CECS 590 – Fall 2018 - Assignment 4

Akshatha Thonnuru Swamygowda & Sarveshwaran Sampathkumar

1. Provide a brief comparison of an RBM and a Hopfield network. What is common and what is different?

Hopfield Network

A Hopfield network is a recurrent neural network having synaptic connection pattern such that there is an underlying Lyapunov function for the activity dynamics. The units in Hopfield nets are binary threshold units, i.e. Hopfield nets normally have units that take on values of 1 or -1. The pair of units connected have a relation by weight which are represented by wij. Thus, Hopfield network can be described as a complete undirected graph, with one or more fully connected recurrent neurons.

Restricted Boltzmann Machines

Helmholtz and Boltzmann machines are stochastic networks, meaning that given an input, the state of the network does not converge to a unique state, but to an ensemble distribution. RBMs are two-layer neural networks that consist of the building blocks on the deep belief networks. The first layer of the RBM is called the visible or the input layer whereas the second layer is called as the hidden layer.

Similarities

No self-connections in both Hopfield networks and RBM. The connections in both Hopfield networks and RBMs are symmetric.

Differences

RBM has one layer of visible units, one layer of hidden units and a bias unit, whereas Hopfield net has one layer of visible units and no hidden layer. Hopfield network suffers from false local minima that form on the energy hypersurface whereas RBM avoid false solutions by adding hidden nodes. Hopfield networks are usually fully connected, whereas RBM can be stacked.

2. Does RBM represent the unsupervised learning? Explain your point of view.

Yes, RBMs represent unsupervised learning. RBMs are usually trained using the contrastive divergence learning procedure given by Hinton in 2002. This requires a certain amount of practical experience to decide how to set the values of numerical meta-parameters such as the learning rate, the momentum, the weight-cost, the sparsity target, the initial values of the weights, the number of hidden units and the size of each mini-batch. There are also decisions to be made about what types of units to use, whether to update their states stochastically or deterministically, how many times to update the states of the hidden units for each training case, and whether to start each sequence of state updates at a data-vector. In addition, it is useful to know how to monitor the progress of learning and when to terminate the training. For any application, the code that was used gives a complete specification of these decisions, but it does not explain why the decisions were made or how minor changes will affect performance. More significantly, it does not provide a novice user with any guidance about how to make good decisions for a new application. This requires some sensible heuristics and the ability to relate failures of the learning to the decisions that caused those failures. Thus, RBM represents unsupervised learning as there are no ground-truth labels in the test set. And the weights must be taken randomly and for the error to be minimum the number of iterations taken can be large.

3. What will be likelihood of setting a state ON if the energy is positive and high? Explain the following statement: "units that are positively connected to each other try to get each other to

share the same state, while units that are negatively connected to each other are enemies that prefer to be in different states".

The probability $p_i = \sigma(a_i)$ where function $\sigma(x) = 1/(1+e^{-x})$ is the logistic function. Here, the activation energy $a_i = \Sigma\, w_{ij} * x_i$
The probability of setting a state ON if the energy is positive and high is the sigmoidal function of the activation energy. Hence, unit i is set ON with a probability p_i.
The probability p_i is the sigmoid function of the activation energy, thus p_i is close to 1 for large positive activation energies and close to 0 for negative activation energies. Thus, we turn unit i ON with probability p_i and turn it OFF with probability $1-p_i$.

4. What is purpose of adding Positive(eij) - Negative(eij) to each edge weight to assure convergence?

Positive(eij) is the association between the ith and jth unit that the network learns from the training examples. Negative(eij) generates when no units are fixed to training data. They along with the product with the learning rate give an idea about how close are we to reduce the error and get it below a certain threshold so that the weights can be updated to the appropriate values. So as the error decreases, Positive(eij) - Negative(eij) gives the assurance of reaching convergence as by adding it to every edge it better matches the reality of the training examples.

5. Consider a training set of inputs: A(1,1,1,0,0,0), B(1,0,1,0,0,0), C(1,1,1,0,0,0), D(0,1,1,1,0,0), E(0,0,1,1,1,0), F(0,0,1,1,1,0).
Explain how the RBM algorithm is applied to this case. Perform a few steps to illustrate the algorithm.

Let us consider, x 1 = A (1, 1, 1, 0, 0, 0)
Let all the weights be w ij =1
And let there be 2 hidden units.
In this example, there are 6 input nodes and 2 hidden nodes. The weights of the connections between the nodes is assumed to be 1.

