RNN(Recurrent Neural Network)

1. Why need RNN
2. What is RNN
3. Architecture of RNN
4. Understand RNN - Forward Propagation
5. Understand RNN - Back propagation
6. Implementation of RNN in TensorFlow

Source code for RNN Recurrent Neural Network

Why need RNN

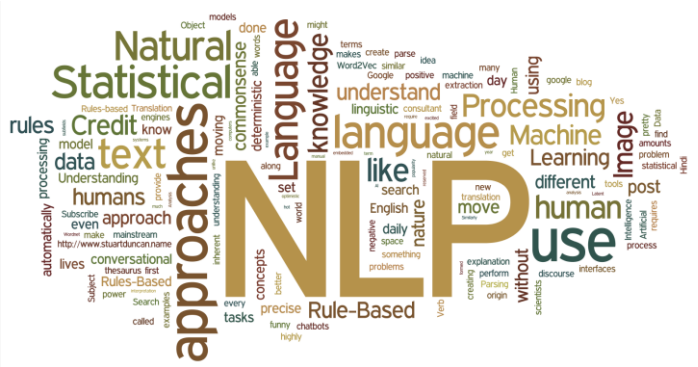
In Natural language processing:

The sequence of words define their meaning

- *Will you do it?*
- *Do you will it?*

- *Woman, without her man, is helpless.*
- *Woman! Without her, man is helpless!*

- *The dragon swallowed the knight.*
- *The knight swallowed the dragon.*



Source code for RNN Recurrent Neural Network

Why need RNN

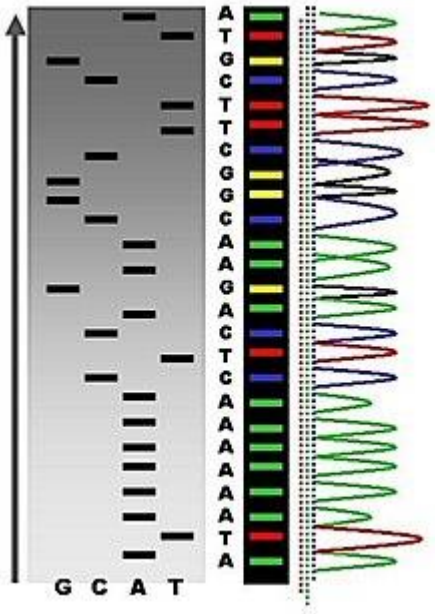
In Bioinformatics:

Genome sequence define nucleotides even life

Order of As, Cs, Gs, and Ts that make up an organism's DNA. The human genome is made up of over 3 billion of these genetic letters.

DNA sequence that has been translated from life's chemical alphabet into our alphabet of written letters might look like this:

AGTCCGCGAATACAGGCTCGGT



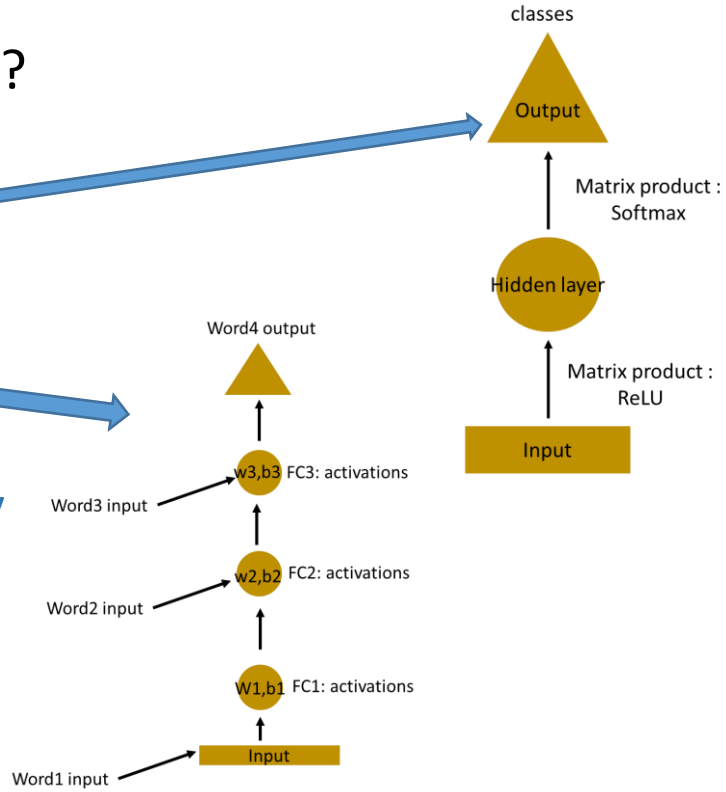
Source code for RNN Recurrent Neural Network

