Artificial Neural Networks
Lecture Notes

Prof. Dr. Sen Cheng

Winter Semester 2019/20

13 Hopfield network

All models discussed so far map some inputs X to qualitatively different outputs Y, and have been viewed as approximating some kind of function $Y = f(X)$. The goal has been to generalize from the training dataset to a novel test dataset. An alternative goal of training neural networks is that we want to store the training dataset (X,Y), and retrieve the y that was associated with an item x. This is called *hetero-associative memory* and is an important function of the brain. This kind of memory is easy to implement using a computer algorithm and without using neural networks. However, another kind of memory that is not so easy to implement is *auto-associative memory*, or *content-addressable memory*, where an item x has to be retrieved based on a partial, or noisy, version x' of the item. The *Hopfield net* is a recurrent neural network that performs this function (Hopfield, 1982). In a way, memory networks severely overfit the training data, but that's exactly what we want them to do.

13.1 Attractor states and attractor networks

We are familiar with gradient descent to find the parameters of a model that minimize the loss function. In the Hopfield net, the same idea is applied to the network states instead, i.e., there is an *energy function* over the states of a network that is minimized by the dynamics of the network (Fig. 1). The local minima are also called *attractors*, and the Hopfield net is called an attractor network. If we could setup the network such that the minima corresponded to

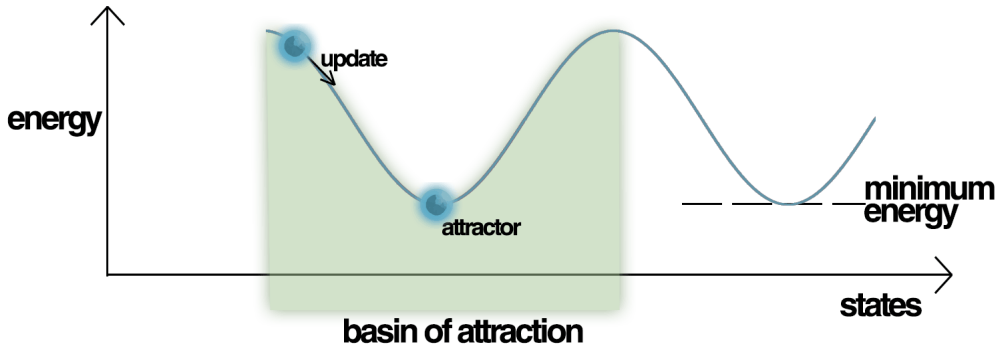

Figure 1: Attractor in energy landscape. The attractor state corresponds to the pattern x that is stored in the network. If the retrieval is initiated with a partial, or noisy, pattern x', which lies within the basin of attraction, the network will evolve towards the attractor state x. Source: Wikimedia Commons, user:Mrazvan22. Licensed under Creative Commons BY-SA 3.0.

the pattern x that we want to store in the network, then the dynamics of the network could retrieve these patterns, if cued with slightly different patterns x'.

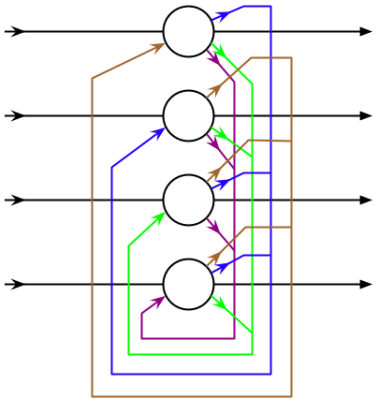

Figure 2: Hopfield net. Source: Wikimedia Commons, user:Zawersh. Licensed under Creative Commons BY-SA 3.0.

13.2 Definition of the Hopfield Network

Given n data vectors $X = (x^1, x^2, \ldots, x^n)$, where each x^i is a binary vector, i.e., $x^i_j \in \{-1, 1\}$, $j = 1, \ldots, m$. The goal of the Hopfield net is to store these patterns in a neural network and retrieve a stored pattern from the network based on a cue x'. This cue could be a, more-or-less, corrupted version of a patterns that was previously stored in the network. The Hopfield net consists of m binary units, i.e., $h_j \in \{-1, 1\}$, that are connected via weights w_{ij}. Two elements are essential in the definition of the network: how the weights are set depending on the inputs and the update rule, which gives rise to the dynamics of the network.

