RECURRENT NETWORKS
&
BOLTZMANN MACHINE

DR. KRISHNENDU GUHA

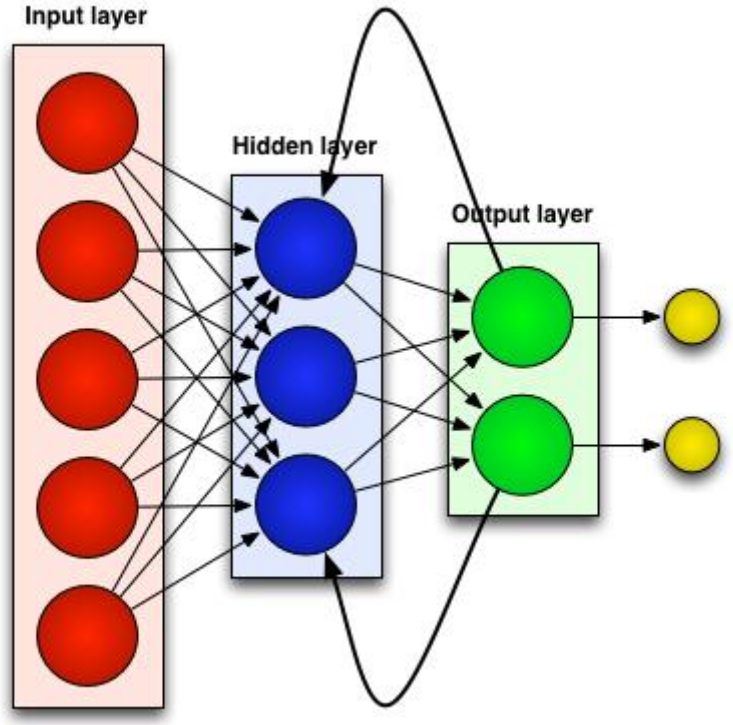
ASSISTANT PROFESSOR (ON CONTRACT)

NATIONAL INSTITUTE OF TECHNOLOGY (NIT), JAMSHEDPUR

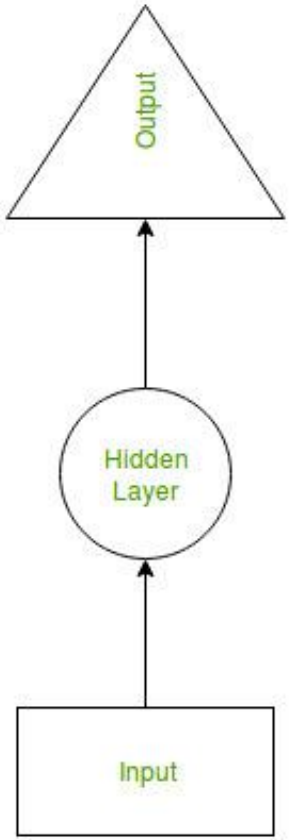
EMAIL: KRISHNENDU.CA@NITJSR.AC.IN

RECURRENT NEURAL NETWORKS (RNN) INTRODUCTION

- **RNN** are a type of Neural Network where the **output from previous step are fed as input to the current step**.
- In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.
- Thus RNN came into existence, which solved this issue with the help of a Hidden Layer.
- The main and most important feature of RNN is **Hidden state**, which remembers some information about a sequence.



- RNN have a **“memory”** which remembers all information about what has been calculated.
- It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output.
- This reduces the complexity of parameters, unlike other neural networks.

