UNIT V

Contents:

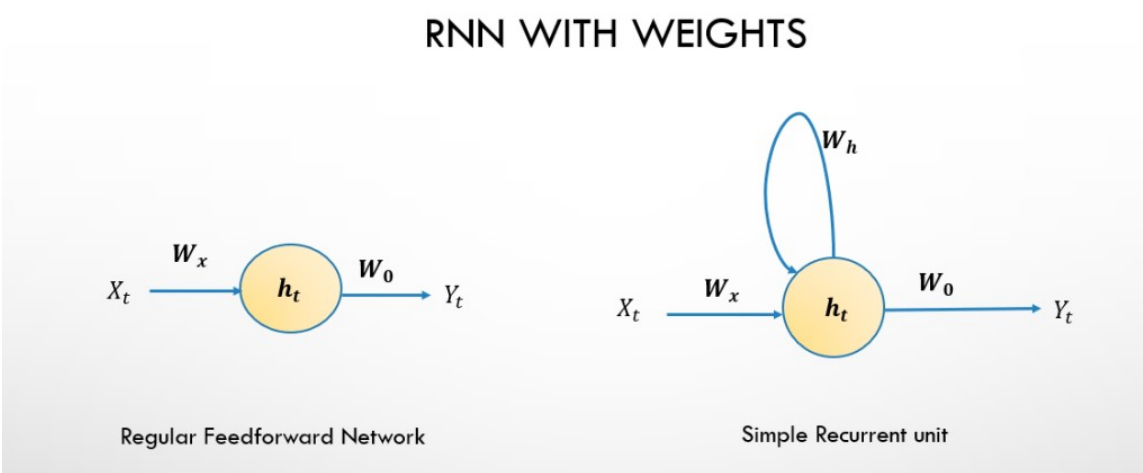
Recurrent Neural Networks: Introduction, Recurrent Neurons, RNN architecture, Bi Directional RNN, LSTM: Properties, LSTM network architecture, LSTM Units, GRU, Encoder-decoder, sequence to sequence architecture, applications of RNN.

Recurrent Neural Network (RNN):

RNNs are a class of neural networks that allow previous outputs to be used as inputs while having hidden states.

RNN has a concept of **"memory"** which remembers all information about what has been calculated till time step t. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.

Architecture of Recurrent Neural Network :-



So from above figure we can write below equations-

$$h_t = f\left(W_h{}^T h_{t-1} + W_x{}^T X_t + b_h \right)$$
$$Y_t = softmax\left(W_0{}^T h_t + b_0 \right)$$

f= Sigmoid , tanh , ReLu

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

Below is how you can convert a Feed-Forward Neural Network into a Recurrent Neural Network:

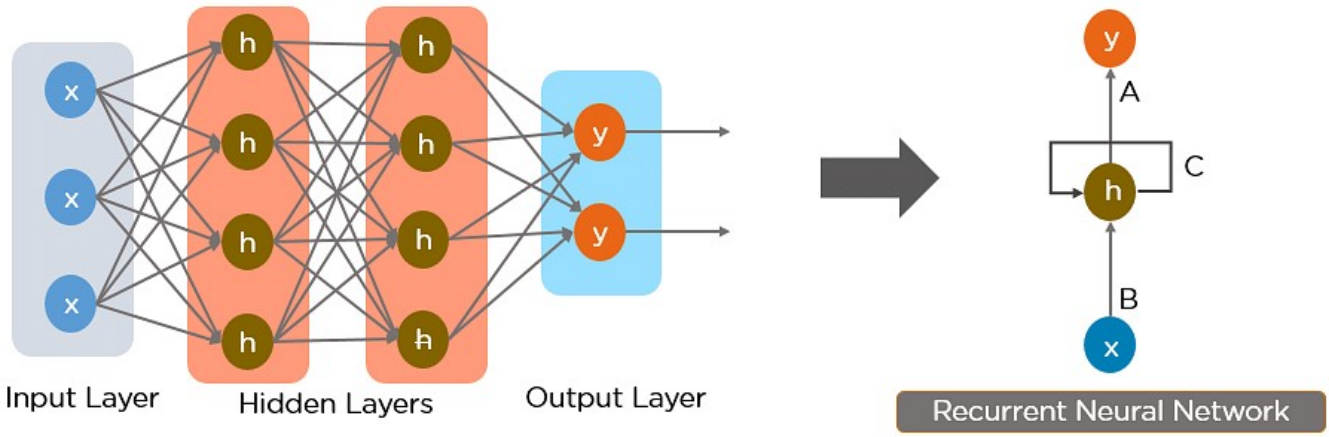
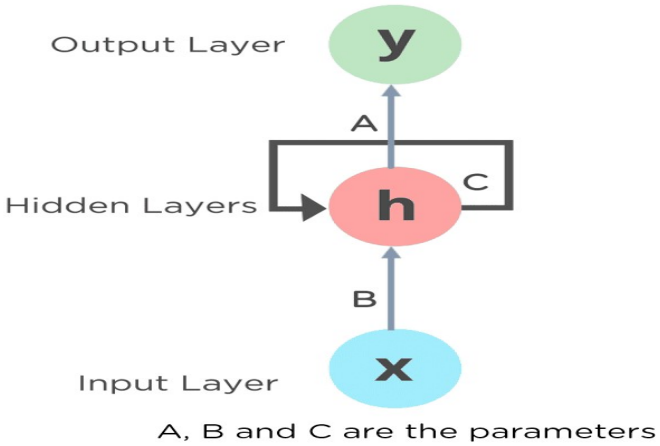