13.2.1 Weights

The weights of the recurrent network (Fig. 2) are given by

$$w_{jk} = \begin{cases} 0 & \text{if } j = k \\ \frac{1}{n} \sum_{i=1}^{N} x^i_j x^i_k & \text{if } j \neq k \end{cases} \tag{1}$$

which is equivalent to writing the following matrix identity:

$$w = \frac{1}{n} \sum_{i=1}^{n} x^i \times x^i - I \tag{2}$$

Eq. 1 can be viewed as the learning rule since it encodes the patterns x into the weight matrix w. It has two desirable properties that are biologically plausible:

- Local: The learning rule is local because the weight update uses only information available to neurons that are connected by the weight.
- Incremental: New patterns are added to the weights as they are encountered and previous patterns do not have to be stored for later stages of training.

13.2.2 Update Rule

To start retrieval we initialize the network with the retrieval cue, i.e. $h = x'$, and iterate the update rule Alg. 13.1.

Algorithm 13.1 Hopfield Net – asynchronous update

Require: w, partial cue h

1: **repeat**

2: Randomly select a unit j

3: Update state of unit j according to

$$h_j \leftarrow \begin{cases} -1 & \text{if } \sum_k w_{jk} h_k + b_j \leq 0 \\ 1 & \text{if } \sum_k w_{jk} h_k + b_j > 0 \end{cases} \tag{3}$$

4: **until** predetermined number of iterations is reached

This update rule changes the state of the network until it reaches a local minimum of the energy function (see below). The bias term can be thought of as some constant external input that is provided to the network during the updating. Most of the times, it will be zero.

For simplicity, we define the function

$$Q(t) = \begin{cases} -1 & \text{if } t \leq 0 \\ 1 & \text{if } t > 0 \end{cases} \tag{4}$$

So, the update equation becomes

$$h_j \leftarrow Q\left(\sum_k w_{jk} h_k + b_j\right) \tag{5}$$

The asynchronous update rule robustly leads to convergence, but is also slow because in each step the activity of only a single unit is updated. To speed the computations, one can update all units at the same time (synchronous update).

Algorithm 13.2 Hopfield Net – synchronous update

Require: w, partial cue h

1: **repeat**

2: Update network state according to

$$h \leftarrow Q(wh + b) \tag{6}$$

3: **until** predetermined number of iterations is reached

From a computational perspective, the disadvantage of the synchronous update rule is that it can lead to oscillations, which do not occur when the asynchronous update rule is used. From a biological perspective, the synchronous update rule seems less realistic since it requires a central clock that determines when the units should be updated and there is little evidence for such a mechanism in the brain.

13.2.3 Examples:

Define a Hopfield net that stores one pattern x:

$$x^1 = \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} \quad \rightarrow \quad w = \begin{bmatrix} 0 & -1 & +1 \\ -1 & 0 & -1 \\ +1 & -1 & 0 \end{bmatrix} \tag{7a}$$

Start with x and $b_j = 0$, and iterate the update algorithm Alg. 13.1 for the pre-set number of iterations. Verify that $x = x^1 \rightarrow x^1$. In the following the box around an element indicates that that element is being updated in the next step.

$$\begin{bmatrix} \boxed{-1} \\ +1 \\ -1 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} -1 \\ +1 \\ \boxed{-1} \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} -1 \\ \boxed{+1} \\ -1 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} -1 \\ +1 \\ 1 \end{bmatrix} \tag{7b}$$

Example for pattern completion:

$$\begin{bmatrix} \boxed{-1} \\ -1 \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ \boxed{-1} \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} \tag{7c}$$

Example for clean-up:

$$\begin{bmatrix} +1 \\ \boxed{+1} \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} \boxed{+1} \\ -1 \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ \boxed{-1} \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix} \tag{7d}$$

Example for *spurious attractor*:

$$\begin{bmatrix} +1 \\ \boxed{+1} \\ +1 \end{bmatrix} \rightarrow \begin{bmatrix} +1 \\ -1 \\ \boxed{+1} \end{bmatrix} \rightarrow \begin{bmatrix} \boxed{+1} \\ -1 \\ +1 \end{bmatrix} \rightarrow \begin{bmatrix} +1 \\ \boxed{-1} \\ +1 \end{bmatrix} \rightarrow \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} \tag{7e}$$

13.2.4 Spurious attractors

Spurious attractors are local minima of the energy of a Hopfield network that do not correspond to one of the patterns stored in the network. There are different types of spurious attractors:

1. The inverse pattern. If x is an attractor then $Q(Wx) = x$. If we define $y = -x$ then

   $$Q(Wy) = Q(W(-x)) = Q(-Wx) = -Q(Wx) = -x = y \tag{8}$$

   So, if x is an attractor of the Hopfield net, then so is $-x$. Since they were not intended to be attractors of the network, they are called spurious attractors.