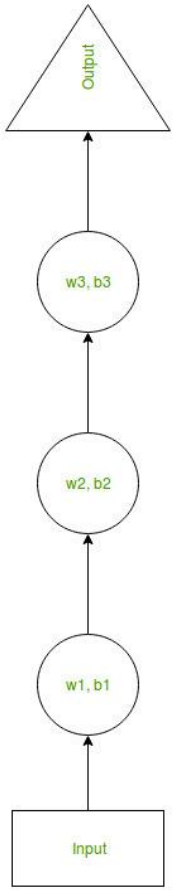


WORKING OF RNN (VIA EXAMPLE)

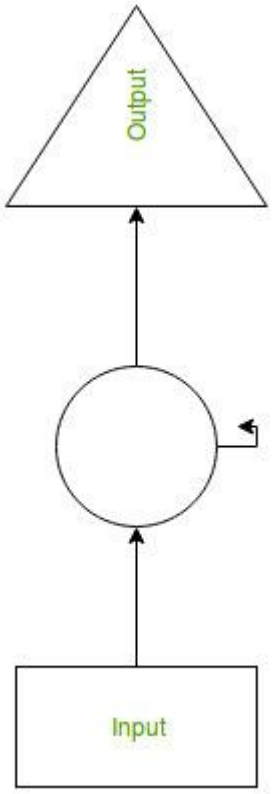
- Suppose there is a deeper network with one input layer, three hidden layers and one output layer.

- Then like other neural networks, each hidden layer will have its own set of weights and biases,
- let's say, for hidden layer 1 the weights and biases are (w1, b1),
- (w2, b2) for second hidden layer and
- (w3, b3) for third hidden layer.

- This means that each of these layers are independent of each other, i.e. they do not memorize the previous outputs.



- Now the RNN will do the following:
- RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer.
- Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer.



FORMULAS

- **Formula for calculating current state:** $h_t = f(h_{t-1}, x_t)$

  where:

  h_t -> current state

  h_{t-1} -> previous state

  x_t -> input state

- **Formula for applying Activation function(tanh):** $h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$

  where:

  w_{hh} -> weight at recurrent neuron

  w_{xh} -> weight at input neuron

- **Formula for calculating output** $y_t = W_{hy}h_t$

Y_t -> output

W_{hy} -> weight at output layer

TRAINING THROUGH RNN

1. A single time step of the input is provided to the network.
2. Then calculate its current state using set of current input and the previous state.
3. The current ht becomes ht-1 for the next time step.
4. One can go as many time steps according to the problem and join the information from all the previous states.
5. Once all the time steps are completed the final current state is used to calculate the output.
6. The output is then compared to the actual output i.e the target output and the error is generated.
7. The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained.

ADVANTAGES AND DISADVANTAGES OF RNN

- **Advantages**

1. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighborhood.

- **Disadvantages**

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

BOLTZMANN MACHINE INTRODUCTION

- These are stochastic learning processes having recurrent structure and
- are the basis of the early optimization techniques used in ANN.

- Boltzmann Machine was invented by Geoffrey Hinton and Terry Sejnowski in 1985.

- Feature of this network is that it uses only locally available information.
- The change of weight depends only on the behavior of the two units it connects, even though the change optimizes a global measure

IMPORTANT POINTS ABOUT BOLTZMANN MACHINE

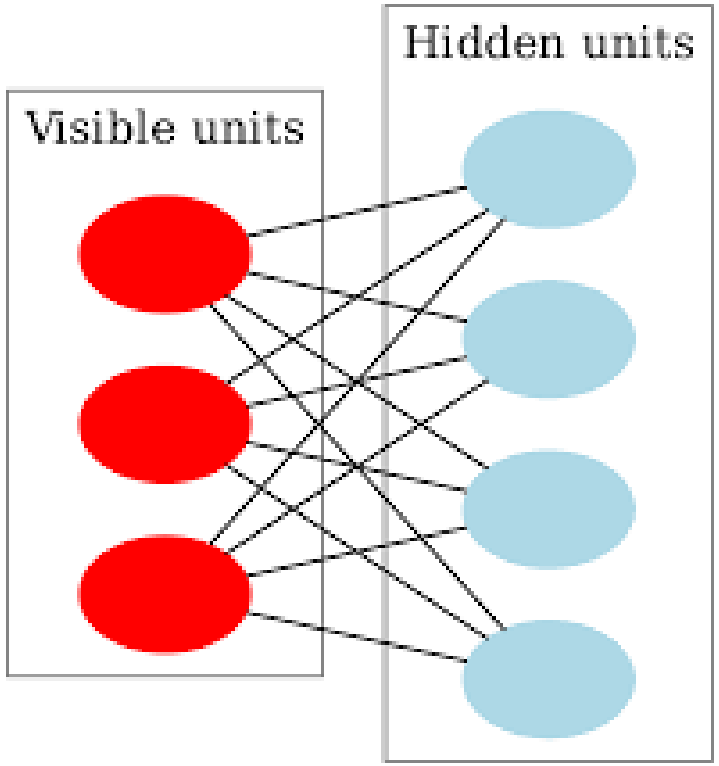
•They use recurrent structure.

