Hedgehog Rock

University of Science and Technology Bannu

Deep Learning

Lesson 7

June,03 2024

Learning Objectives

Need for a Recurrent Neural Network

Why RNN

RNN Architecture

Understanding a Recurrent Neuron in Detail

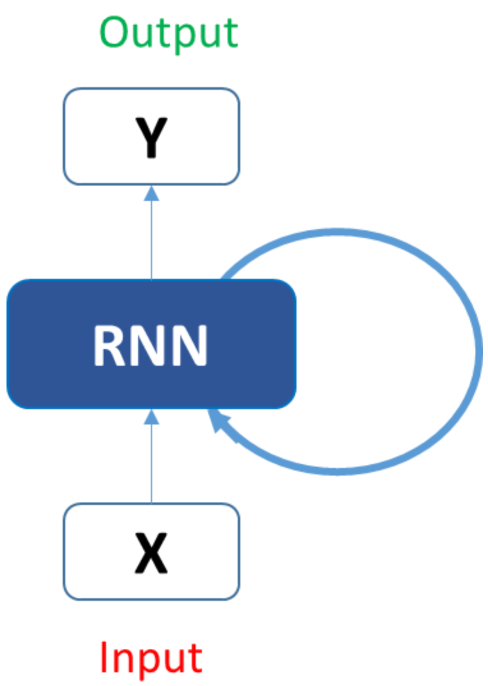
Let's take a simple task at first. Let's take a character level RNN where we have a word **"Hello".**

So we provide the first 4 letters **i.e. h,e,l,l** and ask the network to predict the last letter **i.e.'o'.**

So here the vocabulary of the task is just 4 letters {h,e,l,o}.

In real case scenarios involving natural language processing, the vocabularies include the words in entire wikipedia database, or all the words in a language.

Here for simplicity we have taken a very small set of vocabulary.



RNN

Let's see how the given structure be used to predict the fifth letter in the word "hello".

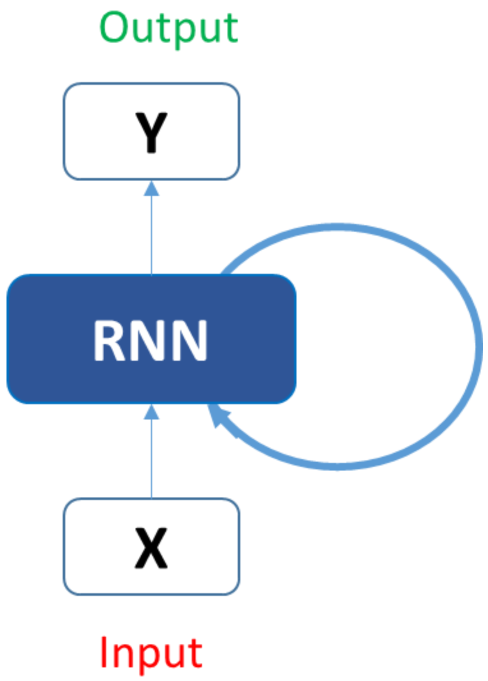
In the given structure, the blue RNN block, applies something called as a recurrence formula to the input vector and also its previous state.

In this case, the letter "h" has nothing preceding it, let's take the letter "e".

So at the time the letter "e" is supplied to the network, a recurrence formula is applied to the letter "e" and the previous state which is the letter "h".

These are known as various time steps of the input. So if at time t, the input is "e", at time t-1, the input was "h".

The recurrence formula is applied to e and h both. and we get a new state.



RNN

The formula for the current state can be written as –

$$h_t = f(h_{t-1}, x_t)$$

Here, Ht is the new state, ht-1 is the previous state while xt is the current input.

We now have a state of the previous input instead of the input itself, because the input neuron would have applied the transformations on our previous input.

So each successive input is called as a time step.

In this case we have four inputs to be given to the network, during a recurrence formula, the same function and the same weights are applied to the network at each time step.

RNN

Taking the simplest form of a recurrent neural network, let's say that the activation function is tanh, the weight at the recurrent neuron is Whh and the weight at the input neuron is Wxh, we can write the equation for the state at time t as –

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

Now, once the current state is calculated we can calculate the output state as-

$$y_t = W_{hy}h_t$$

SUMMARY

Let me summarize the steps in a recurrent neuron for you-

1. A single time step of the input is supplied to the network i.e. xt is supplied to the network
2. We then calculate its current state using a combination of the current input and the previous state i.e. we calculate ht
3. The current ht becomes ht-1 for the next time step
4. We can go as many time steps as the problem demands and combine the information from all the previous states
5. Once all the time steps are completed the final current state is used to calculate the output yt
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 weights and the network is trained

Forward Propagation in a Recurrent Neuron

Let's take a look at the inputs first –

1	0	0	0
0	1	0	0
0	0	1	1
0	0	0	0
h	e	l	l

The inputs are one hot encoded. Our entire vocabulary is {h,e,l,o} and hence we can easily one hot encode the inputs.

RNN

Now the input neuron would transform the input to the hidden state using the weight wxh.

We have randomly initialized the weights as a 3*4 matrix –

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

RNN

Step 1:

Now for the letter “h”, for the hidden state we would need Wxh*Xt. By matrix multiplication, we get it as –

$$\begin{array}{c} \text{wxh} \\ \begin{bmatrix} 0.287027 & 0.84606 & 0.572392 & 0.486813 \\ 0.902874 & 0.871522 & 0.691079 & 0.18998 \\ 0.537524 & 0.09224 & 0.558159 & 0.491528 \end{bmatrix} \end{array} \times \begin{array}{c} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ h \end{array} = \begin{bmatrix} 0.287027 \\ 0.902874 \\ 0.537524 \end{bmatrix}$$

RNN

Step 2:

Now moving to the recurrent neuron, we have Whh as the weight which is a 1*1 matrix as

0.427043

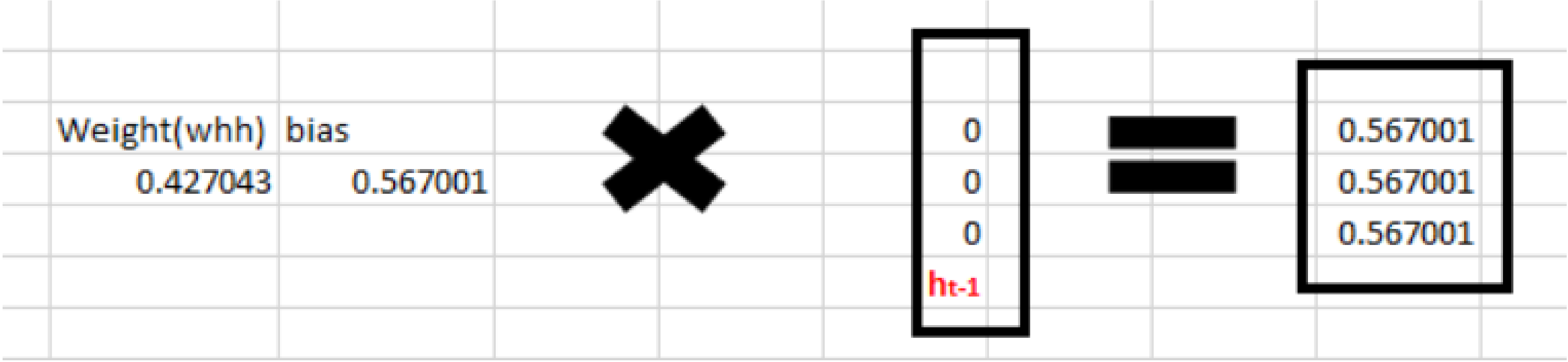
and the bias which is also a 1*1 matrix as

0.56700

For the letter “h”, the previous state is [0,0,0] since there is no letter prior to it.

So to calculate -> (whh*ht-1+bias)

So to calculate -> (whh*ht-1+bias)



Step 3:

Now we can get the current state as –

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

Since for h, there is no previous hidden state we apply the tanh function to this output and get the current state –

$$\begin{bmatrix} 0.287027359 \\ 0.902874425 \\ 0.537523791 \end{bmatrix} + \begin{bmatrix} 0.567001 \\ 0.567001 \\ 0.567001 \end{bmatrix} = \begin{Bmatrix} 0.854028 \\ 1.469875 \\ 1.104525 \end{Bmatrix}$$

$$H_t = \text{TANH}\begin{Bmatrix} 0.854028 \\ 1.469875 \\ 1.104525 \end{Bmatrix} = \begin{matrix} 0.693168 \\ 0.899554 \\ 0.802118 \end{matrix}$$

RNN

Step 4:

Now we go on to the next state. “e” is now supplied to the network. The processed output of ht, now becomes ht-1, while the one hot encoded e, is xt. Let’s now calculate the current state ht.

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

Whh*ht-1 +bias will be –

$$\mathbf{Whh*H_{t\text{-}1}+Bias} = \boxed{0.427043} \times \begin{bmatrix} 0.69316804 \\ 0.89955366 \\ 0.8021184 \end{bmatrix} + \boxed{0.567001} = \begin{bmatrix} 0.863013 \\ 0.951149 \\ 0.90954 \end{bmatrix}$$

Wxh*xt will be –

wxh			
0.287027359	0.84606	0.572392	0.486813
0.902874425	0.871522	0.691079	0.18998
0.537523791	0.09224	0.558159	0.491528

$$\times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ e \end{bmatrix} = \begin{bmatrix} 0.84606 \\ 0.871522 \\ 0.09224 \end{bmatrix}$$

Step 5:

Now calculating ht for the letter “e”,

$$H_t = \text{TANH}\left\{ \begin{bmatrix} 0.863013 \\ 0.951149 \\ 0.90954 \end{bmatrix} + \begin{bmatrix} 0.84606 \\ 0.871522 \\ 0.09224 \end{bmatrix} \right\} = \begin{matrix} 0.93653372 \\ 0.94910403 \\ 0.76234056 \end{matrix}$$

Now this would become ht-1 for the next state and the recurrent neuron would use this along with the new character to predict the next one.

Step 6:

At each state, the recurrent neural network would produce the output as well. Let's calculate yt 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

$\times$

Ht
0.936534
0.949104
0.762341

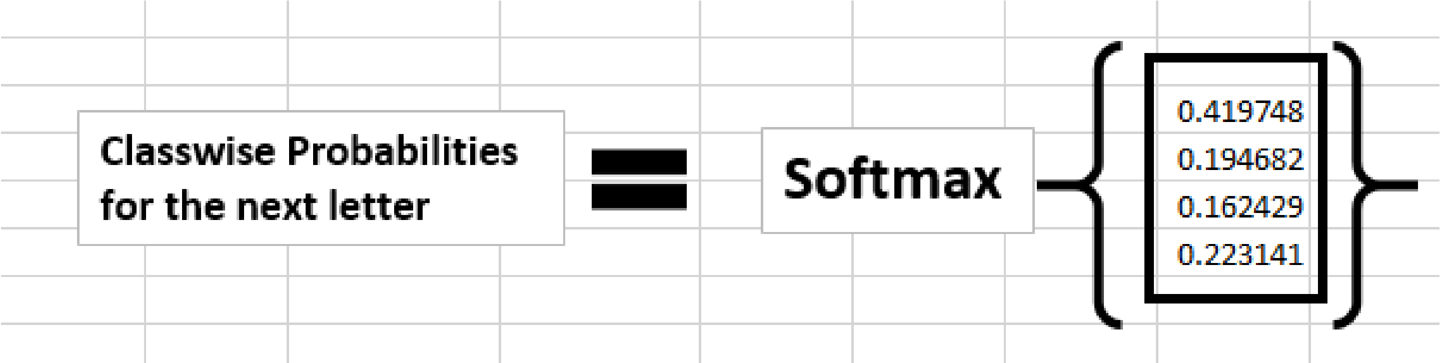
$=$

yt
1.90607732
1.13779113
0.95666016
1.27422602

Step 7:

The probability for a particular letter from the vocabulary can be calculated by applying the softmax function.

so we shall have softmax(yt)



output

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.