CS60010: Deep Learning
Recurrent Neural Network

Sudeshna Sarkar

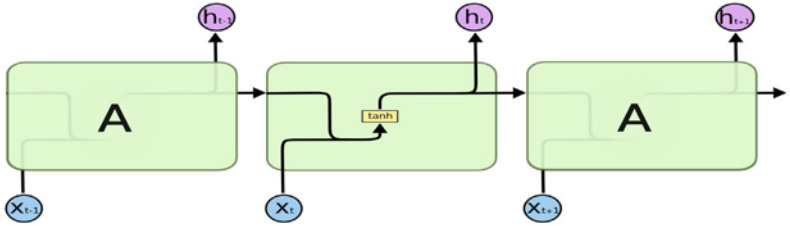
Spring 2018

12 Feb 2018

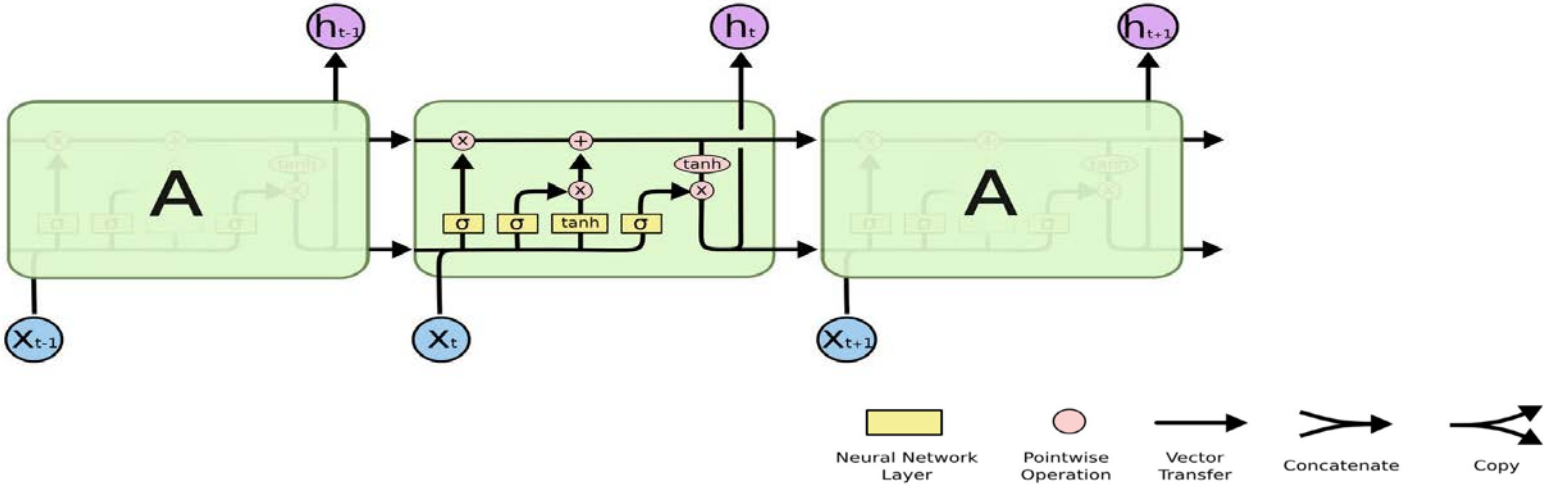
Long Short Term Memory

Hochreiter & Schmidhuber (1997)