What is RNN

How to predict the next word in a sentence?

- Simplest MLP(Multilayer Perceptron)
- Multilayer MLP

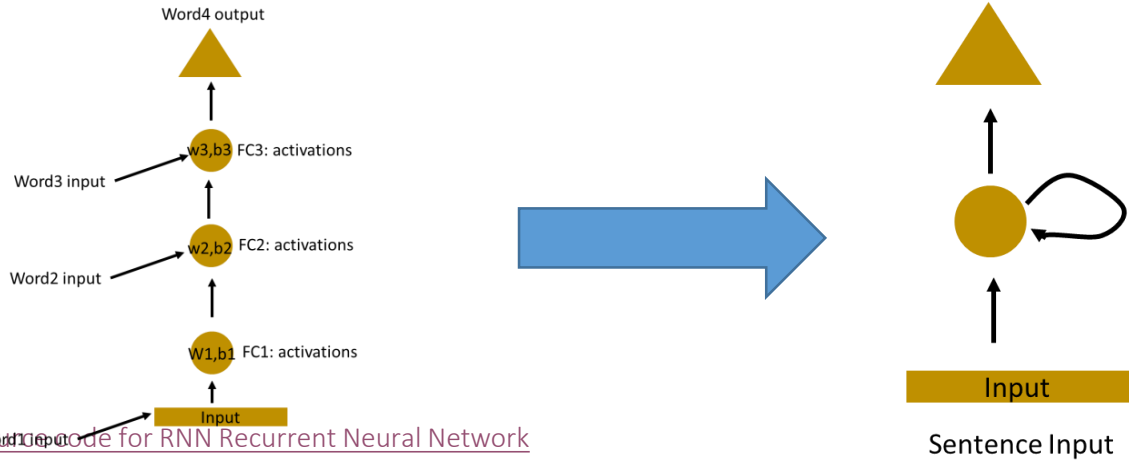
The input layer receives the input, the hidden layer activations are applied and then we finally receive the output.



