Recurrent Neural Networks

⇒ RNNs are very powerful, because they combine two properties

1. Destributed hidden state that allows them to store a lot of information about the past efficiency
2. Non-linear dynamics that allows them to update their hidden state in complicated ways.

⇒ with enough neurons and time, RNNs can compute anything that can be computed by your computer.

Need for RNN:

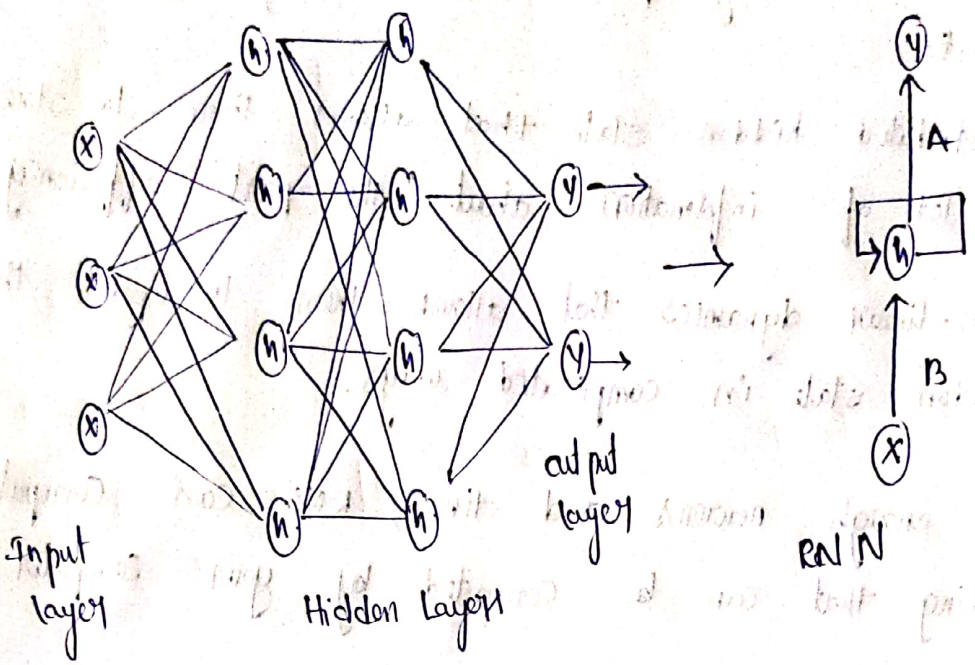
⇒ Normal Networks cannot handle sequential data

⇒ They considers only the current input

⇒ Normal NN cannot memorize previous inputs

⇒ RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

Converting a Full Network into Recurrent Network

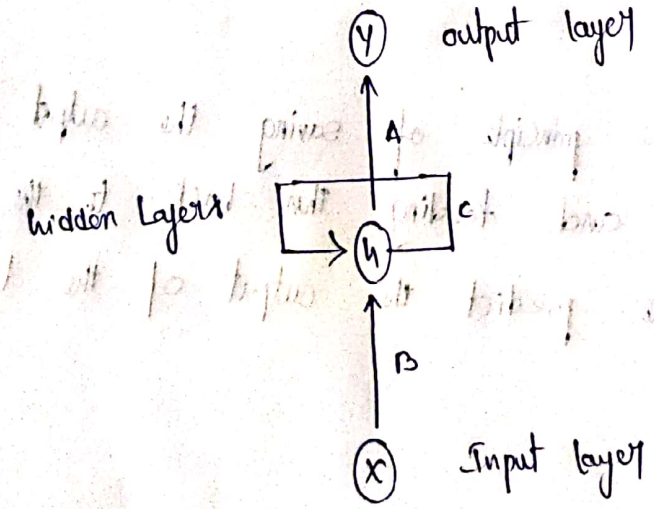


x - input layer

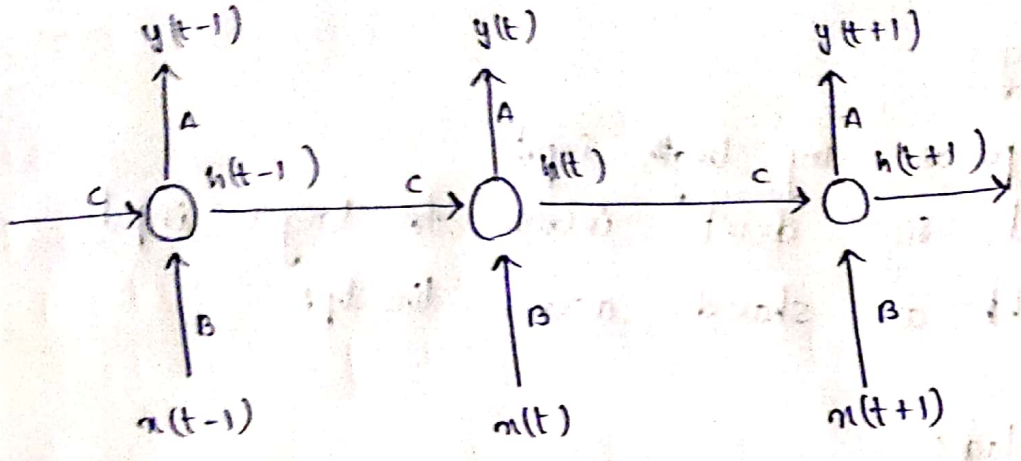
h:- hidden layer

Y - output layer

A, B, C are the network parameters



A) Recurrent Network



B) Fully Connected RNN

$$h(t) = f_c(h(t-1), x(t))$$

$h(t)$ = new state

f_c = function with parameter C

$h(t-1)$ = old state

$x(t)$ = input vector at time step t

Providing Input to RNN

- Specify the initial states of all the units
- Specify the initial states of a subset of the units
- Specify the states of the same subset of the units at every time step.

Providing Targets to RNN

- Specify desired final activities of all the units
- Specify good for learning attractors
- Specify the desired activity of a subset of the units.

Advantages

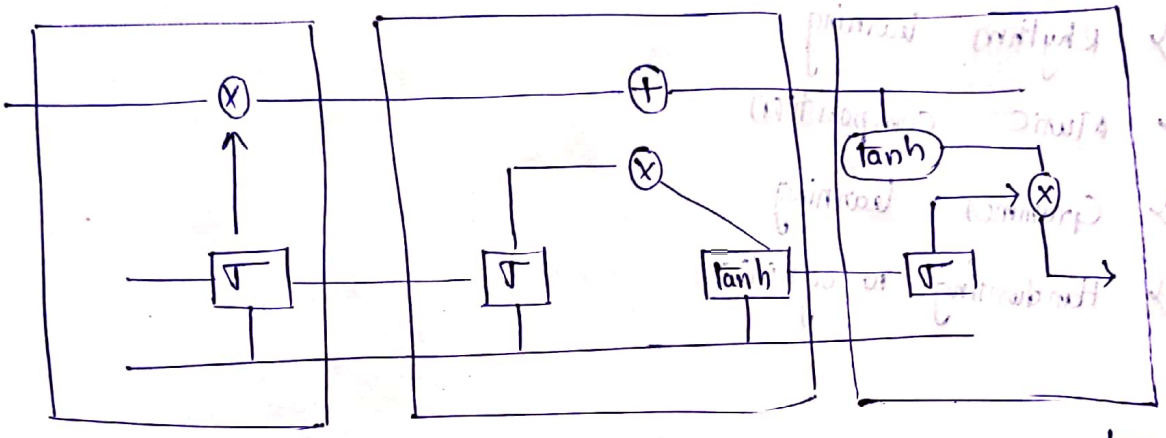
- Can process any length input
- Model size doesn't increase for longer input
- Weights are shared across timesteps

Disadvantages

- Recurrent computation is slow
- In practice, difficult to access information from many steps back.

LSTMs — Long short Term Memory Networks

* A type of RNN architecture that addresses the vanishing gradient problem and allow learning of long-term dependencies



Input gate forget gate output gate

σ – used to removing unwanted data

tanh – used to add the additional information

forget gate : controls what information to throw away from memory

$$f_t = \sigma(w_f [h_{t-1}, x_t] + b_f)$$

Input gate : controls what new information is added to cell state from current input

$$i_t = \sigma(w_i [h_{t-1}, x_t] + b_i)$$

$$\tilde{c}_t = \tanh(w_c [h_{t-1}, x_t] + b_c)$$

Output gate :

$$o_t = \sigma(w_o [h_{t-1}, x_t], b_o)$$

$$h_t = o_t * \tanh(c_t)$$

Applications of LSTM

- Robot control
- Time series prediction
- Speech recognition
- Rhythm learning
- Music composition
- Grammer learning
- Handwriting recognition