Fig: Simple Recurrent Neural Network

But by **<u>using Recurrent neural network concept we can combine all the hidden layers using the same weights and biases</u>**. All these hidden layers are rolled in together in a single recurrent layer.
So from here we can conclude that the recurrent neuron stores the state of a previous input and combines with the current input to maintain the sequence of the input data.



How Recurrent Neural Network works:-

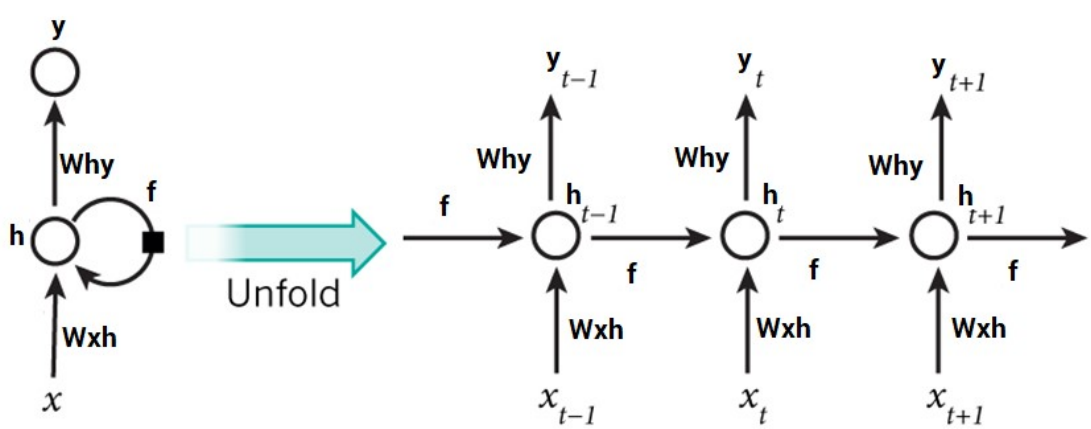
The recurrent neural network works as follows:

1. These all 5 layers of the same weights and bias merge into one single recurring structure.

Back propagation in a Recurrent Neural Network(BPTT)

To imagine how weights would be updated in case of a recurrent neural network, might be a bit of a challenge. So to understand and visualize the back propagation, let's unroll the network at all the time steps. In an RNN we may or may not have outputs at each time step.

In case of a forward propagation, the inputs enter and move forward at each time step. In case of a backward propagation in this case, we are figuratively going back in time to change the weights, hence we call it the Back propagation through time(BPTT).



In case of an RNN, if yt is the predicted value ȳt is the actual value, the error is calculated as a cross entropy loss –

Et(ȳt,yt) = – ȳt log(yt)

$E(\bar{y},y) = -\sum \bar{y}t \log(yt)$

RNN Applications:

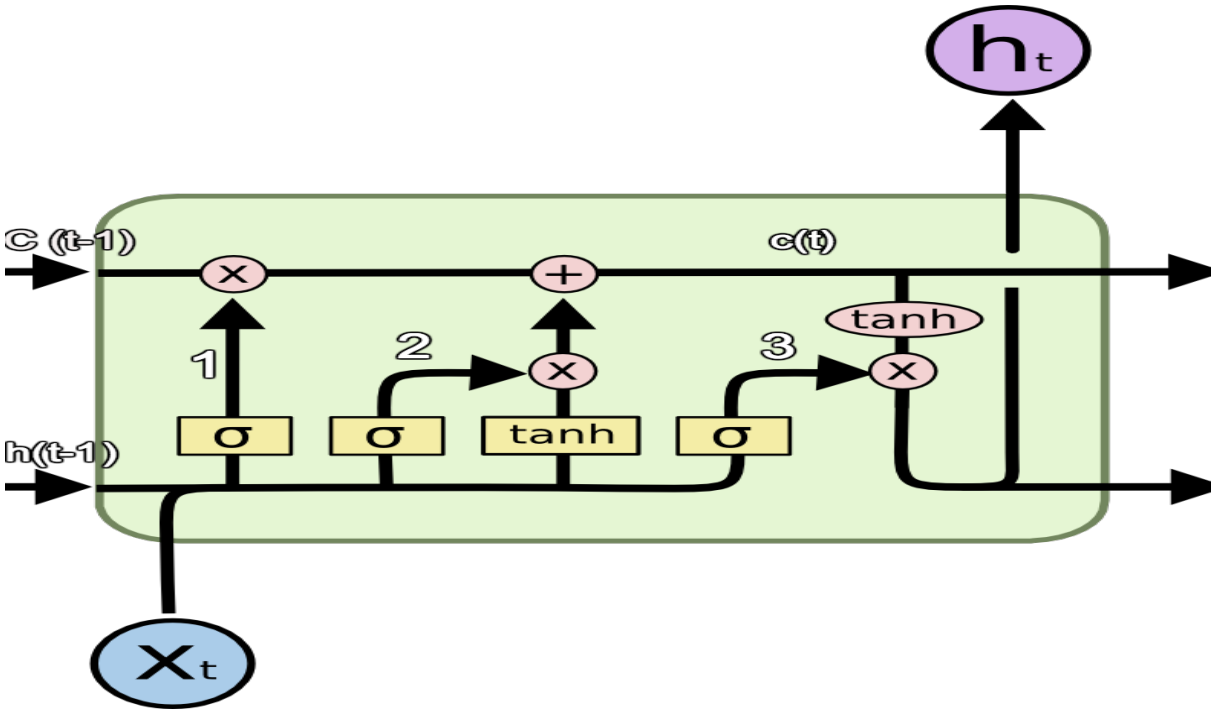
- Prediction problems
- Language Modelling and Generating Text
- Machine Translation
- Speech Recognition
- Generating Image Descriptions
- Video Tagging
- Text Summarization
- Call Center Analysis
- Face detection, OCR Applications as Image Recognition
- Other applications like Music composition

LSTM

LSTM Introduction:

- LSTM stands for Long Short Term Memory networks.
- Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.
- A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.
- LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.
- These have widely been used for speech recognition, language modeling, and sentiment analysis and text prediction.
- LSTM uses gates to control the memorizing process.

LSTM Network Architecture:



The symbols used here have following meaning:
a) X : Scaling of information
b)+ : Adding information
c) σ : Sigmoid layer
d) tanh: tanh layer
e) h(t-1) : Output of last LSTM unit
f) c(t-1) : Memory from last LSTM unit
g) X(t) : Current input
h) c(t) : New updated memory
i) h(t) : Current output

Components of LSTM architecture:

Information passes through many such LSTM units.There are three main components of an LSTM unit which are labeled in the diagram:

1. **Forget Gate:** LSTM has a special architecture which enables it to forget the unnecessary information .The sigmoid layer takes the input X(t) and h(t-1) and decides which parts from old output should be removed (by outputting a 0). In our example, when the input is 'He has a female friend Maria', the gender of 'David' can be forgotten because the subject has changed to 'Maria'. This gate is called forget gate f(t). The output of this gate is f(t)*c(t-1).

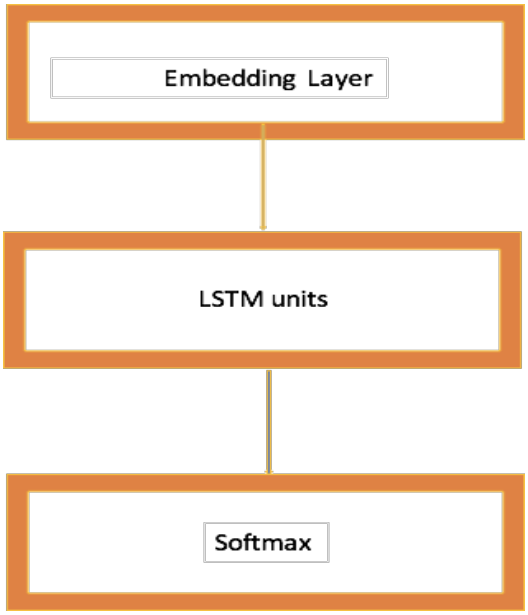
2. **Input Gate:** The next step is to decide and store information from the new input X(t) in the cell state. A Sigmoid layer decides which of the new information should be updated or ignored. A tanh layer creates a vector of all the possible values from the new input. These two are multiplied to update the new cell sate. This new memory is then added to old memory c(t-1) to give c(t). In our example, for the new input ' He has a female friend Maria', the gender of Maria will be updated. When the input is 'Maria works as a cook in a famous restaurant in New York whom he met recently in

a school alumni meet', the words like 'famous', 'school alumni meet' can be ignored and words like 'cook, 'restaurant' and 'New York' will be updated.