Source code for RNN Recurrent Neural Network

What is RNN

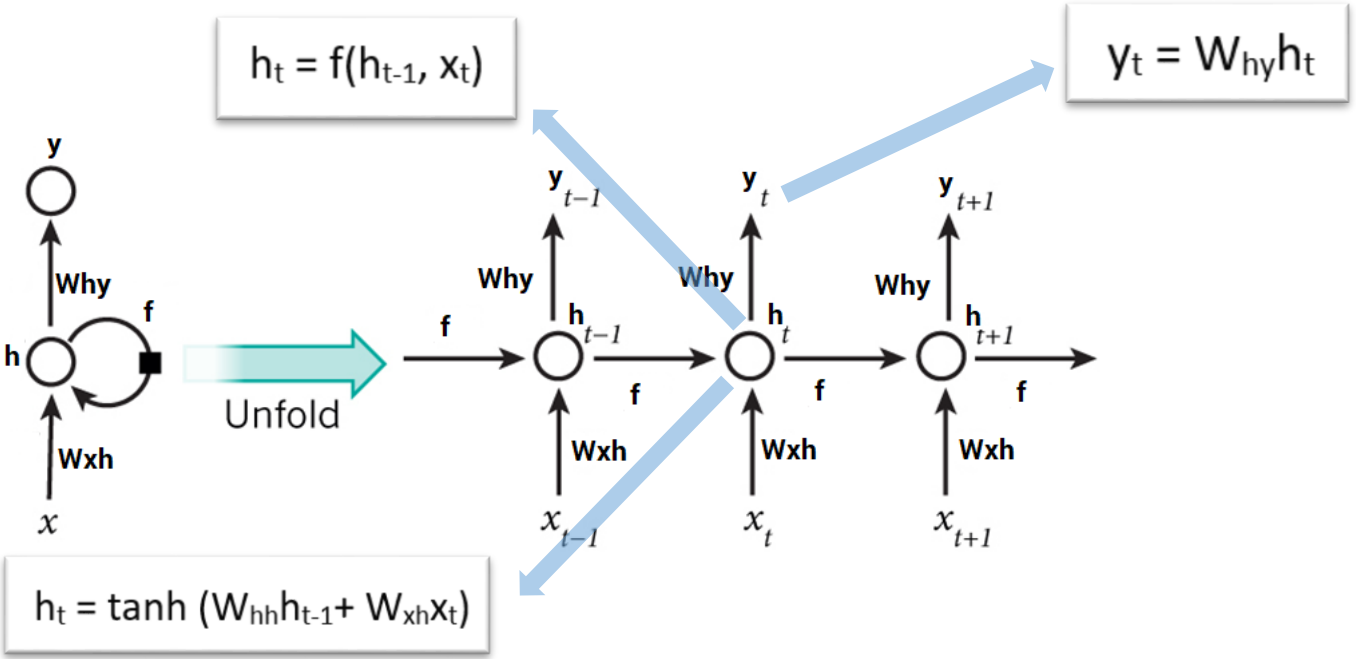
Here, the weights and bias of these hidden layers are different.
And hence each of these layers behave independently and cannot be combined together.
To combine these hidden layers together, we shall have the same weights and bias for these hidden layers.



Source code for RNN Recurrent Neural Network

Architecture of RNN

Unfold the RNN, we can see the architecture of RNN



Source code for RNN Recurrent Neural Network

Understand RNN - Steps

Steps in a recurrent neural network:

If you feel confused for my following example, please review these steps again

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

1. A single time step of the input is supplied to the network i.e. x_t is supplied to the network
2. Calculate current state using a combination of the current input and the previous state i.e. we calculate h_t
3. The current h_t becomes h_{t-1} for the next time step
4. Go as many time steps as the problem demands and combine the information from all the previous states

$$y_t = W_{hy}h_t$$

Source code for RNN Recurrent Neural Network

Understand RNN - Steps

Steps in a recurrent neural network:

If you feel confused for my following example, please review these steps again

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

5. Once all the time steps are completed, the final current state is used to calculate the output y_t
6. The output is then compared to the actual output and the error is generated
7. The error is then backpropagated to the network to update the and the network is trained

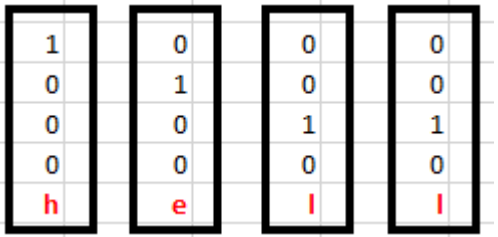
$$y_t = W_{hy}h_t$$

Source code for RNN Recurrent Neural Network

Understand RNN - Forward Propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from 'h', 'e' and 'l'

Input Initialization



One hot encoded Input

wxh			
0.287027	0.84606	0.572392	0.486813
0.902874	0.871522	0.691079	0.18998
0.537524	0.09224	0.558159	0.491528

