xi_j^t + h_i\right]$$

Only one neuron is updated at a time in sequential dynamics.

Where ξ^t is the vector of all neurons' states (or Ising spins). Furthermore, we define the **local field** $\phi\left(\xi^t\right) = J\xi^t + h$.
We denote ξ^{new} as the updated state of ξ.

Remark. We present an alternative interpretation of the update formula (for sequential and parallel dynamics). The direction in fig. 2.2 is a good direction to follow to minimize the function: $-\frac{1}{2}\xi \cdot J\xi - h \cdot \xi$ because its negative gradient is $J\xi + h$.

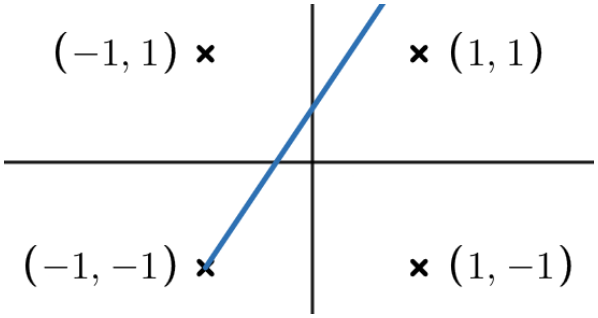


Figure 2.2: The optimal direction is indicated by the blue line, so the next state moves in that direction. The next state can be $(1, 1)$ in parallel dynamics, whereas in sequential dynamics, it can be $(-1, 1)$

2.1.1 Sequential dynamics

Many properties of the Ising model dynamics are obtained using sequential dynamics. In this part of the article, we will focus on this updating rule.

Definition 2.1.3 (Lyapunov's function)**.** Given an autonomous dynamical system $\dot{x} = f(x)$ with equilibrium point x_0, a Lyapunov function is a function V such that:

- $V(x) > 0$ with $x \neq x_0$
- $V(x_0) = 0$
- $\nabla V(x) \cdot f(x) \leq 0$

If V exists then x_0 is stable for Lyapunov.
Remark: for a Lyapunov function it holds that $\dot{V}(x) \leq 0$

The dynamics of the neural network are a discretization of an autonomous dynamical system. In this subsection, we will refer to the function denoted by $L_{J,h}$ as a Lyapunov function, although this is not a rigorous definition. The following lemma defines this function and relates the interpretation in fig. 2.2 to the Lyapunov function in definition 2.1.3.

Lemma 2.1.1 (Decreasing function)**.** *Using Ising's model with sequential dynamics and interaction J such that $J_{ii} \geq 0$ and $J_{ij} = J_{ji}$, the following function:*

$$L_{J,h}(\xi) = -\frac{1}{2}\sum_{ij} \xi_i J_{ij} \xi_j - \sum_i h_i \xi_i$$

decreases over time: $L_{J,h}\left(\xi^{new}\right) \leq L_{J,h}\left(\xi\right)$.

Proof. Sequential dynamics can change only one value of the network so we can suppose that i is the updated neuron: $\xi_k^{\text{new}} = \xi_k$ for all $k \neq i$.
If $\xi_i^{\text{new}} = \xi_i$ there is not any difference, so now we assume $\xi_i^{\text{new}} = -\xi_i$ and study the variation of Lyapunov function $\Delta L_{J,h}$:

$$\begin{aligned}
\Delta L_{J,h}\left(\xi, \xi^{\text{new}}\right) &= -\frac{1}{2}\sum_{k,l} J_{kl}\left(\xi_k^{\text{new}}\xi_l^{\text{new}} - \xi_k\xi_l\right) - \sum_l h_l\left(\xi_l^{\text{new}} - \xi_l\right) \\
&\overset{1}{=} -\sum_{l\neq i} J_{il}\left(\xi_i^{\text{new}}\xi_l^{\text{new}} - \xi_i\xi_l\right) - \left(-2h_i\xi_i\right) \\
&= -\sum_{l\neq i} J_{il}\left(-\xi_i\xi_l - \xi_i\xi_l\right) - \left(-2h_i\xi_i\right) = 2\sum_l J_{il}\xi_i\xi_l + 2h_i\xi_i - J_{ii} \\
&= 2\xi_i\left(\sum_l J_{il}\xi_l + h_i\right) - J_{ii} \overset{2}{=} -2\left|\left(\sum_l J_{il}\xi_l + h_i\right)\right| - J_{ii} \leq 0
\end{aligned}$$

In equivalence 1 we used the symmetry of J and $\xi, \xi^{\text{new}} = \pm 1$.
In equivalence 2 we used the update formula:

$$\xi_i\left(\sum_l J_{il}\xi_l + h_i\right) = \xi_i \operatorname{sgn}\left[\sum_l J_{il}\xi_l + h_i\right]\left|\sum_l J_{il}\xi_l + h_i\right| = \xi_i\xi_i^{\text{new}}\left|\sum_l J_{il}\xi_l + h_i\right|$$

□

Remark (lower bound)*.* We can see that the Lyapunov function is lower bounded:

$$\begin{aligned}
L_{J,h}(\xi) &= -\frac{1}{2}\sum_{i,j} \xi_i J_{ij} \xi_j - \sum_i h_i \xi_i \\
&\geq -\frac{1}{2}\sum_{i,j} J_{ij} - \sum_i h_i \geq -\frac{N^2}{2}|J| - N|h|
\end{aligned}$$

where $|J| = \max_{i,j}|J_{ij}|$ and $|h| = \max_i |h_i|$.
Through the dynamic of the Ising's model, the value of the Lyapunov function decreases until it reaches a minimum L^*.

The last step is to find a fixed point of the updating (in sequential dynamic) and to obtain the lower bound at it. In this way we know that the fixed point is stable for Lyapunov. The idea is to choose J and h such that the final configuration of the system is something meaningful. In particular, we want to store P patterns $(x_1, ..., x_P)$ in the network and given a starting point ξ retrieve some pattern as the final point of the dynamics. Following this perspective we can interpret the value of the Lyapunov function as an energy function for the system that will evolve towards the minimum points.
We need to make these patterns the minimum points of the Lyapunov function to do this analysis. This problem can be solved using Hebb's rule:

Definition 2.1.4 (Hebb's rule)**.** Given $P \leq N$ patterns $(x_1, \ldots, x_P)$ we define Hebbian interaction matrix as:

$$J_{ij} = \frac{1}{N}(1 - \delta_{i,j})\sum_{\mu=1}^{P} x_i^\mu x_j^\mu$$

Lemma 2.1.2 (Pattern as minimum points)**.** *Let J be made with the Hebbian rule using orthogonal patterns (i.e. $\sum_i x_i^\mu x_i^\nu = N\delta_{\mu,\nu}$), every pattern x^μ is a global minimum for the Lyapunov function with $h = 0$.*

Proof.

$$\begin{aligned} L_{J,h}(\xi) &= -\frac{1}{2}\sum_{i,j}\xi_i J_{ij}\xi_j = -\frac{1}{2}\frac{1}{N}\sum_{i,j}(1-\delta_{i,j})\sum_\mu x_i^\mu x_j^\mu \xi_i\xi_j \\ &= -\frac{1}{2}\frac{1}{N}\sum_{i\neq j}\sum_\mu x_i^\mu x_j^\mu\xi_i\xi_j = \frac{1}{2}P - \frac{1}{2}\frac{1}{N}\sum_{i,j}\sum_\mu x_i^\mu x_j^\mu\xi_i\xi_j = \\ &= \frac{1}{2}P - \frac{1}{2}\sum_\mu\left(\frac{1}{\sqrt{N}}\sum_i x_i^\mu\xi_i\right)^2 \geq \frac{1}{2}(P-N) \overset{1}{=} L_{J,h}(x^\nu) \quad \forall x^\nu \in (x_1,\dots,x_P) \end{aligned}$$

the last inequality holds because of the orthogonality of the patterns with same euclidean norm N:

$$\sum_\mu\left(\frac{1}{\sqrt{N}}\sum_i x_i^\mu\xi_i\right)^2 = \|\xi\|^2 \leq N$$

□

2.1.2 Stochastic model

With this framework we already use the network for pattern retrieval, but there are two main problems: the first is that the patterns are not always orthogonal, the second is that we can also retrieve a local minimum of the Lyapunov function, to overcome this problem we introduce the stochastic model. Assume that $U^\star$ is not a deterministic threshold, and that we can express it in this way:

$$\forall i\left(U_i^\star(t) = U_i^\star - \frac{1}{2}Tz_i(t)\right)$$

with $z_i(t) \sim \mathcal{P}$ such that it's symmetric ($\implies \overline{z_i(t)} = 0$) and $\overline{z_i(t)^2} = 1$. The distribution function of $\mathcal{P}$ is denoted with $g(x) = \frac{1}{2} + \int_0^x \mathcal{P}(u)\,du$.
T is called **temperature** and is not negative. The temperature controls the randomness of the process: $T = 0$ implies a deterministic behaviour and as T increases, the behaviour tends to become increasingly random, approaching completely random behaviour in the limit as $T \to \infty$.

Remark. We will use $g(x) = \frac{1}{2}(1 - \tanh(x))$

Definition 2.1.5 (Stochastic sequential dynamics)**.** Consistent with the sequential dynamics of the deterministic process, we define the stochastic update using a non-deterministic threshold.
Let $U_t \sim Unif([N])$, for each component i it holds that:

$$\xi_i^{t+1} := \begin{cases} \operatorname{sgn}\left[\phi_i(\xi^t) + Tz_i(t)\right] & \text{if } U_{t+1} = i \\ \xi_i^t & \text{otherwise} \end{cases}$$

remark: z_i is a random variable for each t.

Lemma 2.1.3 (stochastic derivation)**.** *Using an ising model with sequential dynamics, we can formulate the following thesis:*

$$\mathbb{P}\left[\xi^{t+1} = \xi' \middle| \xi^t = \xi\right] = \frac{1}{N}\sum_i \frac{1}{2}(1 + \xi_i' \tanh(\beta\phi_i(\xi'))\delta_{\xi,\xi'} + \frac{e^{-\beta\xi_i\phi_i(\xi)}}{2\cosh(\beta\phi_i(\xi))}\delta_{F_i(\xi),\xi'}$$

where F_i is the flip operator (it changes the sign of the i^{th} component) and $\beta = \frac{1}{T}$

Proof. We can divide the proof in three cases:

i) $\xi' = \xi$

$$\mathbb{P}\left[\xi^{t+1} = \xi' \middle| \xi^t = \xi\right] = \mathbb{P}\left[\xi^{t+1} = \xi \middle| \xi^t = \xi\right] = \sum_i \mathbb{P}\left[\xi^{t+1} = \xi \middle| \xi^t = \xi, U_{t+1} = i\right] \mathbb{P}\left[U_{t+1} = i\right]$$
$$= \frac{1}{N} \sum_i \mathbb{P}\left[\operatorname{sgn}\left[\phi_i(\xi^t) + T z_i(t)\right] = \xi_i \middle| \xi^t = \xi, U_{t+1} = i\right] = \frac{1}{N} \sum_i \mathbb{P}\left[\operatorname{sgn}\left[\phi_i(\xi) + T z_i(t)\right] = \xi_i\right]$$
$$= \frac{1}{N} \sum_i \mathbb{P}\left[\xi_i \left(\phi_i(\xi) + T z_i\left(t\right)\right) \geq 0\right] \overset{1}{=} \frac{1}{N} \sum_i \mathbb{P}\left[z_i(t) \geq -\beta \xi_i \phi_i\left(\xi\right)\right]$$
$$= \frac{1}{N} \sum_i \frac{1}{2}\left(1 + \xi_i \tanh\left(\beta \phi_i\left(\xi\right)\right)\right)$$

In equivalence 1 we used the relation $\xi_i z_i \sim z_i$ due to the symmetry

ii) $\xi' = F_i(\xi)$

$$\mathbb{P}\left[\xi^{t+1} = \xi' \middle| \xi^t = \xi\right] = \mathbb{P}\left[\xi^{t+1} = F_i(\xi) \middle| \xi^t = \xi\right]$$
$$= \mathbb{P}\left[\xi^{t+1} = F_i(\xi) \middle| \xi^t = \xi, U_{t+1} = i\right] \mathbb{P}\left[U_{t+1} = i\right] + \mathbb{P}\left[\xi^{t+1} = F_i(\xi) \middle| \xi^t = \xi, U_{t+1} \neq i\right] \mathbb{P}\left[U_{t+1} \neq i\right]$$
$$= \frac{1}{N} \mathbb{P}\left[\operatorname{sgn}\left[\phi_i(\xi^t) + T z_i(t)\right] = F_i(\xi)_i \middle| \xi^t = \xi, U_{t+1} = i\right]$$
$$= \frac{1}{N} \mathbb{P}\left[\xi_i \left(\phi_i(\xi^t) + T z_i\left(t\right)\right) \leq 0\right] = \frac{1}{N} \mathbb{P}\left[z_i(t) \leq -\beta \xi_i \phi_(\xi)\right]$$
$$= \frac{1}{2}\left(1 - \xi_i \tanh\left(\beta \phi_i\left(\xi\right)\right)\right) = \frac{e^{-\beta \xi_i \phi_i(\xi)}}{2\cosh(\beta \phi_i(\xi))}$$

iii) otherwise
Sequential dynamics can only change one neuron for each update so the probability is 0.

□

Markov chains In lemma 2.1.3 we proved that only the last state established the new state. This condition defines a Markov chain over $\{-1, 1\}^N$. Define $W_{\xi,\xi'}$ the probability to jump from ξ in ξ':

$$W_{\xi,\xi'} = \frac{1}{N} \sum_i \frac{1}{2}(1 + \xi_i' \tanh(\beta \phi_i(\xi')) \delta_{\xi,\xi'} + \frac{e^{-\beta \xi \phi_i(\xi)}}{2\cosh(\beta \phi_i(\xi))} \delta_{F_i(\xi),\xi'}$$

The matrix $W = (W_{\xi,\xi'})_{\xi,\xi'}$ is the transition matrix of the Markov chain.

Remark. Recall following properties:

- $\exists \tau \left(t \geq \tau \implies W^t > 0\right)$, so it is an ergodic Markov chain. So there exists a unique invariant distribution p_∞ to which the process will converges form any initial state.
- If $\forall \xi, \xi' \left(p(\xi) W_{\xi,\xi'} = p(\xi') W_{\xi'\xi}\right)$ then p is an invariant distribution. The equivalence is called **detailed balance**.

We use stochastic dynamics to escape from non-optimal minima. Now we study a limit probability of the state of the neural network using weights J similar to definition 2.1.4. But before addressing the following theorem let us recall this lemma from graph theory.

Lemma 2.1.4. *Let G graph with vertices V and oriented edges E such that a vertex connected with all vertices exists, called O. Each edge $\vec{e}$ has a value $c(\vec{e})$ and $c(-\vec{e}) = -c(\vec{e})$ (it is called 'flux'). The function $K : V \to \mathbb{R}$ (called potential) is such that $K(v) - K(w) = c(\vec{vw})$ exists if and only for each cycle $\vec{e_1}, \ldots, \vec{e_n}$ holds $\sum c(\vec{e_k}) = 0$ (flow conservation).*

Proof. If K exists, then it's clear that the summation of values over a cycle is 0.
If flow conservation holds, then for each vertices v we can find a path $\vec{e_1}, \ldots, \vec{e_n}$ to O from v with vertices $O = v_0, \ldots, v_n = v$, so:

$$K(v) - K(O) = \sum K(v_{i+1}) - K(v_i)$$

remark: potential exists at less than a constant. □

Theorem 2.1 (*Invariant measure*)

In a sequential stochastic dynamic without self-interactions, i.e. $J_{ii} = 0$, detailed balanced is equivalent to symmetric interactions:

$$J_{ij} = J_{ji} \iff p_\infty(\xi)W_{\xi\xi'} = p_\infty(\xi')W_{\xi'\xi} \quad \forall \xi, \xi'$$

Furthermore, the stationary distribution is:

$$p_\infty(\xi) \propto e^{-\beta \mathcal{H}_{J,h}(\xi)}$$

where $\mathcal{H}_{J,h} = -\frac{1}{2}\sum_{i,j}\xi_i J_{ij}\xi_j - \sum_i h_i\xi_i$ (called Hamiltonian function).

Proof. We know that if p verifies the detailed balance and p is a distribution, then $p_\infty = p$. So we are going to start by finding an explicit form of a distribution that verifies detailed balance. We are going to prove that this distribution exists if J is symmetric, and then the opposite (only if J is symmetric).

