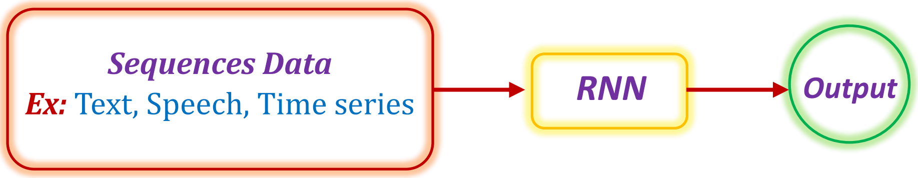
Recurrent Neural Network

1. How Adagrad Optimizer Works
2. α_t^1 Value Calculation
3. α_t Values Calculation
4. Important Graph Explanation
5. Disadvantage

A Recurrent Neural Network (RNN) is a type of artificial neural network designed to recognize patterns in sequences of data, such as text, speech, time series, or any sequence of data points.



RNN ---> Recurrent Neural Network ***Recurrent*** --- > Carrying repeatedly

1. Why we Can't Use ANN for Text Data:

There are mainly two problems with the text data if we use in ANN:

1. High volume of input values
2. Unrecognizable sequence problem

1.1. High Volume of Input Values:

- If we observe artificial neural network hidden neurons and depth is depends on the number of input values.
- But while processing the text data has thousands or hundreds of words, in this case the ANN will be more competitional with huge depth.
- ANN increases the cost and taking too much time to complete the training while processing the text data.

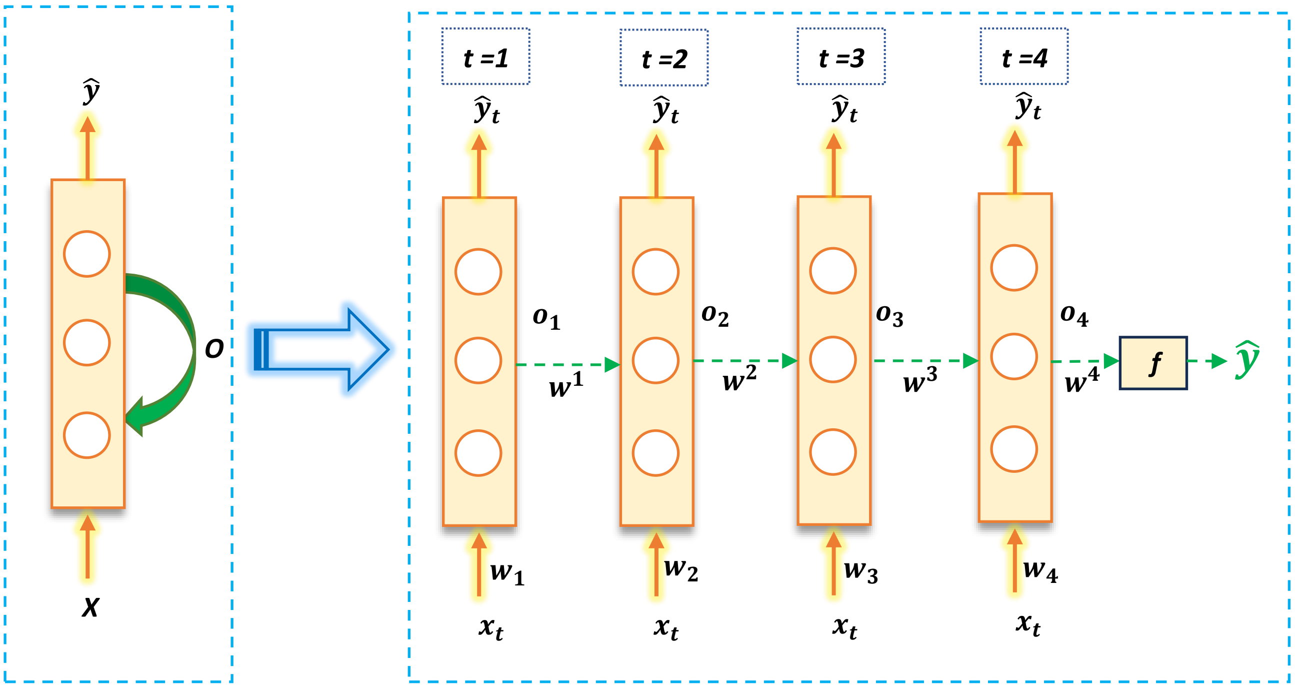
1.2. Unrecognizable Sequence Problem:

- ANN not consider the sequence of the words but in text data processing the sequence of words is very important.

- If we pass two sentences as an input to ANN by changing the words in the two sentences internally, but ANN gives same output for these two sentences because of the words contains two sentences are same.
- Actually, these two sentences meaning is not same, the expected output should not be same.

Recurrent Neural Network can overcome the above problem and it can work with ***sequence data*** and identifying the output based on the sequence along with value**.**

2. RNN Architecture:



Let's consider 4 words (x_1, x_2, x_3, x_4) of sentence X.

$$X = < x_1, x_2, x_3, x_4 >$$

2.1. Forward Propagation:

- At time t = 1 the first word x_1 taking as input along with weight w_1. The first hidden layer processed that word x_1 and generated the output $\hat{y}_1$. Thes first word output O_1 will pass as an input to seconded layer by applying activation function.
- The second layer will take two inputs at time t = 2, one is x_2 along with w_2 and seconded is output of first layer O_1.
- This process continues for all words and final output $\hat{y}$ will come output layer.
- Let's see the output functions in each layer:
  $O_1 = f(x_1\, w_1)$
  $O_2 = f(x_2\, w_2 + O_1\, w^1)$
  $O_3 = f(x_3\, w_3 + O_2\, w^2)$

$O_4 = f(x_4\, w_4 + O_3\, w^3)$

- Based on the above equations, the output O_4 is a depends on the input x_4 and output O_3. Again output O_3 is a depends on output O_2 and the output O_2 is depends on output O_1. It indicates every output considering the previous outputs as sequence. That's why RNN is worked based on the sequence.

2.2. Back Propagation:

- First it will calculate the loss function after got the predicted $\hat{y}$
- Next it will update the weight by using chain rule with back propagation as like as ANN. Let's observe the weight updating equations:

$$\omega_{new} = \omega_{old} - \alpha \frac{\partial L}{\partial \omega_{old}} \dots \dots \dots \dots \quad (1)$$

Updating w^4:

$$\omega^4 = \omega^4 - \alpha \frac{\partial L}{\partial \omega^4} \dots \dots \dots \dots \quad (2)$$

$$\frac{\partial L}{\partial \omega^4} = \frac{\partial\, L}{\partial\, \hat{y}} \cdot \frac{\partial\, \hat{y}}{\partial\, \omega^4}$$

Updating ω_4:

$$\omega_{4(new)} = \omega_4 - \alpha \frac{\partial L}{\partial \omega_4} \dots \dots \dots \dots \quad (3)$$

$$\frac{\partial L}{\partial \omega_4} = \frac{\partial\, L}{\partial\, \hat{y}} \cdot \frac{\partial\, \hat{y}}{\partial\, O_4} \cdot \frac{\partial\, O_4}{\partial\, \omega_4}$$

- The remaining weights also will update as shown above.

2.3. Problems in simple RNN:

- If we use *Sigmoid Activation* function will create varnishing gradient problem because of the derivative terms are very less values (exist between 0 to 1) in chain rule. The multiplication of these less values will create very less change in weights. Finally, it creates vanishing gradient problem.
- If we use *Relu Activation* function will create exploding gradient problem because of the derivative terms are very high values (exist > 1) in chain rule. The multiplication of these high values will create exploding gradient problem.

*The **LSTM** can overcome these **Exploding, Varnishing** gradient problems.*