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
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

- 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

\[ f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \]
Create new information

- 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

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

- 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

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

- Combines the forget and input gates into a single “update gate”
- Merges the cell state and hidden state
- \( z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \)
- \( r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \)
- \( \tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t]) \)
- \( h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \)
GRU intuition

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

\[
\begin{align*}
  z_t &= \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right) \\
  r_t &= \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right) \\
  \tilde{h}_t &= \tanh \left( W x_t + r_t \circ U h_{t-1} \right) \\
  h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t
\end{align*}
\]

- 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:

\[
\begin{align*}
  z_t &= \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right) \\
  r_t &= \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right) \\
  \tilde{h}_t &= \tanh \left( W x_t + r_t \circ U h_{t-1} \right) \\
  h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t
\end{align*}
\]
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.