At the moment, we are rewriting detailed balance in a simple form.
If $\xi' = \xi$ or $\xi' \neq \xi \wedge \forall i\,(\xi' \neq F_i(\xi))$ the detailed balance holds trivially because we get $0 = 0$. For each ξ, ξ' such that $\exists i\,(\xi' = F_i(\xi))$, we want to express the detailed balance in a simpler form. Using lemma 2.1.3:

$$p_\infty(\xi)\frac{e^{-\beta\xi_i\phi_i(\xi)}}{2\cosh(\beta\phi_i(\xi))} = p_\infty(\xi')\frac{e^{-\beta\xi_i'\phi_i(\xi')}}{2\cosh(\beta\phi_i(\xi'))}$$

We remark that there's no self-interaction, so:

$$\phi_i(\xi) = \sum_j J_{ij}\xi_j + h_i = \sum_{j\neq i} J_{ij}\xi_j + h_i = \sum_{j\neq i} J_{ij}\xi_j' + h_i = \sum_j J_{ij}\xi_j' + h_i = \phi_i(\xi')$$

This gives us an equivalent form of detailed balance sheet:

$$\begin{aligned} &\forall \xi, \xi'\,(p_\infty(\xi)W_{\xi\xi'} = p_\infty(\xi')W_{\xi'\xi}) \\ &\iff \forall \xi, \xi', i\left(\xi' = F_i(\xi) \implies p_\infty(\xi)e^{-\beta\xi_i\phi_i(\xi)} = p_\infty(\xi')e^{-\beta\xi_i'\phi_i(\xi')}\right) \end{aligned} \tag{2.4}$$

Now let's try to find a distribution p that verifies the detailed balance. We know that the distribution p_∞ is positive over all states, so our target distribution has to be positive over all states as well. We can write p in exponential form: $p\,(\xi) = \exp\,(-\beta K(\xi))$. This simplifies the equivalent form in eq. (2.4).
Using eq. (2.4), we can verify detailed balance only over the couples ξ, ξ' such that $\exists i\,(\xi' = F_i\,(\xi))$. Simplifying:

$$\begin{aligned} &K(\xi) + \xi_i\phi_i(\xi) = K(\xi') + \xi_i'\phi_i(\xi') \\ &K(\xi') - K(\xi) = 2\xi_i\phi_i(\xi) = 2\xi_i\left(\sum_l J_{il}\xi_l + h_i\right) \end{aligned}$$

Using lemma 2.1.4, it is possible to find an explicit form of K if and only if $2\xi_i\left(\sum_l J_{il}\xi_l + h_i\right)$ is a flux that verifies the flow conservation.
We can express the flow conservation with a simpler and equivalent condition.

Let i, j be different indices, get a path $\mathcal{P}_1 = (v, F_i(v), F_j(F_i(v)))$, we want that the flux over $\mathcal{P}_1$ to be the same flux over $\mathcal{P}_2 = (v, F_j(v), F_i(F_j(v)))$:

$$\begin{aligned}
\text{flux}\,(\mathcal{P}_1) &= 2v_i\left(\sum_l J_{il}v_l + h_i\right) + 2F_i(v)_j\left(\sum_l J_{jl}F_i(v)_l + h_j\right) \\
&= 2v_i\left(\sum_l J_{il}v_l + h_i\right) + 2v_j\left(\sum_l J_{jl}v_l + h_j - 2J_{ji}v_i\right) \\
&= 2v_i\left(\sum_l J_{il}v_l + h_i\right) + 2v_j\left(\sum_l J_{jl}v_l + h_j\right) - 2J_{ji}v_jv_i \\
\text{flux}\,(\mathcal{P}_2) &= 2v_j\left(\sum_l J_{jl}v_l + h_i\right) + 2F_j(v)_i\left(\sum_l J_{il}F_j(v)_l + h_i\right) \\
&= 2v_j\left(\sum_l J_{jl}v_l + h_j\right) + 2v_i\left(\sum_l J_{il}v_l + h_i - 2J_{ij}v_j\right) \\
&= 2v_j\left(\sum_l J_{jl}v_l + h_j\right) + 2v_i\left(\sum_l J_{il}v_l + h_i\right) - 2J_{ij}v_iv_j
\end{aligned}$$

so $\text{flux}\,(\mathcal{P}_1) = \text{flux}\,(\mathcal{P}_2) \iff J_{ij} = J_{ji}$.
The flow does not change when we commute 2 flips. In a cycle we know that if we have a flip F_i then there is another F_i. Using commutation we can remove all flips and so the flux is conservative.
We prove that there is a potential at less than a constant, and in order to get a distribution from K, it's possible to set a particular potential $K(\xi) + \text{const}$ such that p is a distribution. Now we know that the thesis is true, so we can compute K to prove that $p_\infty(\xi) \propto e^{-\beta\mathcal{H}_{J,h}(\xi)}$.
Seeing that $K(\xi') - K(\xi) = 2\xi_i(\sum_l J_{il}\xi_l + h_i)$ we recognize the result in lemma 2.1.1:
$K(\xi') - K(\xi) = \Delta L_{J,h}(\xi, \xi')$
So a good option for the potential K is:

$$K(\xi) = -\frac{1}{2}\sum_{ij}\xi_iJ_{ij}\xi_j - \sum_i h_i\xi_i + \text{const} = \mathcal{H}_{J,h}(\xi) + \text{const}$$

□

2.2 Statistical mechanics perspective

According to theorem 2.1, there is a connection between the dynamics of the network and spin models studied in statistical mechanics, since the state of a neuron can be associated with the alignment of an atom in a magnetic field, or its spin. Statistical mechanics, a branch of physics, connects the individual properties of atoms or molecules to macroscopic observable quantities such as temperature and pressure. This framework provides insight into how the collective behavior of many particles gives rise to the thermodynamic properties we observe. Using this approach, we can analyze how the behavior of the system changes in response to changes in its parameters. The key idea is to interpret the state of the system as a mixture of pure states - a probability distribution over these pure states - and to identify the configuration that minimizes a given energy function. Another essential aspect is to determine specific order parameters on which the probability distribution depends; by analyzing these, we can gain insight into the behavior of the entire system.

Definition 2.2.1. According to statistical mechanics, we can define a state of a system ρ as a convex combination of the set of pure states $\{\rho^1, \dots, \rho^{\mathcal{N}}\}$
Given a state ρ, we define the following quantities:

i) $U(E, \rho) = \sum_i \rho_i E_i$ **internal energy**

ii) $S(\rho) = -K_B \sum_i \rho_i \log(\rho_i)$ **Gibbs entropy**

iii) $f(E, \rho, T) = U(E, \rho) - TS(\rho)$ **free energy**

Definition 2.2.2 (Gibbs distribution)**.** The limit distribution given in theorem 2.1 plays a key role in statistical mechanics. In this setting we can rephrase it as Gibbs distribution:

$$\rho_\beta(\xi) = \frac{e^{-\frac{\beta E(\xi)}{K_B}}}{Z_\beta}$$

where: $Z_\beta = \sum_\xi e^{-\frac{\beta E(\xi)}{K_B}}$ is the normalization factor called **partition function**.

In the context of neural networks:

- $E(\xi) = \mathcal{H}_{J,h}(\xi)$ is the energy of the neural network.
- $K_B = 1$ is a taken factor.

The role of this distribution is given by the following theorem:

Theorem 2.2 (*Gibbs distribution*)

The Gibbs distribution is the state ρ where the free energy reaches its minimum, this point is called thermodynamic equilibrium.

$$\inf_\rho f(E, \rho, T) = f(E, \bar{\rho}, T) = -KT \log(\bar{Z})$$

where $\bar{\rho}$ is the Gibbs distribution

Proof. The free energy is a convex function with respect to ρ because the energy is affine and the Gibbs entropy is concave, so we can find its minimum by finding the point where $\delta f = 0 \ \forall \rho$ variations. Hence we obtain:

$$\begin{cases} \delta f = \sum_i E_i \delta\rho_i + KT(\log(\rho_i) + 1)\delta\rho_i \overset{1}{=} \sum_i (E_i + KT \log(\rho_i))\delta\rho_i = 0 \\ \sum_i \delta\rho_i = 0 \end{cases}$$

In equivalence 1 we use the second equation.
From the first equation we can state that $E_i + KT \log(\rho_i)$ must not depend on i because the system must hold for any variation. So we can say that if $\bar{\rho}$ exists then:

$$E_i + KT \log(\bar{\rho}_i) = \bar{F}$$

from which we get $\bar{\rho}_i$, we have:

$$\bar{\rho}_i = e^{-\frac{E_i - \bar{F}}{KT}}$$

The constant $\bar{F}$ must be taken to satisfy:

$$1 = \sum_i \bar{\rho}_i = \sum_i e^{-\frac{E_i - \bar{F}}{KT}} = e^{\frac{\bar{F}}{KT}} \sum_i e^{-\frac{E_i}{KT}}$$

Using the definition of free energy and the formula for $\bar{\rho}$ we obtain:

$$f(E, \bar{\rho}, T) = \bar{F} e^{\frac{\bar{F}}{KT}} \sum_i e^{-\frac{E_i}{KT}} = \bar{F} \tag{2.5}$$

We can state that $\bar{Z} = e^{-\frac{\bar{F}}{KT}}$. □

We introduced noise to escape unwanted local minima, but in this way it is not possible to assume that the sequence converges to ideal states. So we found the limit distribution and observed that it is the Gibbs distribution (see theorem 2.1).
As the sequential deterministic dynamics reduces the Lyapunov function (see lemma 2.1.1), the sequential stochastic dynamics reduces the free energy. However, the convergence is towards the optimal density (see theorem 2.2).

2.2.1 Curie-Weiss model

To introduce the way of working of statistical mechanics, we begin with an analysis of the Curie-Weiss model.

Definition 2.2.3 (Curie-Weiss model)**.** The Curie-Weiss model is an Ising model in which the Hamiltonian function is:

$$\mathcal{H}_{N,J,h}(\xi) = -\sum_{i \neq j} \frac{J}{2} \xi_i \xi_j - h \sum_i \xi_i$$

Note: This is a special case where $J_{ii} = 0$, J symmetric and J_{ij} is constant where $i \neq j$.

This type of model is called a mean field model because each neuron interacts with all the others in the same way. The goal of the analysis here is to understand the behavior of the magnetization.

Definition 2.2.4 (magnetization)**.** Given a state ξ we call the magnetization the quantity:

$$m_N(\xi) = \frac{1}{N} \sum_i \xi_i$$

The magnetization is what we call an observable quantity because it depends on the state of all the neurons, not just one. An important observation is that we can write the Hamiltonian as a function of the magnetization as:

$$\mathcal{H}_{N,J,h}(\xi) = -\frac{NJ}{2} m_N(\xi)^2 - hN m_N(\xi)$$

We know thanks to the theorem 2.2 that the distribution that minimizes the free energy is the observed one, but we don't know anything about the magnetization, so our goal is to understand how it is effected by the temperature, at least when N is very large. Taking $N \to \infty$ is a key concept in statistical mechanics and it is called thermodynamic limit (TDL) it allows us to simplify some calculations by removing the dependence from the size of the model. To calculate the thermodynamic limit, we need to recall and introduce some quantities:

Definition 2.2.5 (Curie-Weiss quantities)**.** We define quantities for Curie - Weiss model:

- $\rho_{N,\beta,J,h}(\xi) = \frac{e^{-\beta \mathcal{H}_{N,J,h}(\xi)}}{Z_{N,\beta,J,h}}$ **equilibrium distribution**
- $Z_{N,\beta,J,h}$ **partition function**
- $F_{N,\beta,J,h} = -\frac{1}{\beta} \log(Z_{N,\beta,J,h})$ **free energy**
- $f_{N,\beta,J,h} = -\frac{1}{\beta N} \log(Z_{N,\beta,J,h})$ **intensive free energy**
- $f_{\beta,J,h} = \lim_{N \to \infty} f_{N,\beta,J,h}$ **intensive free energy in TDL**

Note: These definitions are addressed with definitions in definition 2.2.1 using the Gibbs distribution $\bar{\rho}$.

Now we are ready to analyze the Curie-Weiss model.
First, we will try to understand the probability distribution of the magnetization when the states follow the equilibrium distribution (definition 2.2.2):

$$\rho_{N,\beta,J,h}(m) \coloneqq \mathbb{P}_{\xi \sim \bar{\rho}} [m_N(\xi) = m] = \sum_{\xi} \rho_{N,\beta,J,h}(\xi) \delta(m_N(\xi) = m)$$

This measure is called **coarse grained measure** and our goal is to understand how it relates to the equilibrium measure in the TDL.
We also introduce the coarse-grained partition function, which is

$$Z_{N,\beta,J,h}(m) \coloneqq \sum_{\xi} e^{-\beta \mathcal{H}_{N,J,h}(\xi)} \delta(m_N(\xi) = m)$$

Definition 2.2.6 (coarse grained measure)**.** We will continue these definitions with other coarse-grained partition functions:

- $\rho_{N,\beta,J,h}(m) = \sum_{\xi} \rho_{N,\beta,J,h}(\xi)\delta(m_N(\xi) = m)$ **probability magnetisation**
- $Z_{N,\beta,J,h}(m) = \sum_{\xi} e^{-\beta \mathcal{H}_{N,J,h}(\xi)}\delta(m_N(\xi) = m)$ **partition function**
- $F_{N,\beta,J,h}(m) = -\frac{1}{\beta}\log(Z_{N,\beta,J,h}(m))$ **free energy**
- $f_{N,\beta,J,h}(m) = -\frac{1}{\beta N}\log(Z_{N,\beta,J,h}(m))$ **intensive free energy**
- $f_{\beta,J,h}(m) = \lim_{N\to\infty} f_{N,\beta,J,h}(m)$ **intensive free energy in TDL**

Proposition 2.1. *Using this formalism we can express the probability of the magnetization from the intensive free energy in TDL:*

$$\begin{aligned}
\rho_{N,\beta,J,h}(m) &= \sum_{\xi} \frac{e^{-\beta \mathcal{H}_{N,J,h}(\xi)}\delta\left(m_N(\xi) = m\right)}{Z_{N,\beta,J,h}} = \frac{Z_{N,\beta,J,h}(m)}{Z_{N,\beta,J,h}} \\
&= \frac{\exp\left(\frac{\beta}{\beta}\log\left(Z_{N,\beta,J,h}\left(m\right)\right)\right)}{\exp\left(\frac{\beta}{\beta}\log\left(Z_{N,\beta,J,h}\right)\right)} = \frac{\exp\left(-\beta F_{N,\beta,J,h}(m)\right)}{\exp\left(-\beta F_{N,\beta,J,h}\right)} = e^{-\beta(F_{N,\beta,J,h}(m) - F_{N,\beta,J,h})} \\
&= e^{-\beta N(f_{N,\beta,J,h}(m) - f_{N,\beta,J,h})}
\end{aligned} \tag{2.6}$$

Remark. In this way we can calculate the initial definitions in definition 2.2.5 in terms of intensive free energy in TDL:

$$\begin{aligned}
Z_{N,\beta,J,h} &= \sum_{\xi} \rho_{N,\beta,J,h}(\xi) = \sum_{m}\sum_{\xi} \rho_{N,\beta,J,h}(\xi)\delta(m_N(\xi) = m) \\
&= \sum_{m} Z_{N,\beta,J,h}(m) = \sum_{m} e^{-\beta N f_{N,\beta,J,h}(m)}
\end{aligned} \tag{2.7}$$

This formulation suggests that the terms that most influence the partition function are those where the value of $f_{N,\beta,J,h}(m)$ is minimized. Moreover, if $f_{\beta,J,h}(m)$ has a minimum in m^*, we can state for the thermodynamic limit, using the Taylor expansion for $f_{\beta,J,h}(m)$, that

$$\begin{aligned}
\lim_{N\to\infty} Z_{N,\beta,J,h} &\overset{1}{=} \lim_{N\to\infty} \sum_{m} e^{-\beta N f_{N,\beta,J,h}(m)} = \lim_{N\to\infty} \int_{-\infty}^{+\infty} e^{-\beta N f_{\beta,J,h}(m)} dm \\
&= \lim_{N\to\infty} \int_{-\infty}^{+\infty} e^{-\beta N\left(f_{\beta,J,h}(m^*) + \frac{f''_{\beta,J,h}(m^*)(m-m^*)^2}{2}\right)} dm \\
&= \lim_{N\to\infty} e^{-\beta N f_{\beta,J,h}(m^*)} \sqrt{\frac{2\pi}{N\beta f''_{\beta,J,h}(m^*)}}
\end{aligned}$$

Where in the equivalence 1 we used a similar conclusion of eq. (2.6), using this relation within the

definition of the intensive free energy in TDL we get:

$$\begin{aligned} f_{\beta,J,h} &= \lim_{N\to\infty} -\frac{1}{\beta N}\log(Z_{N,\beta,J,h}) \\ &= \lim_{N\to\infty} -\frac{1}{\beta N}\log\left(e^{-\beta N f_{\beta,J,h}(m^*)}\sqrt{\frac{2\pi}{N\beta f''_{\beta,J,h}(m^*)}}\right) \\ &= \lim_{N\to\infty} -\frac{-\beta N f_{\beta,J,h}(m^*)}{\beta N} - \frac{\log\left(\sqrt{\frac{2\pi}{N\beta f''_{\beta,J,h}(m^*)}}\right)}{\beta N} \\ &= \lim_{N\to\infty} f_{\beta,J,h}(m^*) = f_{\beta,J,h}(m^*) \end{aligned}$$

So if we can find the minima of the intensive free energy for each N ($m^\star_N$) and then we would like to be able to compute all minima $m^\star$ and thus the intensive free energy for TDL.

