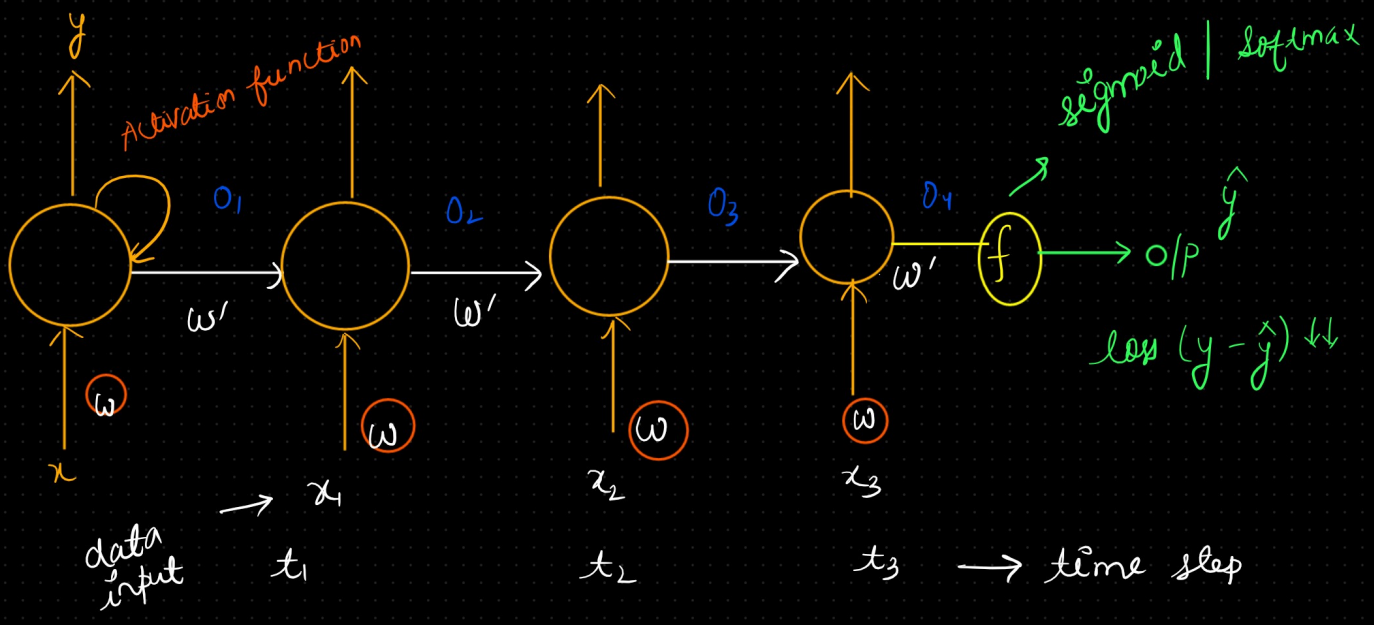
RNN

Forward and Backward Propogation in RNN



x_t "I" "love" "Data" "Science"