2. Linear combination of an odd number of attractor states. (Linear combinations of an even number of attractor states are not attractor states, because the elements ($= \pm 1$) can sum to zero.)

3. For large n, local minima that are not linear combinations of stored patterns.

13.3 Convergence

The energy of a Hopfield network is given by

$$E(w,h) = -\frac{1}{2}\sum_{j,k} w_{jk} h_j h_k - \sum_j b_j h_j \tag{9}$$

$$= -\frac{1}{2} h^T w h - b^T h \tag{10}$$

In analogy to physics, this function is called the energy because every update of the Hopfield network decreases it. Consider what happens to the energy when the network state is updated: $h \rightarrow h'$.

$$\Delta E = E(w,h') - E(w,h) \tag{11a}$$

In the asynchroneous update rule only one unit's activity is updated, while all other units remain the same. Say unit v is updated, then $h'_v \neq h_v$ and $h'_j = h_j$ for all $j \neq v$. Therefore, terms in Eq. 10 that do not include either h'_v or h_v appear identically in both $E(w,h')$ and $E(w,h)$ and therefore cancel out. In other words, only terms where either $j = v$ or $k = v$ survive the substraction. Therefore,

$$\Delta E = -\frac{1}{2}\sum_j w_{jv} h'_j h'_v - \frac{1}{2}\sum_k w_{vk} h'_v h'_k - b_v h'_v + \frac{1}{2}\sum_j w_{jv} h_j h_v + \frac{1}{2}\sum_k w_{vk} h_v h_k + b_v h_v \tag{11b}$$

Since the summation index is arbitrary and the weight matrix is symmetrical ($w_{jk} = w_{kj}$), some of the sums are identical to each other:

$$= -\sum_k w_{vk} h'_v h'_k - b_v h'_v + \sum_k w_{vk} h_v h_k + b_v h_v \tag{11c}$$

Since $w_{vv} = 0$, we can restrict the sums to $k \neq v$

$$= -\sum_{k \neq v} w_{vk} h'_v h'_k - b_v h'_v + \sum_{k \neq v} w_{vk} h_v h_k + b_v h_v \tag{11d}$$

For $k \neq v$, $h'_k = h_k$ and so

$$= -\sum_{k \neq v} w_{vk} h'_v h_k - b_v h'_v + \sum_{k \neq v} w_{vk} h_v h_k + b_v h_v \tag{11e}$$

$$= -(h'_v - h_v) \left(\sum_{k \neq v} w_{vk} h_k + b_v \right) \tag{11f}$$

If the activity of unit v (h_v) didn't change, $(h'_v - h_v) = 0$. Otherwise h'_v and h_v have opposite signs, which means that the first factor $(h'_v - h_v)$ has the same sign as h'_v. The second factor is the summed input to unit v, and is either zero or has the same sign as h'_v (as per Eq. 3). Therefore, ΔE is either zero or both factors in ΔE have the same sign, which means $\Delta E < 0$. Hence,

$$\Delta E \leq 0 \tag{11g}$$

This means that the energy function monotonically decreases. Since there is only a finite number of states, the dynamics of state changes must eventually converge. Note that this proof does not say which pattern the dynamics converges to. We will see in the capacity derivation below that the stored patterns are in fact attractor states of the network.

13.4 Capacity

The first step is to state clearly what is meant by the *memory capacity* of the Hopfield net. It is the maximum number of patterns n that can be stored in the network. This statement in turn has to be made explcit. To say that the network has stored n patterns, at the very least, the following has to be true of all the n patterns: When initialized with a stored pattern x^μ, the updates of the Hopfield net converge to a pattern that is very similar to that stored pattern. We therefore study the probability that a given unit's activity remains faithful, i.e. $h_j = x^\mu_j$. Then we will only state the results for the retrieval of the entire pattern and not derive it because the derivation is quite involved and outside the scope of this class.

We assume n patterns are stored in the network with m units. Since the considerations below are based on statistical arguments, m has to be large. We define the *load* parameter $\alpha = n/m$. Assume that there is no biasing input, i.e., $b = 0$. Use x^μ as input, then the update rule is

$$h_j = Q\left(\sum_k w_{jk} x^\mu_k \right) \tag{12a}$$

$$= Q\left(\frac{1}{n} \sum_{k \neq j} \sum_{i=1}^{n} x^i_j x^i_k x^\mu_k \right) \tag{12b}$$

We split the inner sum into two terms and note that $x^i_j x^i_k x^\mu_k = x^\mu_j$ for $i = \mu$

$$= Q\left(\frac{1}{n} \sum_{k \neq j} \left(x^\mu_j + \sum_{i \neq \mu} x^i_j x^i_k x^\mu_k \right) \right) \tag{12c}$$

$$= Q\left(\frac{m-1}{n} x^\mu_j + \frac{1}{n} \sum_{k \neq j} \sum_{i \neq \mu} x^i_j x^i_k x^\mu_k \right) \tag{12d}$$