From theorem 2.2 we know that we can express intense free energy as

$$f_{N,\beta,J,h}(m) = \frac{E_{N,\beta,J,h}(\rho_{N,\beta,J,h}(m)) - TS(\rho_{N,\beta,J,h}(m))}{N}$$

We consider only the configuration with a fixed magnetization, so the energy is constant and the distribution is uniform over the number of possible configurations, which allows us to say:

$$f_{N,\beta,J,h}(m) = \frac{-\frac{NJ}{2}m^2 - hNm}{N} - T\frac{S(\text{Unif}\,[\Omega_N(m)]))}{N} = -\frac{J}{2}m^2 - hm + T\frac{\log(\Omega_N(m))}{N}$$

where $\Omega_N(m) = \sum_\xi \delta(m(\xi) - m)$ is the number of possible configurations.
In the TDL we get

$$\begin{aligned} f_{\beta,J,h}(m) &= \lim_{N\to\infty} f_{N,\beta,J,h}(m) = -\frac{J}{2}m^2 - hm + \lim_{N\to\infty} T\frac{\log(\Omega_N(m))}{N} \\ &= -\frac{J}{2}m^2 - hm + T\left[\frac{1+m}{2}\log\left(\frac{1+m}{2}\right) + \frac{1-m}{2}\log\left(\frac{1-m}{2}\right)\right] \end{aligned}$$

We are interested in finding the minimum so that we can take the derivative:

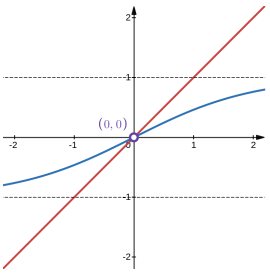
$$f'_{\beta,J,h}(m) = -Jm - h + T\left[\frac{1}{2}\log\left(\frac{1+m}{1-m}\right)\right] = -Jm - h + T\tanh^{-1}(m)$$

So stationary points are characterized by the following equation:

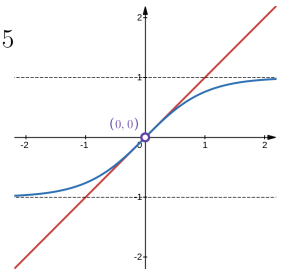
$$m^* = \tanh(\beta(Jm^* + h)) \tag{2.8}$$

This relation is called the self-consistent equation for the Curie-Weiss model and we can't solve it analytically. We will analyze it when $h = 0$ to understand where the minimum of the free energy is. The first observation is that $m^* = 0$ is always a solution and we can find the solution graphically. In a Cartesian graph with the axis x, y we draw the line $\{(x, y)|y = x\}$ and the function $\{(x, y)|y = \tanh(\beta(Jm^*)\}$.

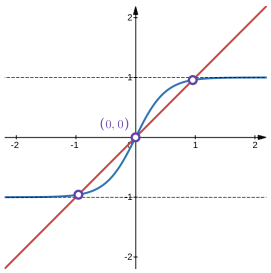
We want to understand when the eq. (2.8) has one solution and when it has three solutions. To do



$\beta J = 0.5$



$\beta J = 1$

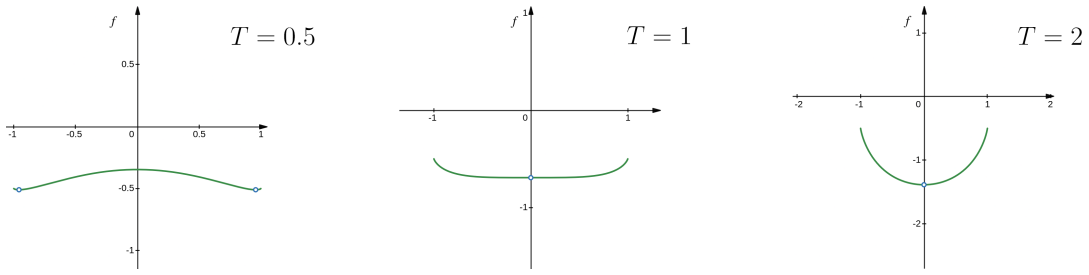


$\beta J = 2$

this, we will analyze the derivative of $\tanh(\beta(Jm)$ in zero:

$$\frac{d\tanh(\beta Jm)}{dm}\bigg|_{m=0} = (1-\tanh^2(\beta Jm))\beta J\bigg|_{m=0} = \beta J$$

From this we can see that the equation has one solution if $\beta J \leq 1$ and three solutions if $\beta J > 1$.
Now, from the graph of the free energy in both cases, we can see that $m^* = 0$ is a minimum if $\beta J \leq 1$ and that the positive and negative solutions are minima if $\beta J > 1$.



The behavior of the system is completely different in the two cases divided by the critical temperature $T_c = J$, in fact for temperatures above T_c randomness prevails and there are only disordered states, instead for temperatures lower than T_c ordered states appear.
Now we want to analyze the magnetization as a function of temperature to understand its behavior near the critical temperature.
To do this we will use the Taylor expansion of the hyperbolic tangent inside the eq. (2.8):

$$m = \beta Jm - \frac{1}{3}(\beta Jm)^3 + o(m^5)$$

Ignoring the higher order terms, we can find the solutions as follows:

$$m^* = \pm\sqrt{\frac{3(\beta J-1)}{(\beta J)^3}} = \pm\sqrt{\frac{3(T_c-T)}{T_c(\beta J)^2}} \sim \sqrt{T_c - T}$$

So we can conclude that the solution can be represented as a continuous function with a discontinuous derivative.
From this model we can understand two key concepts of statistical mechanics:

- **Phase transition**
  We have seen that the magnetization can be written as a function of temperature and that this function has a discontinuous derivative. This phenomenon suggests that the model is in a completely different state depending on the temperature, just as different materials can be gas, solid and liquid. In statistical mechanics this type of behavior is called a phase transition and it is associated with a discontinuity of a derivative of the free energy.

- **Ergodicity breaking**
  In the previous setting we noticed that the Markov chain representing the dynamics of the model is ergodic, so it is possible to replace the space average of a quantity with infinite time average along the trajectory for any starting configuration instead in the TDL different starting point gives different result if the temperature is less than the critical temperature. This is only possible in TDL and makes the system no longer invariant to its symmetries (in CW a symmetry is given by flipping all neurons). This is something we want to transport into a neural network to get different results with different initial conditions.

The last question about the Curie-Weiss model is how it relates to the neural network we defined previously. To understand this connection, we consider a network with a single stored pattern x, $h = 0$ and the interaction matrix J built with definition 2.1.4. We get the following Hamiltonian

$$\mathcal{H}_{N,J}(\xi) = -\sum_{i,j} J_{i,j}\xi_i\xi_j = -\frac{1}{N}\sum_{i\neq j}(\xi_i x_i)(\xi_j x_j) = -Nm_{N,x}(\xi)^2$$

where $m_{N,x}(\xi) = \frac{1}{N}\sum_i \xi_i x_i$ **Mattis magnetization**
So it is equivalent to a Curie-Weiss model where $J = 2$.
Given an initial state, we can use as output the Mattis magnetization after a sufficient number of updates. We can affirm that with a large enough number of neurons (so we can use TDL as a good approximation) and a low temperature, the system will evolve to show a magnetization close to the chosen pattern. It is important to note that there are two main attractors in this type of system, one related to the actual pattern ($m_{N,x}(\xi) \approx 1$) and the other with its negative ($m_{N,x}(\xi) \approx -1$), so this simple network gives as output the pattern or its negative, depending on which is "closer" to the initial configuration.

2.2.2 Hopfield model

n the previous pages we analyzed a network that stores one pattern, now we want to understand networks with more patterns, so we try again to express the Hamiltonian as a function of some order parameter.

Definition 2.2.7 (Mattis magnetizzation)**.** Given the patterns $(x^1, \dots, x^P)$ and a state ξ we define:

$$m_{N,\mu}(\xi) = \frac{1}{N}\sum_i \xi_i x_i^\mu \quad \forall \mu \in \{1, \dots, P\}$$

This definition allows us to write the Hamiltonian with J defined by the Hebb's rule (see definition 2.1.4) as:

$$\mathcal{H}_{N,X}(\xi) = \frac{1}{2}\sum_{i,j}\frac{1}{N}(1-\delta_{i,j})\sum_{\mu=1}^{P} x_i^\mu x_j^\mu \xi_i \xi_j = -\frac{N}{2}\sum_\mu m_{N,\mu}(\xi)^2$$

As we defined before $P \leq N$ and $X \in \mathbb{R}^{N\times P}$ is the design matrix. As before we are interested in the analysis of the system in the TDL and we can have two cases:

- **low load regime** where $\lim_{N\to\infty} \frac{P}{N} = 0$
- **high load regime** where $\lim_{N\to\infty} \frac{P}{N} > 0$

We will analyze the low load regime. We already know that the free energy can be expressed as the logarithm of the partition function, but with this method we do not remove the dependence of the free energy on the patterns X. To remove this dependence we need to define the concept of quenched free energy:

Definition 2.2.8 (Quenched free energy)**.** We define the quenched free energy as the average over all possible pattern realizations of the free energy:

- $Z_{N,\beta,X} = \sum_\xi e^{-\beta \mathcal{H}_{N,X}(\xi)}$ **partition function**
- $f_{N,\beta,X} = -\frac{1}{N\beta}\log\left(Z_{N,\beta,X}\right)$ **intensive free energy**
- $f_{\beta,X} = \lim_{N\to\infty} f_{N,\beta,X}$
- $f^Q_{N,\beta} = \mathbb{E}\left[f_{N,\beta,X}\right] = -\frac{1}{\beta N}\frac{1}{2^{N\times P}}\sum_{X\in\mathbb{R}^{N\times P}}\log(Z_{N,\beta,X})$ **intensive quenched free energy**
- $f^Q_\beta = \lim_{N\to\infty} f^Q_{N,\beta}$ **intensive quenched free energy in TDL**

In the previous definition it is important that the averaging is done before the TDL, in fact it is not true that we can invert these operations.

Remark. To compute the TDL, we use the Fourier representation of the Dirac delta, i.e:

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{ik(x-y)} f(y) dy dk$$

Remark. To compute the TDL, we use the Laplace's method for approximating integrals:

$$\int_a^b \dots \int_a^b e^{Nf(z)} dz \underset{N\to\infty}{\approx} \left(\frac{2\pi}{N|-Hf(z^\star)|}\right)^{\frac{P}{2}} e^{Nf(z^\star)}$$

where f is a holomorphic function, $z^\star \in \mathbb{C}^P$ with $P << N$ (so we can only use it in the low load regime) is a saddle point for f and the path of the integral passes through $z^\star$.
We use this result with $a = -\infty$ and $b = +\infty$
It is noted that the approximation is over the difference, so the difference between two members goes to 0 when $N \to \infty$.

To compute the quenched TDL, we start from an alternative formulation for the partition function obtained with the Fourier representation of the Dirac delta:

$$\begin{aligned}
Z_{\beta,N,X} &= \sum_\xi e^{-\beta \mathcal{H}_{N,X}(\xi)} = \sum_\xi \prod_\mu e^{\frac{\beta N}{2} m_{N,\mu}(\xi)^2} \\
&= \sum_\xi \prod_\mu \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \frac{1}{2\pi} e^{ik_\mu (m_{N,\mu}(\xi) - y_\mu)} e^{\frac{\beta N}{2} y_\mu^2} dy_\mu dk_\mu \\
&= \sum_\xi \prod_\mu \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \left(\frac{dy_\mu dk_\mu}{2\pi}\right) \exp\left(\frac{i}{N}\sum_j k_\mu x_j^\mu \xi_j - i\sum k_\mu y_\mu + \frac{\beta N}{2}\sum y_\mu^2\right)
\end{aligned}$$

We can substitute k_μ with $\frac{k_\mu}{N}$ and multiply the differential by N

$$= \sum_\xi \prod_\mu \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \left(\frac{N dy_\mu dk_\mu}{2\pi}\right) \exp\left(i\sum_j k_\mu x_j^\mu \xi_j - iN k_\mu y_\mu + \frac{\beta N}{2} y_\mu^2\right)$$

This formulation allows us to sum over the configurations because we have linear dependence on the argument of the exponential, hence recalling that $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$ we obtain:

$$\begin{aligned}
&= \int_{\mathbb{R}^{P\times 2}} \prod_\mu \left(\frac{N dy_\mu dk_\mu}{2\pi}\right) \exp\left(-iN\sum_\mu k_\mu y_\mu + \frac{\beta N}{2}\sum_\mu y_\mu^2\right) \sum_\xi \prod_j \prod_\mu \exp\left(ik_\mu x_j^\mu \xi_j\right) \\
&\overset{1}{=} \int_{\mathbb{R}^{P\times 2}} \prod_\mu \left(\frac{N dy_\mu dk_\mu}{2\pi}\right) \exp\left(-iN\sum_\mu k_\mu y_\mu + \frac{\beta N}{2}\sum_\mu y_\mu^2\right) \prod_j \sum_{\xi_j} \prod_\mu \exp\left(ik_\mu x_j^\mu \xi_j\right) \\
&\overset{2}{=} \int_{\mathbb{R}^{P\times 2}} \prod_\mu \left(\frac{N dy_\mu dk_\mu}{2\pi}\right) \exp\left(-iN\sum_\mu k_\mu y_\mu + \frac{\beta N}{2}\sum_\mu y_\mu^2\right) \prod_j \left(2\cos\left(\sum_\mu ik_\mu x_j^\mu\right)\right) \\
&= \int_{\mathbb{R}^{P\times 2}} \prod_\mu \left(\frac{N dy_\mu dk_\mu}{2\pi}\right) \exp\left(\sum_j \log\left(2\cos\left(\sum_\mu k_\mu x_j^\mu\right)\right) - iN\sum_\mu k_\mu y_\mu + \frac{\beta N}{2}\sum_\mu y_\mu^2\right)
\end{aligned} \tag{1}$$

In equivalence 1, we used following state:

$$\begin{aligned}
\sum_\xi \prod_j s_j(\xi_j) &= \sum_{\xi_N} \sum_{\xi_1,\dots,\xi_{N-1}} s_N(\xi_N) \prod_{j=1}^{N-1} s_j(\xi_j) = \sum_{\xi_N} s_N(\xi_N) \sum_{\xi_1,\dots,\xi_{N-1}} \prod_{j=1}^{N-1} s_j(\xi_j) \\
\sum_\xi \prod_j s_j(\xi_j) &= \sum_{\xi_N} s_N(\xi_N) \sum_{\xi_1,\dots,\xi_{N-1}} \prod_{j=1}^{N-1} s_j(\xi_j) = \dots = \prod_j \sum_{\xi_j} s_j(\xi_j)
\end{aligned}$$

In equivalence 2, we used $\forall j (\xi_j = \pm 1)$

Now we can use the Laplace's method to approximate the integrals, at first, we compute the integral with reference to $y = (y_1, \dots, y_P)$. In particular we analyze the problem:

$$\int_{\mathbb{R}^P} dy \exp\left(N\left(-i\sum_\mu k_\mu y_\mu + \frac{\beta}{2}\sum_\mu y_\mu^2\right)\right)$$

$$\begin{aligned}
&f(y) \coloneqq -i\sum_\mu k_\mu y_\mu + \frac{\beta}{2}\sum_\mu y_\mu^2 \\
&\nabla_y f(y) = \left(-ik_\mu + \beta y_\mu\right)_\mu \\
&y^\star = \left(\frac{ik_\mu}{\beta}\right)_\mu \\
&f(y^\star) = -i\sum_\mu k_\mu \frac{ik_\mu}{\beta} + \frac{\beta}{2}\sum_\mu \left(\frac{ik_\mu}{\beta}\right)^2 = \frac{1}{\beta}\sum_\mu \frac{k_\mu^2}{2} \\
&H_y f(y)\big|_{y^\star} = \left(\beta\delta_{\mu,\nu}\right)_{\mu\nu}\big|_{y^\star} = \beta\mathbb{1}_P \\
&\int_{\mathbb{R}^P} dy \exp\left(N\left(-i\sum_\mu k_\mu y_\mu + \frac{\beta}{2}\sum_\mu y_\mu^2\right)\right) \underset{N\to\infty}{\approx} \left(-\frac{2\pi}{N\beta}\right)^{\frac{P}{2}} \exp\left(N\sum_\mu \frac{k_\mu^2}{2\beta}\right)
\end{aligned} \tag{2}$$

Now we can replace the first integral in eq. (1) obtaining:

$$(1) \underset{N\to\infty}{\approx} \left(-\frac{N}{2\pi\beta}\right)^{\frac{P}{2}} \int_{\mathbb{R}^{P\times 2}} dk \exp\left(\sum_j \log\left(2\cos\left(\sum_\mu k_\mu x_j^\mu\right)\right) + N\sum_\mu \frac{k_\mu^2}{2\beta}\right) \tag{3}$$