•They consist of stochastic neurons, which have one of the two possible states, either 1 or 0.

•Some of the neurons in this are adaptive freestate and some are clamped frozenstate.

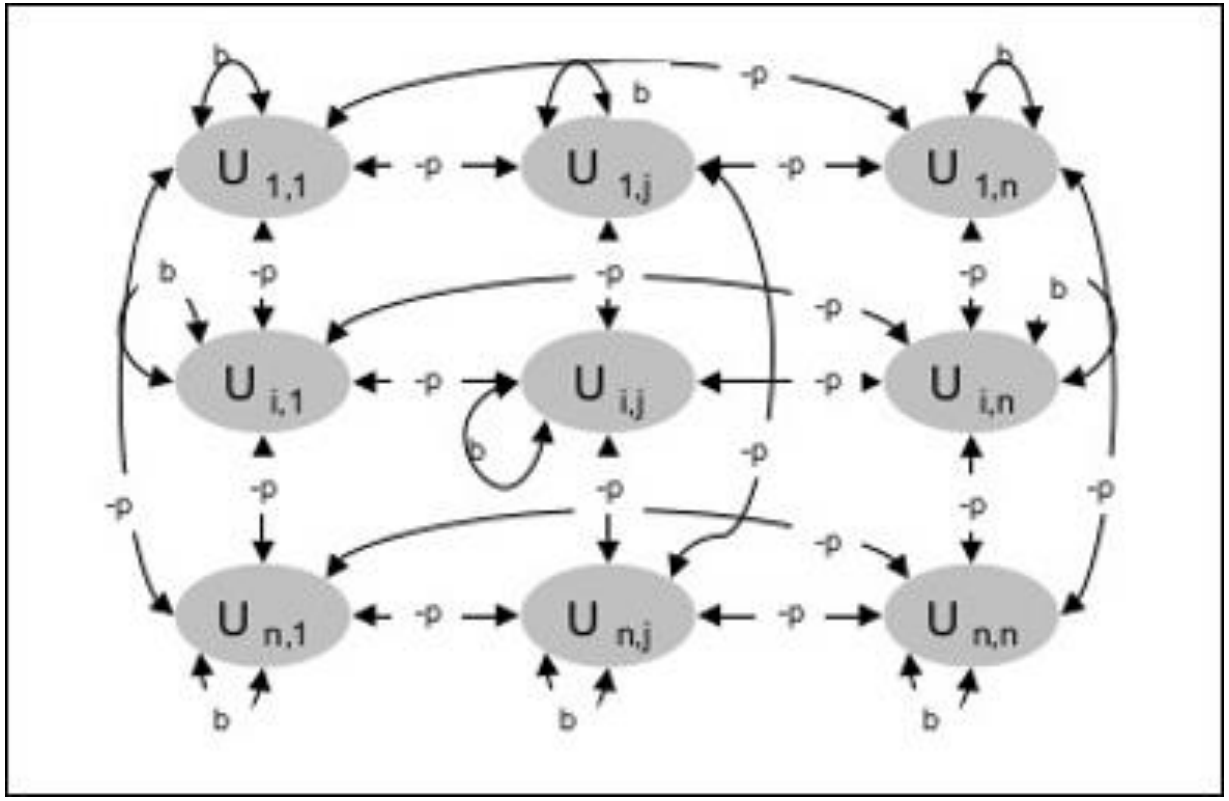
•If we apply simulated annealing on discrete Hopfield network, then it would become Boltzmann Machine.

