LSTM and GRU

EE6350: Artificial Intelligence

Mr. M.W.G.C.K Moremada
Lecturer, Department of Electrical and Information Engineering, Faculty of Engineering, University of Ruhuna

Lecture Outline

1. Problems with RNNs
2. LSTM
    a. The Core Idea
    b. Step-by-Step LSTM Walk Through
    c. PyTorch Implementation
3. Gated Recurrent Unit (GRU)
    a. PyTorch Implementation

Problems with RNNs

Ref: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Text generate image with Bing Image creator powered by DALLE-3

Problems with RNNs

- RNNs brings the previous information to present task. However, RNNs fail to learn long term dependencies.
- E.g., “I grew up in France … **<a lot of text>** … **I speak fluent <u>French</u>**”(Here is the underlined word is the one we are going to predict).
- Recent text suggests the next work should be a name of a **language.**
- **But which language?**

Solution: LSTM - **Long Short Term Memory** Networks

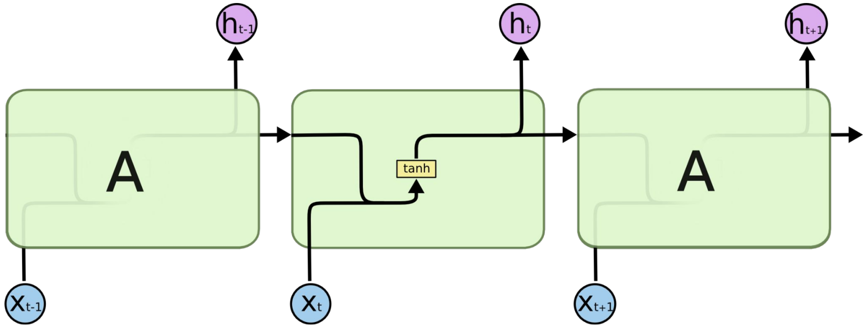
LSTM

Ref: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

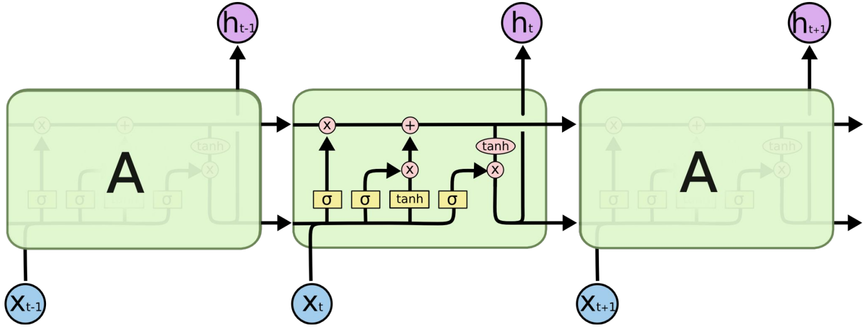
Text generate image with Bing Image creator powered by DALLE-3

LSTM

- LSTMs got the capability to learn long-term dependencies.
- It has been introduced by **Hochreiter & Schmidhuber (1997).**



