Long-Short Term Memory
(LSTM)

A Seminar Report

Submitted by

Mohammed Kashif
(21MA60R39)

MASTER OF TECHNOLOGY

COMPUTER SCIENCE AND DATA PROCESSING

at

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

WEST BENGAL-721302, INDIA

Contents **Page No.**

Figures **Page No.**

1. ABSTRACT

LSTM is an advancement of recurrent neural network (RNN) and it can learn past and future information. Unlike traditional RNN, it can capture a long sequence of text. Sentiment analysis is a very interesting topic as most of the e-commerce companies, social media network platforms, websites, big tech giants, and many more organizations used to work upon them to make decisions based on customer opinion. Earlier so many developers were tried sentiment analysis using machine learning. These days deep learning models have been used to implement sentiment analysis to achieve better performance.

The availability of a large amount of data provides us plenty of opportunities to utilize its insight for the sole purpose of marketing strategies, business development, and many more. LSTM models are trained before the deployment in real-world applications. It needs a data set, which varies with the need of the application. In language modelling/ text generation there involved computation of words when input sequence of words as a text provided. It is also used in image captioning where data in terms of image with captions and concludes its result into a sentence.

The LSTM unit is more intelligent and sophisticated. More precisely, it is composed of the gates that supposedly regulate better the flow of information through the unit. In practice, an LSTM is often used, as opposed to a vanilla (or standard) RNN, because it is more computationally effective. The LSTM was introduced to solve a problem that standard RNNs suffer from, i.e. the vanishing gradient problem.

2. Introduction

The Digital revolution just unlike any other revolution has changed everything around us. Nowadays, organizations of all types and across all industries utilize digital solutions to facilitate work operations, reduce costs, making error-free decisions. People worldwide are using technology to work, study, socialize, entertainment, shopping, or online banking. On a common ground people making interactions with technology, generates a large amount of digital data, combined with the data produced by organizations worldwide create a massive amount of digital data.

The total data created, captured, copied, and consumed worldwide is forecasted to increase rapidly, reaching 59 zettabytes in 2020, According to Statista. Most data are created digitally and never find their way into papers.

Classification is a term that refers to predicting that given data points belong to which class. The type of data analysis that extracts different models and describes important data classes is classification. There are the following machine learning techniques,

1. Logistic Regression
2. Support Vector Machine
3. Decision Tree
4. Random Forest
5. Naïve Bayes
6. K-nearest Neighbor
7. Artificial Neural Network

Depending on the performances of this technique in the different applications they are selected to use. So, in this report, we are discussing one of the Deep Learning techniques named Long Short-term memory.

3. What is Data?

Data is information of a person, facts, statistics that may be qualitative or quantitative. This information of a person is useful or more specifically collectively for a large group of people from the dataset. Let the example of Employees in a company has their information related to their job such as employment Id, Age, Skills, joining date, work profile. These data for faster processing are being converted into different file formats like CSV, Excel, Tabular, Text format, etc.

This data representation can be done in terms of any data structure, a pictorial representation that is in terms of graphs. Many machine learning algorithms employ various statistical, probabilistic, and optimization methods to learn from past known experiences. This learning finds useful patterns from large, unstructured, and complex datasets. These algorithms have a wide range of applications, in marketing just like customer purchase behavior detection, in the medical field for disease modeling, information theory for junk e-mail detection, and spam message detection. These applications are supervised learning variants rather than unsupervised learning. In the supervised learning method of Machine learning, decisions are made by learning dataset. At the learning stage labeling of data is known and this information is used for making decisions of unlabeled examples. The following is the NOIR classification of data,

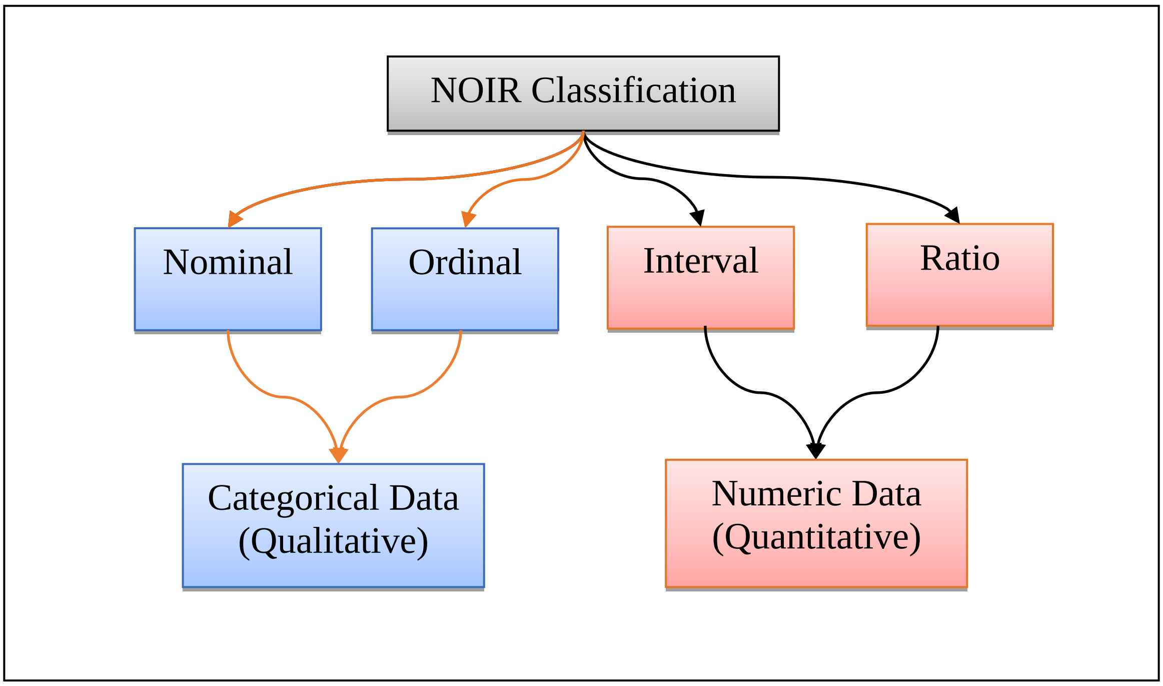


Fig. 1.1: Data Classification

4. Machine Learning:

This is a topic under the recent application useful in many applications. As in recent years or decades, two major changes have boosted machine learning use in a practical scenario. These are,

- Advent of high-speed processors.
- Huge amount of data being generated

Machine learning is the study of computer algorithms in a way that it improves/teaches itself through experience and by the use of data. It has included some parts of computational statistics and it is a part of Artificial intelligence. The algorithms used in machine learning are applicable in various works such as in email filtering, sentiment analysis, customer feedback support (Chat Box), speech and text recognition, and even in object detection techniques. In earlier days this application cannot fulfill the system requirement and was not feasible to develop such an algorithm to perform on computers. In recent processors, these tasks become easier, even though in recent days some machine learning algorithms are taking lots of time for their learning.

