Neural Networks

Homework 4

Juan Jose Mendez Torrero (s3542416)
Sharif Hamed (s2562677)

May 22, 2018

1 Theory questions

1. The two key features are:
   (a) $W_{ii} = 0$, this means that each node does not have a connection with itself.
   (b) $W_{ij} = W_{ji}$, this means that the connections between the nodes are symmetric.
2. The activation of a neuron in a Hopfield network is computed by the multiplication between the neuron vector $\vec{x}$ and the weight vector $\vec{w}$. The size of the vector $\vec{x}$ is equal to the number of neuron, this values can be 1 for an active neuron, or 0 for an inactive neuron. On the other hand, the values of the vector $\vec{w}$ can take any value. These are the weight of the connections between neurons.
3. The type of function used in a Hopfield network is the step function. This means that the neuron will be fired when it gets a value greater or equal to 0. However, if it gets a value smaller than 0, the neuron will not be fired.
4. If we take into account the answer to the previous question, we have assumed that a Hopfield network contains TLU neurons, where the activation is computed the same ways as we have described above.
5. A network with n nodes can store roughly one or two order of magnitude less then n. The following equation describes the ratio:
$$numberOfPatterns < \frac{n}{2\ln n}$$
6. When we look at the queens problem and we choose to use a hopfield network we know the following energy function that solves the TSP:
$$E_1 = \frac{A}{2}\sum_{X}^{N}\sum_{i}^{N}\sum_{j\neq i}^{N} X_{X,i}X_{X,j} + \frac{B}{2}\sum_{i}^{N}\sum_{X}^{N}\sum_{Y\neq X}^{N} X_{X,i}X_{Y,i} + \frac{C}{2}\left(\sum_{X}^{N}\sum\nolimits_{i}^{N} X_{X,i} - N\right)^2$$
This energy function is almost what we want since if we want to solve the 8 queens problem we want to solve the following matrix:
$$\left\{\begin{matrix} 0&0&0&0&0&1&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&1&0&0&0&0&0 \end{matrix}\right\}$$

This matrix is already a solution. A Queen is positioned at every one in the matrix. The queens may not be able to attack each other. The energy function(equation 1) is already made in such a way that makes sure that in every row and column only one queen can be positioned. Now we need to make a additional term that takes care of the diagonals.

$$E_1 = \frac{A}{2}\sum_X^N\sum_i^N\sum_{j\neq i}^N X_{X,i}X_{X,j} + \frac{B}{2}\sum_i^N\sum_X^N\sum_{Y\neq X}^N X_{X,i}X_{Y,i} + \frac{C}{2}\left(\sum_X^N\sum\nolimits_i^N X_{X,i} - N\right)^2 +$$

$$+\frac{D}{2}\sum_X^N\sum_{Y\neq X}^N\sum_i^N\sum_{j=i+(X-Y)^2}^N X_{X,i}X_{Y,j}$$

The extra factor in the second equation is only zero when on every diagonal there is maximal 1 non zero entry.

2 Hopfield network on paper

1. In order to calculate the activation of the network, we have to know that $y_i = \Sigma_k w_{ik}\cdot x_i$, being y_i the activation of the *i-th* neuron.Now we know this, we are going to calculate, first using synchronous, and second using asynchronous update.

   (a) **Synchronous:**
   $y_1 = (x_1 * w_1 + x_2 * w_2 + x_3 * w_3) = (0*0+0*1+0*-1) = 0$
   $y_2 = (0*1+0*0+0*-2) = 0$
   $y_3 = (0*-1+0*-2+0*0) = 0$
   As we can observe, all the activation are greater or equal to 0, hence, the update vector $\vec{x} = [1,1,1]$. Now, we are going to repeat these steps until the vector does not change anymore.
   $y_1 = (1*0+1*1+1*-1) = 0$
   $y_2 = (1*1+1*0+1*-2) = -1$
   $y_3 = (1*-1+1*-2+1*0) = -3$
   Like before, we have to update the vector, this time, only x_3 is not going to be fired. Hence, the vector $\vec{x} = [1,0,0]$. We have to repeat the process again.
   $y_1 = (1*0+0*1+0*-1) = 0$
   $y_2 = (1*1+0*0+0*-2) = 1$
   $y_3 = (1*-1+0*-2+0*0) = -1$
   Now the updated vector is $\vec{x} = [1,1,0]$. We repeat the process:
   $y_1 = (1*0+1*1+0*-1) = 1$
   $y_2 = (1*1+1*0+0*-2) = 1$
   $y_3 = (1*-1+1*-2+0*0) = -3$
   As we can observe, we have reached the equilibrium of the network with $\vec{x} = [1,1,0]$.

