Security and Fairness of Deep Learning

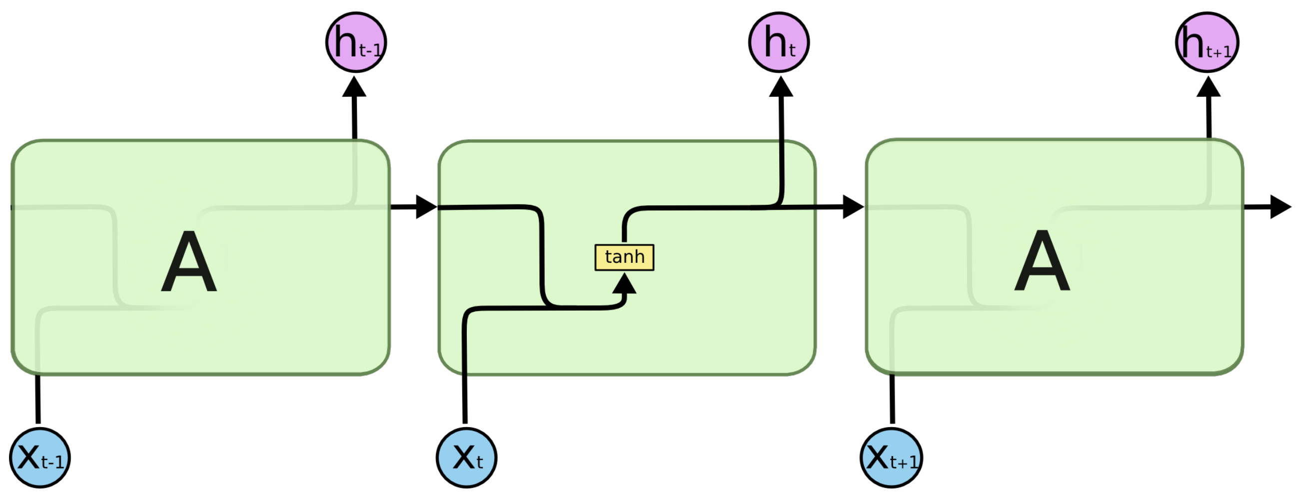
Long Short Term Memory (LSTM) networks

Anupam Datta

CMU

Spring 2019

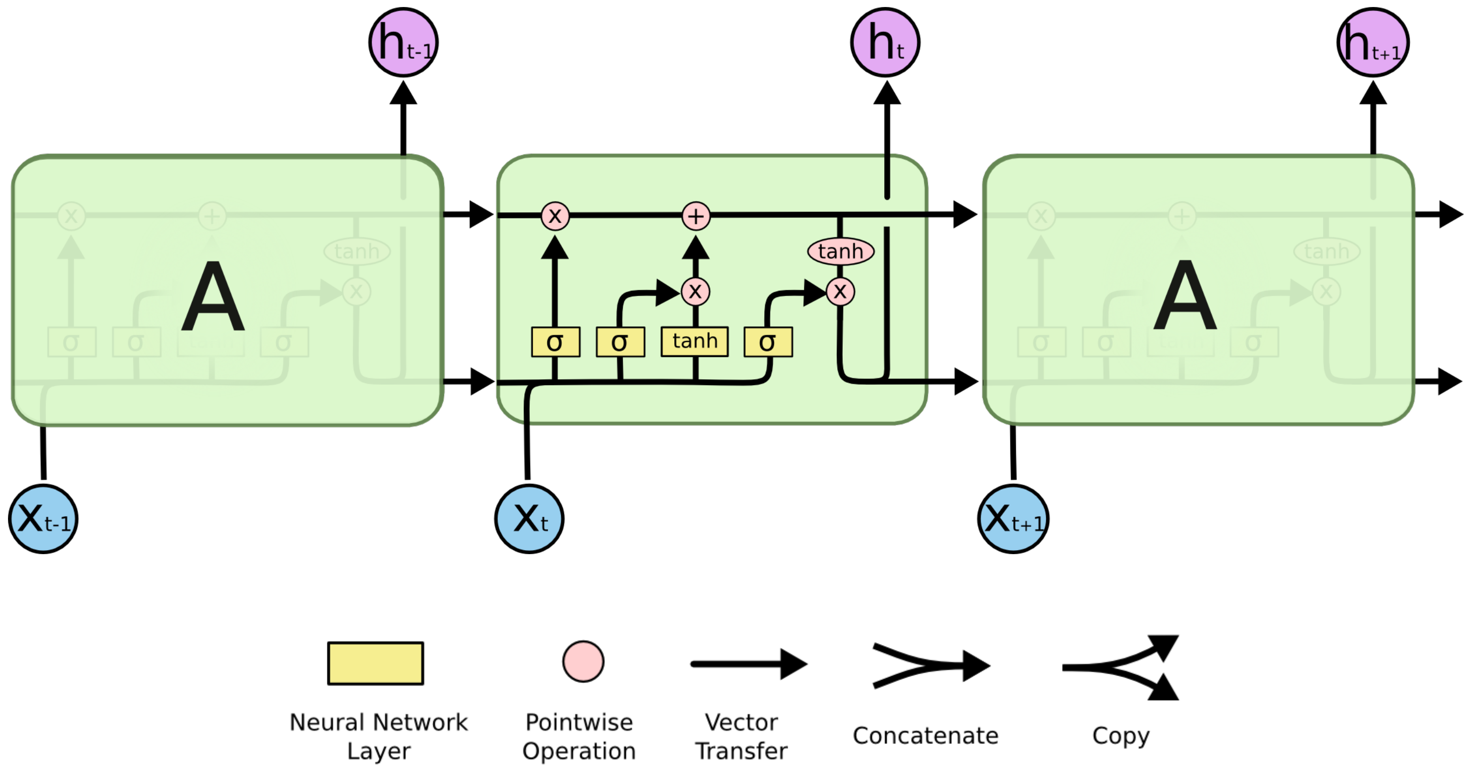
Vanilla RNN



Recall problem

- Vanishing or exploding gradients
  - difficult to learn long term dependencies

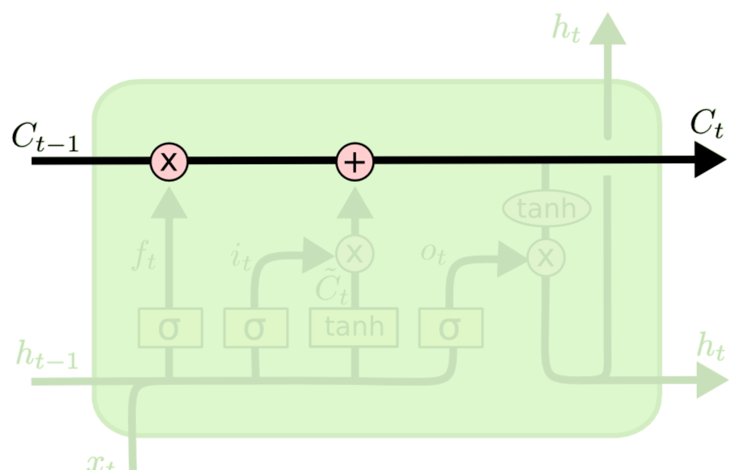
LSTM

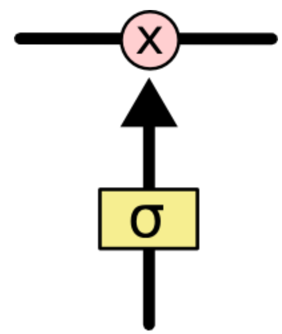


h t-1
h t
h t+1
A
σ
σ
tanh
σ
tanh
A
x t-1
x t
x t+1
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

Core idea behind LSTMs