In nutshell, Machine learning more focuses on making a decision based on predictions made by computers. Not always machine learning statistics, while it also focuses on Data Mining. Data Mining is a field that has roots in exploratory data analysis. One part is machine learning i.e Deep learning which mimics the biological brain. In the field of business analytics, machine learning is often referred to as predictive analysis

Machine learning uses programs in terms of algorithms that modify certain variables present data structure depending on the data provided to it. Thereafter it performs certain useful tasks. For advanced tasks, it could be impossible to create an algorithm manually. From the practical point of view, it will no longer be effective for a person to develop such an algorithm.

5. Neural Network:

As biological brains activated signals depending on input signal and thickness of dendrites.

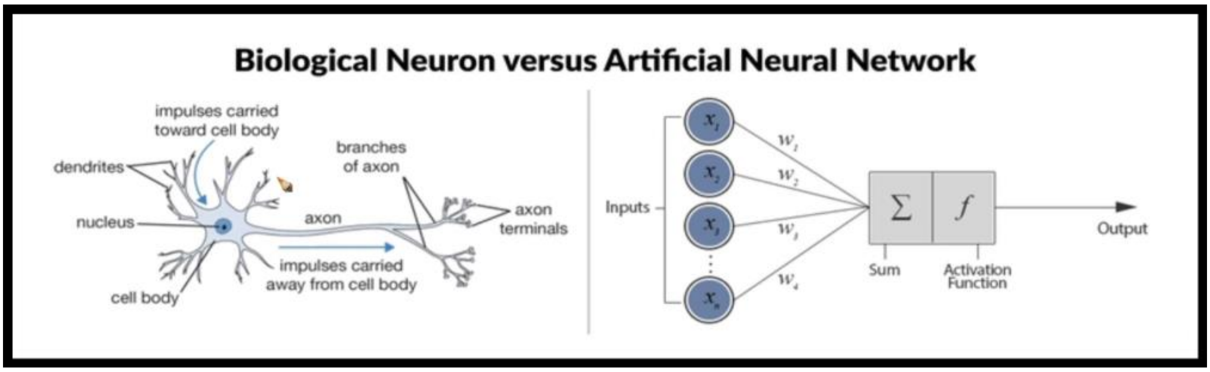


Fig 1.2 Biological Neuron vs Artificial Neural Network

A similar name was suggested by scientists as “Activation function”. It has different layers, Input layers, an Output layer, and intermediary layers. Then as backpropagation failed people understand that we need a very very deep neural network. By deep means, we have more and more layers. What does the biological neural network look like, By using multilayer perceptron we can come up with complex mathematical functions.

Training means finding the best edge weight using training data. Perceptron & LR are single neuron models for classifications. Linear regression the problem is: finding optimal values of weights in the expressions. In the case of linear regression, the activation function is identity functions. Find the weights to have smaller values of error.

Neural Network: A neural network consists of a set of nodes (neurons/units) connected by links.

- Each of its links has a numeric weight
- For poor output, the system alters the weight.

x_i is feature vector, y_i is the output of perceptron. We have a data D = {x_i , y_i}; x_i $\epsilon\mathbb{R}^4$; y_i $\epsilon\mathbb{R}^1$; The fully connected neural network for this data set is given in the following figure,

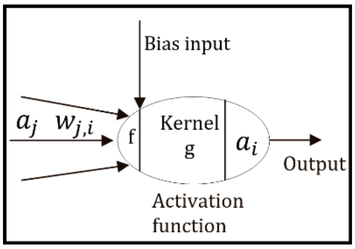


Fig 1.3 Perceptron

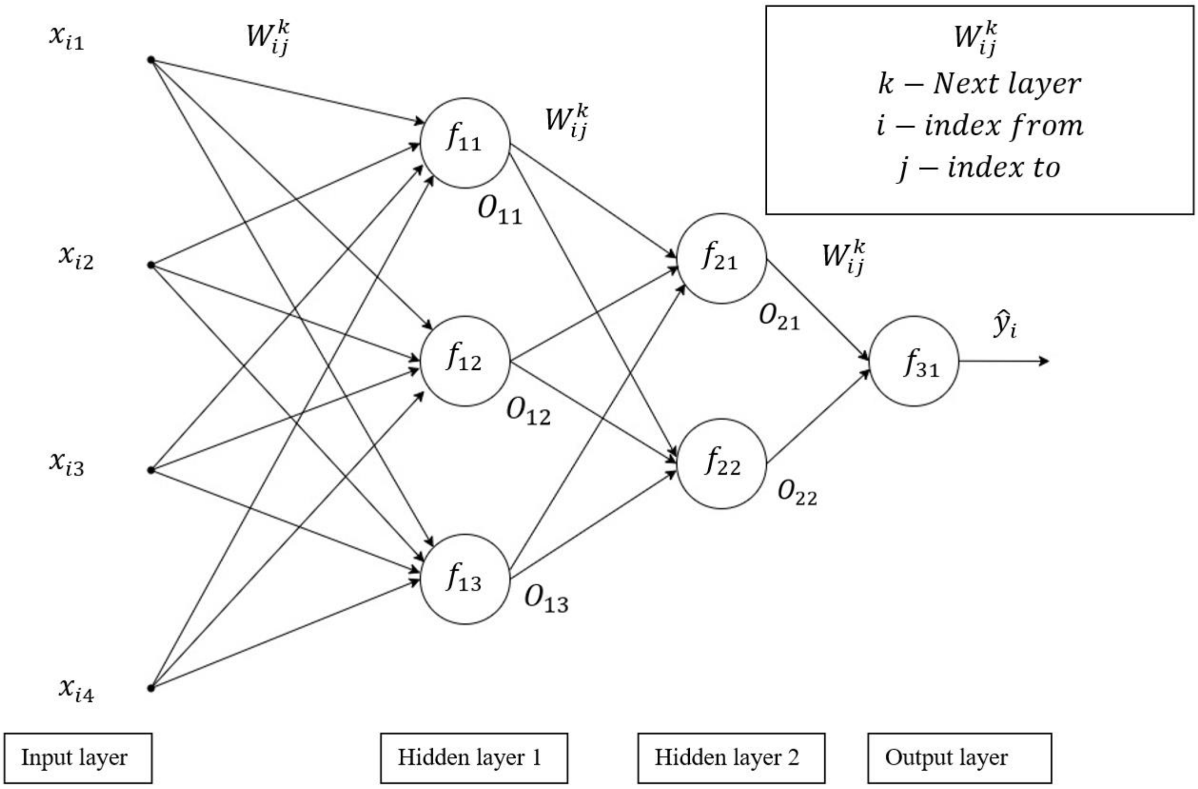


Fig 1.4 Multi-layer perceptron

O_{ij} is the output associated with ith layer and jth activation unit.

O_{ij} –

i = layer from which output is generating

j = index of activation unit

x_{ij} –

i = index of data point (feature vector x_i)

j = jth feature of ith data point (feature)

W_{ij}^{k} –
i = index from previous activation unit
j = index to next activation unit
k = Next layer number

Steps of training of MLP:

1. Define Loss function.
2. Optimization: Write the optimization problems

   Activation function:
   1. Identification function (For Linear regression)
   2. Sigmoid function (For Logistic regression)
   3. Threshold function (Perceptron)
3. Solve the optimization problem
   - a. Initialization of weights and learning rate
   - b. Find vector derivative of Loss
   - c. Find new values of weights using formula.

