Topics: Hopfield Networks - 28.11.2019
Lecturer: Dr. Matthew Cook
Author: Vanessa Leite {vanessa at ini.uzh.ch}

1 Hopfield Networks

The Hopfield Network, or Hopfield Model, was proposed by John Hopfield in the early 80s. He thought about connecting units to each other in an all-to-all pattern. When we look into the brain and how neurons are connected to each other, we do not see always a clear pathway, i.e., is hard to think about the connections in the brain in terms of input-output.

Characteristics of Hopfield Networks

- Every node is connected to every other node but not to itself.
- Connection weights are symmetric.
- $\sum x_i w_i < 0$ represents an inactive unit, $\sum x_i w_i \geq 0$ is active, when using a zero threshold.
- The entire network is in some state at any time. Set of active units of the entire network is important.
- Some states are stable and some are not. While in an unstable state, updating the network leads to a state change.
- Stable state is a local minimum.
- Bias is an unit that is always on.
- Weight symmetry of a connection is correlated to the frequency of neurons firing together (following Hebbian learning).

1.1 Hopfield and Memory

We are used to think about computation as a process that receives an input, does something and generates an output. However, this is not what we "see" in the memorie process, for instance. It seems our memory works with Pattern Completion, also known as Content Addressable Memory or Associative Memory. This memory has no input-output relation: given any piece of it, we can recover the rest. An example is when certain smell can bring us a memory, even though the smell is only part of the memory. Hopfield Networks can be used to give us insights about how the memory works by having a highly connected network that computes with no input-output, but with states.

- A Hopfield Network is an associative type of memory. Information is stored in the stable states as local minima.
- It is important that information is distinct.
- Associative memory has room for error but is still recognizable. Convergence to nearby stable states.
- If some units are retrievable and all others are set randomly, the correct units will eventually set the wrong units right.

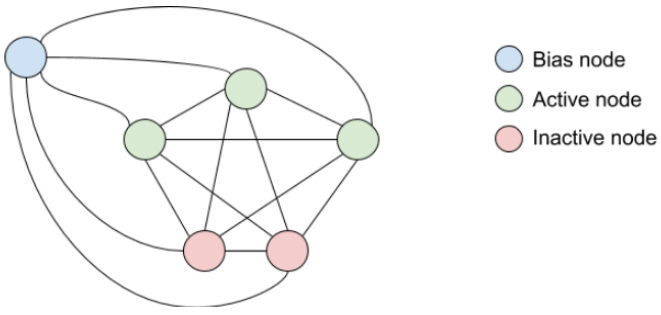

Figure 1: Model of a Hopfield Network. Red units are inactive, green units are active, blue unit is the bias node that is always active.

1.2 Updates and State Dynamics

Hopfield found that these networks always converge to a stable state. Idea: Consider the sum of weights between active units. The update rule is equivalent to always increase this quantity, i.e., we turn an unit active if $\sum x_i w_i >= \theta$, where x_i is the value of the state of each unit, w_i is the weight between the units and θ is the threshold, thus maximizing the sum of active weights (weights between active units) by updating the output of one unit at time. In this setup, the update algorithm of Hopfield Networks can be seen as a greedy algorith to find the MAX-CLIQUE. We update all the units but the bias node (bias node are always active and do not change their state).

Remember, here, inactive neurons don't send inhibitory signal, instead they do not take part in the activation of other neurons. We can also consider active units as having a value of "+1" and inactive units as a "−1". In this new representation, the Hopfield Network dynamic is equivalent to a graph min-cut, i.e., we want to minimize the sum of weights that link active and inactive units.

When we update one unit at time, we are using an asynchronous method. Asynchronous Hopfield Networks always converge to a stable state, regardless of the update order, however, to which stable state is dependent on the update order, see Figure 2.