Repeating module in Standard RNN

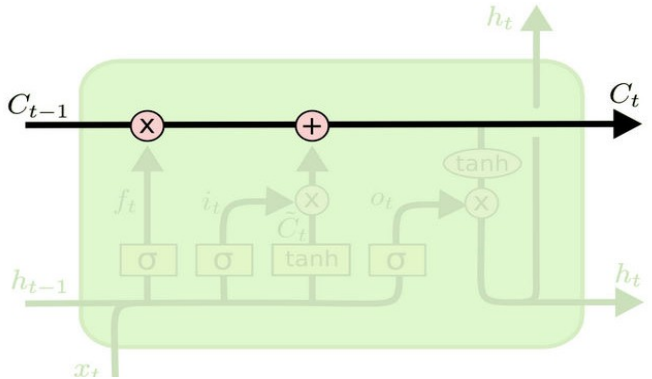


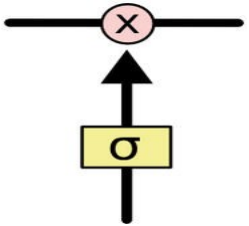
Repeating module in an LSTM



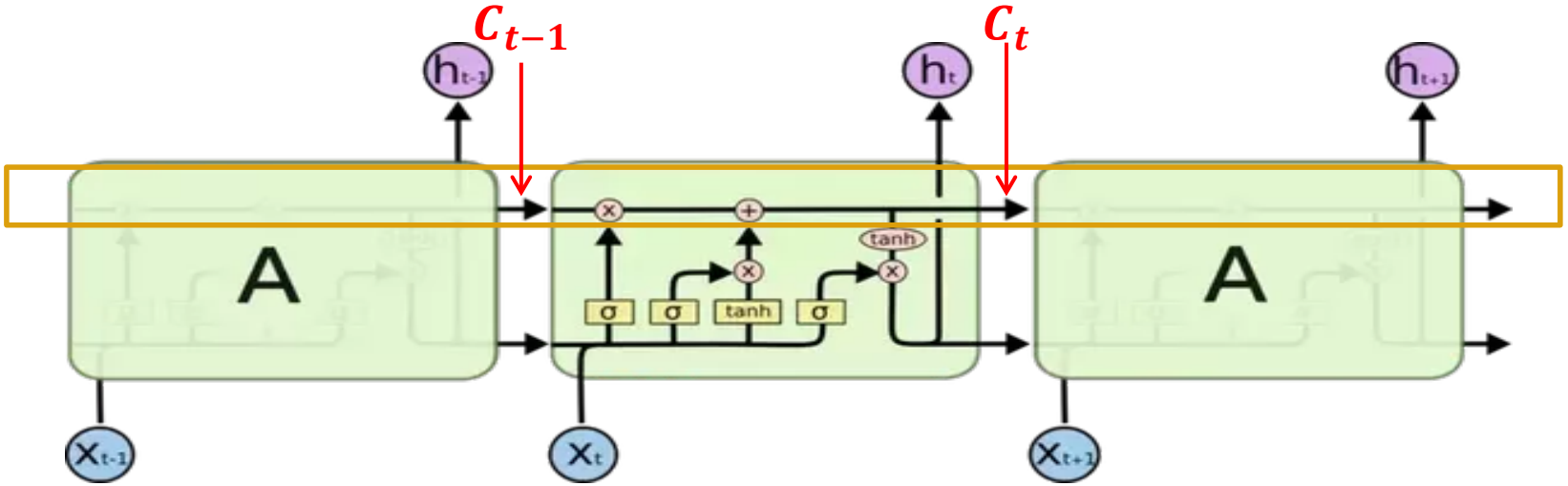
Core idea behind LSTM

- The cell state, C_t , is like a conveyor belt. Runs through entire chain with minor linear interactions.
  It's very easy for information to just flow along it unchanged.
  - LSTM has the ability to remove/add information to cell state regulated by gates.
  - Gates are composed out of a sigmoid neural net layer and a pointwise multiplication operation.





LSTM: Constant Error Carousel

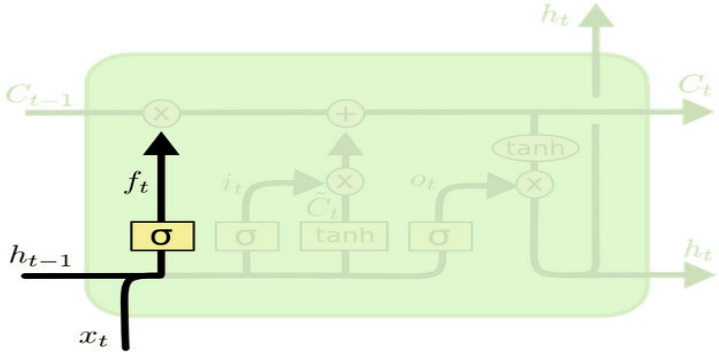


Key Component: A remembered cell state.

