Let’s say our network having only 1 hidden layer, so for RNN we have only 3 weights w, w’, w’’ to update.

Note: While training For a sentence or for batch updation we use same weights, and weights are updated using backpropogation after completion of sentence or batch of sentences.

In RNN backpropogation is performed on time step basis.

Let’s say a sentence have 10 words, so there will be 10 times a word pass through network.

Now for finding Forward propagation just follow the direction of arrow.

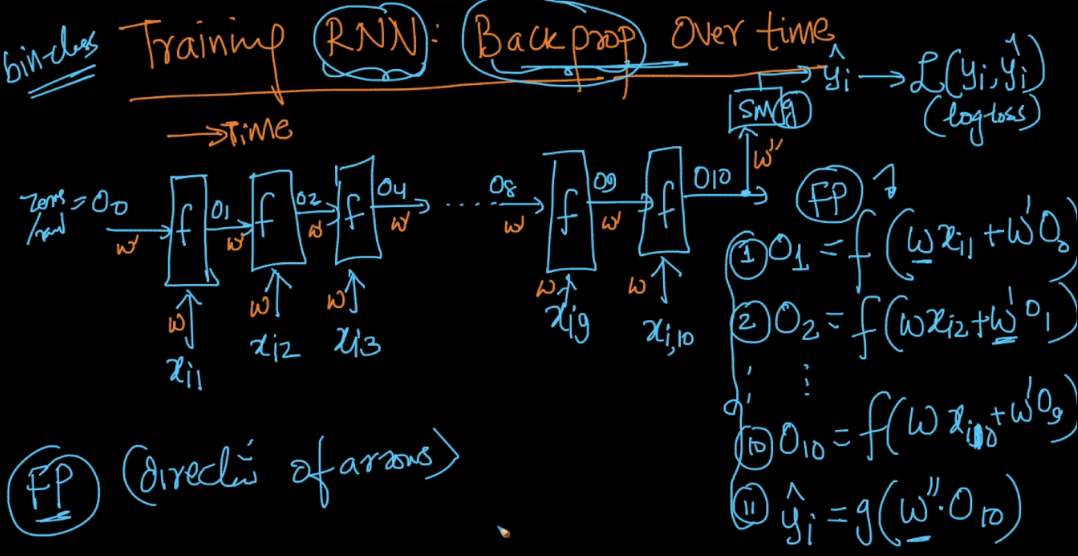
O1 = f( w \* xi1 + w’ \* o0)

O2 = f( w \* xi2 + w’ \* o1)

……

O10 = f( w \* xi10 + w’ \* o9)

Y^ = g(w’’ \* o10)



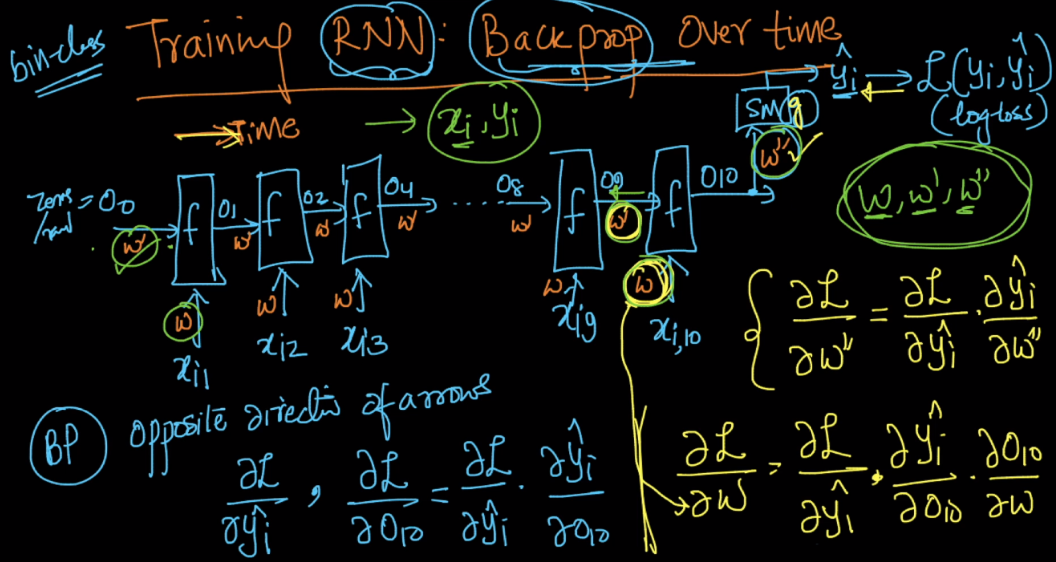
**Backpropogation with time step:**

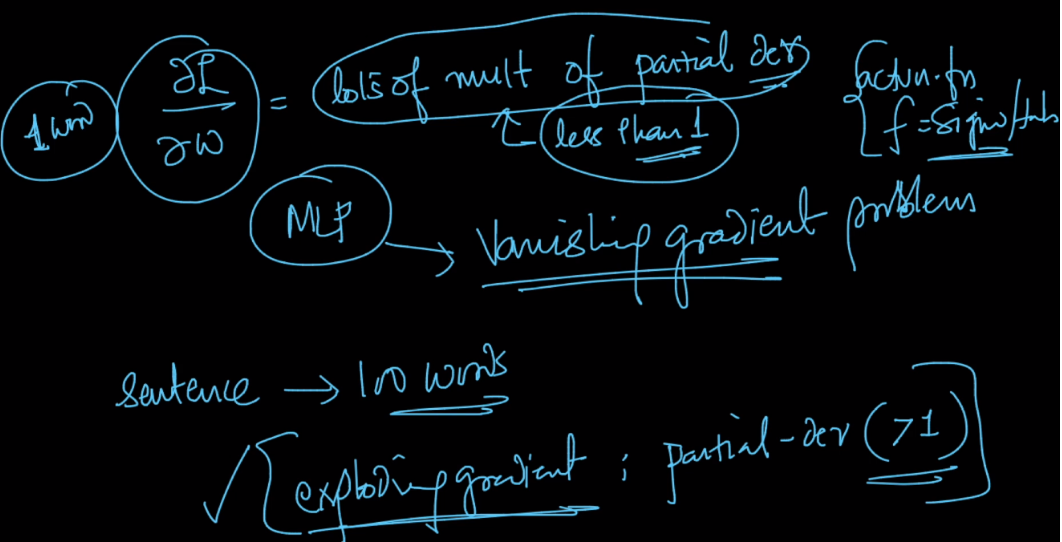
Let’s say our whole sentence is passed and we get a loss. Now we update weights as:

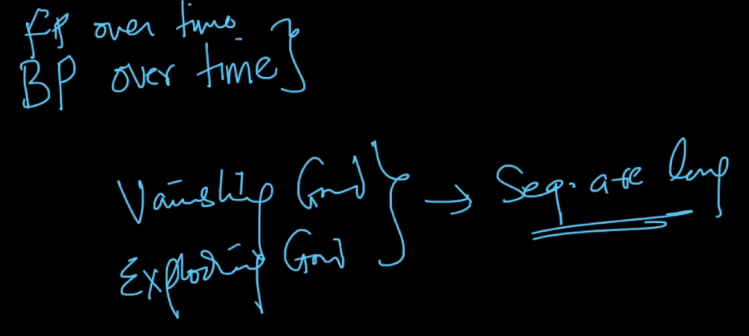
**Updating w’’:** eq given in below image.

Now first we update w associated with 10th word, then we store this w. Now using this w again we update w associated with 9th word. Here w is a same weight matrix we are updating at each time step, but each time we are going backward derivatives are increasing.

Because of this there is a problem of gradient vanishing and gradient exploding, to overcome this we use advance RNN like LSTM, GRU.







<https://towardsdatascience.com/only-numpy-vanilla-recurrent-neural-network-back-propagation-practice-math-956fbea32704>