We define the cross term:

$$C^\mu_j = \sum_{k \neq j} \sum_{i \neq \mu} x^i_j x^i_k x^\mu_k \tag{12e}$$

If the cross term has the same sign as x_j^μ, then h_j has the same sign, too. If the cross term has the opposite sign to x_j^μ, and $|C_j^\mu| > m-1$, then the cross term will flip the sign of h_j. Depending on the sign of x_j^μ, the probability of that flip is equal to

$$P(C_j^\mu > m-1) \text{ or } P(C_j^\mu < -(m-1)) \tag{13}$$

which are equal since C_j^μ is symmetric. To compute this probability, we will assume that $x_j^i x_k^i x_k^\mu \in \{-1,1\}$ is a random variable.

Aside: Central Limit Theorem Given random variables v_i with mean μ and variance σ^2, and large n, then the random variable

$$u = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{14}$$

is normally distributed with mean μ and variance σ^2/n, i.e.,

$$u \sim N\left(\mu, \frac{\sigma^2}{n}\right) \tag{15}$$

Approximating Eq. 12e as

$$C = mn\frac{1}{mn}\sum_{j=1}^{mn} x_j \tag{16}$$

where $x_j \in \{-1,1\}$, $\mu(x_j) = 0$ and $\sigma^2 = 1$, and using the central limit theorem, we obtain

$$C \sim N(0, mn) \tag{17}$$

where the variance is the product of $(mn)^2$ due to the coefficient, and $\frac{1}{mn}$ as per central limit theorem.

$$C' = \frac{1}{\sqrt{mn}}C \sim N(0,1) \tag{18}$$

$$P(C_j^\mu > m-1) \approx P(C > m) = P\left(C' > \sqrt{\frac{m}{n}}\right) = P\left(C' > \frac{1}{\sqrt{\alpha}}\right) = \int_{\frac{1}{\sqrt{\alpha}}}^{\infty} N(x)dx \tag{19}$$

So, the probability of a flip depends only on the load parameter α. The probability is plotted in Fig. 3. The flip

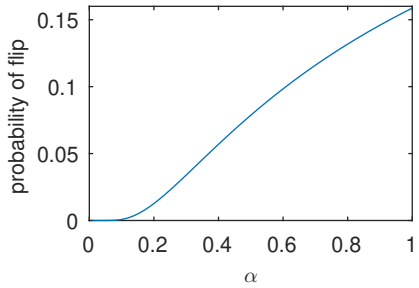

Figure 3: Probability that one unit is flipped incorrectly in one update step of the Hopfield net.

probability is quite low even for high loads of the network. However, the flip probability only looks at one update of an individual unit. Retrieval of the entire pattern requires many updates to different units and errors can accumulate quickly. So, our intuition is that the flip probability of a single unit should be very low, so that the pattern can be retrieved with a high probability. From Fig. 3 we might guess that a load of $\alpha \approx 0.15$ might be the maximum.

Unfortunately, the derivation of the retrieval probability of the entire pattern is rather difficult to follow. We therefore only report the final result in Fig. 4 and note that the theoretical result for the critical value is $\alpha_c = 0.138$ – quite close to our intuitive guess based on the flip probability.

The steep increase of the error probability to 50% in Fig. 4 for $\alpha > \alpha_c = 0.138$ indicates *catastrophic interference*: adding just a few more patterns beyond its capacity will render the network unable to retrieve any patterns correctly. So, the maximum number of patterns that can be stored in a Hopfield net is $n_{max} = 0.138m$. Note that the capacity estimate is based on stochastic arguments, so stable solutions might exist for $\alpha > \alpha_c$ for certain sets of patterns.

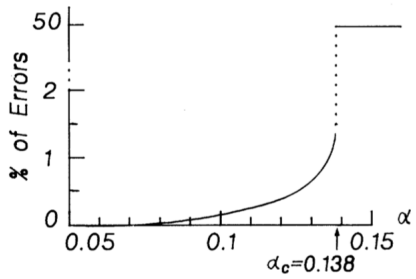

Figure 4: Probability of error in Hopfield net as a function of load (α). Source: Fig. 1, Amit et al. (1985)

References

Amit, D. J., Gutfreund, H., & Sompolinsky, H. (1985). Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks. *Phys. Rev. Lett.*, *55*(14), 1530–1533.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci U A*, *79*(8), 2554–2558.