C_t is the linear history carried by the constant error carousel.

LSTM: Forget Gate

- **Step 1: Forget Gate** decide what information we're going to throw away from the cell state.
- It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1} for whether to forget.

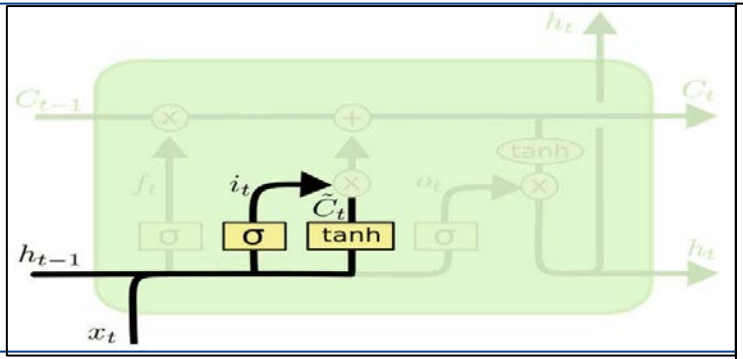


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- Language model: predict next word based on previous ones
  - Cell state may include the gender of the present subject so that the proper pronouns can be used
- When we see a new subject we want to forget old subject

LSTM : Input Gate

Decide what new information we're going to store in the cell state



Two parts:

a) A sigmoid layer called *Input gate layer*: decides which values we will update

b) Next a tanh layer creates a vector of new candidate values $\tilde{C}_t$ that could be added to the state.

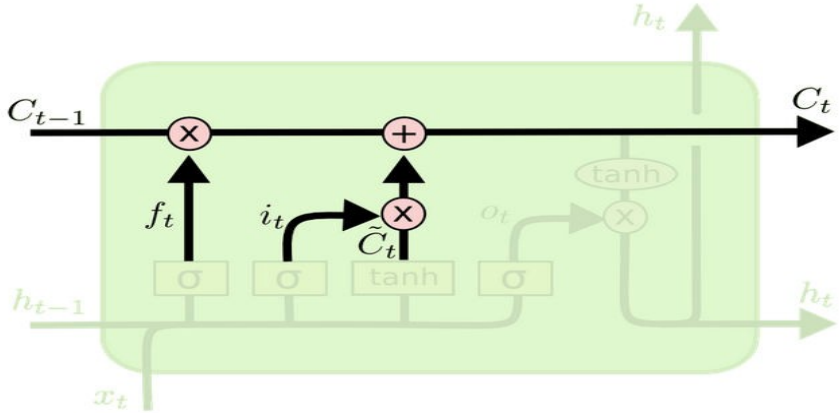
In Step 3 we combine these two to create an update to the state

In the Language model, we'd want to add the gender of the new subject to the cell state, to replace the old one we are forgetting

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

LSTM walk through: Step 3

- Update old cell state C_{t-1} into new cell state C_t.

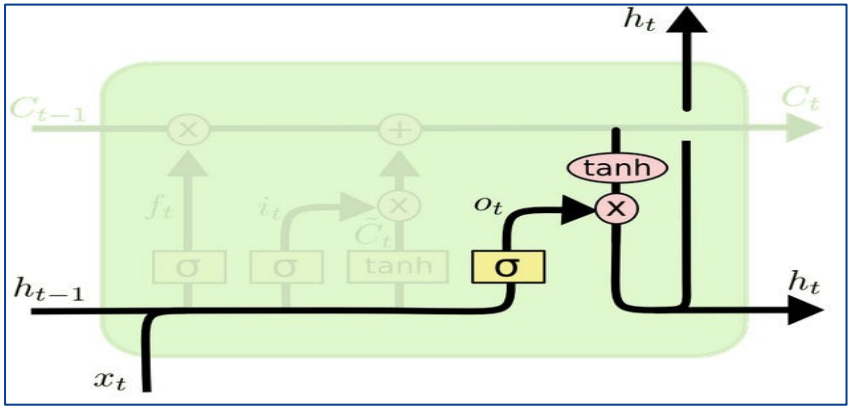


- We multiply the old state by f_t, forgetting the things we decided to forget earlier.
- Then we add $i_t * \widetilde{C}_t$.

