$h_0 = 0$ [RNN Numericals]

Activation function = tanh.

$w_x = [0.6, 0.4]$ (for x_1, x_2)

$w_h = 0.5$

$w_y = 1$

$b = 0$

Input sequence $x_1 = [1, 0]$

$x_2 = [0, 1]$

$y = 0.7$

Hidden state update

$$h_t = \tanh(w_x \cdot x_t + w_h h_{t-1} + b)$$

Output

$$\hat{y} = w_y h_T$$

Loss $L = \frac{1}{2}(\hat{y} - y)^2$.

at t1

$w_x \cdot x_1 = 0.6 * 1 + 0.4 * 0 = 0.6$

$h_1 = \tanh(0.6 + 0.5 * 0) = 0.537$.

at t2.

$w_x \cdot x_2 = 0.6 * 0 + 1 * 0.4 = 0.4$

$h_2 = \tanh(0.4 + 0.5 * 0.537)$

$= 0.584$

Output computation

$y' = w_y h_2 = 1 * 0.584 = 0.584$.

Loss $= \frac{1}{2}(0.7 - 0.584)^2 = 0.0068$.

Chapter-9

What is LSTM?

LSTM (Long Short-Term Memory) is a special type of Recurrent Neural Network (RNN) designed to learn long-term dependencies in sequence data.
It was introduced to overcome the vanishing gradient problem faced by standard RNNs.

LSTMs are widely used in:

- Text and language modeling
- Speech recognition
- Time-series forecasting
- Machine translation

Why do we need LSTM when RNN already exists?

A basic RNN struggles to remember information from far back in the sequence.
LSTM solves this by introducing a memory cell and gating mechanisms that control:

- What to remember
- What to forget
- What to output

LSTM Architecture (Core Components)

An LSTM cell contains:

1. Cell state (Ct) – long-term memory
2. Hidden state (ht) – short-term memory
3. Three gates:
   - Forget gate → decides what to remove
   - Input gate → decides what to add
   - Output gate → decides what to output

This gate-based control makes LSTM very powerful for sequence learning.

Working Principle of LSTM (Long Short-Term Memory)

An LSTM processes sequence data one time step at a time while maintaining two states:

- Cell state (C_t) → long-term memory
- Hidden state (h_t) → short-term output

The key idea is to control the flow of information using gates so that important information is remembered for long durations and irrelevant information is forgotten.

① Forget gate
→ It takes care of, which information to be removed from the memory
at each time step 't', it takes
→ input – x_t
→ previous state – h_{t-1} (hidden state)
→ previous cell state – C_{t-1}

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

$f_t = [0-1]$ 0 → remove inf^n.
1 → keep inf^n.

② Input state
→ It takes care of, which information to be added to the memory.

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

W_i & b_i – are weight & bias – learnable parameter.

Here a candidate cell state is created $\bar{C}_t$

$$\bar{C}_t = \tanh[W_c(h_{t-1}, x_t) + b_c]$$

③ Update long-term memory

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \bar{C}_t.$$

It preserves important information for long term purpose.

Step-by-Step Working of LSTM

At each time step t, the LSTM takes:

- Input vector x_t
- Previous hidden state h_{t-1} (Short-term)
- Previous cell state C_{t-1} (Long-term)

Step 1: Forget Gate – Remove Irrelevant Information

The **forget gate** decides which information from the previous cell state should be discarded.

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

- Output values range between **0 and 1**
- 0 → forget completely
- 1 → retain completely

σ= sigmoidal activation function

What is W_f?

- W_f is the **weight matrix of the forget gate**
- It contains **trainable parameters**
- It learns **how strongly each element of the input and previous hidden state influences forgetting**

What is b_f?

- b_f is the **bias vector of the forget gate**
- Also **trainable**
- Helps shift the activation so the gate can learn better behavior

Step 2: Input Gate – Select New Information to Store

The input gate determines **what new information** should be added to memory.