Now we can use the Laplace's method to approximate the integrals with reference to $k = (k_1, \dots, k_P)$. In particular we analyze the problem:

$$\int_{\mathbb{R}^{P\times 2}} dk \exp\left(N\left(\frac{1}{N}\sum_j \log\left(2\cos\left(\sum_\mu k_\mu x_j^\mu\right)\right) + \sum_\mu \frac{k_\mu^2}{2\beta}\right)\right)$$

$$\begin{aligned}
&f(k) = \frac{1}{N}\sum_j \log\left(2\cos\left(\sum_\mu k_\mu x_j^\mu\right)\right) + \sum_\mu \frac{k_\mu^2}{2\beta} \\
&\nabla f(k) = \left(-\frac{1}{N}\sum_j \tan\left(\sum_\mu z_\mu x_j^\mu\right) x_j^\nu + \frac{k_\nu}{\beta}\right)_\nu
\end{aligned}$$

$k^\star$ satisfies:

$$\begin{aligned}
&\forall\nu \left(k_\nu^\star = \frac{\beta}{N}\sum_j \tan\left(\sum_\mu k_\mu^\star x_j^\mu\right) x_j^\nu\right) \\
&f(k^\star) = \frac{1}{N}\sum_j \log\left(2\cos\left(\sum_\mu k_\mu^\star x_j^\mu\right)\right) + \sum_\mu \frac{\left(k_\mu^\star\right)^2}{2\beta}
\end{aligned} \tag{4}$$

Now we can replace the approximation in eq. (3) obtaining:

$$(3) \underset{N\to\infty}{\approx} \left(\frac{1}{\beta|Hf(k^\star)|}\right)^{\frac{P}{2}} \exp\left(\sum_j \log\left(2\cos\left(\sum_\mu k_\mu^\star x_j^\mu\right)\right) + N\sum_\mu \frac{\left(k_\mu^\star\right)^2}{2\beta}\right)$$

We would like, using eq. (2), to write eq. (4) and $f(k^\star)$ respect $y^\star := \frac{i}{\beta}k^\star$:

$$\begin{aligned}
&\forall\nu \left(-i\beta y_\nu^\star = \frac{\beta}{N}\sum_j \tan\left(-i\beta\sum_\mu y_\mu^\star x_j^\mu\right) x_j^\nu \right) \\
&\iff \forall\nu \left(y_\nu^\star = \frac{1}{N}\sum_j \tanh\left(\beta\sum_\mu y_\mu^\star x_j^\mu\right) x_j^\nu \right) \qquad (5)\\
&f(k^\star) = \frac{1}{N}\sum_j \log\left(2\cos\left(-i\beta\sum_\mu y_\mu^\star x_j^\mu\right)\right) - \sum_\mu \frac{\beta\left(y_\mu^\star\right)^2}{2} \\
&\quad = \frac{1}{N}\sum_j \log\left(2\cosh\left(\beta\sum_\mu y_\mu^\star x_j^\mu\right)\right) - \sum_\mu \frac{\beta\left(y_\mu^\star\right)^2}{2}
\end{aligned}$$

now we can express eq. (3) respect $y^\star$:

$$(3) \underset{N\to\infty}{\approx} \left(\frac{1}{\beta|Hf(k^\star)|}\right)^{\frac{P}{2}} \exp\left(\sum_j \log\left(2\cosh\left(\beta\sum_\mu y_\mu^\star x_j^\mu\right)\right) - N\sum_\mu \frac{\beta\left(y_\mu^\star\right)^2}{2}\right)$$

Now we are ready to calculate the TDL of the free energy:

$$\begin{aligned}
f_{\beta,X} &= \lim_{N\to\infty} f_{N,\beta,X} = \lim_{N\to\infty} -\frac{1}{\beta N}\log(Z_{N,\beta,X}) \\
&= \lim_{N\to\infty} -\frac{1}{\beta N}\sum_j \log\left(2\cosh\left(\beta\sum_\mu y_\mu^\star x_j^\mu\right)\right) + \sum_\mu \frac{(y_\mu^\star)^2}{2}
\end{aligned}$$

Theorem 2.3 (*Quenched free energy low load regime*)

The quenched free energy of the Hopfield model in TDL is:

$$f_\beta^Q = \frac{1}{2}\sum_\mu (m_\mu^*)^2 - \frac{1}{\beta}\mathbb{E}\left[\log\left(2\cosh\left(\beta\sum_\nu m_\nu^* x^\nu\right)\right)\right] \qquad (2.9)$$

where m^* satisfy the self consistent equations:

$$m_\mu^* = \mathbb{E}\left[x^\mu \tanh\left(\beta\sum_\nu m_\nu^* x^\nu\right)\right] \qquad (2.10)$$

remark: this notation is justified by eq. (2.8) and eq. (5).

We can write the self consistent equation in a vector notation as:

$$m = \mathbb{E}(X\tanh(\beta m\cdot X))$$

Moreover, if there is a single pattern to store, we obtain the self-consistent equation for the Curie-Weiss model eq. (2.8) because the hyperbolic tangent is an odd function.

Lemma 2.2.1. *If $\beta < 1$ then $m = 0$ is the only solution for eq. (2.10)*

Proof.

$$\begin{aligned}
m^2 &= \sum_\mu m_\mu m_\mu = \sum_\mu m_\mu \mathbb{E}(x^\mu\tanh(\beta m\cdot X)) = \mathbb{E}(m\cdot X\tanh(\beta m\cdot X)) \\
&= \mathbb{E}(|m\cdot X|\tanh(\beta|m\cdot X|)) \le \mathbb{E}(|m\cdot X|\beta|m\cdot X|) \\
&= \beta\mathbb{E}(\sum_{\mu,\nu}(m_\mu x^\mu)(m_\nu x^\nu)) = \beta\sum_{\mu,\nu} m_\mu m_\nu \mathbb{E}(x^\mu x^\nu)
\end{aligned}$$

in the TDL the average of the dot product of two different pattern is 0 then we can replace $\mathbb{E}(x^\mu x^\nu)$ with $\delta_{\mu,\nu}$

$$= \beta \sum_{\mu,\nu} m_\mu m_\nu \delta_{\mu,\nu} = \beta \sum_{\mu} m_\mu m_\mu = \beta m^2$$

that is true if and only if $m = 0$.

□

The previous lemma states that if the temperature is greater than $T_c = 1$, the only solution is $m = 0$, so no pattern is obtained, similar to the Curie-Weiss model.
On the other hand, if $\beta < 1$, i.e. the temperature is less than the critical value, the self-consistent equation has more solutions.

Lemma 2.2.2. *If $\beta > 1$ then a solution of eq. (2.10) can be expressed as:*

$$m^n = m_n(1, \dots, 1, 0, \dots, 0)$$

where m^n contains n ones and $P - n$ zeros and $m_n > 0$.

Proof. then let $\mu > n$ then:

$$(m^n)_\mu = 0 = \mathbb{E}(x^\mu)\mathbb{E}(\tanh(\beta m_n \sum_{\nu \le n} x^\nu)) = \mathbb{E}(x^\mu \tanh(\beta m_n \sum_{\nu \le n} x^\nu)) = \mathbb{E}(x^\mu \tanh(\beta m^n X))$$

where the third equality is the true only if $\mu > n$ and is the independence of the random variables x^μ.
Let $\mu \le n$ then m^n is a solution if the line $y = m_n$ intersects the function

$$y = \mathbb{E}(x^\mu \tanh(\beta m^n X)) = f(m_n)$$

.
At zero both functions have the same value and for $m_n \to \infty$ the second one goes to one while the line goes to infinity, then if $f(m_n) > m_n$ for some $m_n > 0$ a solution exists.
We will prove this by showing that $f'(0) > 1$.

$$\begin{aligned} f'(0) &= \frac{d\mathbb{E}(x^\mu \tanh(\beta m^n X))}{dm_n}\Big|_{m_n=0} = \mathbb{E}(x^\mu \frac{d\tanh(\beta m^n X)}{dm_n}\Big|_{m_n=0}) \\ &= \mathbb{E}(x^\mu(1 - \tanh(\beta m^n X)^2)\beta X\big|_{m_n=0}) = \beta\mathbb{E}((x^\mu)^2 + \sum_{\nu \neq \mu} x^\mu x^\nu) = \beta > 1 \end{aligned}$$

□

With the previous lemma we showed that there are many more false solutions for the eq. (2.10), every permutation of a solution is still a solution, so there are exponentially many solutions w.r.t. the number of patterns. Not all of these solutions are minimum points for the free energy and are not configurations reached by the dynamics of the system.

Lemma 2.2.3. *Given a solution of the eq. (2.10) expressed as $m^n = m_n(1, \dots, 1, 0, \dots, 0)$ the Hessian of the free energy is:*

$$Hf(m_n)_{\mu.\nu} = \begin{cases} 1 - \beta + \beta Q & \mu = \nu \\ 0 & \mu \neq \nu \wedge (\mu > n \vee \nu > n) \\ \beta R_{\mu,\nu} & \mu \neq \nu \wedge \mu, \nu \le n \end{cases}$$

where $Q = \mathbb{E}(\tanh^2(\beta m^n X))$, $R_{\mu,\nu} = \mathbb{E}(x^\mu x^\nu \tanh^2(\beta m^n X))$.

Proof. The free energy is $f(m) = \frac{1}{2}\sum_\mu (m_\mu)^2 - \frac{1}{\beta}\mathbb{E}(\log(2\cosh(\beta m X)))$ then:

$$(\nabla f(m))_\mu = m_\mu - \mathbb{E}(\tanh(\beta m X)x^\mu)$$

$$\begin{aligned}(Hf(m))_{\mu,\nu} &= \delta_{\mu,\nu} - \beta\mathbb{E}((1-\tanh^2(\beta m X))x^\mu x^\nu) = \delta_{\mu,\nu} - \beta\mathbb{E}(x^\mu x^\nu) + \beta\mathbb{E}(\tanh^2(\beta m X)x^\mu x^\nu) \\ &= \delta_{\mu,\nu} - \beta\delta_{\mu,\nu} + \beta\mathbb{E}(\tanh^2(\beta m X)x^\mu x^\nu)\end{aligned}$$

The thesis follows because we can factor the expectation in the second case and get zero. □

Furthermore, after proving the lemma, we can state that

$$(Hf(0))_{\mu,\nu} = \begin{cases} 1-\beta & \mu = \nu \\ 0 & \mu \neq \nu \end{cases}$$

So $m = 0$ is the only minimum if $\beta < 1$.
Now let us analyze whether m^1 is a minimum:
$Hf(m^1)$ is a diagonal matrix, so it is definitely positive if its diagonal values are greater than zero.

$$(Hf(m^1))_{\mu,\mu} = 1 - \beta + \beta\mathbb{E}(\tanh^2(\beta m^1 X)) = 1 - \beta + \beta\tanh^2(\beta m_1)) = 1 - \beta(1 - \tanh^2(\beta m_1)))$$

where the second equality is true because the hyperbolic tangent is odd and the patterns have only $-1, 1$ as possible values. To prove that $(Hf(m^1))_{\mu,\mu}$ is positive we start with the eq. (2.10) for m^1:

$$m_1 = (m^1)_1 = \mathbb{E}\left[x^1 \tanh\left(\beta m^1 X\right)\right] = \tanh(\beta m_1)$$

Now deriving both members with respect to β we get

$$\begin{aligned}&\frac{\partial m_1}{\partial \beta} = \frac{\partial \tanh(\beta m_1)}{\partial \beta} = (1-\tanh^2(\beta m_1))(m_1 + \beta\frac{\partial m_1}{\partial \beta}) \quad \iff \\ &\frac{\partial m_1}{\partial \beta}(1-\tanh^2(\beta m_1)) = m_1(1-\tanh^2(\beta m_1)) \quad \iff \\ &(1-\tanh^2(\beta m_1)) = \frac{m_1}{\frac{\partial m_1}{\partial \beta}}(1-\tanh^2(\beta m_1)) > 0\end{aligned}$$

The last inequality holds because $\beta > 1$ when m^1 is a solution.
Thus we obtain that all pure solutions are points of minima for the free energy.
It can also be shown that:

- m^n is a minimum if n is odd and $f^Q_\beta(m^1) < f^Q_\beta(m^3) < \dots$
- m^n is not a minimum if n is even

The energy landscape contains many minima, with this perspective we can see the positive influence of the noise, in fact it prevents the network to fall into a local minima of a spurious and never escape from it.

Chapter 3

Continuous Hopfield Neural Networks

Because a Hopfield neural network works on discrete input, its process is discontinuous. This can make it difficult to use in a continuous problem, such as classifying photos of cats and dogs, or in a feed-forward neural network.
In this chapter we introduce continuous Hopfield neural networks, following on from the article [3]. These neural networks accept real values and update them with continuous and derivable transformations. This way it's possible to use them in continuous problems and train the neural networks to get good patterns for a data set.

3.1 Derivation

In chapter 2 we define a discrete Hopfield neural network as a predictor that receives an input in $\{-1,+1\}^d$ and recovers a pattern in the same space.
We denoted the patterns to be recovered as $x_1,\dots,x_P \in \{-1,+1\}^N$ or using the matrix $X \in \mathbb{R}^{N\times P}$ in general, the current state as ξ and a sequence of states as $(\xi^t)_t$.
We have defined the energy $E(\xi,X) = -\sum_i^P F\left(\xi^T x_i\right)$ and the asynchronous update:

$$\begin{aligned}\xi_j^{\text{new}} &\coloneqq \operatorname{sgn}\left[\sum_{i=1}^{P} F\left(x_{ij}+\sum_{l\neq j} x_{il}\xi_l\right) - F\left(-x_{ij}+\sum_{l\neq j} x_{il}\xi_l\right)\right] \\ &= \operatorname{sgn}\left[-E\left(\xi \mid \xi_j = 1, X\right) + E\left(\xi \mid \xi_j = -1, X\right)\right]\end{aligned}$$

The mean value theorem gives a value $v \in [-1,1]$ such that:

$$E\left(\xi \mid \xi_j = 1, X\right) - E\left(\xi \mid \xi_j = -1, X\right) = 2\frac{\partial E\left(\xi, X\right)}{\partial \xi_j}\left(\xi \mid \xi_j = v, X\right) \tag{3.1}$$

To derive the continuous Hopfield neural network we use the interaction function $F : x \mapsto e^{\beta x}$, where β^{-1} is called **temperature**, a positive value. Thus the updating formulation can be written from

definition 2.1.2 as

$$\begin{aligned}
\xi_j^{\text{new}} &= \operatorname{sgn}\left[-E\left(\xi \mid \xi_j = 1, X\right) + E\left(\xi \mid \xi_j = -1, X\right)\right] \\
&= \operatorname{sgn}\left[\exp\left(\ln\left(\sum_{i=1}^{P} \exp\left(\beta\left(\xi \mid \xi_j = 1\right)^T x_i\right)\right)\right) - \exp\left(\ln\left(\sum_{i=1}^{P} \exp\left(\beta\left(\xi \mid \xi_j = -1\right)^T x_i\right)\right)\right)\right] \\
&= \operatorname{sgn}\left[\beta^{-1} \ln\left(\sum_{i=1}^{P} \exp\left(\beta\left(\xi \mid \xi_j = 1\right)^T x_i\right)\right) - \beta^{-1} \ln\left(\sum_{i=1}^{P} \exp\left(\beta\left(\xi \mid \xi_j = -1\right)^T x_i\right)\right)\right] \qquad (3.2) \\
&= \operatorname{sgn}\left[2\frac{\partial E\left(\xi, X\right)}{\partial \xi_j}\left(\xi \mid \xi_j = v, X\right)\right] = \operatorname{sgn}\left[\frac{\sum_{i=1}^{P} x_{ij} \exp\left(\beta\left(\xi \mid \xi_j = v\right)^T x_i\right)}{\sum_{k=1}^{P} \exp\left(\beta\left(\xi \mid \xi_j = v\right)^T x_k\right)}\right] \\
&= \operatorname{sgn}\left[\left[X \operatorname{softmax}\left(X^T\left(\beta\xi \mid \xi_j = v\right)\right)\right]_j\right] \qquad (3.3)
\end{aligned}$$

Hence, in general $\xi^{\text{new}} = \operatorname{sgn}\left[X \operatorname{softmax}\left(X^T\left(\beta\xi \mid \xi_j = v\right)\right)\right]$

In view of the above, let us define a Continuous Hopfield Neural Network according to the following definitions:

Definition 3.1.1 (Energy)**.** We define the energy using eq. (3.2) as

$$E_\beta\left(\xi, X\right) = -\beta^{-1} \ln\left(\sum_{i=1}^{P} \exp\left(\beta \xi^T x_i\right)\right) + \frac{1}{2}\xi^T\xi + \text{const} = -\operatorname{lse}\left(\beta, X^T\xi\right) + \frac{1}{2}\xi^T\xi + \text{const}$$

we observe that: $\left(\xi \mid \xi_j = 1\right)^T\left(\xi \mid \xi_j = 1\right) = \left(\xi \mid \xi_j = -1\right)^T\left(\xi \mid \xi_j = -1\right)$.
To achieve more specific properties, we say:

$$E_\beta\left(\xi, X\right) = -\operatorname{lse}\left(\beta, X^T\xi\right) + \frac{1}{2}\xi^T\xi + \beta^{-1}\ln P + \frac{1}{2}M^2 \qquad (3.4)$$

where $M^2 = \max\|x_i\|^2$

Definition 3.1.2 (updating)**.** We define the update using eq. (3.3) as

$$\xi^{\text{new}} = X \operatorname{softmax}\left(\beta X^T \xi\right) \qquad (3.5)$$