$$C_t = f_t * C_t + i_t * \widetilde{C}_t$$

In the Language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in previous steps

LSTM: Output and Output gate



This output will be based on our cell state, but will be a filtered version. First we run a sigmoid layer which decides what parts of cell state we're going to output. Then we put the cell state through tanh (to push values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we output only the parts we decided to

For the Language model, Since it just saw a subject it might want to output information relevant to a verb

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

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

LSTM Equations

$\boldsymbol{i}$: input gate, how much of the new information will be let through the memory cell.

f: forget gate, responsible for which information should be thrown away from memory cell.

$\boldsymbol{o}$: output gate, how much of the information will be passed to expose to the next time step.

g: self-recurrent which is equal to standard RNN

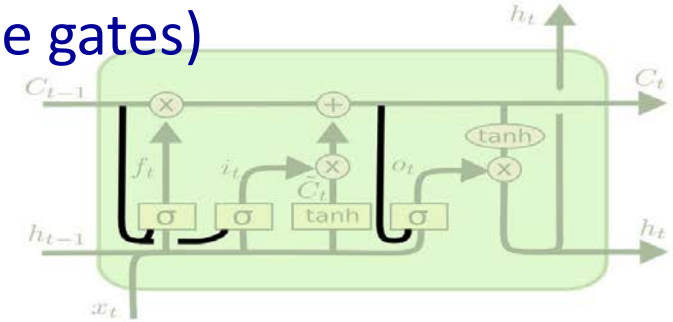
C_t: internal memory of the memory cell

s_t: hidden state

$\mathbf{y}$: final output

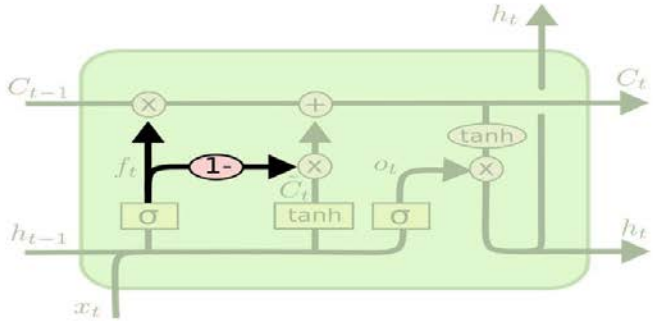
Variants of LSTM

LSTM with “peephole” connections (let the cell directly influence the gates)



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] + b_o\right)$$

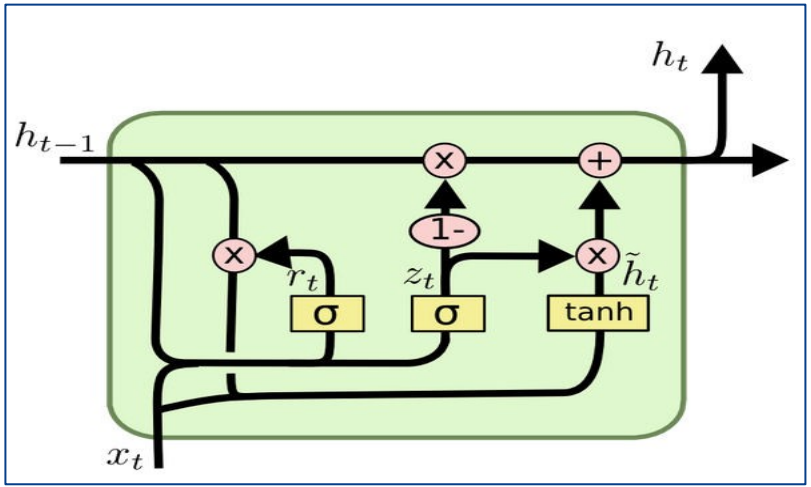
Coupled forget and input gates



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

Gated Recurrent Unit (GRU)