•The main purpose of Boltzmann Machine is to optimize the solution of a problem.
•It is the work of Boltzmann Machine to optimize the weights and quantity related to that particular problem



ARCHITECTURE

- it is a two-dimensional array of units.
- Here, weights on interconnections between units are **–p** where **p > 0**.
- The weights of self-connections are given by **b** where **b > 0**.



TRAINING AND TESTING FOR BOLTZMANN MACHINE

As we know that Boltzmann machines have fixed weights, hence there will be no training algorithm as we do not need to update the weights in the network.

However, to test the network we have to set the weights as well as to find the consensus function CF.

Boltzmann machine has a set of units $\mathbf{U_i}$ and $\mathbf{U_j}$ and has bi-directional connections on them.
- We are considering the fixed weight say $\mathbf{w_{ij}}$.
- $\mathbf{w_{ij} \neq 0}$ if $\mathbf{U_i}$ and $\mathbf{U_j}$ are connected.
- There also exists a symmetry in weighted interconnection, i.e. $\mathbf{w_{ij} = w_{ji}}$.
- $\mathbf{w_{ii}}$ also exists, i.e. there would be the self-connection between units.
- For any unit $\mathbf{U_i}$, its state $\mathbf{u_i}$ would be either 1 or 0.

The main objective of Boltzmann Machine is to maximize the Consensus Function CF which can be given by the following relation

$$CF = \sum_{i} \sum_{j \leqslant i} w_{ij} u_i u_j$$

- Now, when the state changes from either 1 to 0 or from 0 to 1, then the change in consensus can be given by the following relation −

$$(1 - 2u_i) = \begin{cases} +1, & U_i \; is \; currently \; off \\ -1, & U_i \; is \; currently \; on \end{cases}$$

- Here $\mathbf{u_i}$ is the current state of $\mathbf{U_i}$.
- The variation in coefficient ($\mathbf{1 - 2u_i}$) is given by the following relation −

$$(1 - 2u_i) = \begin{cases} +1, & U_i \; is \; currently \; off \\ -1, & U_i \; is \; currently \; on \end{cases}$$

- Generally, unit $\mathbf{U_i}$ does not change its state, but if it does then the information would be residing local to the unit. With that change, there would also be an increase in the consensus of the network.
- Probability of the network to accept the change in the state of the unit is given by the following relation −

$$AF(i, T) = \frac{1}{1 + exp[-\frac{\Delta CF(i)}{T}]}$$

- Here, **T** is the controlling parameter. It will decrease as CF reaches the maximum value.

- Testing Algorithm
- **Step 1** − Initialize the following to start the training −
- Weights representing the constraint of the problem
- Control Parameter T
- **Step 2** − Continue steps 3-8, when the stopping condition is not true.
- **Step 3** − Perform steps 4-7.
- **Step 4** − Assume that one of the state has changed the weight and choose the integer **I, J** as random values between **1** and **n**.
- **Step 5** − Calculate the change in consensus as follows −

$$\Delta CF = (1 - 2u_i)(w_{ij} + \sum_{j \neq i} u_i w_{ij})$$

- **Step 6** − Calculate the probability that this network would accept the change in state

$$AF(i,T) = \frac{1}{1 + exp[-\frac{\Delta CF(i)}{T}]}$$

- **Step 7** − Accept or reject this change as follows −
- **Case I** − if **R < AF**, accept the change.
- **Case II** − if **R ≥ AF**, reject the change.
- Here, **R** is the random number between 0 and 1.
- **Step 8** − Reduce the control parameter temperature as follows:

$$T\ new = 0.95T\ old$$

- **Step 9** − Test for the stopping conditions which may be as follows −
- Temperature reaches a specified value
- There is no change in state for a specified number of iterations


thank
you