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

A candidate memory is created using a tanh layer:

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

Step 3: Update Cell State – Maintain Long-Term Memory

④ Output gate - generate output

→ Produces next hidden state

→ else output layer output

$$O_t = \sigma\left[W_o\left[h_{t-1}, x_t\right] + b_o\right]$$

$$h_t = O_t * \tanh(C_t)$$

The cell state is updated by combining old memory and new information:

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

This step enables the LSTM to **preserve important information over long sequences.**

◆ **Step 4: Output Gate – Generate Output**

The output gate decides which part of the cell state should be exposed as the hidden state.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

The hidden state is passed to:

- The next LSTM time step
- The output layer (if required)

Information Flow Summary

Input (x_t)

↓

Forget gate → removes unnecessary memory

↓

Input gate → adds new relevant information

↓

Cell state (long-term memory)

↓

Output gate → produces hidden state (h_t)

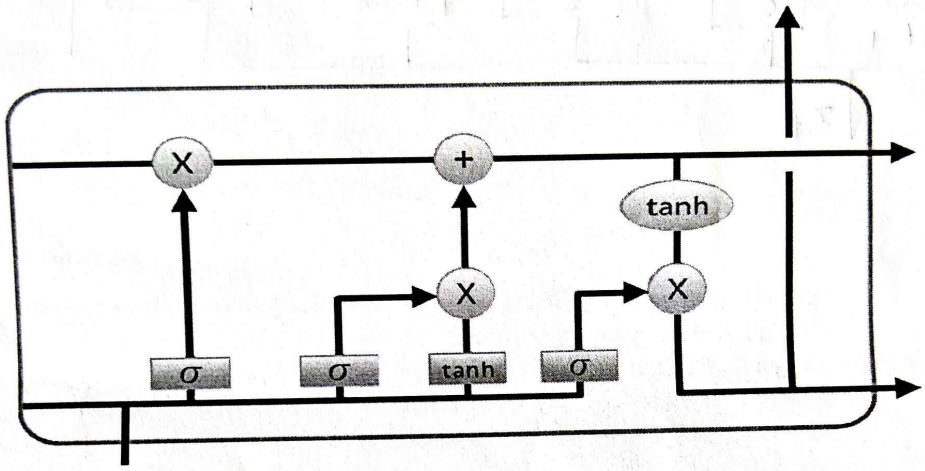
Why This Working Principle Is Effective

- Prevents **vanishing gradient problem**
- Allows **selective remembering and forgetting**
- Maintains stable memory across long sequences
- Learns long-term dependencies efficiently

Architecture of LSTM

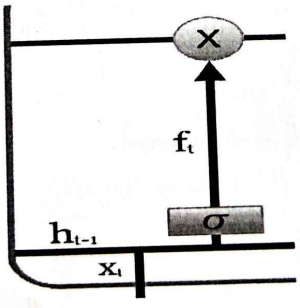
Structure of LSTM

The basic difference between the architectures of RNNs and LSTMs is that the hidden layer of LSTM is a gated unit or gated cell. It consists of four layers that interact with one another in a way to produce the output of that cell along with the cell state. These two things are then passed onto the next hidden layer. Unlike RNNs which have got only a single neural net layer of tanh, LSTMs comprise three logistic sigmoid gates and one tanh layer. Gates have been introduced in order to limit the information that is passed through the cell. They determine which part of the information will be needed by the next cell and which part is to be discarded. The output is usually in the range of 0-1 where '0' means 'reject all' and '1' means 'include all'.



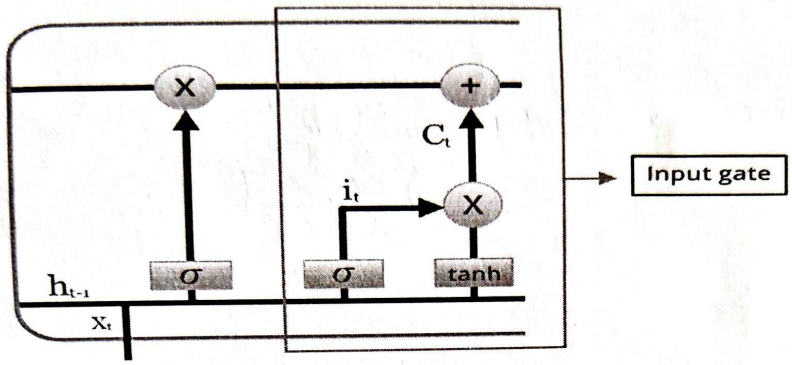
Forget Gate

The information that is no longer useful in the cell state is removed with the forget gate. Two inputs x_t (input at the particular time) and h_t-1 (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state, the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.



Input gate

The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs h_t-1 and x_t. Then, a vector is created using the tanh function that gives an output from -1 to +1, which contains all the possible values from h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to obtain useful information.



Output gate

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying the tanh function on the cell. Then, the information is regulated using the sigmoid function and filtered by the values to be remembered using inputs h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.