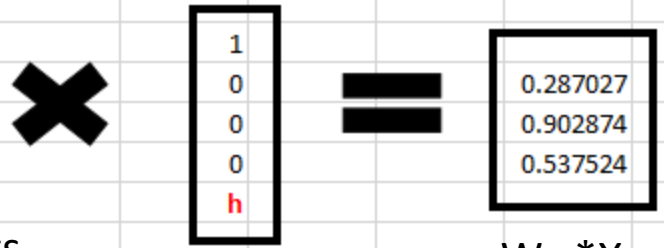
Randomly initialized weights

Step 1: Calculate $W_{xh} * X_t$

Source code fo

wxh			
0.287027	0.84606	0.572392	0.486813
0.902874	0.871522	0.691079	0.18998
0.537524	0.09224	0.558159	0.491528

Randomly initialized weights

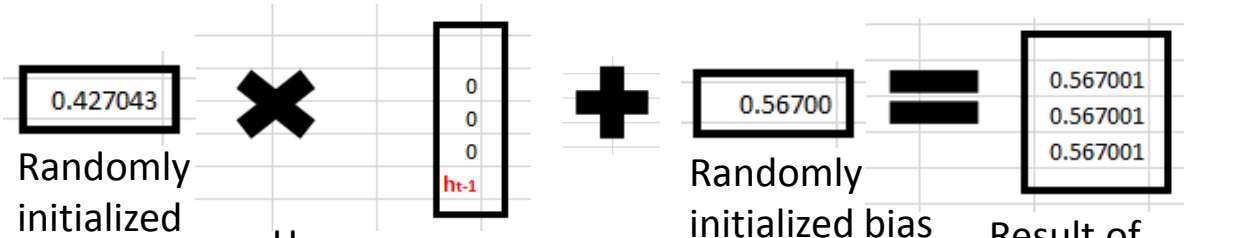


First input x_t $W_{xh} * X_t$

Understand RNN - Forward Propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from 'h', 'e', and 'l'

Step 2: calculate ($w_{hh}*h_{t-1}$+bias)



Randomly initialized W_{hh}

H_{t-1} previous state is [0,0,0] since there is no letter prior to it.

Randomly initialized bias

Result of ($w_{hh}*h_{t-1}$+bias)

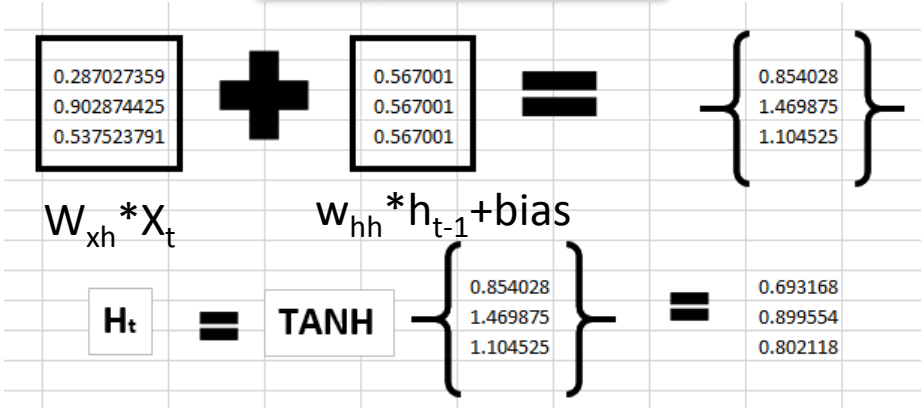
Source code for RNN Recurrent Neural Network

Understand RNN - Forward Propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

