Unit 7:

Recurrent neural network:

→ It is a neural network where the output from previous state step is fed as input to current step.

→ It uses sequential data and time series data

→ It is distinguished by their memory as they take information from prior inputs to influence the current input and output.

→ It has hidden state and allows past output used as input.

RNN architecture:-

(i) I/p layer :- layers recieves initial element of data.

(ii) Hidden layer :- Heart of RNN contains set of interconnected neuron.

→ Each neuron contains current input along the information of previous hidden state

(iii) Activation function :- Introduces non linearity in network, enabling it to learn complex pattern

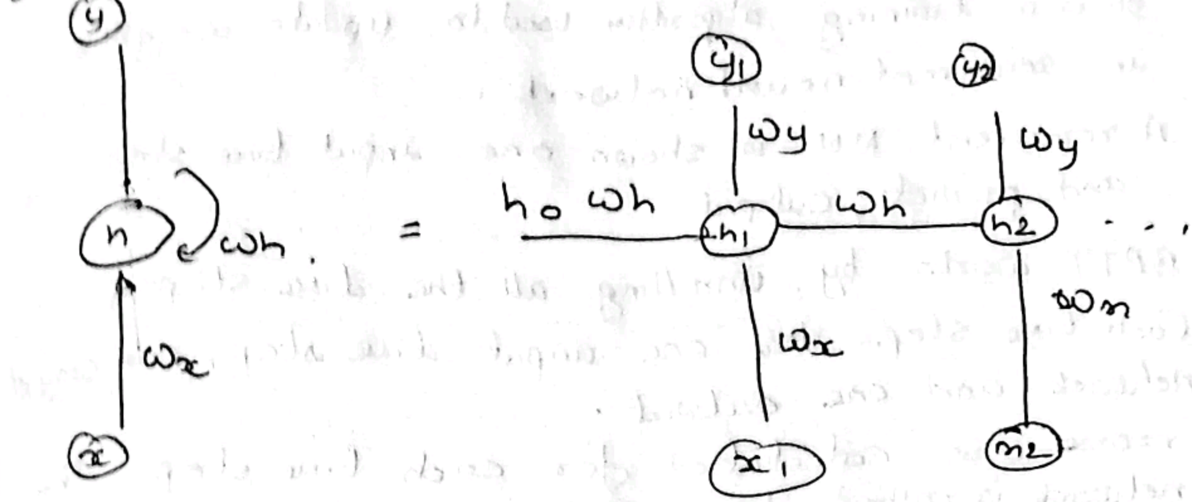
(iv) Output layer :-

→ Generates network prediction based on processed information.

Recurrent connection :-

It allows hidden state information to next time step.

Computation



Computation of hidden state

$h_t = f(W_h h_{t-1} + W_x . x_t)$

Computation of output

$y_t = f(W_y h_t)$

Application of RNN

(i) Time series prediction
(ii) Sequence prediction
(iii) NLP
(iv) Speech processing.

Back propogation through time :-

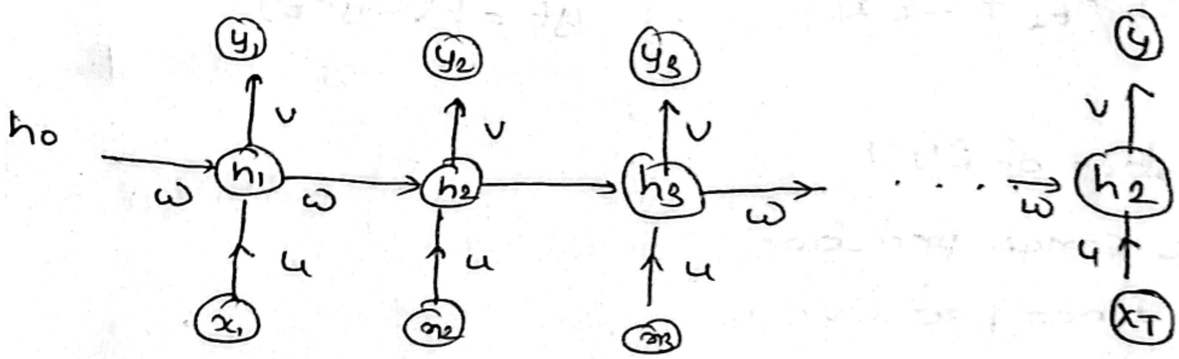
It is a training algorithm used to update weights in recurrent neural network.