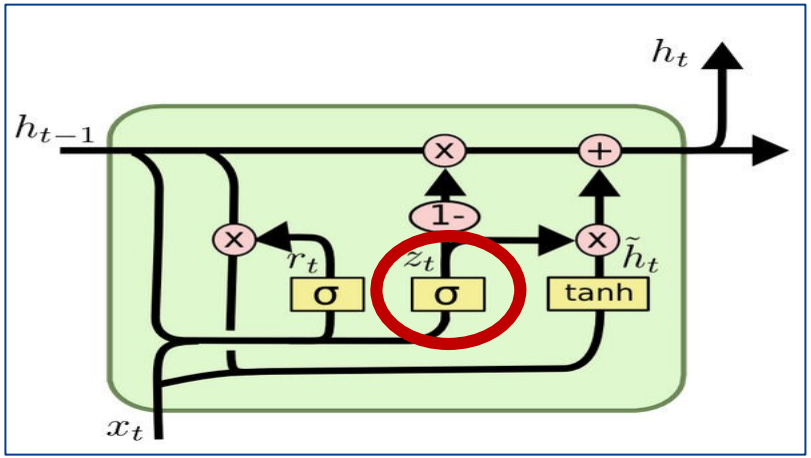
- A dramatic variant of LSTM
  - It combines the forget and input gates into a single update gate
  - It also merges the cell state and hidden state, and makes some other changes
  - The resulting model is simpler than LSTM models
  - Has become increasingly popular



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

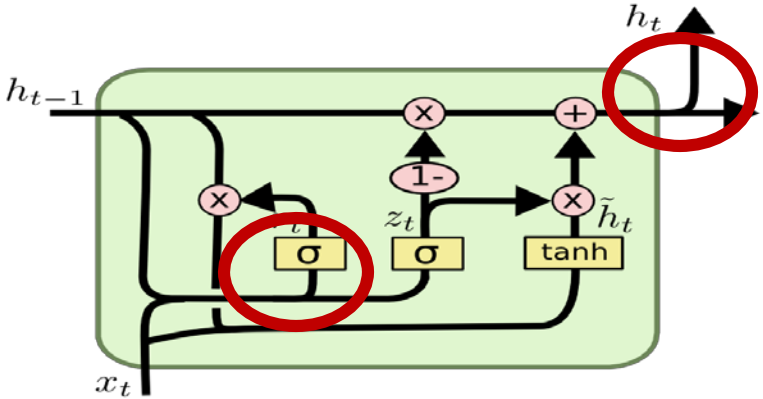
Gated Recurrent Unit (GRU)

- Combine forget and input gates
  - In new input is to be remembered, then this means old memory is to be forgotten
    - Why compute twice?



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

Gated Recurrent Units



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

- Don’t bother t o separately maintain compressed and regular memories
- But compress it before using it to decide on the usefulness of the current input!

Gated RNN

- Gated RNNs are based on the idea of creating paths through time that have derivatives that neither vanish nor explode.

- Accumulating Information over Longer Duration
- Gated RNN:
- Instead of manually deciding when to clear the state, we want the neural network to learn to decide when to do it

LSTM

- Contribution of LSTMs
  - introducing self-loops to produce paths where the gradient can flow for long durations
  - make weight on this self-loop conditioned on the context, rather than fixed
    - By making weight of this self-loop gated (controlled by another hidden unit), time-scale can be changed dynamically
    - Even for an LSTM with fixed parameters, time scale of integration can change based on the input sequence
      - Because time constants are output by the model itself
- LSTM found extremely successful in:
  - Unconstrained handwriting recognition, Speech recognition
  - Handwriting generation, Machine Translation
  - Image Captioning, Parsing