Step 3: Get current State

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

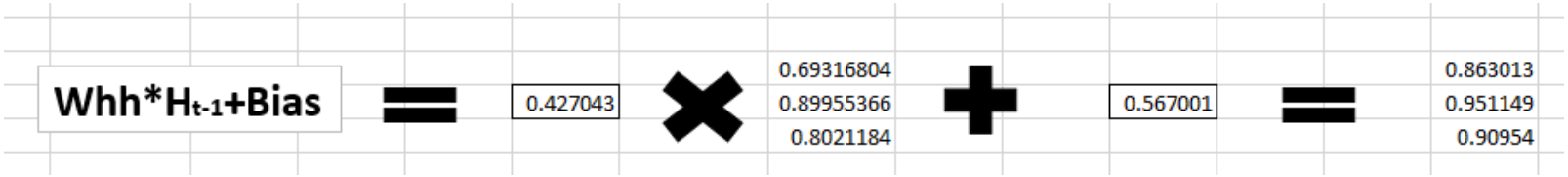


Source code for RNN Recurrent Neural Network

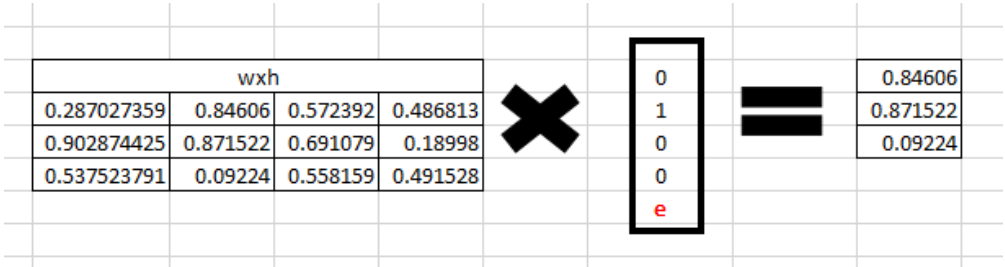
Understand RNN - Forward Propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

Step 4: h_t become h_{t-1} and 'e' is supplied



$W_{xh} * X_t$

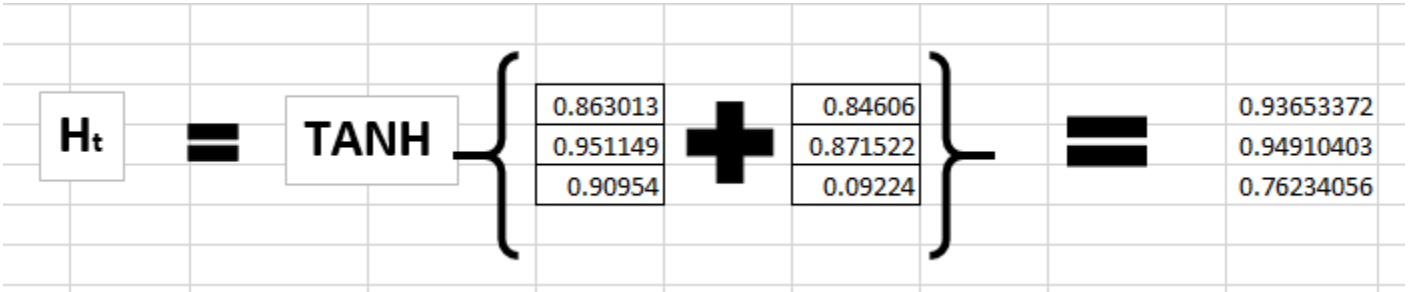


Source code for RNN Recurrent Neural Network

Understand RNN - Forward Propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

Step 5: calculating h_t for the letter "e",



Source code for RNN Recurrent Neural Network

Understand RNN - Forward Propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

Step 6: calculate y_t for the letter 'e'

$$y_t = W_{hy} h_t$$

why		
0.37168	0.974829459	0.830034886
0.39141	0.282585823	0.659835709
0.64985	0.09821557	0.334287084
0.91266	0.32581642	0.144630018

