Neural Networks
Lecture Notes
Hopfield networks

Marcel van Gerven

1 Learning Goals

After studying the lecture material you should:

1. understand what is meant by a Hopfield network
2. be able to update the states of a simple Hopfield network
3. be able to write down the energy function of a Hopfield network
4. know what is meant by combinatorial optimisation and associative memories
5. be able to derive what Hopfield network can solve the multiflop problem
6. be able to write down the weight update for a Hopfield network with bipolar units
7. understand the proof that Hebbian learning works in case of training on one input pattern
8. know what is meant by network capacity
9. know what is meant by spurious patterns and the ways to handle these

2 Notes

2.1 Hopfield networks

The Hopfield network (HN) was developed by John Hopfield [Hop82]. They are inspired by concepts in statistical physics. A (discrete) Hopfield network is composed of binary threshold units. Units have states x_i and biases θ_i. They have symmetric recurrent connections between them such that $w_{ij} = w_{ji}$ with $w_{ii} = 0$.

Network states are updated by computing the input activations $a_i = \sum_j w_{ij} x_j + \theta_i$ and then feeding this into a threshold unit such that

$$x_i = f(a_i) = \begin{cases} 1 & \text{if } a_i \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Note that each unit conditional on all other units is equivalent to a perceptron.

Hopfield nets are guaranteed to converge to a stable state. This guarantee relies on:

- no self-connection ($w_{ii} = 0$)
- symmetric weights ($w_{ij} = w_{ji}$)
- asymmetric updates (i.e. one randomly chosen unit is updated at each iteration)

Hopfield networks have been used to solve optimization problems but also as a model system for associative memory.

2.2 Energy function

Recurrent networks of non-linear units are generally very hard to analyze. They can behave in many different ways:

- Settle to a stable state
- Oscillate
- Follow chaotic trajectories that cannot be predicted far into the future.

Hopfield realized that if the connections are symmetric, there is a global energy function: Each "configuration" of the network has an energy. The binary threshold activation function causes the network to settle to an energy minimum.

Hopfield nets have a scalar value associated with each state of the network referred to as the *energy*, E, of the network, where:

$$\begin{aligned} E(\mathbf{x}) &= -\frac{1}{2}\sum_{i,j} w_{ij}x_i x_j - \sum_i \theta_i x_i \quad (1) \\ &= -\frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} - \mathbf{x}^T\boldsymbol{\theta} \quad (2) \end{aligned}$$

When units are randomly chosen to update, the energy E will either lower in value or stay the same. Under repeated updating the network will eventually converge to a state which is a local minimum in the energy function (stable state). At that state, the updating will not causes any changes in the states anymore.

2.3 Solving combinatorial optimization problems

Hopfield nets have been used to solve combinatorial optimization problems. An *optimization problem* is the problem of finding the best solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete. An optimization problem with discrete variables is known as a *combinatorial optimization problem*.

Combinatorial optimization problems may be solved by making the HN energy function equivalent to the objective function via manual specification where the problem constraints are included in the energy function as penalty terms. Consider for example the multiflop problem, where we want to obtain vectors with all zeros except for a single one. We would like to solve this "problem" as follows:

- Run an n-variable Hopfield net until convergence
- Produce vectors with exactly one active unit

How should we set the weights and biases to enforce this behaviour?

We can specify this problem in terms of the energy function. We want to find a minimum of:

$$E(\mathbf{x}) = \left(\sum_{i=1}^{n} x_i - 1\right)^2$$

since this minimum is zero if we have exactly one active unit. Otherwise, it is larger than zero. This expression can also be written as

$$E(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + \sum_{i \neq j} x_i x_j - 2\sum_{i=1}^{n} x_i + 1$$

For binary states, $x_i = x_i^2$ and therefore

$$E(\mathbf{x}) = \sum_{i \neq j} x_i x_j - \sum_{i=1}^{n} x_i + 1$$

Ignoring the irrelevant constant 1, this can be rewritten as

$$E(\mathbf{x}) = -\frac{1}{2} \sum_{i \neq j} (-2) x_i x_j - \sum_{i=1}^{n} x_i$$

Recalling Equation (2), this implies we have a Hopfield network with energy function

$$E(\mathbf{x}) = -\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x} - \mathbf{x}^T \boldsymbol{\theta}$$

with $w_{ij} = -2$ for $i \neq j$, $w_{ii} = 0$ and $\theta_i = 1$.

This example forms the basis for solving more complex problems such as the n-queens problem, traveling salesman problem, vehicle routing problem, minimum spanning tree problem, etc. Combinatorial optimization embodies many important problems in Artificial Intelligence! This does not mean that formulation as a Hopfield network is the state of the art in solving combinatorial optimization problems!

2.4 Associative memory

Hopfield networks have also been used to model auto-associative memory. To work as an auto-associative memory, a network has to solve the following problem:

> "Store M patterns $\mathbf{x}_1, \ldots, \mathbf{x}_M$ such that when presented with a new pattern $\mathbf{x}$, the network returns the stored pattern $\mathbf{x}_i$ that is closest in some sense to $\mathbf{x}$."

Auto-associative memories can be contrasted with hetero-associative memory where the aim is to retrieve one pattern upon presentation of another associated pattern (see e.g. bidirectional associative memory, BAM, network).