3.2 Properties

A first simple property is:

$$E_\beta\left(\xi, X\right) = -\beta^{-1} \ln\left[\frac{1}{P}\sum_{i=1}^{P} \exp\left(-\frac{1}{2}\beta\|\xi - x_i\|^2\right) \exp\left(-\frac{1}{2}\beta\left(M^2 - \|x_i\|^2\right)\right)\right] \qquad (3.6)$$

This formula emphasises that the energy uses the density of P symmetric Gaussian variables chosen with equal probability but with different centres.
Using eq. (3.6) it's easy to prove the following lemma:

Lemma 3.2.1 (energy lower bound)**.** *from Lemma.A1 in [3]*
For all states, the energy is formally non-negative:

$$\forall \xi \in \mathbb{R}^N \left(E_\beta\left(\xi, X\right) \geq 0\right)$$

Proof. For all $\xi \in \mathbb{R}^N$, it holds that:

$$\begin{aligned} E_\beta(\xi, X) \geq 0 &\iff \frac{1}{P}\sum_{i=1}^{P} \exp\left(-\frac{1}{2}\beta\|\xi - x_i\|^2\right) \exp\left(-\frac{1}{2}\beta\left(M^2 - \|x_i\|^2\right)\right) \leq 1 \\ &\impliedby \forall i \in \{1,\dots,P\} \left(\exp\left(-\frac{1}{2}\beta\|\xi - x_i\|^2\right) \exp\left(-\frac{1}{2}\beta\left(M^2 - \|x_i\|^2\right)\right) \leq 1\right) \\ &\iff \forall i \in \{1,\dots,P\} \left(\left(-\frac{1}{2}\beta\|\xi - x_i\|^2\right) + \left(-\frac{1}{2}\beta\left(M^2 - \|x_i\|^2\right)\right) \leq 0\right) \\ &\iff \forall i \in \{1,\dots,P\} \left(\|\xi - x_i\|^2 + \left(M^2 - \|x_i\|^2\right) \geq 0\right) \end{aligned}$$

□

A second simple property is that the vector **softmax** in eq. (3.5) is a probability distribution. Let $p \coloneqq (p_1,\dots,p_P) = \text{softmax}\left(\beta X^T \xi\right)$, the new state ξ^{new} is a convex combination of the patterns in X:

$$\xi^{\text{new}} = Xp = p_1 x_1 + \cdots + p_P x_P$$

With $\langle x_1,\dots,x_P\rangle$ the set $\{v|\exists p_1,\dots,p_P\,(v = p_1x_1 + \cdots + p_Px_P)\}$.
Hence the sequence $\xi^1, \xi^2, \dots \in \langle x_1,\dots,x_P\rangle$. Now we compute an upper bound for the energy, whether or not it is temperature dependent.

Lemma 3.2.2 (energy upper bound)**.** *from Lemma.A1 in [3]*
For all $\xi \in \langle x_1,\dots,x_P\rangle$, it holds that:

i) $E_\beta(\xi, X) \leq 2M^2$

ii) $E_\beta(\xi, X) \leq \beta^{-1}\ln P + \frac{1}{2}M^2$

Proof. exp is a convex function over $\mathbb{R}$ so, for all $v_1,\dots,v_P \in \mathbb{R}$, it holds that:

- $\forall p_1,\dots,p_P \geq 0 \left(p_1 + \cdots + p_P = 1 \implies \sum_{i=1}^P p_i \exp(v_i) \geq \exp\sum_{i=1}^P p_i v_i\right)$
- $\forall p_1,\dots,p_P \geq 0 \left(p_1 + \ldots + p_P = 1 \implies \max_i \exp(v_i) \geq \exp\sum_{i=1}^P p_i v_i\right)$

Proof of the first relation:

$$\begin{aligned} E_\beta(\xi, X) &= -\beta^{-1}\ln\left(\sum_{i=1}^{P}\exp\left(\beta X^T\xi\right)\right) + \frac{1}{2}\xi^T\xi + \beta^{-1}\ln P + \frac{1}{2}M^2 \\ &= -\beta^{-1}\ln\left(\frac{1}{P}\sum_{i=1}^{P}\exp\left(\beta\xi^T x_i\right)\right) + \frac{1}{2}\xi^T\xi + \frac{1}{2}M^2 \\ &\leq -\beta^{-1}\left(\frac{1}{P}\sum_{i=1}^{P}\beta\xi^T x_i\right) + \frac{1}{2}\xi^T\xi + \frac{1}{2}M^2 \\ &= -\xi^T m_x + \frac{1}{2}\xi^T\xi + \frac{1}{2}M^2 \leq \|\xi\|\,\|m_x\| + \frac{1}{2}\|\xi\|^2 + \frac{1}{2}M^2 \leq 2M^2 \end{aligned}$$

using $m_x \coloneqq \frac{1}{P}\sum_{i=1}^P x_i$
Proof of the second relation:

Let $p_1, \ldots, p_P \geq 0$ be such that $p_1 + \cdots + p_P = 1$ and $\xi = \sum_{i=1}^{P} p_i x_i$.

$$\begin{aligned} E_\beta(\xi, X) &= -\beta^{-1} \ln\left(\sum_{i=1}^{P} \exp\left(\beta \xi^T x_i\right)\right) + \frac{1}{2}\xi^T\xi + \beta^{-1}\ln P + \frac{1}{2}M^2 \\ &\leq -\beta^{-1} \ln\left(\max_i \exp\left(\beta \xi^T x_i\right)\right) + \frac{1}{2}\xi^T\xi + \beta^{-1}\ln P + \frac{1}{2}M^2 \\ &\leq -\beta^{-1} \sum_{i=1}^{P} p_i \beta \xi^T x_i + \frac{1}{2}\xi^T\xi + \beta^{-1}\ln P + \frac{1}{2}M^2 \\ &= -\frac{1}{2}\xi^T\xi + \beta^{-1}\ln P + \frac{1}{2}M^2 \leq \beta^{-1}\ln P + \frac{1}{2}M^2 \end{aligned}$$

□

3.2.1 Convergence

Before we talk about convergence, we observe some simple properties of the energy minimization problem. Now we will talk about the lse function:

Remark. from Lemma.22 in [3]
The Jacobian of softmax is positive semi-definite matrix.

$$\begin{aligned} \nabla_{v_i}\left[\text{softmax}(v)\right]_j = \nabla_{v_i}\frac{\exp v_j}{\sum_{k=1}^{P}\exp v_k} &= \frac{\left(\nabla_{v_i}\exp v_j\right)\sum_{k=1}^{P}\exp v_k - \exp v_j \nabla_{v_i}\sum_{k=1}^{P}\exp v_k}{\left(\sum_{k=1}^{P}\exp v_k\right)^2} \\ = \frac{\delta_{ij}\exp v_i \sum_{k=1}^{P}\exp v_k - \exp v_j \exp v_i}{\left(\sum_{k=1}^{P}\exp v_k\right)^2} &= \delta_{ij}\frac{\exp v_i}{\sum_{k=1}^{P}\exp v_k} - \frac{\exp v_j \exp v_i}{\left(\sum_{k=1}^{P}\exp v_k\right)^2} \end{aligned}$$

Hence, if we define $p(v) = \text{softmax}(v)$:

$$\nabla_v p(v) = \text{diag}(p(v)) - p(v)p(v)^T$$

Proof that the Jacobian of softmax has eigenvalue zero with eigenvector $\mathbb{1}$:

$$\nabla_v p(v)\mathbb{1} = (\text{diag}(p(v)) - p(v)p(v)^T)\mathbb{1} = p(v) - p(v) = 0$$

Furthermore $\forall h \in \mathbb{R}^P$ holds:

$$\begin{aligned} h^T\nabla_v p(v) h = h^T(\text{diag}(p(v)) - p(v)p(v)^T)h &= \left(\sum_{k=1}^{P} p(v)_k h_k^2\right) - \left(\sum_{k=1}^{P} p(v)_k h_k\right) \\ = \left(\sum_{k=1}^{P} p(v)_k h_k^2\right)\left(\sum_{k=1}^{P} p(v)_k\right) - \left(\sum_{k=1}^{P} p(v)_k h_k\right) &\geq 0 \end{aligned}$$

where the last inequality is Cauchy-Schwartz applied to a, b such that $a_k = \sqrt{p(v)_k}h_k$ and $b_k = \sqrt{p(v)_k}$

Remark. The gradient of $\text{lse}(\beta, v)$ with respect to v is $\text{softmax}(\beta v)$.

$$\nabla_{v_i}\text{lse}(\beta, v) = \nabla_{v_i}\beta^{-1}\ln\sum_{k=1}^{P}\exp(\beta v_k) = \frac{\exp(\beta v_i)}{\sum_{k=1}^{P}\exp(\beta v_k)} = \left[\text{softmax}(\beta v)\right]_i$$

So the Hessian of $\text{lse}(\beta, v)$ respect v is positive semi-definite, and so $\text{lse}(\beta, v)$ is a convex function respecting v.

The objective function is a sum of convex and concave functions respecting ξ:

$$E_\beta(\xi, X) = E_\beta^-(\xi, X) + E_\beta^+(\xi, X) \tag{3.7}$$

where:

- $E_\beta^- (\xi, X) = -\operatorname{lse}\left(\beta, X^T \xi\right)$ a concave function
- $E_\beta^+ (\xi, X) = \frac{1}{2}\xi^T \xi + \beta^{-1} \ln P + \frac{1}{2} M^2$ a convex function

We observe that the objective function does not describe a convex problem, but the domain is the convex $G \coloneqq \langle x_1, \ldots, x_P \rangle$. We want to prove that the sequence defined in eq. (3.5) reduces the energy and converges to local minimum.

Proposition 3.1. *from Theorem.A1 in [3]*
The updating formula $\xi^{t+1} = X \operatorname{softmax}\left(X^T \xi^t\right)$ *solves the problem:*

$$\xi^{t+1} \in \operatorname{argmin}_{\xi \in G} \xi^T \nabla_{\xi^t} E_\beta^- \left(\xi^t, X\right) + E_\beta^+ (\xi, X)$$

Proof. This is a convex problem and we can solve it by finding the value where the gradient is zero:

$$\nabla_\xi \left(\xi^T \nabla_{\xi^t} E_\beta^- \left(\xi^t, X\right) + E_\beta^+ (\xi, X)\right) = \nabla_{\xi^t} E_\beta^- \left(\xi^t, X\right) + \nabla_\xi E_\beta^+ (\xi, X) = \nabla_{\xi^t} E_\beta^- \left(\xi^t, X\right) + \xi$$

We know that $\nabla_{\xi^t} E_\beta^- (\xi^t, X) = -X \operatorname{softmax}(\beta X^T \xi^t)$, so:

$$0 = \nabla_\xi \left(\xi^T \nabla_{\xi^t} E_\beta^- \left(\xi^t, X\right) + E_\beta^+ (\xi, X)\right) \iff \xi = X \operatorname{softmax}(\beta X^T \xi^t)$$

□

Remark. We can linearize the energy formulation eq. (3.7) only over the concave function:

$$f_\beta \left(\xi, \xi^t, X\right) \coloneqq E_\beta^+ (\xi, X) + E_\beta^- \left(\xi^t, X\right) + \left(\xi - \xi^t\right)^T \nabla_{\xi^t} E_\beta^- \left(\xi^t, X\right)$$

This function is convex and has other important properties.

Proposition 3.2 (The updating reduces the energy)**.** *from Theorem.A1 in [3]*

$$\forall t \in \mathbb{N}_0 \left(E_\beta \left(\xi^{t+1}, X\right) \leq E_\beta \left(\xi^t, X\right)\right)$$

Proof. We noticed that:

- $f_\beta (v, v, X) = E_\beta (v, X)$
- $\xi^{t+1} \in \operatorname{argmin}_{\xi \in G} f_\beta \left(\xi, \xi^t, X\right)$ from proposition 3.1
- $f_\beta (v, w, X) \geq E_\beta (v, X)$, because $E_\beta^- (v, X)$ is concave over v and so:
  $E_\beta^- (w, X) + (v - w)^T \nabla_w E_\beta^- (w, X) \geq E_\beta^- (v, X)$

So for all $t \in \mathbb{N}_0$, it holds that:

$$E_\beta \left(\xi^{t+1}, X\right) \leq f_\beta \left(\xi^{t+1}, \xi^t, X\right) \leq f_\beta \left(\xi^t, \xi^t, X\right) = E_\beta \left(\xi^t, X\right)$$

□

If the updating reduces the energy and the energy is non-negative, the sequence $\left(E_\beta \left(\xi^t, X\right)\right)_t$ converges to $E^\star$, which depends on the starting point.
Furthermore, the sequence of states $\left(\xi^t\right)_t$ is a compact set in G. So there are accumulation points (denoted as $\xi^\star$) for the sequence of states $\left(\xi^t\right)_t$ and for them the energy is $E^\star$.

Theorem 3.1 (*stationary points*)

from Theorem.A2 in [3]
For all states ξ*, then it holds that:*

- $\nabla_\xi E_\beta (\xi, X) = 0 \iff \xi = \xi^{new}$
- $E_\beta (\xi, X) = E_\beta \left(\xi^{new}, X\right) \iff \xi = \xi^{new}$

Proof. We observe that the convex problem $\operatorname{argmin}_v f_\beta(v, w, X)$ is the computation of the minimum of a quadratic form, in fact the Hessian of f over v is the identity:

$$\frac{\partial^2 f_\beta(v, w, X)}{(\partial v)^2} = I_d$$

so the solution is unique in $\mathbb{R}^d$.
Let ξ be such that $\nabla_\xi E_\beta(\xi, X) = 0$:

$$\nabla_v f_\beta(v, \xi, X)\big|_{v=\xi} = \nabla_v E_\beta^+(v, X)\big|_{v=\xi} + \nabla_\xi E_\beta^-(\xi, X) = \nabla_\xi E_\beta(\xi, X) = 0$$

so ξ is a valid solution and so is the unique solution, hence $\xi^{\text{new}} = \xi$.
Let ξ be such that $\xi^{\text{new}} = \xi$:

$$0 = \nabla_v f_\beta(v, \xi, X)\big|_{v=\xi} = \nabla_v E_\beta^+(v, X)\big|_{v=\xi} + \nabla_\xi E_\beta^-(\xi, X) = \nabla_\xi E_\beta(\xi, X)$$

so ξ is a stationary points, hence $\nabla_\xi E_\beta(\xi, X) = 0$.
Furthermore, is obviously that $E_\beta(\xi, X) = E_\beta(\xi^{\text{new}}, X)$ if $\xi = \xi^{\text{new}}$
Let ξ be such that $\xi^{\text{new}} \neq \xi$. Since the result of the convex problem is unique, then:

$$E_\beta\left(\xi^{\text{next}}, X\right) \leq f_\beta\left(\xi^{\text{next}}, \xi, X\right) < f_\beta(\xi, \xi, X) = E_\beta(\xi, X)$$

Hence, the statement of this theorem holds. □

Lemma 3.2.3. *Let a sequence* $(v_n)_n$ *be such that* $2 \leq |\operatorname{acc}\{v_n\}_n| < \infty$.

$$\exists v_1^\star, v_2^\star \in \operatorname{acc}\{v_n\}_n \; \exists (v_{n_i})_i \subseteq (v_n)_n \left(v_{n_i} \to v_1^\star \wedge v_{n_i+1} \to v_2^\star \wedge v_1^\star \neq v_2^\star\right)$$

Proof. Get a sequence $\left(v_{n_i^j}\right)_i$ for each accumulation point $v^j \in \operatorname{acc}\{v_n\}_n$.
Without loss of generality $\forall i, a\exists j, b\left(a \neq b \wedge n_i^a + 1 = n_j^b\right)$
Let $S \coloneqq \{v^j\}_j \times \{v^j\}_j$ be the set of couples of accumulation points.
Let $h : (a, b) \in S \to |\{i \in \mathbb{N} | n_i^a + 1 = n_i^b\}| \in \mathbb{N}_\infty$ be a counter of jumps from sub-sequence of v^a to sub-sequence of v^b. Clearly $\forall a\,(h(a, a) = 0)$ and $\sum_{a,b\in S} h(a, b) = \infty$.
Let a, b be such that $a \neq b$ and $h(a, b) = \infty$, so there exists a sub-sequence $\left(n_{i_k}^a\right)_k \subseteq (n_i^a)$ such that $\left(n_{i_k}^a + 1\right)_k \subseteq \left(n_i^b\right)_i$.
In this way $v_{n_{i_k}^a} \to v^a$ and $v_{n_{i_k}^a+1} \to v^b$ □

Theorem 3.2 (*accumulation points*)

from Theorem.A2 in [3]

$$|\operatorname{acc}\{\xi^t\}_t| < \infty \implies \operatorname{acc}\{\xi^t\}_t = \{\xi^\star\} \wedge \nabla_{\xi^\star} E_\beta(\xi^\star, X) = 0$$

Proof. **Ad absurdum** $|\operatorname{acc}\{\xi^t\}_t| \geq 2$.
Let $\xi_1^\star, \xi_2^\star$ be different accumulation points such that $\exists (\xi^{t_i})_i \subseteq (\xi^t)_t \left(\xi^{t_i} \to \xi_1^\star \wedge \xi^{t_i+1} \to \xi_2^\star\right)$ (see lemma 3.2.3).
Get 2 sub-sequences of $(\xi^t)_t$ over indexing $(n_i)_i$, $(m_i)_i$ such that:

- $m_i = n_i + 1$
- $\xi_{n_i} \to \xi_1^\star$, $i \to \infty$
- $\xi_{m_i} \to \xi_2^\star$, $i \to \infty$

For all $i \in \mathbb{N}_0$ hold $\xi_{m_i} = \xi_{n_i+1} = X \operatorname{softmax}\left(\beta X^T \xi_{n_i}\right)$, hence $\xi_2^\star = X \operatorname{softmax}\left(\beta X^T \xi_1^\star\right)$
We already know that $E^\star = E_\beta(\xi_1^\star, X) = E_\beta(\xi_2^\star, X)$
With theorem 3.1 it holds that $\xi_1^\star = \xi_2^\star$ ↯

It is proved that $\text{acc}\{\xi^t\}_t = \{\xi^\star\}$
The sub-sequence $\left(\xi^{t+1}\right)_t$ converges to $\xi^\star$.
Hence $\xi^\star = X\,\text{softmax}\left(\beta X^T \xi^\star\right)$ and finally $\nabla_\xi E_\beta\left(\xi^\star, X\right) = 0$ (using theorem 3.1). □

Corollary. *The theorem 3.2 also proves that the local energy minima are separated and that the sequence starting at ξ^0 converges to one of them.*

3.3 Analysis

We have proved that given a starting point ξ_0 the sequence defined by the updating rule converges to some minimum of the energy; the next question to analyze is whether the starting point has an influence on the point of convergence.
To study this, we write the updating rule as: $\xi^{\text{new}} = f(\xi)$. From a previous remark we can compute the Jacobian of f as:

$$J = \frac{\partial f(\xi)}{\partial \xi} = \beta X\left(\text{diag}(p(\beta X T\xi)) - p(\beta X T\xi)p(\beta X T\xi)^T\right)X^T = \beta J_s$$

From now on we write $p = p(\beta X T\xi)$ to simplify the notation.
We also write: $\mathbb{E}_p\left[f(x)\right] = \sum_i p_i f(x_i)$.