×

Ht
0.936534
0.949104
0.762341

=

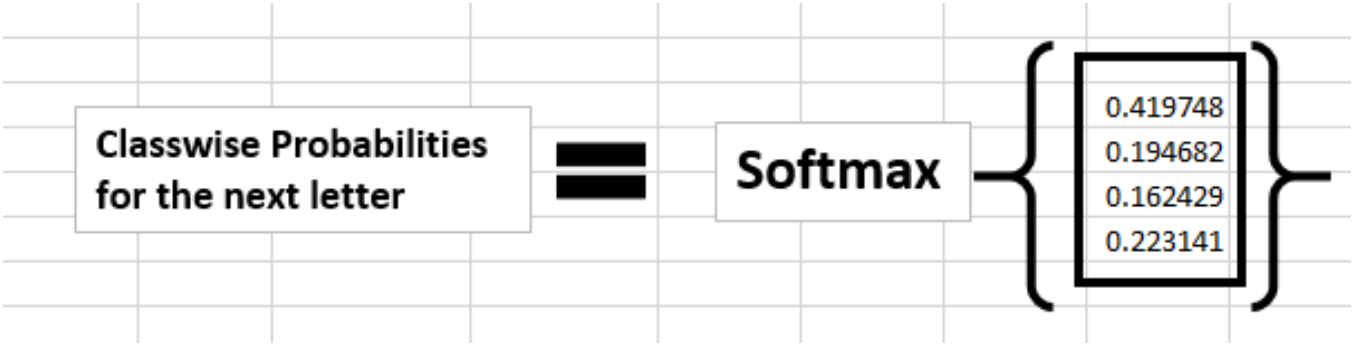
yt
1.90607732
1.13779113
0.95666016
1.27422602

Source code for RNN Recurrent Neural Network

Understand RNN - Forward Propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

Step 7: Calculate probability using soft-max function



Source code for RNN Recurrent Neural Network

Understand RNN - Back propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

If we convert these probabilities to understand the prediction, we see that the model says that the letter after "e" should be h,

since the highest probability is for the letter "h". Does this mean we have done something wrong?
No, so here we have hardly trained the network. We have just shown it two letters. So it pretty much hasn't learnt anything yet.

Now the next BIG question that faces us is how does Back propagation work in case of a Recurrent Neural Network. How are the weights updated while there is a feedback loop?

Source code for RNN Recurrent Neural Network

Understand RNN - Back propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

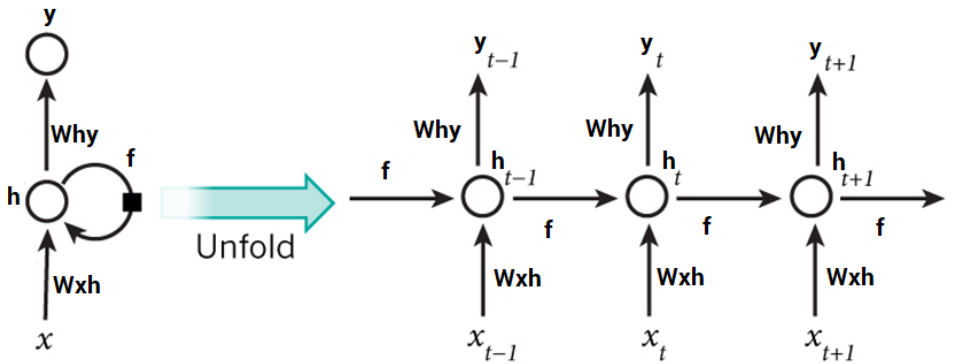
y_t is the predicted value

$\bar{y}_t$ is the actual value

The error is calculated as a cross entropy loss

$$E_t(\bar{y}_t, y_t) = -\bar{y}_t \log(y_t)$$

$$E(\bar{y}, y) = -\sum \bar{y}_t \log(y_t)$$



Source code for RNN Recurrent Neural Network

Understand RNN - Back propagation

To understand RNN, let's see how Forward Propagation works:
For the word 'hello', we predict the last letter 'o' from h, e, l and l

The steps of back propagation:

1. The cross entropy error is computed using the current output and the actual output
2. Network is unrolled for all the time steps
3. For the unrolled network, the gradient is calculated for each time step with respect to the weight parameter
4. Weight is the same for all the time steps the gradients can be combined together for all time steps
5. The weights are then updated for both recurrent neuron and the dense layers

Source code for RNN Recurrent Neural Network

RNN(Recurrent Neural Network)

This is the end

Source code for RNN Recurrent Neural Network