The weight update rule for Gradient descent (Single Layer)

$$E = \frac{1}{2}(\hat{y}_t - y_t)^2$$

$$W_j = W_j - \alpha \times \frac{\partial E}{\partial W_j}$$

α is Learning rate.

6. Deep Learning:

Earlier in 1980 till 2006 people used to built neural networks of very few layers like 2-3. The major problem was the vanishing gradient. In a deep neural network as we have 1000 weights while data points are considerably lower leads to overfitting problems. So lots of deep neural networks and shallow networks were very popular till 2006, but we have very little data, and too little computing problem undergoes an overfitting problem. If we have thousands of weights to learn, and in each iteration minibatch of data is provided for learning. These mini-batches of data are used to update the weights, compute the gradients/derivatives. This large computation in each iteration may take lots of time. For epoch and multiple epochs, it takes lots of computation time and hence it is cumbersome due to less computing power. There we even have mathematical problems like vanishing gradients.

By the time of 2010, we have lots of data (these are labeled data). This leads to having a high quality of data. There was extremely powerful computation in the computer is possible due to high performing CPU, in addition, we have highly powerful GPUs. These GPUs were found by scientists to be very very useful for deep learning networks. GPUs are almost 100 times faster than CPUs.

Machine learning has algorithms like SVM, the random forest has theoretical studies. In Modern deep learning algorithm as compared to earlier machine learning algorithms scientist tries to experiment then build theories. These trial-and-error methods are easy as we are having lots of data and computation power.

The first model was designed in the year 1957 [Rosenblatt]. It has seemed that it is similar to logistic regression. Alan Turing father of modern computing said what kind of neuron this is. How can we increase the intelligence of machines? A vague understanding that how the brain works. Neural has something called the basic building block of the brain. It has a biological nucleus that has lots of inputs and it sends signals further to other neurons. It takes signals from other dendrites and passed them to the next neuron. Imagine that I have some inputs, and provide these inputs with corresponding input. We have to decide which input is given higher importance to which input depends on the weights given to them. Then perform some mathematical operations on them using Activation functions.

Then came the era in the actual brain a neuron does not exist like this. One neural is connected to another neuron. Let's consider the structure of neurons. It is like building a structure of neurons. What we have is the network of neurons. There are networks of neurons that are interconnected there were lots of attempts. One of the works was by Hinton and others in 1986, which was Backpropagation which is just like the chain rule of derivatives. Around 1986 Artificial neural networks were a concept but at that time we don't have good computation resources computation power and enough data to work upon these concepts.

Later on, in the 1990s this hype went down, this time is called an AI-winter period. Later many persons were working on other techniques around 1995-2009 (SVM, RF, GBDT), jeff Hinton worked almost 28 years of life to work on NN, in 2006 he presented a paper on how to train Neural Networks. In 2012(Watershed moment of Deep Learning) there was a competition of Imagenet. We have to use a large data set. A team Using this large data set applied and designed a deep neural network that has beaten all other algorithms. This result then blew air to research under the Deep Neural network. Various companies started noticing deep learning. These companies have lots of data as well as computational resources.

They have started investing huge money in the application of AI. Lots of scientists noticed that deep learning has great and great applications. Siri, Cortana, Alexa, and Google Assistant. Self-driving cars, skin cancer diagnostic used smartphones camera. There were lots many applications. Deep learning has algorithms like CNN, which were old later advanced processors and large data lead to enhance applications of CNN.

7. Activation Function:

Activation functions used in neural networks must be differentiable and easily differentiable. Deep neural networks have a large number of activation units. While the computation of derivatives in the learning process through backpropagation. It consumes a significant amount of time. If these functions are easily differentiable, then computation can be saved. Also, some other techniques like Memoization are used to save time to some extent.

1. Linear function:

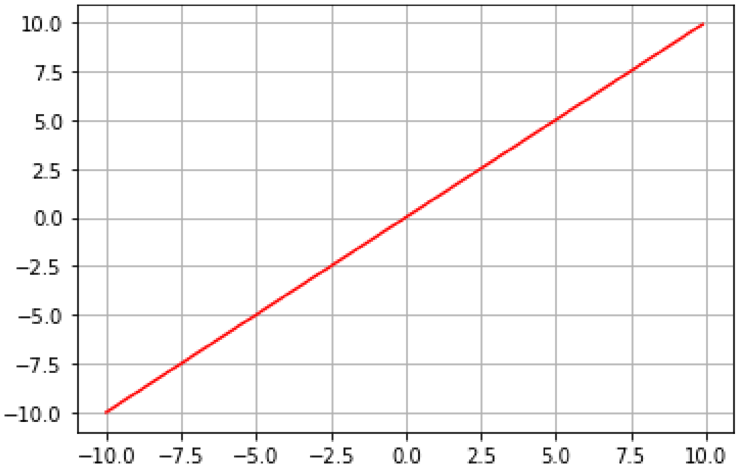


Fig 1.5 Linear activation

The linear activation function is the identity function,

Let,

$$z = w_{ij}^{k} \cdot x_{ij}$$
$$f(z) = z$$
$$f'(z) = 1$$

2. Sigmoid function

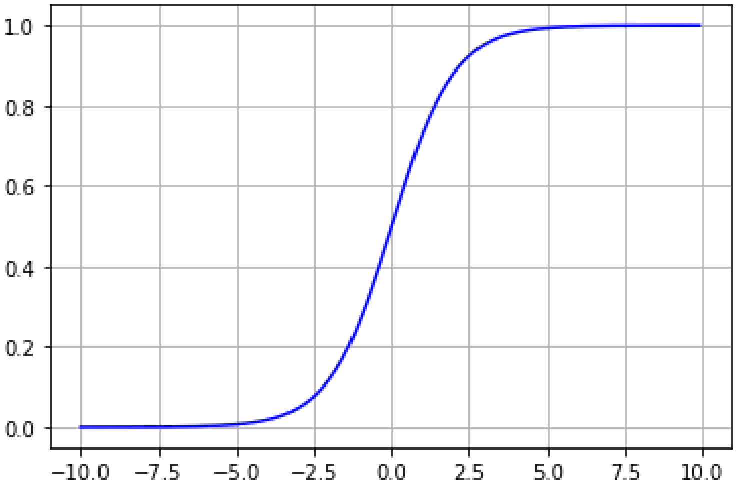


Fig 1.6 Sigmoid (Logistic) activation

It transforms input values in the range of [0,1], we use the following function,

$$f(z) = \sigma(z) = \frac{1}{1 + e^{z}}$$
$$f'(z) = \sigma(z)(1 - \sigma(z))$$

3. tanh (hyperbolic tangent):