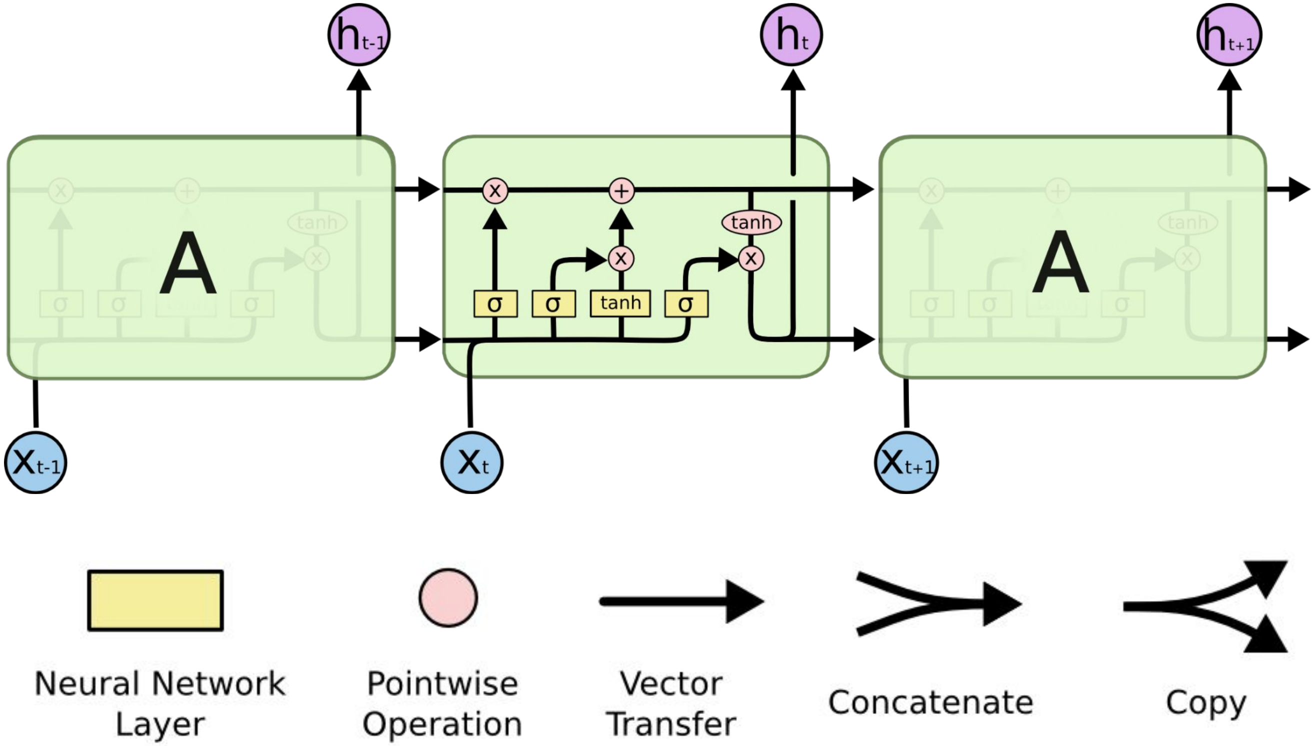
Repeating Module in Standard RNN



Repeating Module in LSTM

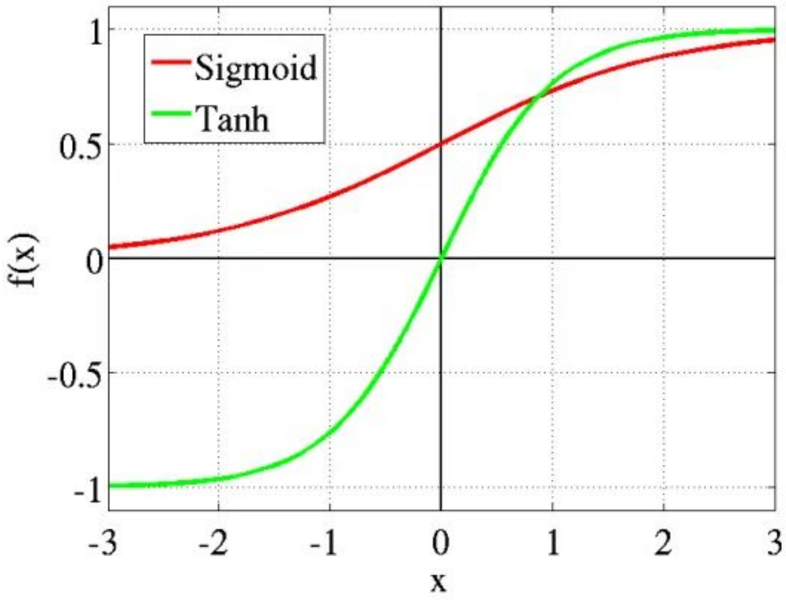
LSTM

4-Layers in LSTM instead of one in standard RNN.