Lemma 3.3.1. $\text{Var}_p(x) = J_s$

Proof.

$$\begin{aligned}\text{Var}_p\left(x\right) &= \mathbb{E}_p\left[xx^T\right] - \mathbb{E}_p\left[x\right]\mathbb{E}_p\left[x\right]^T = X\mathbb{E}_p\left[\mathbb{1}\right]X^T - Xp(Xp)^T \\ &= X(\text{diag}(p))X^T - Xp(Xp)^T = X(\text{diag}(p) - pp^T)X^T = J_s\end{aligned}$$

where the last equality holds for a remark made in the previous section. □

The parameter β plays a central role in the convergence of the method; in fact, if its value is too high, the softmax result is similar to a uniform distribution for any given input, and this can cause the network to lose information about the input.

Definition 3.3.1. We define:

$$\begin{aligned} m_X &= \frac{1}{N}\sum x^i \quad \textbf{patterns mean} \\ \bar{m} &= \sum p_i x^i \quad \textbf{patterns weighted mean} \\ m^2_{\max} &= \max_i \left\|x^i - m_X\right\|^2 \end{aligned}$$

Proposition 3.3. *from lemma.A3 in [3]*

$$\|J\|_2 < \beta m^2_{\max}$$

So if $\beta m^2_{\max} < 1$, the sequence defined by the continuous Hopfield neural network converges to a fixed point for any given input.

Proof.

$$\begin{aligned}\|J_s\|_2 = \|\text{Var}_p\left(x\right)\|_2 = \left\|\mathbb{E}_p\left[(x-\bar{m})(x-\bar{m})^T\right]\right\|_2 &= \left\|\sum_i p_i (x^i - \bar{m})(x^i - \bar{m})^T\right\|_2 \\ \leq \sum_i p_i \left\|(x^i - \bar{m})(x^i - \bar{m})^T\right\|_2 = \sum_i p_i \left\|x^i - \bar{m}\right\|^2 \leq \sum_i p_i \left\|x^i - m_X\right\|^2 &\leq m^2_{\max}\end{aligned}$$

where in the penultimate inequality we used:

$$\min_m \sum_i p_i \left\|x^i - m\right\|^2 = \sum_i p_i \left\|x^i - \bar{m}\right\|^2$$

To prove the second part of the statement we can use the Banach fixed point theorem and note that the sequence defined by the network is inside the compact set $\langle x_1, \dots, x_P \rangle$ □

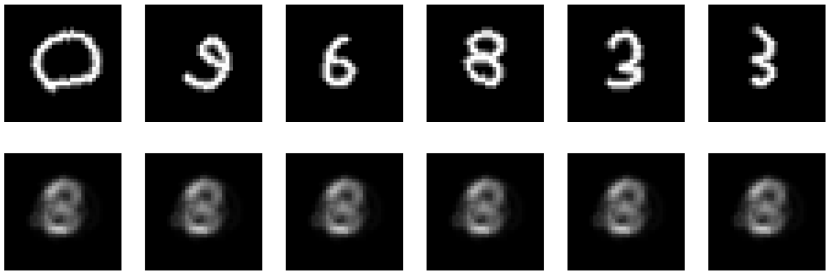


Figure 3.1: The figure shows how, with a small value of β, each input gives the same output.

Remark. $\|J_s\| = e_1$ where e_1 is the largest eigenvalue
Also holds:

$$\|J_s\| = e_1 = \mathrm{Tr}\,(J_s) - \sum_{i=2}^{n} e_i = \sum_i p_i \mathrm{Tr}\left((x^i - \bar{m})(x^i - \bar{m})^T\right) - \sum_{i=2}^{n} e_i = \sum_i p_i \left\|x^i - \bar{m}\right\|^2 - \sum_{i=2}^{n} e_i$$

Thus, the bound presented in the previous proposition is high when other eigenvalues are small.

The previous proposition states that we cannot properly use the Continuous Hopfield Neural Network if the temperature is too high, so it is necessary to formulate some condition for which the network has more than just one attractor point.

Definition 3.3.2 (separation)**.** Given a pattern matrix X, we define the pattern separation of the i-th pattern as:

$$\Delta_i = \min_{j \neq i} (x^i)^T x^i - (x^i)^T x^j$$

Furthermore, an input ξ is at least separated from x^i while being separated from x^j with $j \neq i$ if:

$$i = \mathrm{argmax}_k \min_{j \neq k} (\xi^T x^k - \xi^T x^j), \quad c = \min_{j \neq i} (\xi^T x^i - \xi^T x^j) > 0$$

Lemma 3.3.2. *from lemma.A4 in [3]*
If ξ is at least separated from x^i while being separated from all x^j with $j \neq i$, then:

$$\left\|x^i - f(\xi)\right\| \leq 2\epsilon M$$

where $M = \max_i \left\|x^i\right\|$ and $\epsilon = (N-1)e^{-\beta c}$

Proof.

$$\begin{aligned}
\left[\mathrm{softmax}(\beta X^T \xi)\right]_i &= \frac{\exp(\beta x^i \xi)}{\sum_j \exp(\beta x^j \xi)} = \frac{1}{1 + \sum_{j \neq i} \exp(\beta \xi^T x^j - \beta \xi^T x^i)} \geq \frac{1}{1 + \sum_{j \neq i} \exp(-\beta c)} \\
&= \frac{1}{1 + (N-1)\exp(-\beta c)} = 1 - \frac{(N-1)\exp(-\beta c)}{1 + (N-1)\exp(-\beta c)} \\
&= 1 - \frac{\epsilon}{1 + (N-1)\exp(-\beta c)} \geq 1 - \epsilon
\end{aligned}$$

given k $\neq i$ then:

$$\begin{aligned}\left[\text{softmax}(\beta X^T\xi)\right]_k &= \frac{\exp(\beta x^k\xi)}{\sum_j \exp(\beta x^j\xi)} = \frac{\exp(\beta\xi^T x^k - \beta\xi^T x^i)}{1+\sum_{j\neq i}\exp(\beta\xi^T x^j - \beta\xi^T x^i)} \\ &\leq \frac{\exp(-\beta c)}{1+\sum_{j\neq i}\exp(\beta\xi^T x^j - \beta\xi^T x^i)} \leq \exp(-\beta c) = \frac{\epsilon}{N-1}\end{aligned}$$

Now using the definition of f, we can prove the proposition:

$$\begin{aligned}\left\|x^i - f(\xi)\right\| &= \left\|x^i - X\,\text{softmax}(\beta X\xi)\right\| = \left\|(1-[\text{softmax}(\beta X^T\xi)]_i)x^i - \sum_{k\neq i} x^k[\text{softmax}(\beta X^T\xi)]_k\right\| \\ &\leq \left(1-\left[\text{softmax}(\beta X^T\xi)\right]_i\right)\left\|x^i\right\| + \sum_{k\neq i}\left[\text{softmax}(\beta X^T\xi)\right]_k\left\|x^k\right\| \\ &\leq \epsilon M + \sum_{k\neq i}\frac{\epsilon}{N-1}M = 2\epsilon M\end{aligned}$$

□

The previous lemma suggests that separate patterns and β not too small will make the sequence defined by the model closer to a particular pattern.

Definition 3.3.3. Given a pattern matrix X we define:

$$S_i = \left\{\xi \middle| ||\xi - x^i|| \leq \frac{1}{\beta NM}\right\}$$

Proposition 3.4. *from lemma.A5 in [3]*
If $\xi \in S_i$ and $\Delta_i \geq \frac{2}{\beta N} + \frac{1}{\beta}\log(2(N-1)N\beta M^2)$ then: $f(S_i) \subseteq S_i$

Proof. We define $\widetilde{\Delta_i} = \min_{j\neq i}\xi^T x^i - \xi^T x^j$
Using the Cauchy-Schwarz inequality, we get: $|\xi^T x^j - (x^i)^T x^j| \leq ||\xi - x^i||\,||x^j|| \leq ||\xi - x^i||M$
So we can say that:

$$\begin{aligned}\widetilde{\Delta_i} &= \min_{j\neq i}\xi^T x^i - \xi^T x^j = \min_{j\neq i}\xi^T x^i - (x^i)^T x^i + (x^i)^T x^i - \xi^T x^j + (x^i)^T x^j - (x^i)^T x^j \\ &\geq \min_{j\neq i}((x^i)^T x^i - ||\xi - x^i||M) - ((x^i)^T x^j + ||\xi - x^i||M) \\ &= -2||\xi - x^i||M + \min_{j\neq i}(x^i)^T x^i - (x^i)^T x^j = \Delta_i - 2||\xi - x^i||M \\ &\geq \Delta_i - \frac{2}{\beta N}\end{aligned}$$

Where the last inequality holds because $\xi \in S_i$.
From lemma 3.3.2 we can affirm that:

$$\begin{aligned}||x^i - f(\xi)|| \leq 2\epsilon M &= 2M(N-1)\exp(-\beta\widetilde{\Delta_i}) \leq 2M(N-1)\exp(-\beta(\Delta_i - \frac{2}{\beta N})) \\ &\leq 2M(N-1)\exp(-\beta(\frac{2}{\beta N} + \frac{1}{\beta}\log(2(N-1)N\beta M^2) - \frac{2}{\beta N})) \\ &= 2M(N-1)\frac{1}{2(N-1)N\beta M^2} = \frac{1}{\beta NM}\end{aligned}$$

□

Using the previous proposition and theorem 3.1 we can state that if the input is inside a sphere S_i, than the accumulation points that are point of minima for the energy are inside the sphere that allows us to use the network properly, with the right choice of the temperature, because different input gives different output.

Remark. We can always find a value for β that satisfies the condition of the previous proposition $\forall i$ solving:

$$\min_i \Delta_i \geq \frac{2}{\beta N} + \frac{1}{\beta}\log(2(N-1)N\beta M^2)$$

which is equivalent to solving:

$$\beta \geq \frac{2}{N \min_i \Delta_i} + \frac{\log(2(N-1)NM^2)}{\min_i \Delta_i} + \frac{\log(\beta)}{\min_i \Delta_i}$$

which solution is: $\beta \in (0, \beta_-] \cup [\beta_+, \infty)$ where β_-, β_+ are the point of intersection of $y = x$ and $y = \frac{2}{N \min_i \Delta_i} + \frac{\log(2(N-1)NM^2)}{\min_i \Delta_i} + \frac{\log(x)}{\min_i \Delta_i}$.
For retrieval, the smallest solutions are useless because they correspond to the global convergence state of the network, as proved in proposition 3.3. However, the largest ones are useful because they can give disjoint S_i.

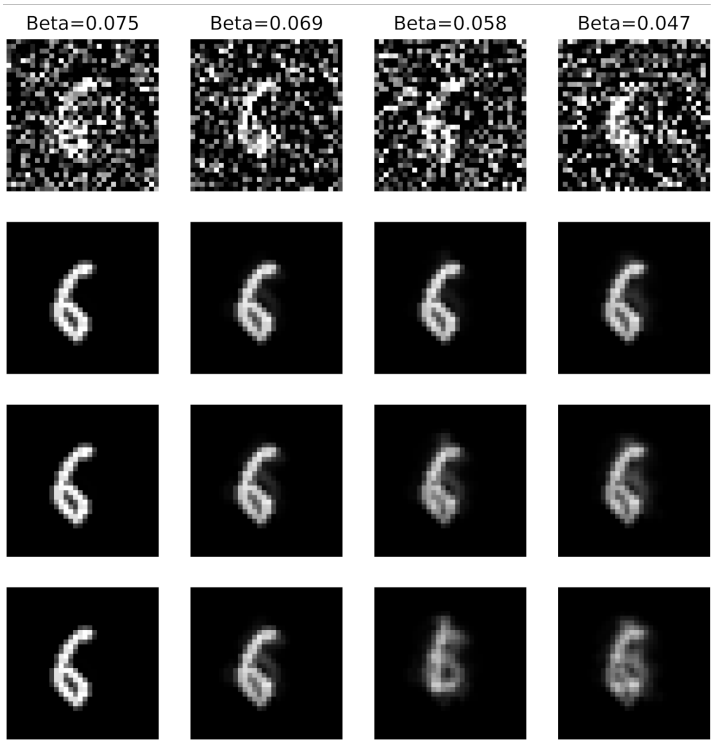


Figure 3.2: The figure shows how the same network gives different outputs with different T, larger values bring to a convergence point that is a mixed pattern instead smaller values give local convergence to the stored pattern. The input image is obtained as a random perturbation of a stored pattern.

3.4 Comparison with discrete Hopfield network

In this section we want to analyse the differences and analogies between the discrete and the continuum Hopfield networks.

- **Network structure**
  Both networks receive an input image, represented as an array of numbers, and process it by correlation with stored patterns. By iteratively updating the input, each network aims to transform it into something closer to a stored pattern.
- **Energy minimization**
  An energy function can be defined for both networks, expressed in terms of correlations with stored patterns. This function decreases along the network dynamics and guides the system towards energy minima.

- **Stochasticity and temperature**
  Both networks have a parameter called temperature, which can be tuned to facilitate convergence. However, this parameter affects each network differently: in discrete networks, temperature controls the amount of random noise in the updating rule, introducing a degree of stochasticity. In continuous networks, the temperature affects how information is integrated within the softmax operation, so there is no randomness. Despite this difference, the two networks show similar behaviour with respect to temperature. In both cases, high temperatures lead to uninformative outputs, where each input is transformed into a generic, uninformative result. Conversely, too low a temperature can also be problematic, leading to trivial outputs in both networks.

- **Pattern and training**
  In both networks, the energy function and update rules are defined in terms of stored patterns. However, due to the discrete nature of the neurons, the dynamics of discrete networks cannot be distinguished from the pattern vectors. In contrast, continuous networks allow the computation of derivatives, which allows them to be integrated into larger deep neural networks and trained with backpropagation. Discrete networks, on the other hand, lack any form of learning because the patterns are fixed and remain unchanged.

Chapter 4

Applications

In this chapter, we will introduce basic implementations and applications of Continuous Hopfield Neural Networks. The results of the contents of this chapter are replicable using the open source repository [2]. We recall contents, techniques and interpretations of the course notes of Deep Learning and Applied Artificial Intelligence of Rodolà [4].