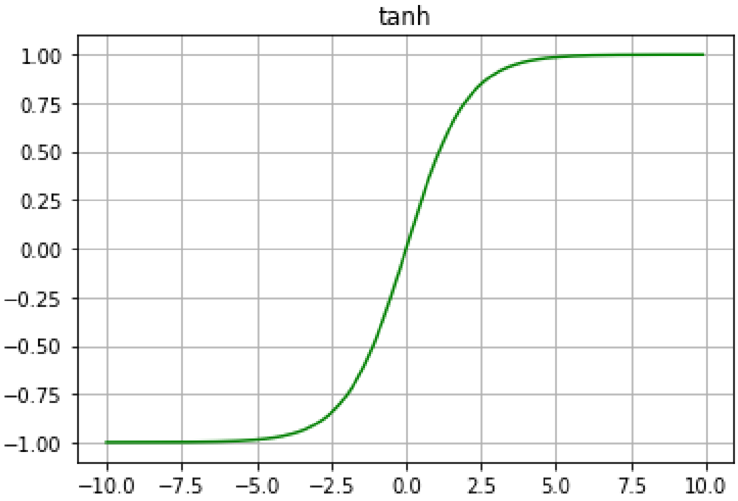


Fig 1.7 tanh activation

$$f(z) = \frac{1 - e^{-2z}}{1 + e^{2z}} = \tanh(z)$$

$$f'(z) = 1 - \tan h^2\ (z)$$

4. ReLU (Rectified Linear Unit):

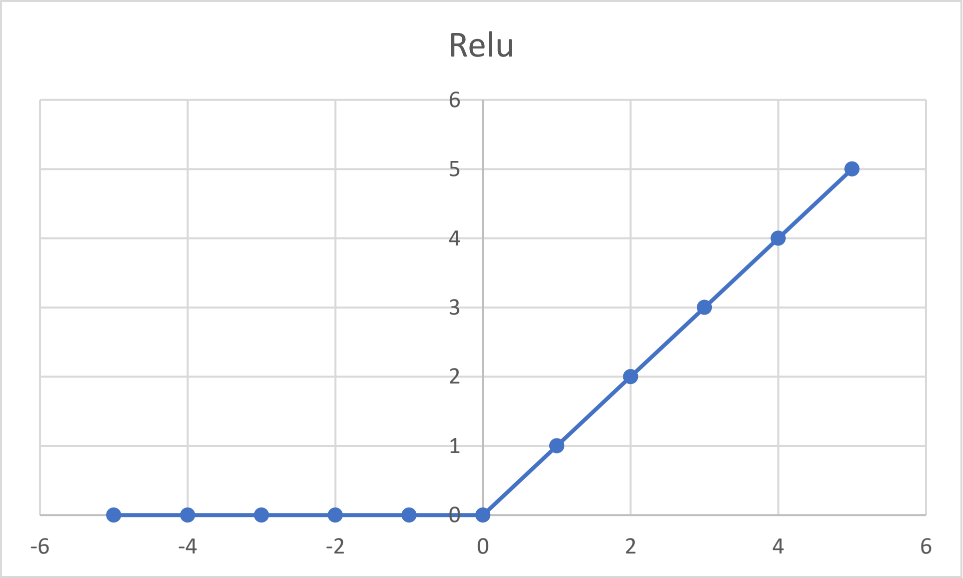


Fig 1.8 Rectified Linear activation

$$f(z) = \max(0, z)$$
$$f'(z) = \ 1 \ \ z > 0$$
$$= \ 0 \ \ z \leq 0$$

But this function is not differentiable at z=0, hence we use softplus function,

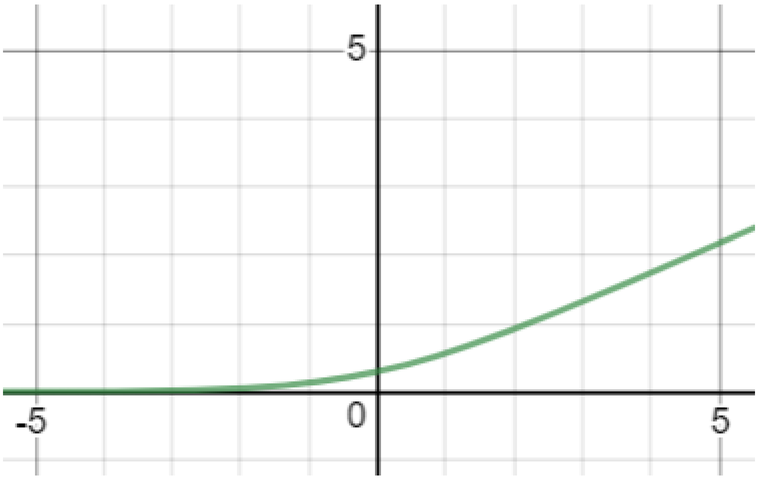


Fig 1.9 Softplus activation

$$f(z) = \log(1 + e^{z})$$

5. Non-linear ReLU:

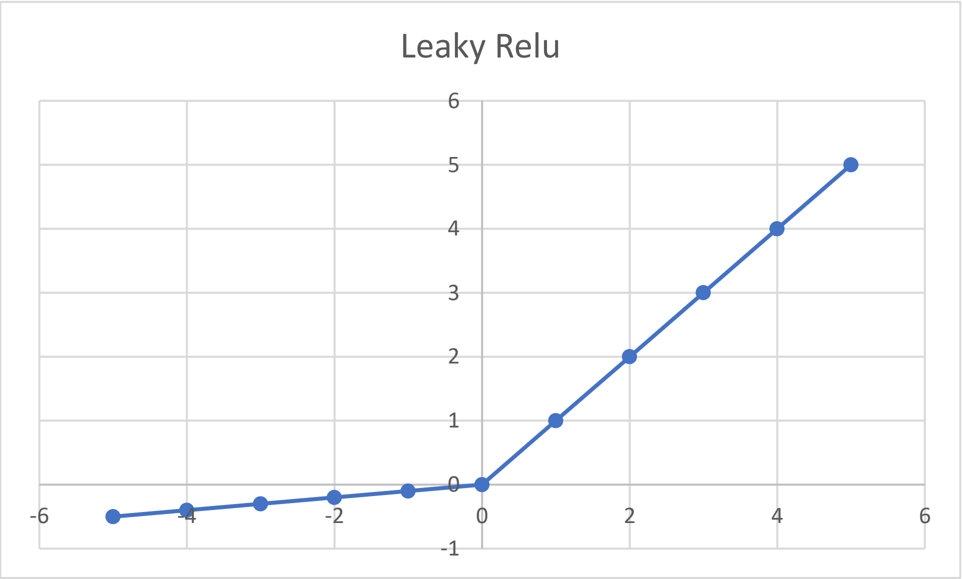


Fig 1.10 Leaky ReLU

$$\begin{aligned} f'(z) &= z \quad z > 0 \\ &= az \;\; z \leq 0 \end{aligned}$$

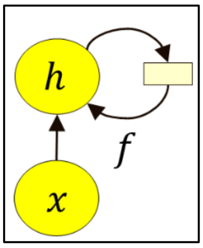
$$\begin{aligned} f'(z) &= 1 \;\; z > 0 \\ &= a \;\; z \leq 0 \end{aligned}$$

8. Recurrent Neural Network:

The sequence of values/words is very important, let two sentences, R1: This phone is very fast; R2: This very is phone fast. R1 is grammatically correct, while R2 is incorrect. MLPs do not guarantee retaining sequence information. Hence there is the development of RNNs. Examples of sequence data are Time series data, Machine translation, Speech recognition, Image caption.

Recurrent Neural Network: *It is a collection of neural Networks for processing sequential data that arrives at a different time interval.*

- The learned model has the same input size, that can be dealt with the same transition function (f).
- RNNs could output their understanding depending on past information.