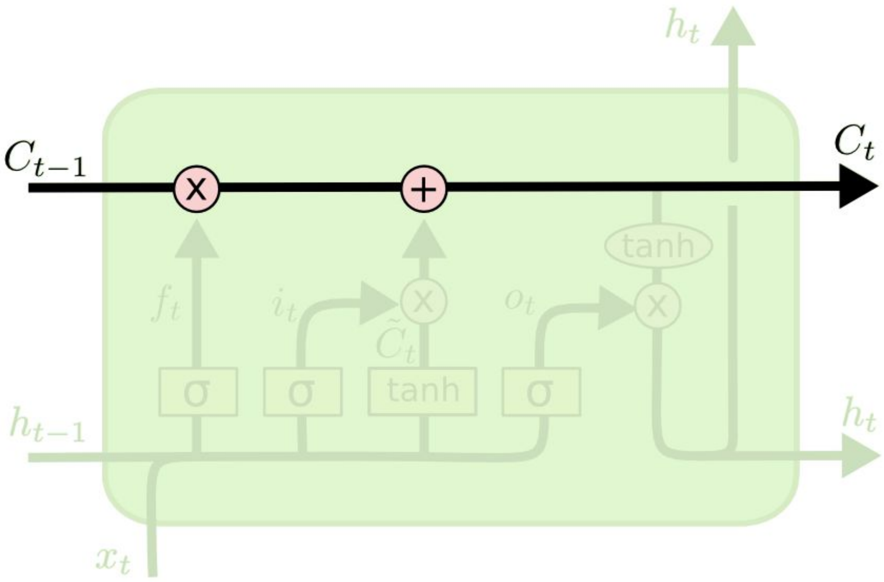
Recap

Sigmoid Vs Tanh Activation Functions



LSTM: The Core Idea

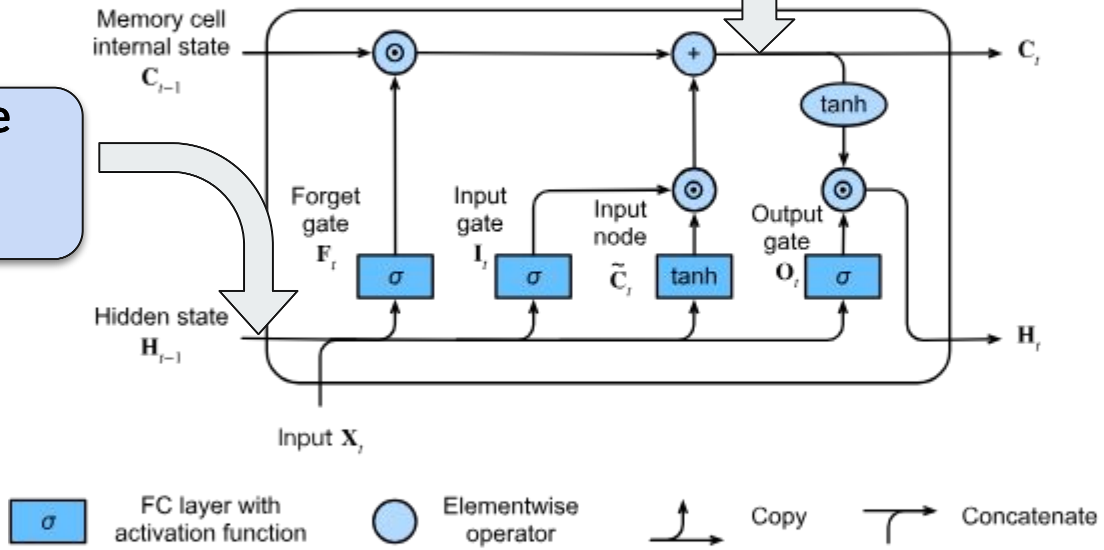
- Key in LSTM: **Cell State**
- Runs straight down the entire chain with only some **minor linear interactions.**
- In LSTM there are 3 gates to **protect and control the cell state.**



LSTM: The Core Idea

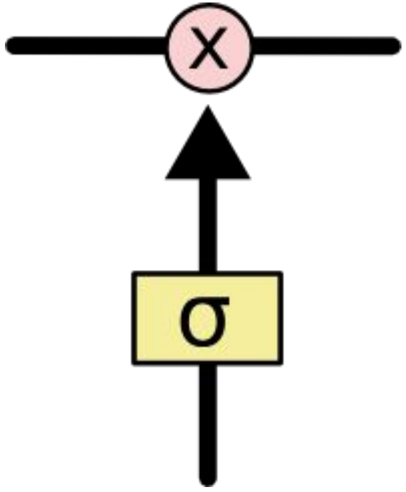
Represents the long-term memory

Represents the short-term memory



LSTM: The Core Idea

- LSTM remove or add information to the cell state through the structures called GATES.
- Gates optionally let information through.
- Consist of sigmoid neural network layer and pointwise multiplication.



LSTM: Step-by-Step LSTM Walk Through

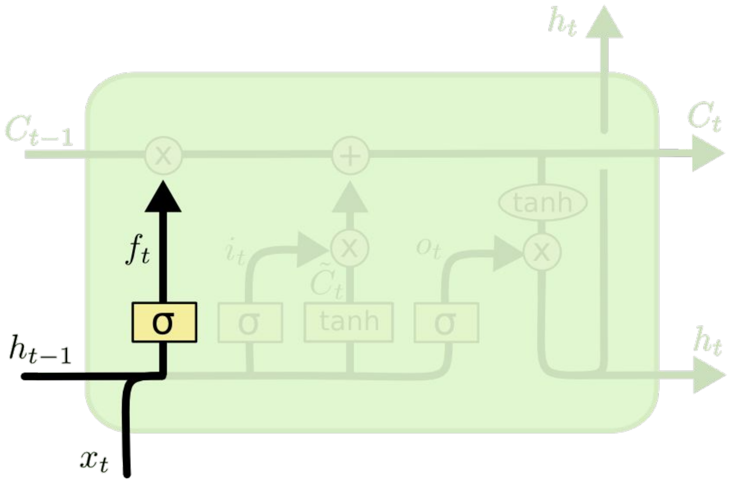
1

Step: 01

Purpose: Decide what information to throw away from the sell state.