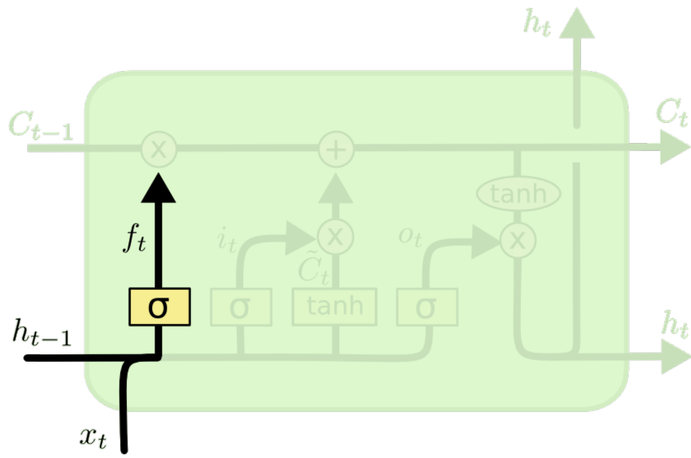
- Cell state + gates
  - Cell state stores long-term information
  - Gates remove and add information to the cell state





LSTM walk-through

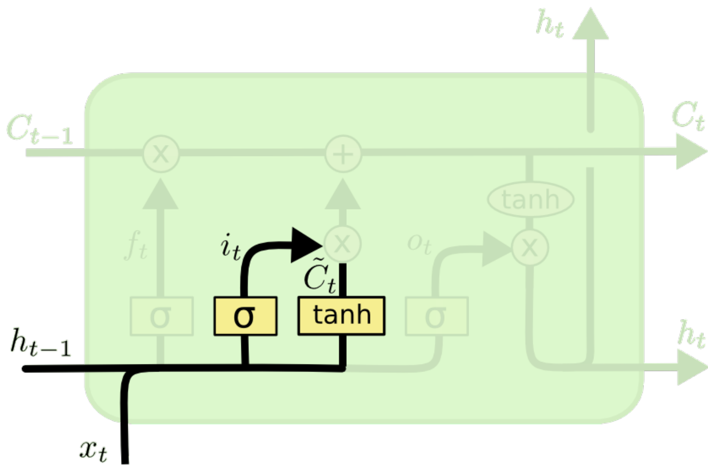
Forget information



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

- What information are we going to forget from the cell state?
- sigmoid output in [0,1]; if output 0, then forget completely
- Language model example: Forget gender of old subject when model sees new subject