$h^{(t)} = f\ (h^{(t-1)}, x^{(t)}; \theta)$

$\theta -$ Parameters

$h^{(t)}$ – Next output

$h^{(t-1)}$ – Previous output

$x^{(t)}$ – Input at time t

Fig 1.11 Simple RRN unit

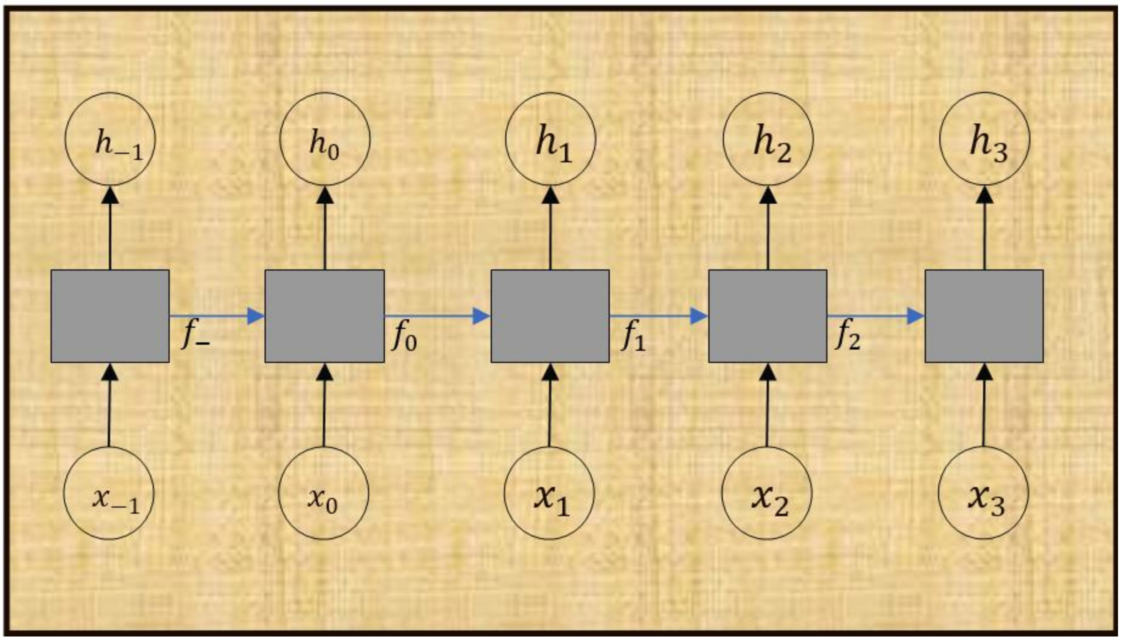


Fig 1.12 RNN at different time steps

9. LSTM:

The LSTM network is an advanced version of a recurrent neural network. The RNN is the network that tries to model sequence or time-dependent data like language, stock market prediction, speech synthesis. As simple RRNs cannot take care of long-term dependencies. In LSTM we use cell state block to take past information and transfer it to the desired period. This led to a new feature in RNNs of long-term dependencies.

Embedding:

The embedding used in LSTM to convert word into machine-readable vectors. Let's say there is a sequence of words as a comment or review. These words are stored in sequence and transferred to the LSTM model at different time intervals but not directly. Embeddings are most commonly used for working with text data. The one-hot encoded vectors have large dimensions, which are reduced to smaller dimensions with floating points numbers. Even the vocabulary of 50K words is mapped to a floating-point numbered valued vector of size 0.1K. These vectors now become very handy.

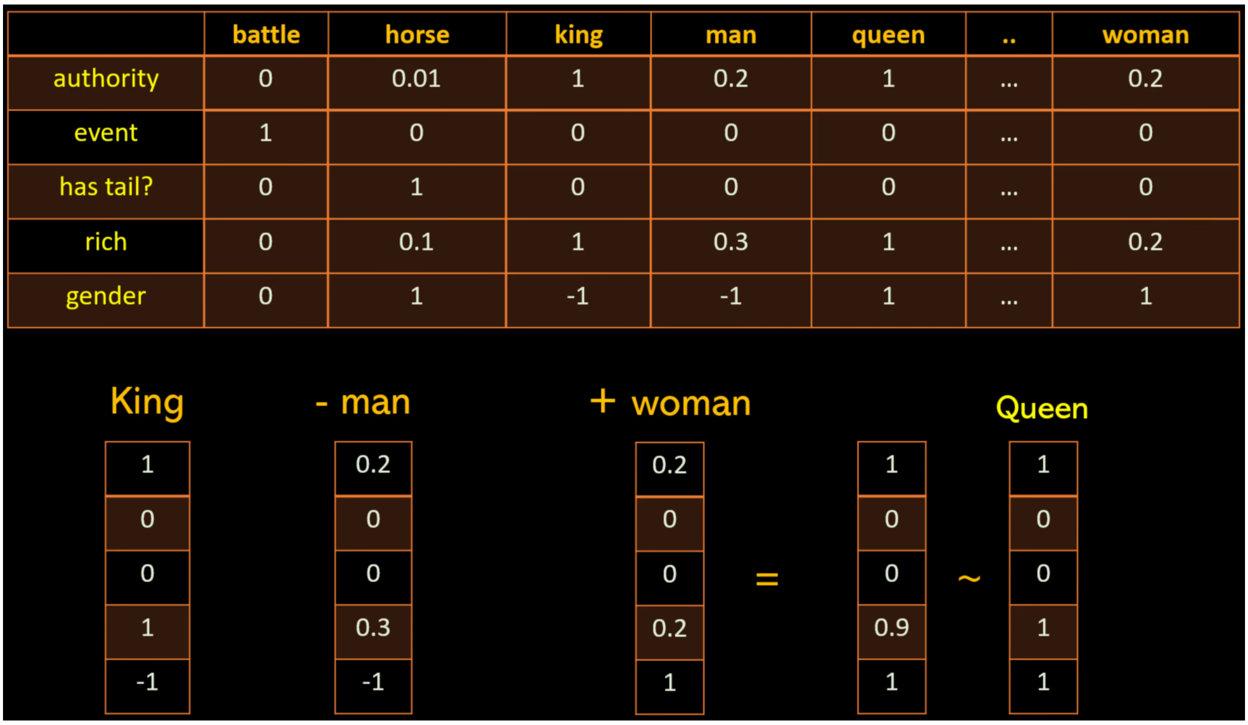
Let this is given vector of word 'Good':

5	→	0.9
0.1	→	0.009
-1	→	-0.46

Vectors undergo many transformations due to various math operations. There are various methods to do embeddings such as CBOW, Skipgram, TFID, etc. Word2vect is the technique that allows you to do mathematics with the words. So, you can give this vector to the computer for further engineering analysis. There are methods to convert different words into vectors. We can train over the network to predict sentiments. These features will be learned by the network by adjusting its weights. As of the usual case, here is also learning efficiency depends on the quantity and quality of data.