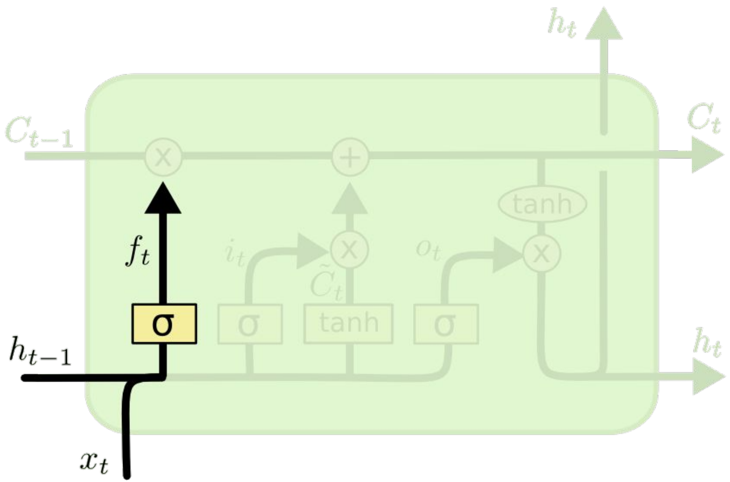
Done using "**Forget Gate Layer**".

Identify what percentage of the long-term memory will be remembered...



LSTM: Step-by-Step LSTM Walk Through

1



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

1

LSTM: Step-by-Step LSTM Walk Through

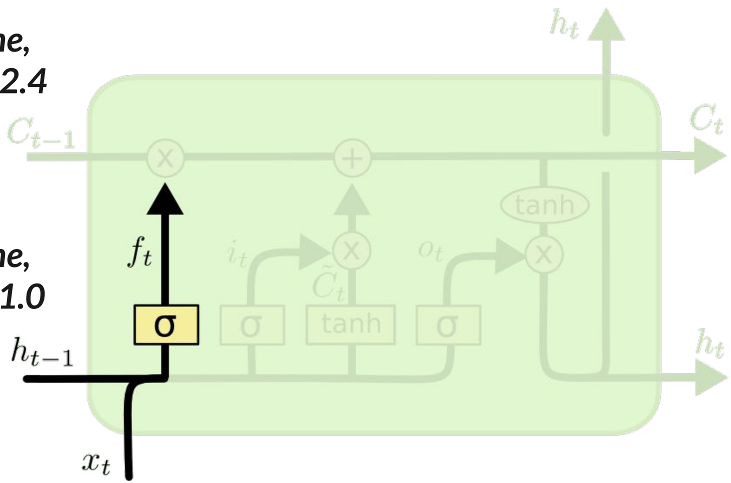
f_t

0.98

0.02

Assume,
$C_{t-1} = 2.4$

Assume,
$h_{t-1} = 1.0$



$x_t = 1.2, 1.5, 2.7, 5.3...$

Values for illustration purposes only. Not associated with any actual example/calculation. Furthermore, under the actual scenarios computation are happening as matrix multiplications.

1

LSTM: Step-by-Step LSTM Walk Through

Example: Predict next word based on previous information.

- Cell state might include the gender of the present subject. So that the correct pronoun can be used.
- Once found a new subject, we want to **forget the gender of the old subject.**

E.g.,
Mr. Sam is a teacher and **he** has a daughter. **She** is 6 years old and **her**…

2

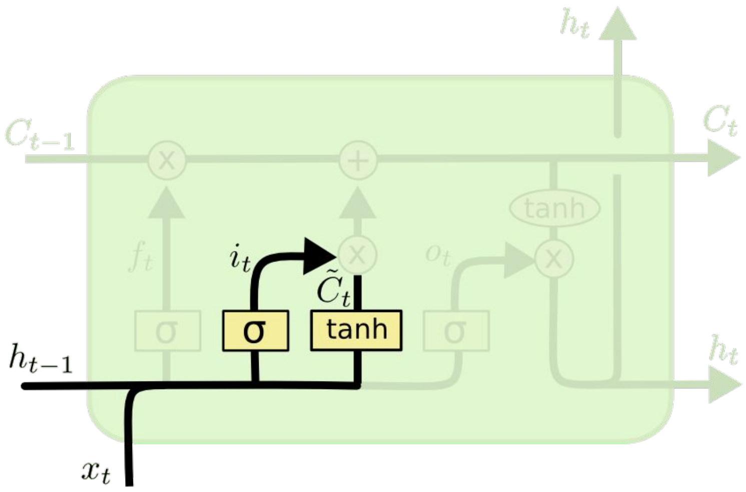
LSTM: Step-by-Step LSTM Walk Through

Step: 02

Purpose: Decide what new information we are going to store in the cell state.