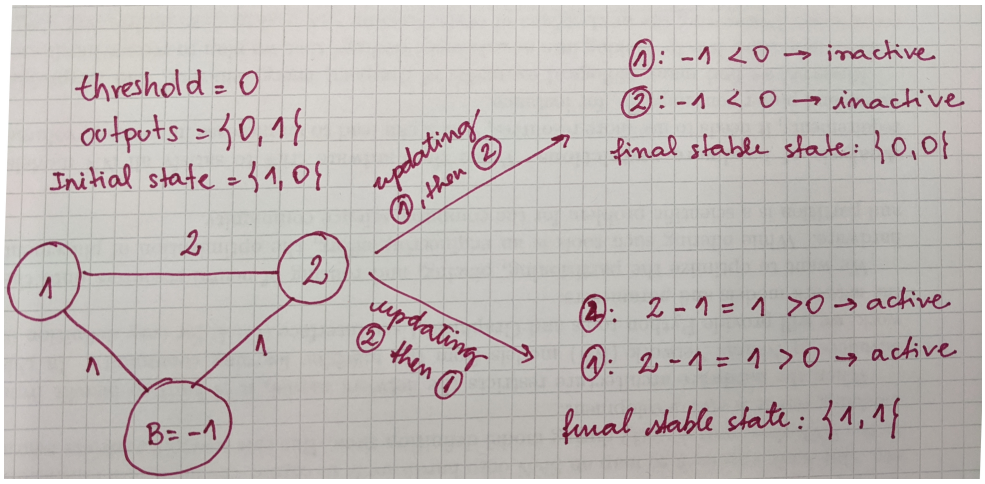

Figure 2: Considering a Hopfield network with two units connected with weight = 2, threshold zero, and a bias unit of −1. Starting in a state with 1 active and 2 inactive, we can see how the order of update can lead to different stable states. By updating first unit 1 then unit 2, the stable state is with both units inactive. However, updating unit 2 then unit 1, we arrive to a stable state where both units are active.

Let's now consider a synchronous case, i.e., when we update all units at the same time: for this, active units for time $t + 1$ are computed based on active units at time t. Consider the network on Figure 3, starting with units 1 and 2 active, in the next step of update (after all units), unit 3 and 4 should be active. It may be confusing to see this by updating one unit per time, let's say we update units in the order$1, 2, 3, 4$: after updating unit 3 it will become active but, when updating unit 4, we shouldn't consider unit 3, yet.

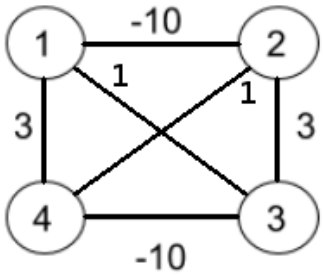

Figure 3: Starting with 1 and 2 actives, in the next step 3 and 4 will be active. This network doesn't have a unique stable state but converges to a cycle between two states.

Trick for analysis: make a larger asynchronous network based on the network we want to analyse. Duplicate the units in two columns and only use non-zero weights between them.

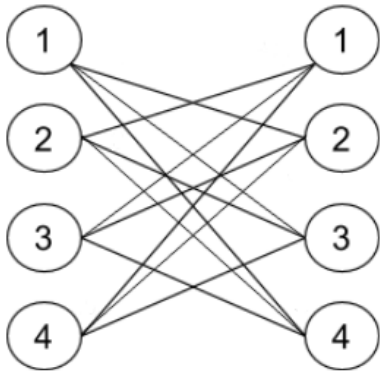

Figure 4: Starting with 1 and 2 actives, we will end up with 3 and 4 active in the second column. First column represents $t = 0$, second $t = 1$. With synchronous updates, a Hopfield Network converges to a cycle of length 2 or to a stable state.

- Nodes can be updated synchronously or asynchronously.
- State: Set of units that are active, it can be seen as the activity vector.
- Dynamics: Units update their activity level.
- When a node is updated, weights are considered from all other active nodes, like with a perceptron.
- Asynchronous updates (greedy algorithm) converge to a stable state (sequential), but the converged state can depend on update order.
- Asynchronous is either in max-clique state if activity is in $\{0, 1\}$ or min-cut if activities are in $\{-1, 1\}$.
- Synchronous, parallel updates either also go to a stable state, just like asynchronous, or can get stuck in a pair of patterns (flipping or cyclic).