Word	**King**	**Horse**
Authority	1	0
Rich	1	0
Tail	0	1
Gender	-1	-4
Vector-	**[1,1,0,-1]**	**[0,0,1,-1]**

Table 1.1 Feature relation in word2vect



	battle	horse	king	man	queen	..	woman
authority	0	0.01	1	0.2	1	...	0.2
event	1	0	0	0	0	...	0
has tail?	0	1	0	0	0	...	0
rich	0	0.1	1	0.3	1	...	0.2
gender	0	1	-1	-1	1	...	1

King
- man
+ woman
Queen
1 0 0 1 -1
0.2 0 0 0.3 -1
0.2 0 0 0.2 1
=
1 0 0 0.9 1
~
1 0 0 1 1

Fig 1.13 Word Embedding

Authority, event, has tail?, rich, gender features. While King, man, woman, Queen are feature vectors in machine learning. In actual practice we don't know features, there are corresponding values as each element of the feature vector. Vector size decides based on weight matrices of MLPs used in CBOW or Skipgram techniques.

- Conversion of words and documents in machine useful/understandable format.
- It allows us to capture relationships among words.
- Keras uses word2vect embeddings.
- Embedding is not handcrafted. Instead, they are learned during neural network training.

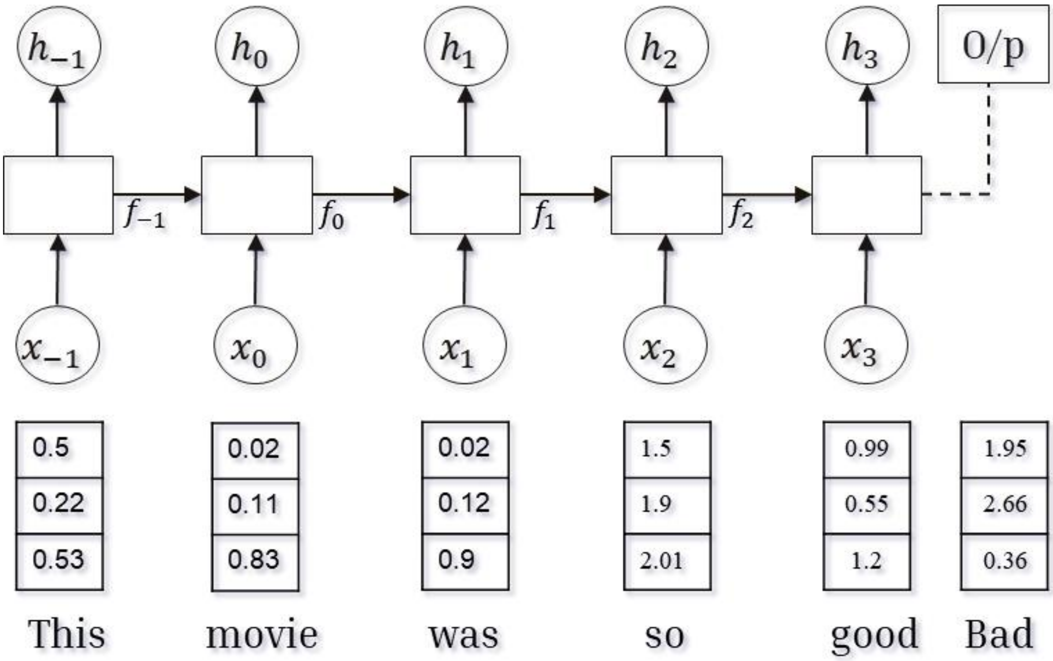


Fig 1.14 Input word vector to RNN/LSTM

One Hot Encoding: It is the technique to convert categorical data into numerical data. Here each vector has a dimension equal to dictionary size.

Steps involved in One Hot encoding:

1. Create a dictionary: Set of all the words in these reviews. Here in this, we have words like pasta this is very…. All unique words.

2. construct a vector: lets us have n reviews and d unique words. Vector V is created such that ||V|| =d, where each element will be either 1 or 0. Where 1 denotes the word is present in the review and 0 denotes it is not present.

Let us consider each word is of a different dimension.

R1= This pasta is very tasty and affordable.

V1=vector corresponding to review r1.

R1	a	an	any	affordable	pasta	this	very	watch
v1	0	0	0	1	1	1	1	0
Index	0	1	2	.	.	.	d-2	d-1

Table 1.2 One Hot Encoded vector

First row represents dictionary words in dictionary order, second row have value that corresponds to occurrences of words. Third row denotes numbering of dictionary words. So that $V_i \in \mathbb{R}^d$, has objective to create a vector of closer values when their text is similar i.e similar in meaning.

In code, we used following method for embedding. For n reviews, d is dictionary size,

1. Create dictionary.
2. Sort in descending order of dictionary.
3. Padding.
4. Embedding (by using user defined vector length).

Index	Word	Frequency
0	a	4850
1	The	4900
.	.	.
d-1	.	.
d	as	5001

Table 1.3 Dictionary Obtained

Index	Word	Frequency
0	as	5001
1	The	4900
.	.	.
d-1	.	.
d	a	4850

Table1.4 Sorted dictionary wrt frequency

$$x_1 = < x_{11}, x_{12}, \ldots, x_{1|x_1|} >$$

$$x_2 = < x_{21}, x_{22}, \ldots, x_{2|x_2|} >$$

.

.

$$x_n = < x_{n1}, x_{n2}, \ldots, x_{n|x_n|} >$$

These n vectors are of different sizes. So, perform padding by considering size of reveiws in terms of words.

10. Working of LSTM:

In LSTM there are three Gates, one cell state as an information carrier, and number pointwise operators.

Cell State:

Think of it as a memory of a network. It reduces the effect of short-term memory. While these data travels through sequentially arranged units, the information is added or removed to the cell state via input and forget gates respectively.

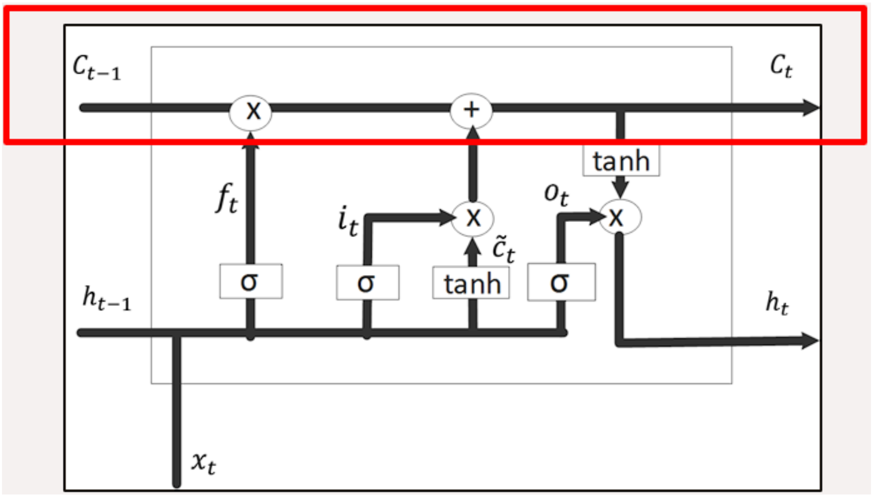


Fig 1.15 Cell state

Gates:

Gates are different neural networks that decide which information is allowed on the cell state. Gates learns what information is relevant to keep or forget during training. Gates contains sigmoid activation. The vectors provided to the sigmoid activation function are having data that is of varying values. However sigmoid activation squishes value in the range of 0 to 1. That is helpful to update or forget data because any number getting multiplied by ‘0’ is ‘0’ causing the value to disappear or be forgotten and any number multiplied by ‘1’ is the same value. Therefore, that value stays the same or is kept as it is.