Create new information

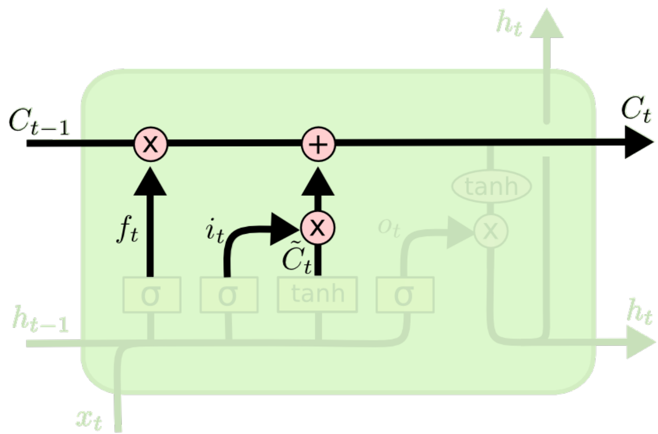


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

- input gate layer decides which values we'll update
- tanh layer creates a vector of new candidate values that could be added to the state
- Language model example: the gender of the new subject

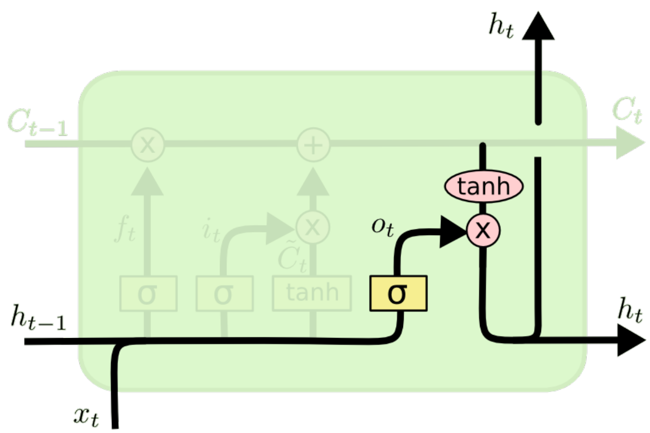
Store new information



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

- Update state by forgetting some info and adding new info
- Language model example: drop the information about the old subject's gender and add information about new subject's gender

Output new hidden state



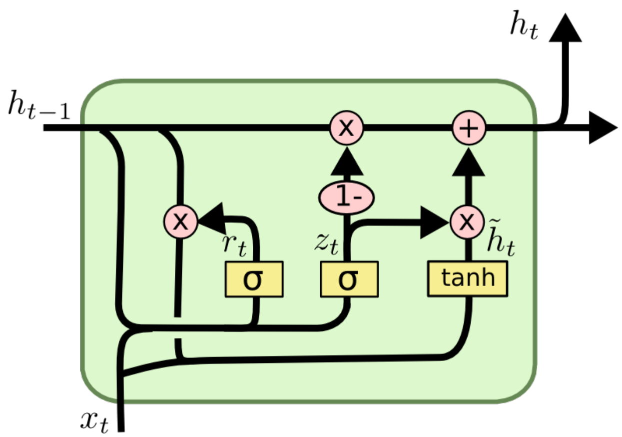
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

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

- Language model example: Since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.

LSTM variant: Gated Recurrent Unit (GRU)

GRU



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

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

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

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

- Combines the forget and input gates into a single “update gate”
- Merges the cell state and hidden state
- …

GRU intuition

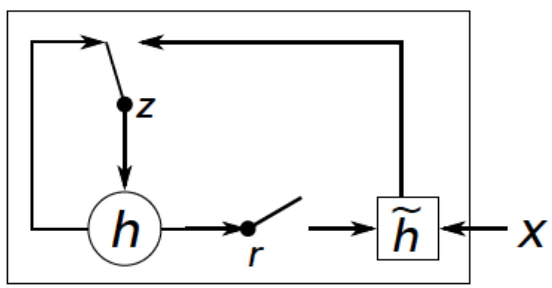
- If reset is close to 0, ignore previous hidden state → Allows model to drop information that is irrelevant in the future

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$
$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$
$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

- Update gate z controls how much of past state should matter now.
  - If z close to 1, then we can copy information in that unit through many time steps! **Less vanishing gradient**!
- Units with short-term dependencies often have reset gates very active

GRU intuition

- Units with long term dependencies have active update gates z
- Illustration:



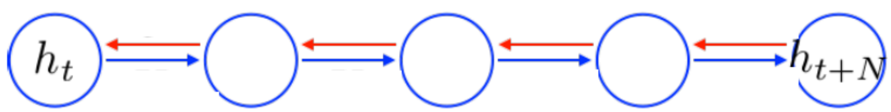
$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$
$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$
$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

How GRUs fix vanishing gradients problem

- Is the problem with standard RNNs the naïve transition function?

$$h_t = f\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right)$$

- It implies that the error must backpropagate through all the intermediate nodes:



- Perhaps we can create shortcut connections.