A recurrent NN is shown one input time step and predict output.

BPTT works by unrolling all the time steps.

Each time steps has one input time step, one copy of network and one output.

Errors are calculated for each time step. The network is rolled back and weights are updated.



Let L is the loss in RNN then weights w, u, v are updated as

$$w = w - \alpha \frac{dL}{\delta w}, \quad v = v - \alpha \frac{dL}{\delta v}, \quad u = u - \alpha \frac{\delta L}{du}$$

we need to compute $\frac{\partial L}{\delta w}$, $\frac{dL}{\delta v}$, $\frac{\delta L}{du}$ for updating weights

Let L_t is loss as t^{th} step.

$$L = \sum_{t=1}^{T} L_t$$

Computation of hidden state and output in RNN is given by.

$h_t = f(wh_{t-1} + ux_t)$

$y(t) = g(vh_t)$

Assume $t = 3$

$h_3 = f(wh_2 + ux_3)$ $\quad y_3 = g(vh_3)$

also

$L_3 = \frac{1}{2}(d_3 - y_3)^2$. d is desired output and y is predicted output

Now,

$$\frac{dL_3}{dv} = \frac{\delta L_3}{dy_3} \cdot \frac{dy_3}{dv} = -(d_3 - y_3) \cdot h_3$$

$$\frac{\delta L_3}{dw} = \frac{\delta L_3}{dy_3} \cdot \frac{dy_3}{dh_3} \cdot \frac{dh_3}{dw} = -(d_3 - y_3) \cdot v f'(z_3) \cdot \frac{dz_3}{dw}$$

$$= -(d_3 - y_3) \cdot v f'(z_3) \left[h_2 + w \frac{dh_2}{dw} \right]$$

here $\quad z_3 = wh_2 + ux_3$

$$\frac{dL_3}{dw} = \frac{dL_3}{dy_3} \cdot \frac{dy_3}{dh_3} \cdot \frac{dh_3}{du} = -(d_3 - y_3) \cdot v f'(z_3) \left[x_3 + w \frac{\delta h_2}{du} \right]$$

In above relation δh_2 needs to computed. recursively in terms of h_1.

Since the derivatives $\frac{\delta L}{dv}$, $\frac{dL}{du}$ needs to calculated by back propogation through time named algorithm BPTT.

Vaniching Gradient and Truncated BPTT

Vanishing Gradient problem is common issues in training deep neural networks ie RNN.

→ It occurs when gradient steps used to update weights of NN become entremly small. As a result the weights stops changing and networks stops.

→ It occurs particularly in RNN because they involve repeatedly multiplying gradients through time steps, which causes gradients to shrink exponentially.

Truncated BPTT :-

→ Technique used to make training RNN's more feasable by addressing the computational efficency and vanishing gradient problem.

→ Instead of back propogating through entire sequence BPTT splits the sequence into smaller segments and performs back propogation through these smaller segments.

Long short term Memory:-

- It is improved version of RNN.

In RNN single hidden state is passed through time which can make it difficult for n/w to learn long term dependencies. This Model. address this problem by introducing Memory cell. which is a container that can hold a information for extended peroid.