For the RBM algorithm, the first step is to calculate the activation energies of the hidden nodes which is done as follows:
Activation Energy for the first hidden unit i.e. j=1
a 1 = Σ w ij * x i = 1*1+1*1+1*1+1*0+1*0+1*0 = 3
For second hidden unit, i.e. j=2
a 2 = Σ w ij * x i = 1*1+1*1+1*1+1*0+1*0+1*0 = 3

The next step is to calculate the probabilities using the result of the activation energy. Probability p i is the sigmoidal function of the activation energy.
p 1 = 1 / (1 + exp (-3)) = 1
p 2 = 1 / (1 + exp (-3)) = 1

Now, for the visible units, the activation energy is calculated with a backward pass i.e. sending the input from the hidden units to the visible units.
a 1 = Σ w ij * x i = 1*1+1*1 = 2
a 2 = Σ w ij * x i = 1*1+1*1 = 2
a 3 = Σ w ij * x i = 1*1+1*1 = 2
a 4 = Σ w ij * x i = 1*1+1*1 = 2
a 5 = Σ w ij * x i = 1*1+1*1 = 2

a 6 = Σ w ij * x i = 1*1+1*1 = 2

Probability p i for each of the input nodes is calculated using the activation energy calculated previously.
p 1 = 1 / (1 + exp (-2)) = 1
p 2 = 1 / (1 + exp (-2)) = 1
p 3 = 1 / (1 + exp (-2)) = 1
p 4 = 1 / (1 + exp (-2)) = 1
p 5 = 1 / (1 + exp (-2)) = 1
p 6 = 1 / (1 + exp (-2)) = 1

The x i and x j values are used to calculate the Positive (e ij) which are the measures of association between the i-th and j-th unit that the network should learn from the training examples.
Positive (e ij) = x i * x j

For the first hidden unit, i.e. j=1
Positive (e 11) = 1*1 = 1
Positive (e 21) = 1*1 = 1
Positive (e 31) = 1*1 = 1
Positive (e 41) = 0*1 = 0
Positive (e 51) = 0*1 = 0
Positive (e 61) = 0*1 = 0

For the second hidden unit, i.e. j=2
Positive (e 12) = 1*1 = 1
Positive (e 22) = 1*1 = 1
Positive (e 32) = 1*1 = 1
Positive (e 42) = 0*1 = 0
Positive (e 52) = 0*1 = 0
Positive (e 62) = 0*1 = 0

Using the new input x i, the activation energies of the hidden units are again calculated.
For j=1,
a 1 = 1*1+1*1+1*1+1*1+1*1+1*1 = 6
For j=2,
a 2 = 1*1+1*1+1*1+1*1+1*1+1*1 = 6
Also, the probabilities p i are calculated,
p 1 = 1 / (1 + exp (-6)) = 1
p 2 = 1 / (1 + exp (-6)) = 1

Further the Negative (e ij) is calculated which measures the association that the network itself generates when no units are fixed to training data.
Negative (e ij) = x i * x j

For the first hidden layer, i.e. j=1
Negative (e 11) = 1*1 = 1
Negative (e 21) = 1*1 = 1
Negative (e 31) = 1*1 = 1
Negative (e 41) = 1*1 = 1
Negative (e 51) = 1*1 = 1
Negative (e 61) = 1*1 = 1

For the second hidden layer, i.e. j=2

Negative (e 12) = 1*1 = 1
Negative (e 22) = 1*1 = 1
Negative (e 32) = 1*1 = 1
Negative (e 42) = 1*1 = 1
Negative (e 52) = 1*1 = 1
Negative (e 62) = 1*1 = 1

The Positive (e ij) and Negative (e ij) along with the learning rate L is used to update the weights w ij.
w ij =w ij +L∗ (Positive (e ij) – Negative (e ij))
Here, L is the learning rate, let's assume L = 0.5

For hidden layer 1, i.e. j=1,
w 11 = 1 + 0.5(1-1) = 1
w 21 = 1 + 0.5(1-1) = 1w 31 = 1 + 0.5(1-1) = 1
w 41 = 1 + 0.5(0-1) = 0.5
w 51 = 1 + 0.5(0-1) = 0.5
w 61 = 1 + 0.5(0-1) = 0.5

For hidden layer 2, i.e. j=2,
w 12 = 1 + 0.5(1-1) = 1
w 22 = 1 + 0.5(1-1) = 1
w 32 = 1 + 0.5(1-1) = 1
w 42 = 1 + 0.5(0-1) = 0.5
w 52 = 1 + 0.5(0-1) = 0.5
w 62 = 1 + 0.5(0-1) = 0.5

The above steps are repeated for the remaining training set until a minimum error is reached through appropriate weights for the network to give appropriate outputs.

The remaining training sets are:
x 2 = B (1,0,1,0,0,0),
x 3 = C (1,1,1,0,0,0),
x 4 = D (0,1,1,1,0,0),
x 5 = E (0,0,1,1,1,0),
x 6 = F (0,0,1,1,1,0).