4.1 Implementations

In definition 3.1.2, we defined the updated state of a Hopfield neural network as:

$$\xi^{\text{new}} = X \operatorname{softmax}\left(\beta X^T \xi\right)$$

This neural network is used to classify inputs, so it is more informative to study the evolution of the logits. A **logit** represents the degree to which an input adheres to a particular feature. For example, the image of a '1' may resemble that of a '7', meaning that the '1' may have a high logit for feature '7'. Typically, the softmax of all logits represents the probability that the 1 is classified as a 7 in this case. Note that the vector of logits is proportional to the vector of magnetisations for each pattern, in particular $l_\mu = N\beta m_{N,\mu}(\xi)$ from definition 2.2.7.
An equivalent way to express the update formula using logits is

$$l^{\text{new}} = \beta X^T \xi^{\text{new}} = \beta X^T X \operatorname{softmax}(l)$$

We define $A_\beta \coloneqq \beta X^T X$, so $l^{\text{new}} = A_\beta \operatorname{softmax}(l)$. A Hopfield neural network can be thought of as a cyclic `FCL` with softmax activation:

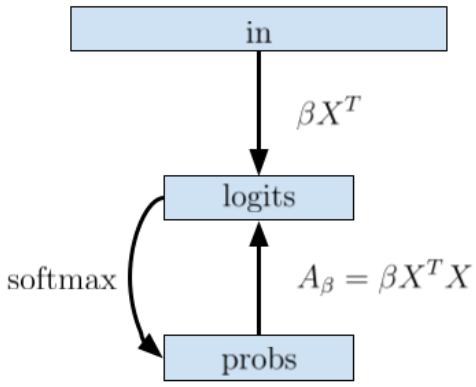


Figure 4.1: Diagram of the Hopfield neural network. The input (on the top) passes through a fully connected layer with parameters βX^T and zero bias. The logits are then fed into a cyclic loop, and the output is obtained. To reconstruct the image, we can use the *probs* layer; $Xp = \xi$, so we can connect an FC layer without bias and weights X to the *probs* layer.

Convolutions A Hopfield neural network looks at the whole image and analyses the image as a whole. To restrict the analysis to a smaller Hopfield neural network, we can define a convolution of the Hopfield neural network. This allows us to reconstruct small patterns in more small parts of the image. A convolution returns a new image with more (or less) channels with logits, and each logit indicates the presence of a pattern in the image.
Let H be a Hopfield neural network that takes $C_i \times S$ neurons as input and returns C_o logits, where S is a shape (e.g. 3×3) with n dimensions. A convolution applies H to each region of shape S in data with C_i channels and n dimensions.
For example, if we have an image of size 32×32 with 3 channels (`RGB`), we can apply a convolution with a 4×4 filter. In this case H operates on $3 \times 4 \times 4 = 48$ neurons and detects 5 patterns. The convolution returns data with 5 channels and a shape of 29×29, where each position represents the response to a pattern for that region of the image.
In fig. 4.2 we can see that a convolution works in a similar way to a classical Hopfield neural network.

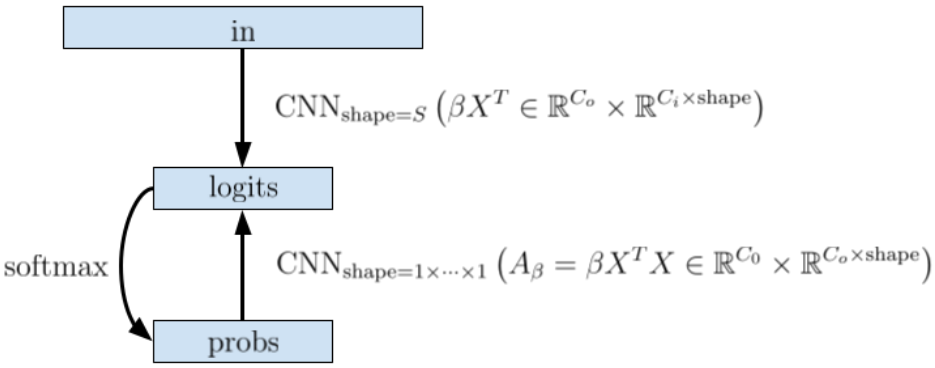


Figure 4.2: The neural network receives data in the form of $C_i \times S$. A convolutional layer computes the logit vector for each region, producing data in the form of $C_o \times S$. A pointwise softmax is then applied to each logit vector. Finally, each logit vector is linearly transformed and a pointwise convolutional layer is applied, producing data of the form $C_o \times S$.

Remark (Generalized Continuous Hopfield Neural Networks). During training, the parameters of a Hopfield neural network can become difficult to interpret. Is it possible to train a number of parameters that is not directly proportional to the number of neurons?
The matrix A_β is not arbitrary. In fact, it must satisfy the following properties:

- A_β is a real, square, and symmetric matrix.
- A_β is non-negative: $v^T A_\beta v = \beta v^T X^T X v = \beta \|Xv\|^2 \geq 0$.

Therefore, if interpretability is not an issue, we can train over a matrix $M \in \mathbb{R}^{C \times C}$ and set $A_\beta = M^T M$. In this way the Hopfield neural network can be seen as a composition of: a `FC` layer with parameter βX^T and bias 0, a cycle with $A_\beta = M^T M$, and another `FC` layer with parameter X and bias 0.
This observation also applies to convolutional Hopfield nets.

Autoencoders Note that the number of parameters P is much smaller than the number of neurons N, so image reconstruction using Hopfield neural networks is a special case of **autoencoder**. In fact, the Hopfield neural network reduces the dimensionality of the input (with N neurons), then it recalls the pattern in the loop and reconstructs the image from it, in detail:

$$XH_{\beta,X}(\xi) = XH'_{A_\beta}\left(\beta X^T \xi\right)$$

where H is the main Hopfield neural network with N neurons and P patterns, H' is a Hopfield neural network with P neurons and P patterns. βX^T makes the code of ξ (encoding), this code is cleaned in the loop (in a manifold) and finally it is rebuilt (decoding).
Now we derive a variational autoencoder from Hopfield neural networks and *PCA*. We observe that a Hopfield neural network is a variant of a *PCA* because it uses a matrix and its transpose to encode

and decode information, the only difference being an additional softmax layer, an internal loop, and an initial and final multiplicative constant[1]. So we get a *PCA* similar to a Hopfield net:

- **encoding** : The input is multiplied by $\sqrt{\beta}X^T$.
- **decoding** : The code is multiplied by $\sqrt{\beta}X$.

Now we turn this autoencoder into a variational autoencoder. So the encoder returns information about some probability distribution and the decoder samples from that distribution and reconstructs the input, this is a **VAE**[2]. We can see that a Hopfield neural network has the same architecture, but it has no sampling. The final architecture is shown in fig. 4.3.

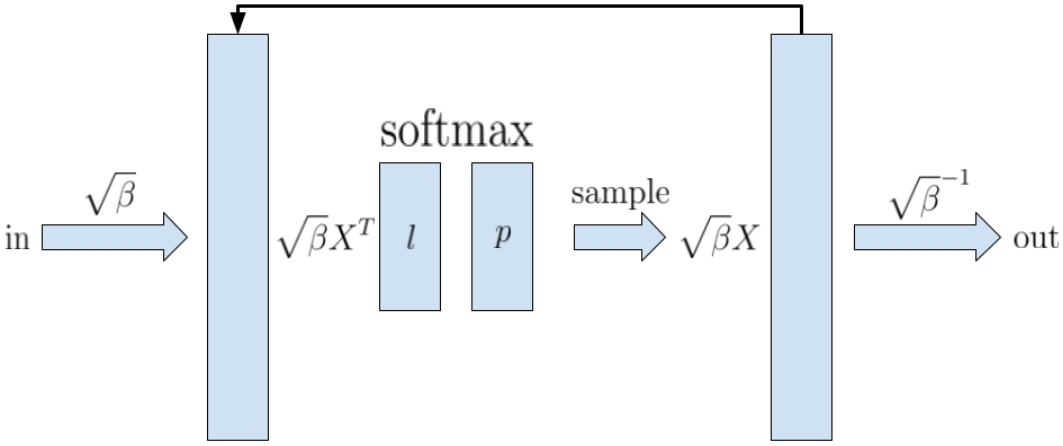


Figure 4.3: The input is passed to the autoencoder with a simple multiplication ($\sqrt{\beta}$), then the autoencoder multiplies its input by M^T, this produces a vector of P logits l, after a softmax we get a discrete distribution p. Now it's possible to simulate this distribution with one or more samples. This distribution is then passed to the decoder which multiplies it by M. This is a single loop and we can answer it using the output as a new input.

4.2 DeepHNN

First, we define a simple use of Hopfield neural networks. A **DeepHNN** is a Hopfield neural network with more patterns given the number of target features. In fact, using 3 patterns per feature may be better than using only one pattern per feature. Thus, the logit returned by a DeepHNN is the maximum logit for any subset of patterns returned by a Hopfield neural network.
We have added a **final bias** for each pattern in the same feature. The bias can assign a confidence about a feature, it is very useful to interface with other layers. In a classical Hopfield neural network the bias is the same for each feature, but when there are other layers and in a more complex neural network it is better to add a new degree of freedom. In algorithm 4.1 we show an excerpt of the code used for the forward pass. In particular, the implemented class `DeepHNN` uses more channels, a separate neural network for each channel.

[1]For more about PCA, see https://www.cs.umd.edu/class/spring2019/cmsc422-0101/materials/lecture20-sp19.pdf
[2]For more about VAEs, see https://www.ibm.com/think/topics/variational-autoencoder

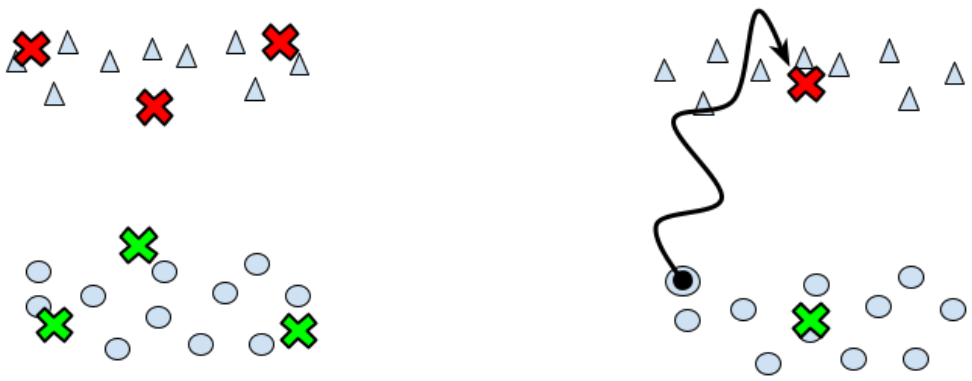


Figure 4.4: We have two features: triangles and circles. By using 3 patterns per feature, we can better capture the data with better coverage. However, the result may be more confusing.

Listing 4.1: DeepHNN forward pass

```python
import torch
from torch import nn
from torch.nn import functional

""" method of DeepHNN
attributes:
    iterations: int
    channels, features, deep, neurons: int
        positive values
    _logbeta: nn.Parameter
        with shape (channels,)
    _bias: nn.Parameter
        with shape (channels, features,)
    _patterns: nn.Parameter
        with shape (channels, deep * features, neurons,)
"""
def forward(self, x: torch.Tensor) -> torch.Tensor:

    # Shape of x must be (batch, channels, neurons)

    L = torch.exp(self._logbeta).view(self.channels, 1, 1) * self._patterns
    A = torch.einsum('cin , cjn-> cij', L, self._patterns)  # i,j are logits

    # main algorithm
    x = torch.einsum('cln, bcn -> bcl', L, x)  # get logits
    for _ in range(self.iterations):
        x = functional.softmax(x, dim=2)  # get probs
        x = torch.einsum('clp, bcp -> bcl', A, x)  # get logits

    # max reduction
    x = x.view(-1, self.channels, self.features, self.deep)
    x = torch.max(x, dim=3).values
    x = x + self._bias.view(1, self.channels, self.features)

    return x
```

A Simple Experiment The first experiment applies a Hopfield neural network with predefined patterns directly to an MNIST image. We manually define 3 patterns for each digit between 0 and 9, and then implement a Hopfield neural network over 32×32 neurons. In fig. 4.5 we observe that the neural network has trouble recognizing the true digit. To solve this problem, we can try to train the neural network on MNIST and then analyze the results.
The neural network was trained using the cross-entropy loss for 40 epochs with an adaptive learning rate using the current accuracy. At the end of training, the cross-entropy loss was 1.8, and in the test phase, the neural network showed 32% accuracy. The patterns with high attribution influence were not changed.

DeepHNN as MLP A single Hopfield neural network with the whole image as input is often not the best choice, because the output of the neural network depends entirely on the dot product with the vector patterns, i.e. moving all the pixels to the right (or left, or up, or down) will give a completely different output. Furthermore, we can observe that within a Hopfield neural network there is no

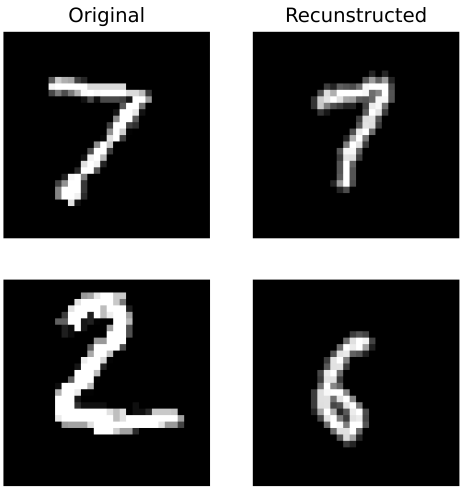


Figure 4.5: Applying a Hopfield neural network to two MNIST images. The Hopfield neural network had $\beta = 0.07$ and ran for 2 iterations.

notion of locality, in fact patterns are retrieved if and only if they are in the same position of the stored patterns. To overcome these problems, we observe that more smaller neural networks are better than only one, because they can analyze a small part of the images.
A convolution can create more "versions" of the same image, in particular it highlights a set of patterns in the image. For example, a convolution that detects horizontal lines and vertical lines will show, with high values, where there are horizontal lines and vertical lines. Thus, a single-channel image becomes a multichannel image (one for each feature). In addition, the number of patterns is usually not too large in relation to the number of input channels and the size of the kernel. Typically, the number of patterns is a quarter of the number of neurons in a kernel (e.g., 8 input channels, 4×4 kernel size, 32 output channels).
We define a neural network with a linear CNN that has 5 channels with 4×4 kernels and 2 steps. Then we use a DeepHNN with 5 channels and 10 features with deep 5. Finally, we add a trainable convex reducer that merges all channels into only one and obtains 10 logits. This neural network has 56 440 trainable parameters. We trained this neural network for 40 epochs with a learning rate of 0.01: the final accuracy obtained is $\approx 87\%$ with a cross entropy loss of 0.7. In fig. 4.6 it's possible to see details about the training, but kernels and patterns are not visible.

4.3 ConvHNN2d

We define a convolutional layer with Hopfield as a small Hopfield neural network that detects patterns in all parts of the image (see fig. 4.2). In this experiment, we use a $2d$ convolution to use over MNIST. The main difference between a typical linear convolution and this convolution is how it highlights the patterns. A linear convolution pattern works over logits, so if it looks for the pattern $(1, 1)$, it will highlight the pattern $(5, 5)$ with value 10 (excellent compatibility) and the pattern $(1, 1)$ with value 2 (good, but not excellent). So it works not in terms of distance, but in terms of the scalar product (intensities and directions) of patterns. A Hopfield neural network looks at patterns in a different way and does not work in terms of distance or scalar products. The Hopfield neural network tries to recall a pattern so that the final image is a mixture of recalled patterns for each region of the image, so it can clean the image from noise, and this behaviour is very useful when this neural network is the first convolutional layer. In the algorithm 4.2 we show the forward pass of this layer.

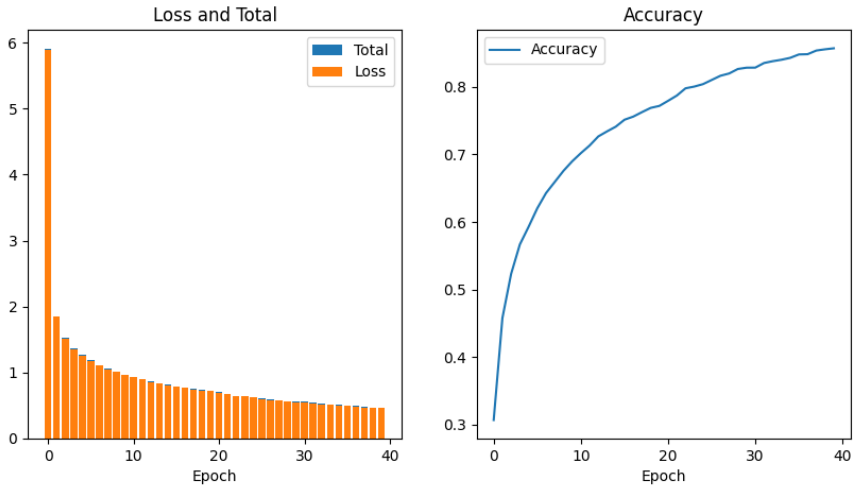


Figure 4.6: On the left the loss evolution during training (in orange the cross entropy loss, in blue the part due to the regularization), on the right the accuracy evolution.