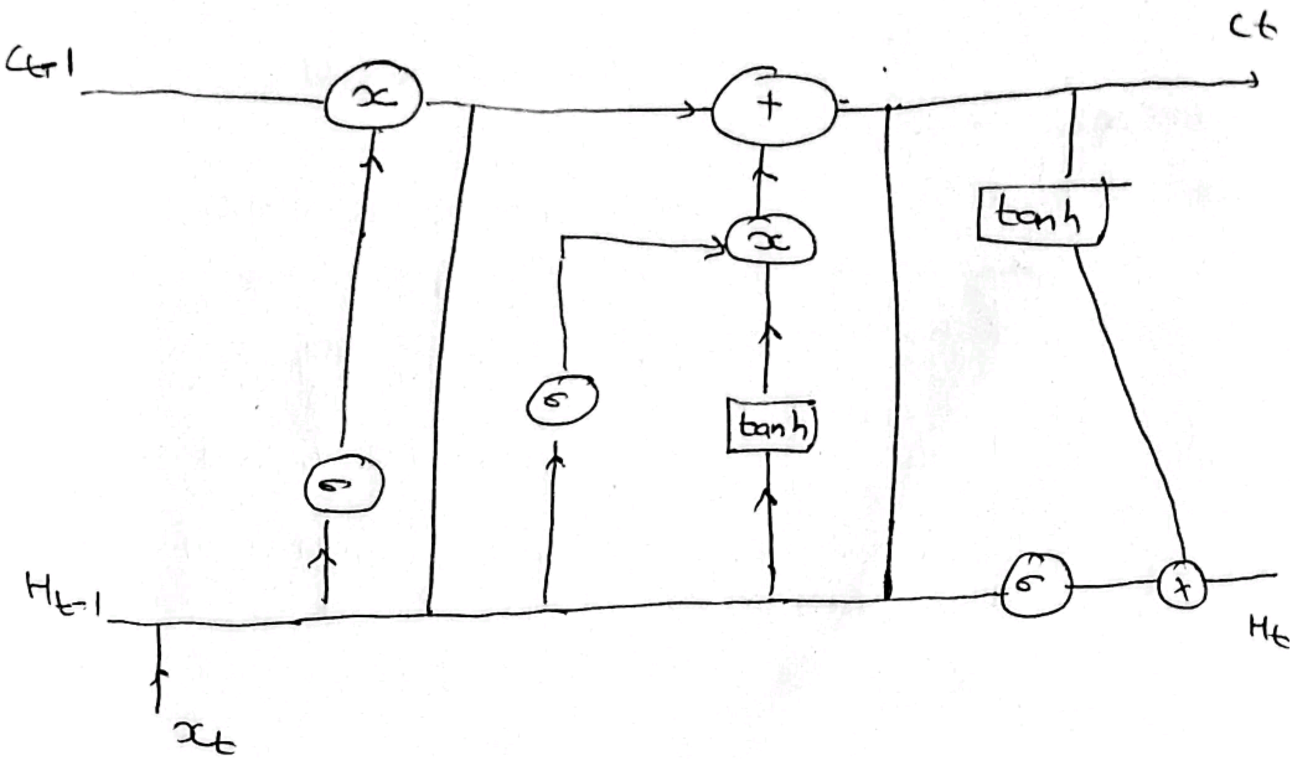
LTSM architecture:-

Its architecture involves the memory cells which is controlled by three gates

(i) Input gate controls what information added to memory cell

(ii) Forget gate controls what information remo-ved from memory cell

(iii) Output get: controls what information is output from memory cell.



Forget gate :

$$F_t = \sigma(x_t W_f + H_{t-1} U_f)$$

Input gate

$$I_t = \sigma(x_t W_i + H_{t-1} U_i)$$

$$H'_t = \tanh(x_t W_c + H_{t-1} U_c)$$

$$C_t = (C_{t-1} \times f_t + I_t \cdot H'_t)$$

Output gate

$$O_t = \sigma(x_t W_o + H_{t-1} U_o)$$

$$H_t = \tanh(C_t) \times O_t .$$