Understanding LSTM Networks

Chris Olah
5.14.17

Summary by Justin Chen

Summary:

- Purpose of RNNs is to related past information and present information
- RNNs suffer from two problems: exploding gradients, and vanishing gradients
    - Exploding gradient
        - Continuously multiply by values greater than 1, which causes the values to amplify as they are propagated back through the network making the computation increasingly expensive and difficult to store
    - Vanishing gradient
        - Continuously multiply by values between 0 and 1, which causes the values to diminish towards 0 disallowing the network to learn anything
- LSTM designed to handle vanishing gradient
    - Gates and decision layers allow the network to pick and choose what values to forget, remember, and emit
- LSTMs consist of four decision layers: forget, input, output, candidate, and three memory gates
    - Decision layers and memory gates are not the formal terms, but I use them here for clarity
- At each timestep
    1. Takes in input, $\mathbf{x}_{t,}$ hidden state, $\mathbf{h}_{t-1}$, cell state $\mathbf{C}_{t-1}$
    2. Emits a cell state, $\mathbf{C}_t$, and hidden state, $\mathbf{h}_t$
- Cell state
    a. Linear combination of memories to remember and forget.
        - How much of previous cell state to remember plus amount of candidate memory to store from current timestep
- Hidden state contains
    a. Applies tanh nonlinearity to current cell state and then takes linear combination with vector from output decision layer
    b. This hidden vector emitted as the output/ prediction for the current timestep, and a copy is also fed back into the network for the next timestep.
- Input vector contains
    a. Vector representing current timestep
- Computation:
    1. Forget gate
        a. Emit a value between [0, 1] for each memory from previous cell state
    2. Input gate
        a. Decide which values to update

3. Candidate layer
    a. Create a candidate vector to add to memory
    b. Uses tanh nonlinearity
4. Output gate
    a. Decides which memories to output from cell state

Questions:

1. Why use tanh for output gate? What advantageous properties does the range [-1, 1] have?
2. How deep and wide are the network gates?
    a. Gates are single layer MLPs
    b. Networks are all same width since they all accept the same input $\mathbf{x}_t$
    c. Widths are as wide as the longest input at each timestep
3. Sigmoid output of forget, input, and output gates are not rounded to 0 or 1, so partial information can persist

Other LSTM variants:

1. LSTM with peepholes
    a. Allows the gates to look at the cell state when deciding
        i. This means you concatenate the previous cell state $\mathbf{C}_{t-1}$ to the previous hidden state $\mathbf{h}_{t-1}$, and the current input, as one long vector, and then multiply that by the connection matrix for the gate and then add the bias. This is done for the <u>forget</u> $\mathbf{f}_t$, <u>input</u> $\mathbf{i}_t$, and <u>output gates</u> $\mathbf{o}_t$.
        e.g. $f_t = sig(W_f * [C_{t-1}, h_{t-1}, x_t] + b_f)$
2. Gated Recurrent Unit (GRU)
    a. Combines the forget and input gates into one decision/one network
    b. Also merges the cell state and hidden state
3. Depth Gated RNN
4. Clockwork RNNs
5. Greff, et al. find that all variants perform more or less the same
    a. //Expected since they’re supposed the achieve the same thing…