Hopfield proposed that memories could be energy minima of a neural net. The binary threshold decision rule can then be used to "clean up" incomplete or corrupted memories. This gives a content-addressable memory in which an item can be accessed by just knowing part of its content Applications: models of biological memory, denoising, hand-written digit recognition, face recognition, information retrieval in database systems,...

2.5 HN learning algorithm

Learning the weights in a Hopfield network is easy:

- Instantiate each of the patterns onto the network (clamping states 0 and 1)
- Update the (initially zero) network weights after each kth pattern according to

$$w_{ij} \leftarrow w_{ij} + 4(x_i^{(k)} - 0.5)(x_j^{(k)} - 0.5)$$

Biases can be represented with weights from a unit that is always on.

Instead of unipolar units (0 vs 1) we could also use bipolar units (-1 vs 1). The update function in case of bipolar units is given by:

$$x_i = \operatorname{sgn}\left(\sum_j w_{ij} x_j - \theta_i \right)$$

If we use bipolar units then the weight update is given by

$$w_{ij} \leftarrow w_{ij} + x_i^{(k)} x_j^{(k)}$$

This is an example of a Hebbian learning rule.

Consider a Hopfield network with bipolar units and an n-dimensional pattern $\mathbf{x}_1$. We want to show that after Hebbian learning the minimum of the resulting energy function is located at $\mathbf{x}_1$. Update the (initially zero) network weights after a pattern k according to

$$w_{ij} \leftarrow w_{ij} + x_i^{(k)} x_j^{(k)} .$$

After presentation of a pattern $\mathbf{x}_1$, the (initially zero) weight matrix is given by:

$$\mathbf{W}_1 = \mathbf{x}_1 \mathbf{x}_1^T - \mathbf{I}$$

Recall that

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T \mathbf{W} \mathbf{x} - \mathbf{x}^T \boldsymbol{\theta}$$

We can ignore the bias term if it is represented as a constant in $\mathbf{x}$. For $\mathbf{W}_1 = \mathbf{x}_1 \mathbf{x}_1^T - \mathbf{I}$ it holds that

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T \mathbf{W}_1 \mathbf{x} = -\frac{1}{2}(\mathbf{x}^T \mathbf{x}_1 \mathbf{x}_1^T \mathbf{x} - \mathbf{x}^T \mathbf{x})$$

This is equivalent to

$$E(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}^T \mathbf{x}_1)(\mathbf{x}_1^T \mathbf{x})^T + \frac{n}{2}$$

since $\mathbf{x}^T \mathbf{x} = n$ for bipolar units and $(ab)^T = b^T a^T$. The minimum does not depend on the second term. Hence

$$E(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}^T \mathbf{x}_1)(\mathbf{x}_1^T \mathbf{x})^T$$

is minimized when the scalar $\mathbf{x}^T \mathbf{x}_1 = \mathbf{x}_1^T \mathbf{x}$ is maximized. Since $\mathbf{x}^T \mathbf{x}_1$ can be at most n for bipolar units and $\mathbf{x}_1^T \mathbf{x}_1 = n$, it holds that Hebbian learning locates the minimum of the energy function at $\mathbf{x}_1$, quod erat demonstrandum.

2.6 Spurious patterns and network capacity

Patterns that the network uses for training, called retrieval states, become attractors of the system. Repeated updates would eventually lead to convergence to one of the retrieval states. Sometimes the network will converge to *spurious patterns*, that are different from the training patterns. The energy of these spurious patterns is also a local minimum. We have the following types of spurious patterns:

- For each stored pattern $\mathbf{x}$, the reversed state $-\mathbf{x}$ is a spurious pattern.
- A spurious pattern can also be a mixture state; a linear combination of an odd number of retrieval states.
- A spurious state can also be unrelated to the original patterns; these are called spin glass states.[1]

Network capacity of a Hopfield network is defined as the number of patterns that can be reliably stored Number of patterns that can be stored depends on the number of nodes. Each time we memorize a pattern we hope to create a new energy minimum. However, nearby minima may merge to create one minimum at an intermediate location (spurious pattern) This limits storage

[1]A *spin glass* is a disordered magnet, where the magnetic spin of the component atoms (the orientation of the north and south magnetic poles in three-dimensional space) are not aligned in a regular pattern.

capacity of the Hopfield net. The number of patterns that can be stored using Hebbian learning is on the order of $0.138N$ where N is the number of nodes in the network.[2]

There are various ways to reduce the appearance of spurious patterns. One way is to apply *unlearning*, where we let the network settle at a random initial states and use:

$$w_{ij} \leftarrow w_{ij} - \epsilon x_i^{(k)} x_j^{(k)} \,.$$

This gets rid of deep spurious minima and increases memory capacity. Dream sleep has been proposed as an unlearning process. Another approach is to transformation a Hopfield learning problem to a set of perceptron learning problems. Yet another approach is to use simulated annealing based on the Boltzmannn machine.

3 Reading material

For more background info on Hopfield networks, study the following chapter.

References

[Hop82] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.*, 79(8):2554–2558, 1982.

[2]Suppose we assume the human brain acts as an associative memory with about 10^8 neurons. Then there are about 10^7 patterns to store. Would this be sufficient?