The network will learn what data should be forgotten or what data is important to keep. We have 3 different gates that regulate information flow in an LSTM cell. These all are essential components of this model with definite significance.

1. Forget Gate
2. Input Gate
3. Output Gate

1. Forget gate:

This gate decides which information should be thrown away or kept, information from the current input is passed through the sigmoid (0-1) activation function.

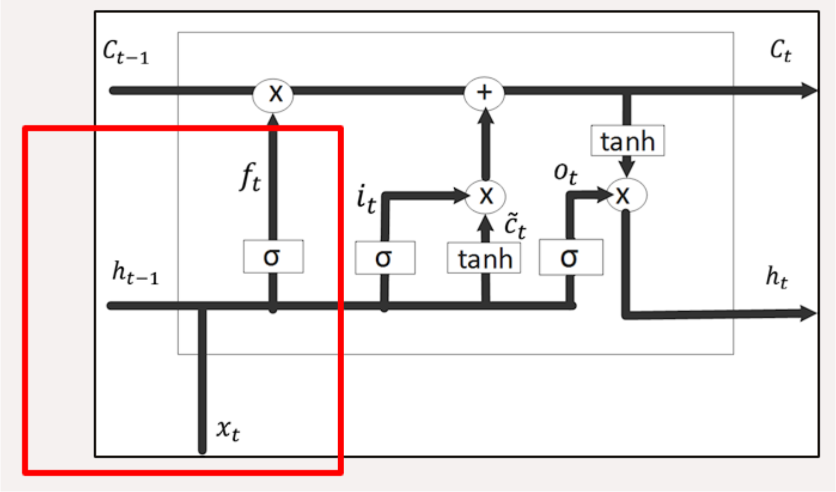


Fig 1.16 Forget gate

h_{t-1} : output of previous state at t= -1

x_t : input to LSTM network at t=0

W_f : weight matrix at forget gate (adjust in backpropagation)

b_f : fixed biased value at forget gate

f_t : forget gate vector

$$f_t = \sigma\,[\,W_f\,.\;\;[h_{t-1}, x_t] + b_f]$$

2. Input gate:

The input x_t and h_{t-1} concatenated and passed through tanh activation unit after vector multiplication of with weight matrix W_i. These values are squished in [-1,1]. To update cell state, we have an input gate first we pass hidden states value & current input to a sigmoid function that decides which value will be updated by transforming value to be between 0 & 1.

Now we have enough information to calculate self-state.

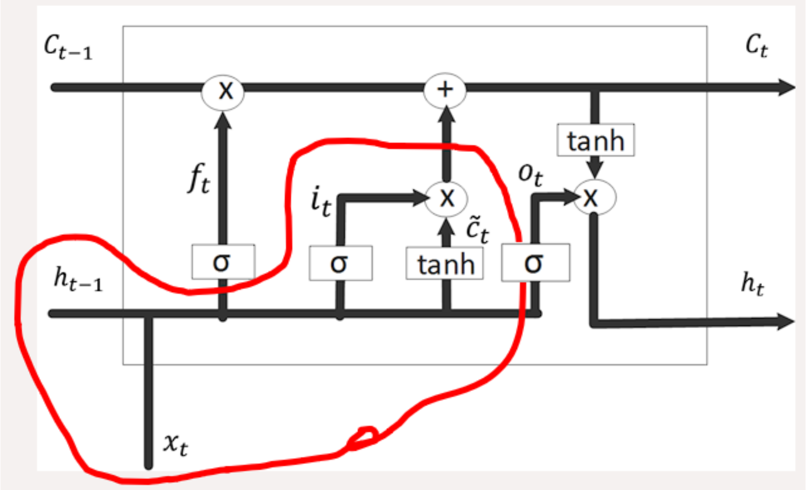


Fig 1.17 Input gate

Important Terms:

W_i : weight matrix at input gate (adjusted in backpropagation)

b_i : fixed biased value at input gate

i_t : input gate vector

$$i_t = \sigma\,[\,W_i\,.\;[h_{t-1}, x_t] + b_i]$$

W_C : weight matrix at input gate for tanh activation function (adjusted in backpropagation)
b_C : fixed biased value at input gate for the tanh activation function
$\tilde{C}_t$: input gate vector

$$\tilde{C}_t = tanh\,[\,W_C\,.\,[h_{t-1}, x_t] + b_C]$$

The term $f_t * C_{t-1}$ is the calculated value has possibility of dropping value in a cell state if its value not zero then we can take output from input gate and data pointwise addition which update, cell state to new values. While these updated values further updated as,

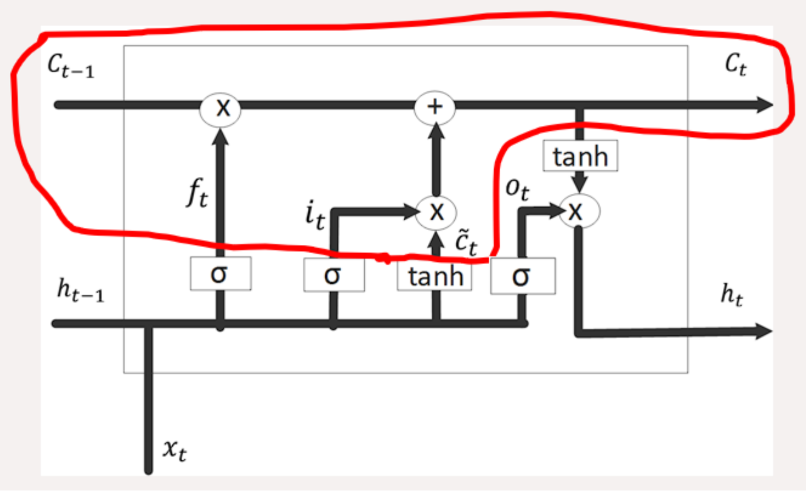


Fig 1.18 Cell state information modification

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

While the term $i_t * \tilde{C}_t$ is updates through input gate, where '*' is the pointwise multiplication.

3. Output Gate:

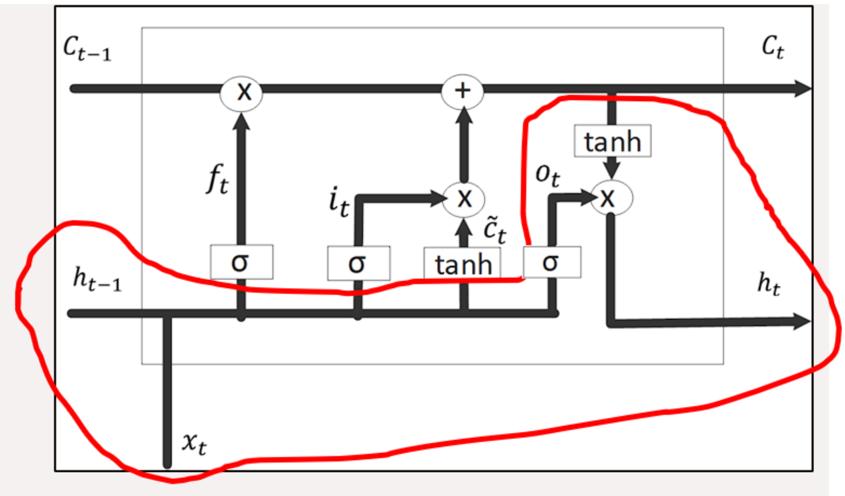


Fig 1.19 Output gate

Output gate decide what information should h_t contains of previous inputs. First, we pass previous hidden state and current input until a sigmoid function to get O_t. Newly made 'C_t' to tanh function. We multiply tanh output to sigmoid output to decide which information the h_t should carry. The output is h_t, C_t and h_t are carried over to the next step. This tanh used here is not activation unit. we apply the elements wise tanh Operation on our new cell state and multiply it with O_t to produce the new output. Here we apply tanh operation just to regulate the values, to prevent it from getting in exploding or vanishing gradients problem.