x_1 x_2 x_3 x_4

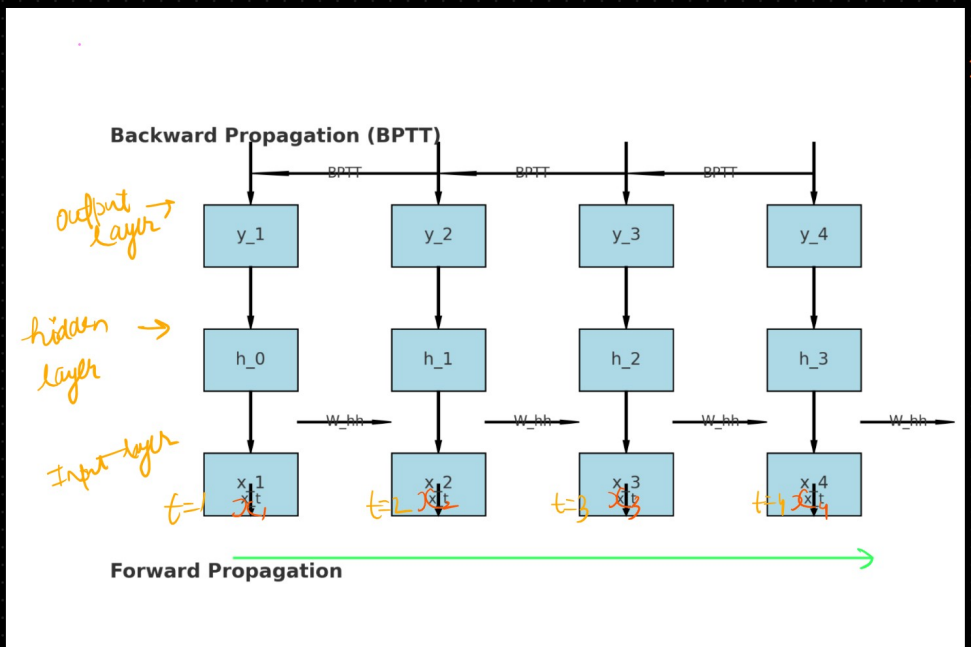
$t=1$ $t=2$ $t=3$ $t=4$

Forward Propogation

In an RNN, forward propagation involves processing an input sequence one step at a time. At each time step, the RNN takes the current input and the hidden state from the previous time step to compute the current hidden state and output.

→ Each time step processes the input and updates the hidden state, capturing information from the sequence.

BPTT arrows indicate the backward flow of errors through time.



$x_1, x_2, x_3 \rightarrow x_t$ ↓ Input Sequence

where t → time step

x_t → Input at a time step t

h_i → hidden state at time step t

y_t → Output at time t

Forward propagation arrows show the flow of information from inputs through hidden states to outputs.

① Input Sequence

Each input x_t is fed into a network one a time. (eg: x_1, x_2, x_3)

② Hidden state

The hidden state h_t is updated using the previous state h_{t-1} and the current input x_t.

③ Output Calculation

The output y_t is calculated from the current hidden state h_t.

Eg → Predicting the next character in a sequence

Input sequence → "hello"

① Initialization

- Input sequence → "h", "e", "l", "l", "o"
  (x_1, x_2, x_3, x_4, x_5)

input at a time step t → x_t

hidden state → h_0

② Forward Propogation

At $t = 1$, input $x_1 \rightarrow$ "h"

- Compute $h_1 = f(W_{xh} \cdot x_1 + W_{hh} \cdot h_0 + b_h)$

Compute output $y_1 = g(W_{hy} \cdot h_1 + b_y)$

At $t = 2$, input $x_2 \rightarrow$ "e"

$$h_2 = f(W_{xh} \cdot x_2 + W_{hh} \cdot h_1 + b_h)$$

$$y_2 = g(W_{hy} \cdot h_2 + b_y)$$

⋮

$t = 5$

Continue this process for each time step in the sequence

∴ The hidden state h_t at each time step captures information from the previous input and is used to predict the output y_t

$f \rightarrow$ Activation func^n (tanh, Relu)

$g \rightarrow$ output Activation func^n (softmax, sigmoid)

$W_{xh}, W_{hh}, W_{hy} \rightarrow$ weight matrices

$b_h, b_y \rightarrow$ biases

Backward Propogation in RNN (Backpropogation Through Time - BPTT)

→ Backward propagation in RNN involves calculating the gradients of the loss with respect to the weights by propagating errors backward through time.

→ Errors are propagated backward through the network, allowing the model to learn dependencies across time steps.

Backward Propogation Through time (BPTT)

① Loss Function

The loss is computed at each time step based on the difference between the predicted output and the target output. $(y - \hat{y})$

② Error Propogation

Errors are propagated backward through the network, updating the weights at each time step. This process involves computing the gradients and updating the weights to minimize the loss.

Backward Propogation Steps

At $t = T$ (last time step)

→ Compute the gradient (derivative) of the loss w.r.t y_t (output)

$$\frac{\partial L}{\partial y_t}$$

→ Compute the gradient of the loss w.r.t h_t (hidden state)

$$\frac{\partial L}{\partial h_t}$$

At $t = T-1$

$\therefore$ Output $\rightarrow \frac{\partial L}{\partial y_{t-1}}$

hidden $\rightarrow \frac{\partial L}{\partial h_{t-1}}$

$\therefore$ Weight update Formula

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial y_t}$$

$\therefore$ Propagate the error backward through time

$\frac{\partial L}{\partial h_{T-1}} \rightarrow$ include the error $\frac{\partial L}{\partial h_T}$

Gradient Calculation

W_{xh}, W_{hh}, W_{hy} and b_h, b_y using chain Rule

Adam $\rightarrow$ Update the weight using gradient descent.

By iteratively updating the weights based on the gradients, the RNN learns to make better predictions for each time step in the sequence.