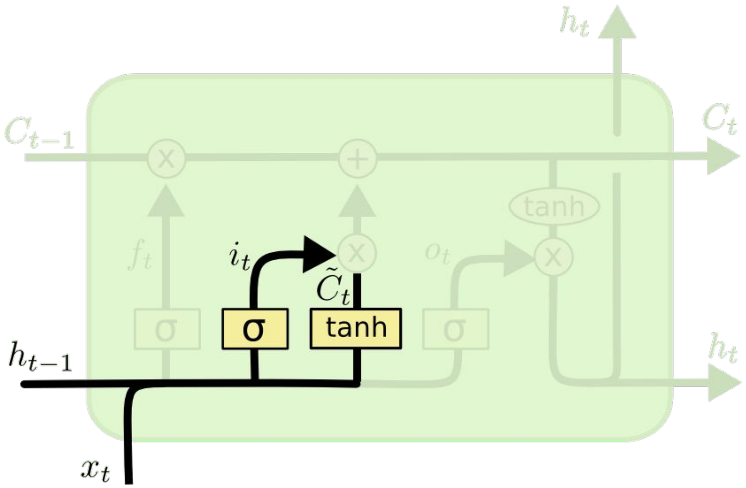
Done using two parts:

- **Input Gate Layer (Sigmoid)**: Decides which values we'll update.
- Next the **tanh layer** creates a vector of new candidate values that could be added to the state.



2

LSTM: Step-by-Step LSTM Walk Through



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

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

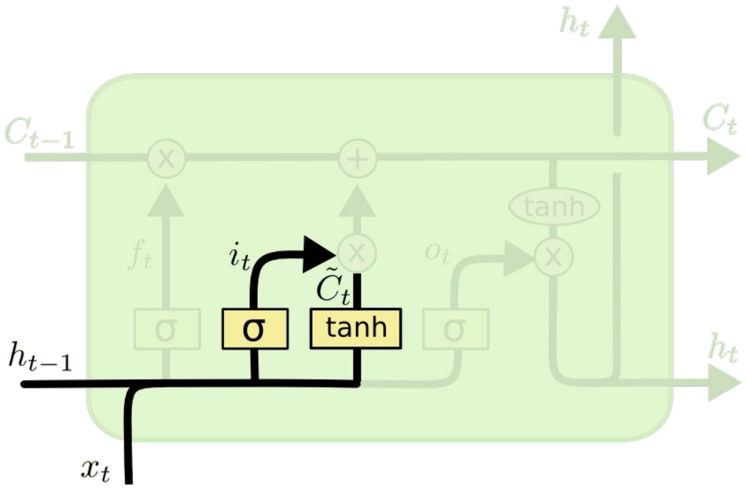
LSTM: Step-by-Step LSTM Walk Through

2

i_t

0.98

0.02



$\tilde{C}_t$

0.85

Values for illustration purposes only. Not associated with any actual example/calculation. Furthermore, under the actual scenarios computation are happening as matrix multiplications.

LSTM: Step-by-Step LSTM Walk Through

2

Example: Predict next word based on previous information.

- Cell state might include the gender of the present subject. So that the correct pronoun can be used.
- Once found a new subject, we want to forget the gender of the old subject.
- **Add the gender of the new subject to the cell state to replace the old one we are forgetting.**

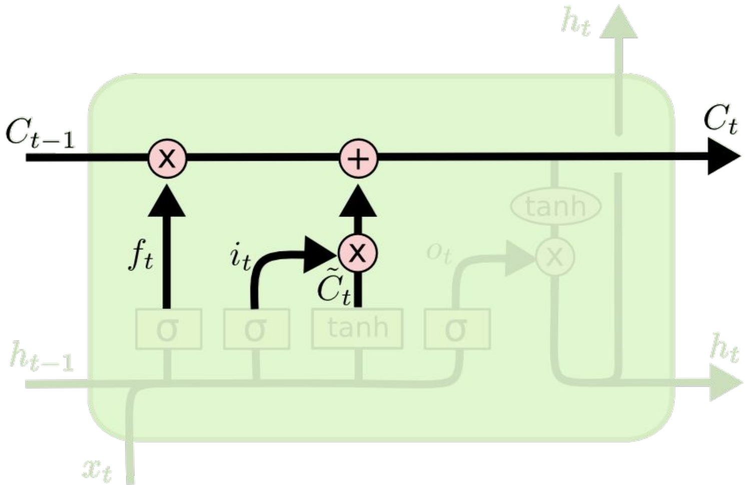
E.g.,
Mr. Sam is a teacher and **he** has a daughter. **She** is 6 years old and **her**…