   (b) **Asynchronous:**This time, we are going to update the vector $\vec{x}$ each time that we had calculated the activation.
   $y_1 = (x_1 * w_1 + x_2 * w_2 + x_3 * w_3) = (0*0+0*1+0*-1) = 0 \longrightarrow \vec{x} = [1,0,0]$
   $y_2 = (1*1+0*0+0*-2) = 1 \longrightarrow \vec{x} = [1,1,0]$
   $y_3 = (1*-1+1*-2+0*0) = -3 \longrightarrow \vec{x} = [1,1,0]$
   We repeat again the vector until it does not change anymore. $y_1 = (1*0+1*1+0*-1) = 1 \longrightarrow \vec{x} = [1,1,0]$
   $y_2 = (1*1+1*0+0*-2) = 1 \longrightarrow \vec{x} = [1,1,0]$
   $y_3 = (1*-1+1*-2+0*0) = -3 \longrightarrow \vec{x} = [1,1,0]$
   As we can observe, we have reached the equilibrium for this network with an asynchronous update.

2. In order to use the Hebbian learning, we have to know that $\Delta w = \alpha \cdot v_i \cdot v_j$. After this, we have to iterate throught the network three times.

   (a) First iteration:
   $w_1 = 0,5 \cdot -1 \cdot 1 = -0,5$, hence, $\Delta w_1 = w_1 - 0,5 = 0,5$
   $w_2 = 0,5 \cdot 1 \cdot -1 = -0,5$, hence, $\Delta w_2 = w_2 - 0,5 = -2,5$
   $w_3 = 0,5 \cdot -1 \cdot -1 = 0,5$, hence, $\Delta w_3 = w_3 + 0,5 = -0,5$

   (b) Second iteration:
   $\Delta w_1 = w_1 - 0,5 = 0$
   $\Delta w_2 = w_2 - 0,5 = -3$
   $\Delta w_3 = w_3 - 0,5 = 0$

   (c) Third iteration:
   $\Delta w_1 = w_1 - 0,5 = -0,5$
   $\Delta w_2 = w_2 - 0,5 = -3,5$
   $\Delta w_3 = w_3 - 0,5 = 0,5$
   Now we can observe that the resulted weight matrix is now:

0	-0,5	0,5
-0,5	0	-3,5
0,5	-3,5	0

3. As before, we are going to apply the synchronous update to get the equilibrium for the network. Hence, we are going to calculate, first the activation of all the inputs, and then we are going to update them.
   $y_1 = (x_1 * w_1 + x_2 * w_2 + x_3 * w_3) = (0 * 0 + 0 * -0,5 + 0 * 0,5) = 0$
   $y_2 = (0 * -0,5 + 0 * 0 + 0 * -3,5) = 0$
   $y_3 = (0 * 0,5 + 0 * -3,5 + 0 * 0) = 0$
   As we observe, the activation for all the nodes will be fired, hence we know that the input vector now will be : $\vec{x} = [1, 1, 1]$. We can see, as well, that the network does not reach the equilibrium, so we are going to continue with the updating.
   $y_1 = (1 * 0 + 1 * -0,5 + 1 * 0,5) = 0$
   $y_2 = (1 * -0,5 + 1 * 0 + 1 * -3,5) = -4$
   $y_3 = (1 * 0,5 + 1 * -3,5 + 1 * 0) = -3$
   Likewise, we update the vector $\vec{x}$ into $[1, 0, 0]$. We do not reach the equilibrium, hence we have to continue.
   $y_1 = (1 * 0 + 0 * -0,5 + 0 * 0,5) = 0$
   $y_2 = (1 * -0,5 + 0 * 0 + 0 * -3,5) = -0,5$
   $y_3 = (1 * 0,5 + 0 * -3,5 + 0 * 0) = 0,5$
   We observe that now, the updated vector will be $\vec{x} = [1, 0, 1]$. As before, we do not reach the equilibrium, hence we have to keep updating.
   $y_1 = (1 * 0 + 0 * -0,5 + 1 * 0,5) = 0$
   $y_2 = (1 * -0,5 + 0 * 0 + 1 * -3,5) = -4$
   $y_3 = (1 * 0,5 + 0 * -3,5 + 1 * 0) = 0,5$
   Now, the updated vector is $\vec{x} = [1, 0, 1]$. As we can observe, we have reached the equilibrium for the network.

3 Essay questions

1. Imagine that we have an noisy input, if we use a feedworard network, it will be so difficult to the network to be right on the output. That means that if our output has a lot of noise, it will be a better option to use a RNN. This is due to the way that RNN works. The input is evaluated inside a "loop" that compute and update the weight between the connections, being this method better when we have a noisy input.

2. The way that BTT works is:

    (a) First, it creates a stack with identical copies of the RNN.
    (b) Secondly, it redirects all the connections within the network, in order to obtain new connection between the subsequent copies.
    (c) Finally, the RNN now looks like a feedback network, hence, we can apply the backpropagation algorithm.

    This type of approach may be it is not ideal, that is due to the complexity of the algorithm($O(TN^2)$, being N the number of units.), that means, the process time will be greater that if we use RNN. Hence, we can say that BPTT is not a feasible approach.

3. The main difference between ESN and Hopfield networks is the way they train their inputs and how they generate their output. One of that differences is that the ESN "forces" their output to get a value similar or equal to the desired output. This step is called "teacher-forced". Both of them use a RNN. Furthermore, the ESN training add normalizes the weights with an spectral ratio, which gave to the untrained network an echo state network.