Listing 4.2: ConvHNN2d forward pass

```python
import torch
from torch import nn
from torch.nn import functional

""" method of ConvHNN2d
attributes:
    channels_in, channels_out: int
        positive values
    kernel_size, padding, stride, dilation
        parameters of classical convolution
    iterations: int
        number of neurons of theoretical Hopfield neural network
    _logbeta: nn.Parameter
        with shape (1,)
    _bias: nn.Parameter
        with shape (channels_out,)
    _patterns: nn.Parameter
        with shape (channels_out, channels_in, kernel_size)
"""
def forward(self, x: torch.Tensor) -> torch.Tensor:

    # Shape of x must be (batch, channels_in, #, #)

    weight_loop = (
        torch.exp(self._logbeta)
        * torch.einsum(
            'in, jn -> ij',  # i,j are logits (channels_out)
            self._patterns.view(self.channels_out, -1),
            self._patterns.view(self.channels_out, -1)
        )
    ).view(self.channels_out, self.channels_out, 1, 1)

    x = nn.functional.conv2d(
        x,
        torch.exp(self._logbeta) * self._patterns,
        stride=self.stride,
        padding=self.padding,
        dilation=self.dilation,
    )
    for _ in range(self.iterations):
        x = nn.functional.softmax(x, dim=1)
        x = nn.functional.conv2d(x, weight_loop)

    return x + self._bias.view(1,self.channels_out,1,1)
```

In this experiment, we define a simple neural network that attempts to reconstruct an image using only convolutions. In particular, the input image has Gaussian noise, so we will use ConvHNN2d to remove the noise and reconstruct the original image. A Hopfield convolution produces an image with

C_{out} logits per pixel, so for each vector of logits per pixel we can recall the pattern using this formula:

$$\text{recalled pattern} = X \operatorname{softmax}(\text{logits})$$

Thus, the transposed convolution combines all the reconstructed patterns and obtains an approximation of the original convolution. In our architecture, a ConvHNN2d uses 9 patterns with kernel size $(5, 5)$. Thus, it produces a new image with 9 channels of logits, the bias penalizes the patterns related to the noise and rewards the real patterns. In this way, the transposed convolution reconstructs only good patterns.

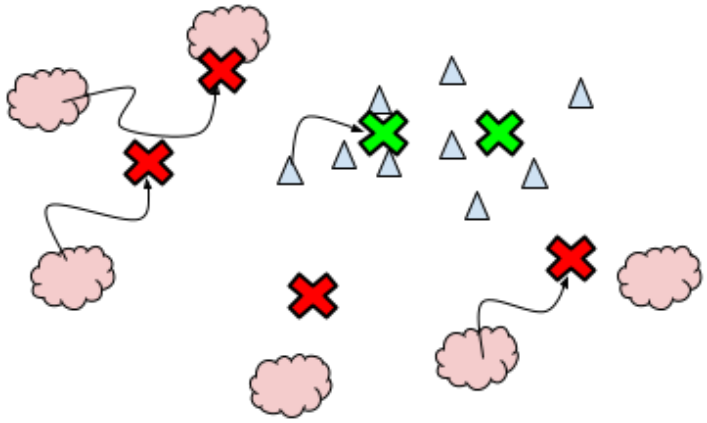


Figure 4.7: Patterns with low bias (in red) are ignored as logits. In this way these patterns protect correct patterns with higher bias.

The architecture of the neural network is detailed:

- `ConvHNN2d` with 9 patterns and 5×5 kernel size and a `SiLU` activation
- `AutoEncoder` with: a linear convolution of 9 channels to 3 channels and 3×3 kernel size as encoder, a transposed convolution of 9 channels to 2 channels and 3×3 kernel size as decoder.
- Transpose convolution with Hopfield parameters.

This neural network was trained to reconstruct images, so its input was an image with Gaussian noise (with variance 1.0). After training 20 epochs, the MSE loss over the test set is 4×10^{-2} with only 742 trainable parameters.
Now we see how this neural network cleans up the images. In fig. 4.8 we show the evolution of an image through the neural network.

4.4 Variational Hopfield Neural Network (VHNN)

In fig. 4.3 we showed the architecture of the Variation Hopfield Neural Network, we observe that the number of samples defines the incidence of noise. If we get a single sample from the distribution p we will have maximum noise, if we get a lot of samples the neural network will look like a typical Hopfield neural network.
We observe that $p = \mathbb{E}_p[M^n]$ where M^n is the estimated distribution from n Bernoulli samples from the distribution p. Let M_k^n be the kth estimate associated with the kth sample (or the kth component of the softmax operation) and σ_k be the variance of the Bernoulli sample from the kth component.

$$\frac{M_k^n - p_k}{\sigma_k/\sqrt{n}} = \frac{\frac{1}{n}\sum^n \text{Be}(p_k) - p_k}{\sigma_k/\sqrt{n}} \xrightarrow{D} \mathcal{N}(0,1) \ , \qquad \sigma^2 = \text{Var}[\text{Be}(p_k)] = p_k(1-p_k)$$

We can reformulate the noise with an approximation:

$$S^\eta \coloneqq p + \text{diag}\left(\sqrt{p_k(1-p_k)}\right)\varepsilon \, , \qquad \varepsilon \sim \mathcal{N}(0, \eta^2 \mathbb{1}) \ , \quad \eta \in \mathbb{R}_+$$

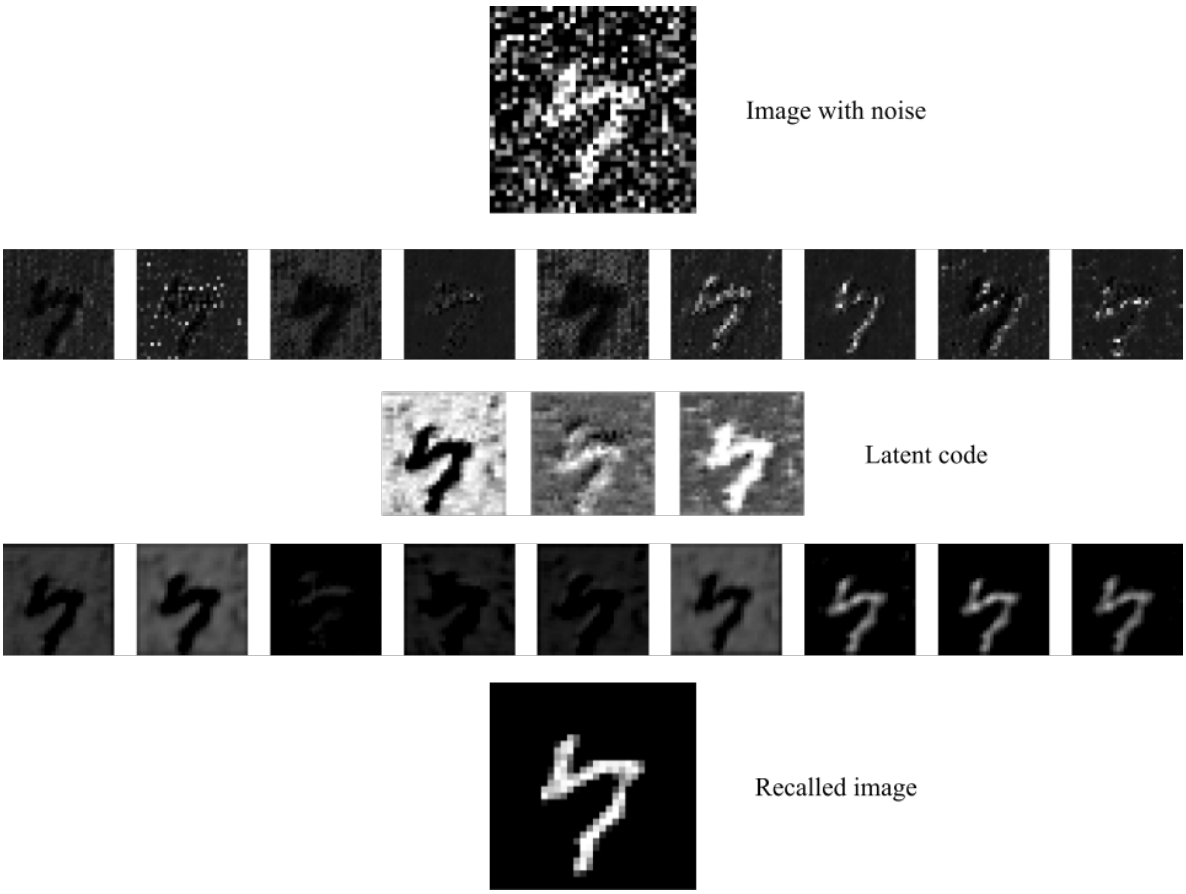


Figure 4.8: Above is an image from MNIST with Gaussian noise (variance: 1.44). The Hopfield convolution layer highlights certain patterns in the image (see white pixels). A linear CNN finds 3 patterns from logits. An activation function then removes the residual noise so that a transposed CNN reconstructs the logits of the Hopfield convolution. Finally, the patterns from the Hopfield convolution are used to reconstruct the original image without noise.

where S^η is the sampled data from the distribution p in **VHNN**. Note that the noise η is a continuous and eventually trainable value that replaces the iper parameter $\frac{1}{\sqrt{n}}$.
This formulation is very similar to the common formulation of VAE, which uses normal samples from the mean and variance given by the encoder.
In algorithm 4.3 we show the forward pass of layer VHNN.

Listing 4.3: VHNN forward pass

```python
import torch
from torch import nn
from torch.nn import functional

""" method of ConvHNN2d
attributes:
    channels, features, neurons: int
        positive values
    iterations: int
        number of neurons of theoretical Hopfield neural network
    _logbeta, _lognoise: nn.Parameter
        with shape (channels,)
    _patterns: nn.Parameter
        with shape (channels, features, neurons)
"""
def forward(self, x: torch.Tensor) -> torch.Tensor:

    # Shape of x must be (batch, channels, neurons)

    L = torch.exp(self._logbeta/2).view(-1, 1, 1) * self._patterns

    x = torch.exp(self._logbeta/2).view(1, -1, 1) * x
    for _ in range(self.iterations):
        x = torch.einsum('cfn, bcn -> bcf', L, x)
        x = nn.functional.softmax(x, dim=2)
        epsilon = torch.randn_like(x, device=x.device, requires_grad=False)
        xdet = x.detach().requires_grad(False)
        x = x + torch.exp(self._lognoise).view(1,-1,1) * (xdet*(1-xdet))**0.5 * epsilon
        x = torch.einsum('cfn, bcf -> bcn', L, x)

    x = torch.exp(-self._logbeta/2).view(1, -1, 1) * x
    return x
```

A Simple Experiment Using the noise, we study how the state evolution falls into and escapes from a S_i (see proposition 3.4).
Now we analyze the pattern set S with 30 patterns in the previously used $\mathbb{R}^{32\times 32}$. We calculate the values M and $(\Delta_i)_i$:

$$M = \max_\mu \|x^\mu\| = 31.3933$$

$$(\Delta_i)_i = \left(\min_{j\neq i}(x^i)^T x^i - (x^i)^T x_j\right)_i = \text{ see table 4.1}$$

digit	pattern 1	pattern 2	pattern 3
0	88.0986	80.1672	131.736
1	53.4877	69.0222	59.4534
2	63.2126	63.9695	69.7267
3	65.0894	55.4366	46.0532
4	66.9539	62.2354	46.1037
5	41.8603	39.9262	37.5255
6	45.8450	60.8577	49.2501
7	55.7401	51.3837	35.1906
8	52.2477	56.2899	69.4982
9	107.129	74.5176	51.1973

Table 4.1

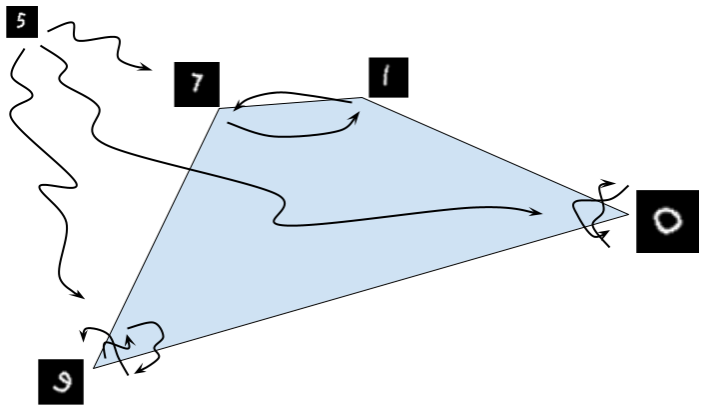
So we want to find a value β such that

$$\forall i \left(\Delta_i \geq \frac{2}{\beta N} + \frac{1}{\beta}\log\left(2\left(N-1\right)N\beta M^2\right)\right)$$

We calculate the minimum value $\Delta_{\min} = \min_i \Delta_i = 37.5255$ and simplify the equation:

$$\beta' \geq a + \log \beta'$$

where: $\beta' = \beta \Delta_{\min}$, $a = \frac{2}{N} + \log \frac{2(N-1)NM^2}{\Delta_{\min}}$
If $a \leq 1$ then the statement holds for any β', otherwise we solve the equation $\beta' = a + \log \beta'$. The equation has solutions $\beta'_- < 1$ and $\beta'_+ > 1$. We can use a fixed point iteration to find β'_+ with starting point $u_0 = 2a$ and K iterations, so we know that $\beta = \frac{u_K}{\Delta_{\min}}$ is a good value for β, but we could choose a larger one. Now we can calculate the radius of each S_i: $R(S_i) \coloneqq \frac{1}{\beta NM}$.
So we calculate $\beta = 0.0727$, but in this experiment we use $\beta = 0.063$, which reduces the ability of the neural network to detect patterns. We set the noise to $\beta = 0.3$ (high noise) so that we can see a change in opinion very quickly. The VHNN works over 4 different states: for two patterns the noise does not allow an easy exit from local convergence, while for two patterns the neural network state jumps from one to the other. Empirically, with random inputs we have only these solutions, even if we start from a different S_i.
This is a stochastic process where the state of the neural network is a continuous distribution in the affine space of the patterns. In this way, the neural network can recall other patterns in a manner similar to a classical Markov chain. In a training process, the neural network can choose the position of the patterns to create a particular distribution, and the noise can take values that allow for fast training.



Chapter 5

Conclusion

Although discrete Hopfield networks are a fundamental model in the field of neural networks, they have structural limitations that limit their applicability in modern deep learning architectures. However, their ability to store and retrieve even partially damaged patterns makes them useful in contexts such as pattern recognition and error correction, especially in resource-constrained environments. Acting as "associative memories", they store and retrieve patterns in the presence of partial or noisy inputs, making them particularly effective in areas such as image recognition or error correction in telecommunications. Their rapid stabilization toward an equilibrium state also makes them useful in models of biological behavior. However, their limited memory capacity and inability to be trained by backpropagation reduce their effectiveness in more complex applications.

Despite their advantages, discrete Hopfield networks suffer from significant limitations in their storage capacity and lack of trainability via backpropagation methods, making them unsuitable for more complex applications. Pattern storage is limited by the fact that the maximum number of patterns that can be stored increases linearly with the number of neurons, making the model ineffective on a large scale. In addition, the lack of gradient descent trainability prevents optimization for more sophisticated applications. These limitations have led to the development of continuous Hopfield networks, which offer greater flexibility, better generalization, and a wide range of applications, especially when integrated with other neural layers.

To overcome the limitations of discrete networks, continuous Hopfield networks have proven to be a powerful tool, especially when integrated with layers that exploit their properties. In our analysis, we have experimented with several structures that apply the principles of continuous networks to achieve remarkable results. The `DeepHNN` layer allows multiple patterns to be associated with a single feature, improving the robustness of information retrieval, while `ConvHNN` overcomes the problem of pattern translation through local analysis, which is more effective than traditional convolutions at removing noise. The `VHNN model`, which combines continuous Hopfield networks and variational autoencoders (VAE), allows more efficient training and provides theoretical insights into the search for energy minima in stochastic processes. Although continuous Hopfield networks are simpler than other neural architectures, their small number of parameters makes them ideal for resource-constrained environments. In addition, their clustering capability makes them more resistant to overfitting, which improves generalization. However, the training speed is slower than for more complex models and requires optimizers with high learning rates and moderate batch sizes for good convergence, with computational cost increasing with the number of iterations.

A possible future development concerns the theoretical analysis of the behavior of variational Hopfield networks (VHNNs), which raises numerous theoretical questions, such as Is there a limit distribution for stochastic processes in Hopfield networks? What is the waiting time before the network changes state, and how is it related to the energy associated with a state? How do Hopfield networks behave as the number of neurons (or pixels) tends to infinity? Answering these questions could lead to significant theoretical developments in the field of complex dynamical systems and stochastic neural networks, improve the training efficiency of Hopfield networks, and open new avenues for applications in systems theory, optimization, and theoretical machine learning.

Bibliography

[1] Elena Agliari. Mathematical models for neural networks, 2024. Università degli Studi di Roma La Sapienza.

[2] Stefano Magrini Alunno. Mathematical models for neural networks, 2024. URL https://github.com/StefanoMagriniAlunno/MathematicalModelsForNeuralNetworks.

[3] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need, 2021. URL https://arxiv.org/abs/2008.02217.

[4] Emanuele Rodolà. Deep learning and applied artificial intelligence, 2024. URL https://erodola.github.io/DLAI-s2-2024/. Università degli Studi di Roma La Sapienza.