LSTM: Step-by-Step LSTM Walk Through

Step: 03

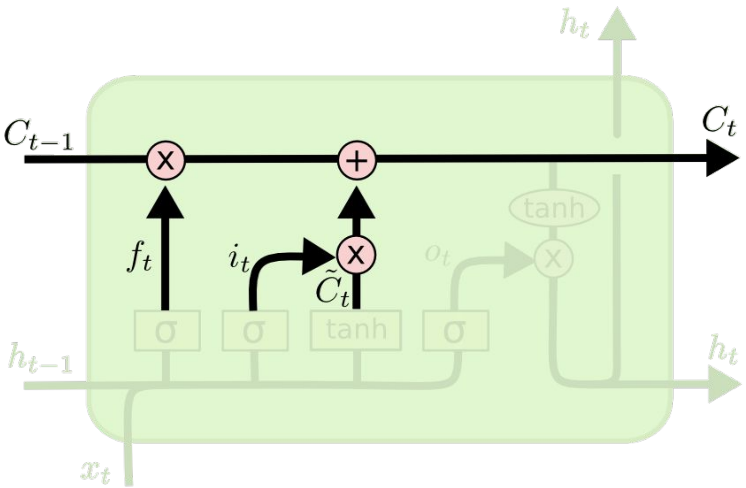
Purpose: The old cell state c_{t-1} will be updated into the new cell state c_t.

- In the previous two stages we have decided what to forget and what to be added to the cell state.



3

LSTM: Step-by-Step LSTM Walk Through



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

LSTM: Step-by-Step LSTM Walk Through

3

f_t

0.98

0.02

i_t

0.98

0.02

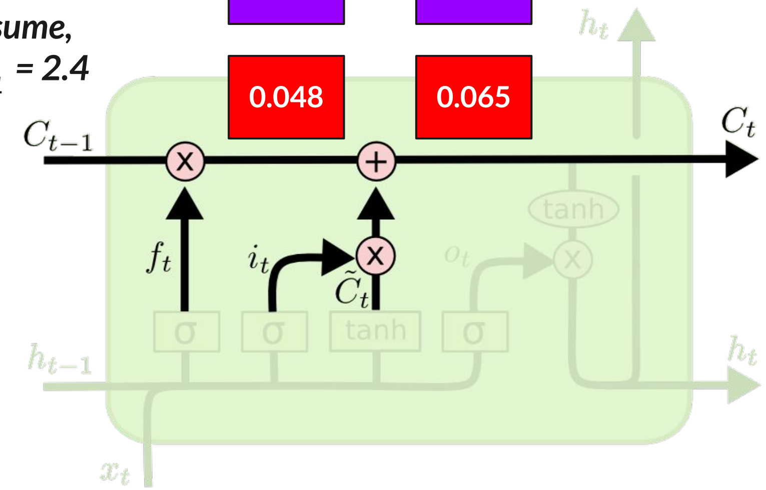
Assume,
C_{t-1} = 2.4

2.352

3.185

0.048

0.065



$\tilde{C}_t$

0.85

Values for illustration purposes only. Not associated with any actual example/calculation. Furthermore, under the actual scenarios computation are happening as matrix multiplications.

3

LSTM: Step-by-Step LSTM Walk Through

Example: Predict next word based on previous information.

- Cell state might include the gender of the present subject. So that the correct pronoun can be used.
- Once found a new subject, we want to forget the gender of the old subject.
- Add the gender of the new subject to the cell state to replace the old one we are forgetting.
- **This is actually where we drop the information about the old subject's gender and add the new information.**

E.g.,
Mr. Sam is a teacher and **he** has a daughter. **She** is 6 years old and **her**…

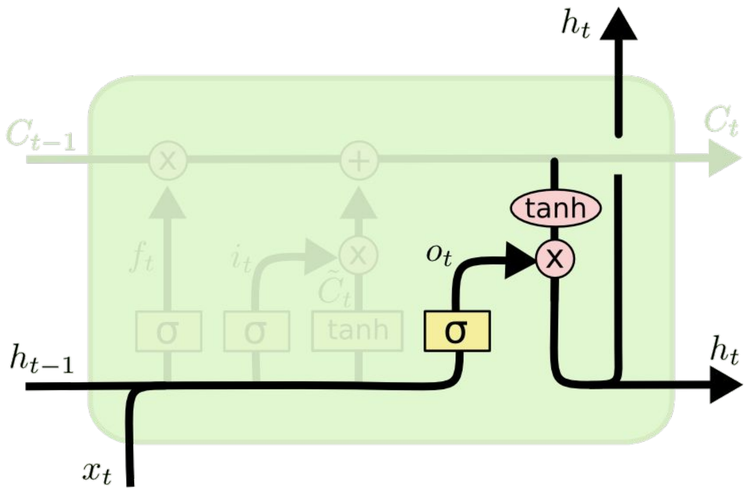
LSTM: Step-by-Step LSTM Walk Through

4

Step: 03

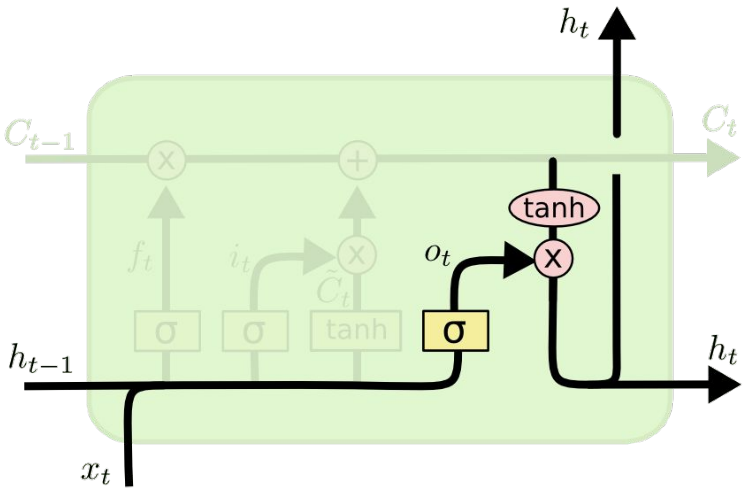
Purpose: Decide what to output.

- Output based on the the cell state (But will be a filtered version).
- This is also known as the **Output Gate.**



4

LSTM: Step-by-Step LSTM Walk Through



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

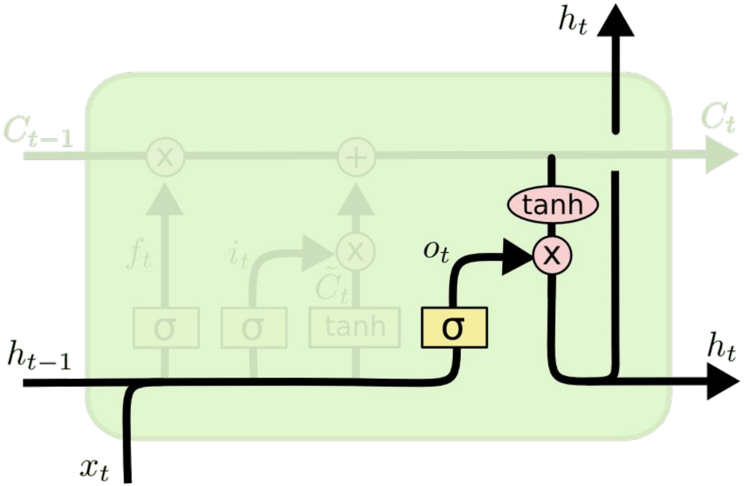
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM: Step-by-Step LSTM Walk Through

4

o_t

0.92



Values for illustration purposes only. Not associated with any actual example/calculation. Furthermore, under the actual scenarios computation are happening as matrix multiplications.

$h_t = 0.92 \times \tanh(3.185) = 0.91$

3

LSTM: Step-by-Step LSTM Walk Through

Example: Predict next word based on previous information.

- Cell state might include the gender of the present subject. So that the correct pronoun can be used.
- Once found a new subject, we want to forget the gender of the old subject.
- Add the gender of the new subject to the cell state to replace the old one we are forgetting.
- This is actually where we drop the information about the old subject's gender and add the new information.
- **Output whether subject is singular or plural so that we know what form a verb should be conjugated into if that's what follows next.**

E.g.,
Mr. Sam is a teacher and **he** has a daughter. **She** is 6 years old and **her**…

LSTM: PyTorch Implementation

CLASS torch.nn.LSTM(*self*, *input_size*, *hidden_size*, *num_layers=1*, *bias=True*, *batch_first=False*, *dropout=0.0*, *bidirectional=False*, *proj_size=0*, *device=None*, *dtype=None*) [SOURCE]

Parameters

- **input_size** – The number of expected features in the input *x*
- **hidden_size** – The number of features in the hidden state *h*
- **num_layers** – Number of recurrent layers. E.g., setting `num_layers=2` would mean stacking two LSTMs together to form a *stacked LSTM*, with the second LSTM taking in outputs of the first LSTM and computing the final results. Default: 1
- **bias** – If `False`, then the layer does not use bias weights *b_ih* and *b_hh*. Default: `True`
- **batch_first** – If `True`, then the input and output tensors are provided as (*batch, seq, feature*) instead of (*seq, batch, feature*). Note that this does not apply to hidden or cell states. See the Inputs/Outputs sections below for details. Default: `False`
- **dropout** – If non-zero, introduces a *Dropout* layer on the outputs of each LSTM layer except the last layer, with dropout probability equal to `dropout`. Default: 0
- **bidirectional** – If `True`, becomes a bidirectional LSTM. Default: `False`
- **proj_size** – If `> 0`, will use LSTM with projections of corresponding size. Default: 0

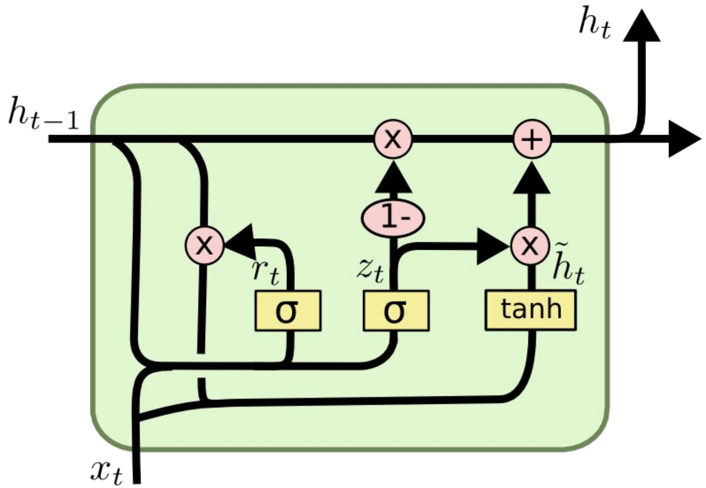
Gated Recurrent Unit (GRU)

Ref: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Text generate image with Bing Image creator powered by DALLE-3

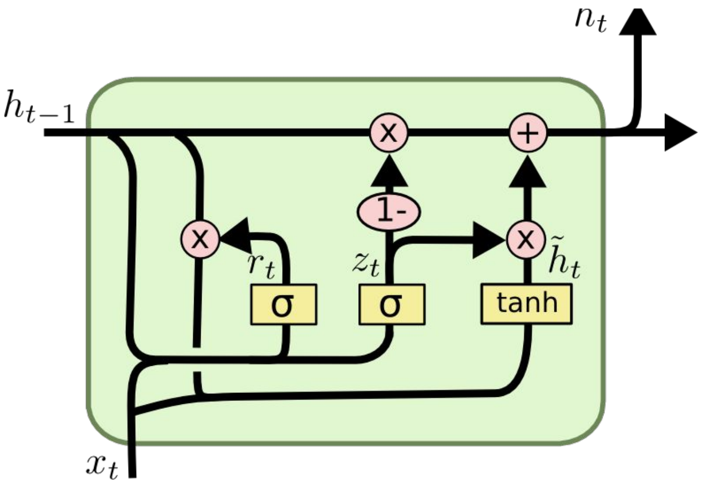
GRU

- Combines the forget and input gates to single update gate.
- Also merges the cell state and the hidden state.
- Ultimately the GRU model has become much simpler than the LSTM.



GRU

Try to investigate how these functions have been derived and overall working of GRU...



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU: PyTorch Implementation

CLASS torch.nn.GRU(*self, input_size, hidden_size, num_layers=1, bias=True, batch_first=False, dropout=0.0, bidirectional=False, device=None, dtype=None*) [SOURCE]

Parameters

- **input_size** – The number of expected features in the input *x*
- **hidden_size** – The number of features in the hidden state *h*
- **num_layers** – Number of recurrent layers. E.g., setting `num_layers=2` would mean stacking two GRUs together to form a *stacked GRU*, with the second GRU taking in outputs of the first GRU and computing the final results. Default: 1
- **bias** – If `False`, then the layer does not use bias weights *b_ih* and *b_hh*. Default: `True`
- **batch_first** – If `True`, then the input and output tensors are provided as (*batch, seq, feature*) instead of (*seq, batch, feature*). Note that this does not apply to hidden or cell states. See the Inputs/Outputs sections below for details. Default: `False`
- **dropout** – If non-zero, introduces a *Dropout* layer on the outputs of each GRU layer except the last layer, with dropout probability equal to `dropout`. Default: 0
- **bidirectional** – If `True`, becomes a bidirectional GRU. Default: `False`

LSTM vs GRU

Investigate the differences between the LSTM and GRU architectures and their advantages and disadvantages…

Recommended Reading: Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

References

1. C. Olah, “Understanding LSTM Networks,” Github.io, Aug. 27, 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
2. S. SHARMA, “Activation Functions in Neural Networks,” Medium, Sep. 06, 2017. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
3. “Long Short-Term Memory (LSTM), Clearly Explained,” www.youtube.com. https://youtu.be/YCzL96nL7j0 (accessed Apr. 05, 2024).
4. “9.2. Long Short Term Memory (LSTM) — Dive into Deep Learning 0.14.4 documentation,” d2l.ai. https://d2l.ai/chapter_recurrent-modern/lstm.html.

Thank You