LSTM is good for processing sequence, while suffering from short-term memory, and to mitigate short-term memory we used the mechanism of all gates. Gates are just neural networks that regulate the information being passed from one-time step to the next. LSTM is used in state-of-the-art applications like speech recognition, speech synthesis, natural language understanding.

Limitations:

LSTMs have become popular as it is capable to solve the vanishing gradient problem. However, it is found that they failed to remove it completely. The problem is present in the traversal path where data still has to move from cell to cell for its evaluation. LSTMs are models from the applied recurrent neural network and are popular as they could solve the major problem of vanishing gradients.

There are various gates used and additional neural networks, which lead to the increased complexity of LSTMs. Lots of computation time and resources are needed to get trained our model, to become ready for real-world applications. Higher configuration of hardware requirements is another problem, for satisfactorily training of the model. They needed high memory and bandwidth of linear layers in each cell. So, hardware-wise LSTM is quite inefficient.

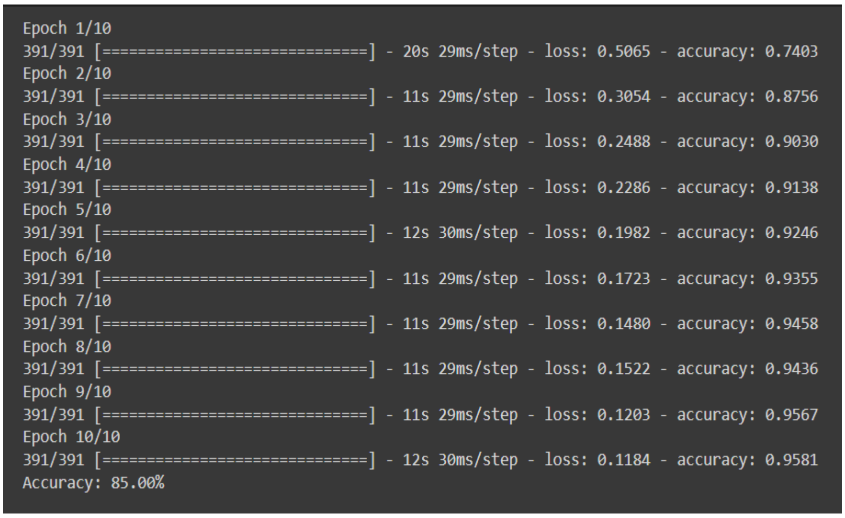
Due to huge data, scientists are searching for models that can remember the past information for a longer duration of time than LSTM could be. LSTM has a huge impact due to random weight initialization hence behaves similar to that of simple RNN, while small weight initializations suppress this effect.

Dropout is a regularization method while training a network, the recurrent connections to LSTM units are probabilistically excluded from activation and weight updates. LSTMs are prone to overfitting and it is difficult to apply the dropout algorithm to curb this issue.

11. Results:

Parameters	Values
Epoch	10
LSTM layers(m)	100
Vocab_size	5000
max_review_length	600
embedding_vecor_length (n)	32

1.Max_review_length
 = 600 is padding length.
2.No. of hyperparameters (LSTM)
 = $4(m^2 + mn + m)$ = 53200
3.No. of hyperparameter (Embedding)
= Vocab_size * n = 160000



```
Epoch 1/10
391/391 [==============================] - 20s 29ms/step - loss: 0.5065 - accuracy: 0.7403
Epoch 2/10
391/391 [==============================] - 11s 29ms/step - loss: 0.3054 - accuracy: 0.8756
Epoch 3/10
391/391 [==============================] - 11s 29ms/step - loss: 0.2488 - accuracy: 0.9030
Epoch 4/10
391/391 [==============================] - 11s 29ms/step - loss: 0.2286 - accuracy: 0.9138
Epoch 5/10
391/391 [==============================] - 12s 30ms/step - loss: 0.1982 - accuracy: 0.9246
Epoch 6/10
391/391 [==============================] - 11s 29ms/step - loss: 0.1723 - accuracy: 0.9355
Epoch 7/10
391/391 [==============================] - 11s 29ms/step - loss: 0.1480 - accuracy: 0.9458
Epoch 8/10
391/391 [==============================] - 11s 29ms/step - loss: 0.1522 - accuracy: 0.9436
Epoch 9/10
391/391 [==============================] - 11s 29ms/step - loss: 0.1203 - accuracy: 0.9567
Epoch 10/10
391/391 [==============================] - 12s 30ms/step - loss: 0.1184 - accuracy: 0.9581
Accuracy: 85.00%
```

Fig 1.20 Model Accuracy

12. Conclusion:

It has been found efficient in terms of time and space complexity. The long-term decencies made it useful in many applications like Language Translation, Music generation, Speech and Handwriting Recognition, Image processing, Language modeling, or text generation. The long term dependency is primarily due to the cell state. The cell state is passed through each time step(it does get updated at every time step) as a fraction of its previous time step.

The LSTM architecture is designed in such a way that it addresses the vanishing gradient problem by introducing a memory cell that can preserve the state over long periods. Here, by changing the values of outputs of input gate, output gate, and update gate, one can control the amount of information the network retains/discards over the entire input series as well as control dependency on individual inputs. In the recurrency of the LSTM, the activation function is the identity function (similar to ReLU) with a derivative of 1.0. In the backpropagated gradient neither does the gradient explode nor does it vanish.

However, there is exploding gradient problem with LSTMs. Large updates to weights during training can cause a numerical overflow or underflow often referred to as “exploding gradients.” The problem of exploding gradients is more common with recurrent neural networks, such as LSTMs given the accumulation of gradients unrolled over hundreds of input time steps.

13. References:

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/
- https://www.hindawi.com/journals/misy/2021/6654029/
- https://www.hindawi.com/search/all/lstm/
- https://medium.com/@dcameronsteinke/tf-idf-vs-word-embedding-a-comparison-and-code-tutorial-5ba341379ab0
- Deep Learning, NLP, and Representations - colah's blog
- Improved Time Series Prediction Using LSTM Neural Network for Smart Agriculture Application. Authors: Budi Cahyo Suryo Putro S , I Wayan Mustika, Oyas Wahyunggoro, Hutomo Suryo Wasisto.
- Applying Deep Learning Approach to Targeted Aspect-based Sentiment Analysis for Restaurant Domain. Authors: Win Lei Kay Khine1 , Nyein Thwet Thwet Aung, University of Information Technology, Yangon, Myanmar.