3. **Output Gate:** Finally, we need to decide what we're going to output. A sigmoid layer decides which parts of the cell state we are going to output. Then, we put the cell state through a tanh generating all the possible values and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to. In our example, we want to predict the blank word, our model knows that it is a noun related to 'cook' from its memory, it can easily answer it as 'cooking'. Our model does not learn this answer from the immediate dependency, rather it learnt it from long term dependency.

We just saw that there is a big difference in the architecture of a typical RNN and a LSTM. In LSTM, our model learns what information to store in long term memory and what to get rid of.

LSTM Network:



mathematical formulation of LSTM units:

$$i^{(t)} = \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)}) \qquad \text{(Input gate)}$$

$$f^{(t)} = \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)}) \qquad \text{(Forget gate)}$$

$$o^{(t)} = \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)}) \qquad \text{(Output/Exposure gate)}$$

$$\tilde{c}^{(t)} = \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)}) \qquad \text{(New memory cell)}$$

$$c^{(t)} = f^{(t)} \circ \tilde{c}^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} \qquad \text{(Final memory cell)}$$

$$h^{(t)} = o^{(t)} \circ \tanh(c^{(t)})$$

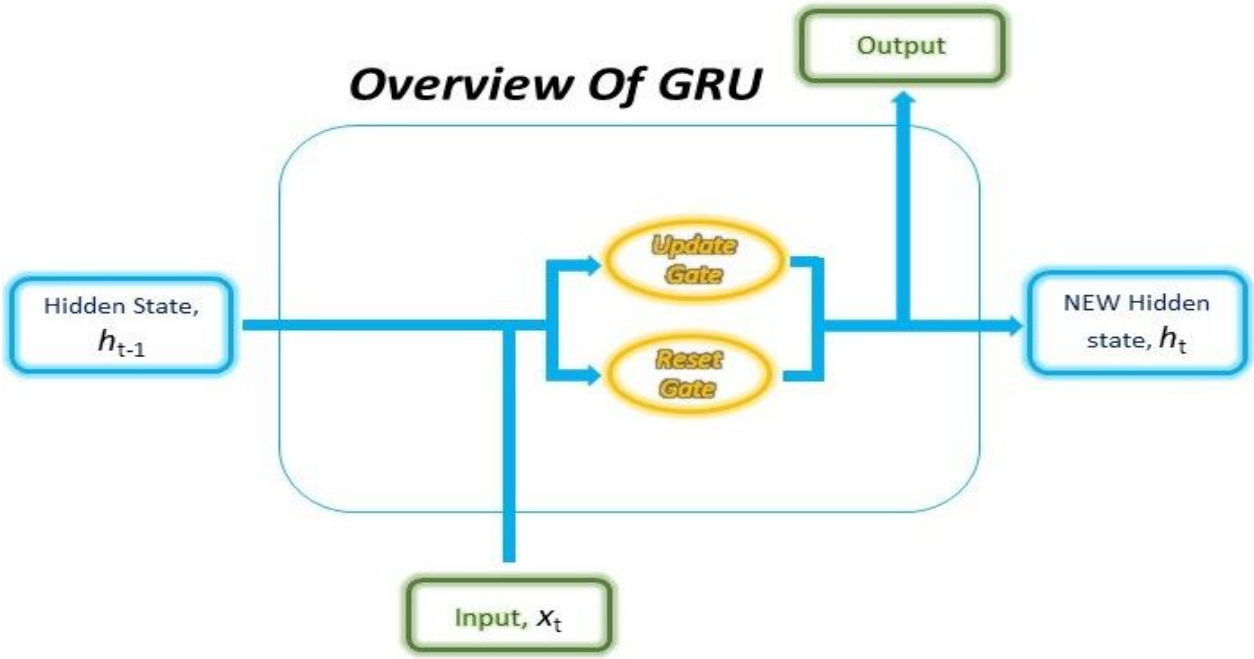
GRU networks:

GRUs are improved version of standard recurrent neural network.

A gated recurrent unit (GRU) is part of a specific model of recurrent neural network that intends to use connections through a sequence of nodes to perform machine learning tasks associated with memory and clustering, for instance, in speech recognition. Gated recurrent units help to adjust neural network input weights to solve the vanishing gradient problem that is a common issue with recurrent neural networks.

To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, **update gate and reset gate**. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

Structure of GRU:



GRU vs LSTM: