Recurrent Neural Networks 2

CS 287

(Based on Yoav Goldberg's notes)

Review: Representation of Sequence

- Many tasks in NLP involve sequences

$$w_1, \ldots, w_n$$

- Representations as matrix dense vectors $\mathbf{X}$
  (Following YG, slight abuse of notation)

$$\mathbf{x}_1 = \mathbf{x}_1^0 \mathbf{W}^0, \ldots, \mathbf{x}_n = \mathbf{x}_n^0 \mathbf{W}^0$$

- Would like fixed-dimensional representation.

Review: Sequence Recurrence

- Can map from dense sequence to dense representation.
- $\mathbf{x}_1, \dots, \mathbf{x}_n \mapsto \mathbf{s}_1, \dots, \mathbf{s}_n$
- For all $i \in \{1, \dots, n\}$

$$\mathbf{s}_i = R(\mathbf{s}_{i-1}, \mathbf{x}_i; \theta)$$

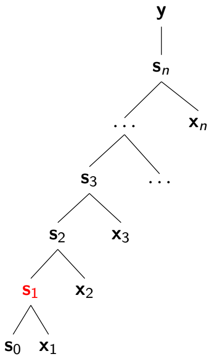
- θ is shared by all R

Example:

$$\begin{aligned}
\mathbf{s}_4 &= R(\mathbf{s}_3, \mathbf{x}_4) \\
&= R(R(\mathbf{s}_2, \mathbf{x}_3), \mathbf{x}_4) \\
&= R(R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)
\end{aligned}$$

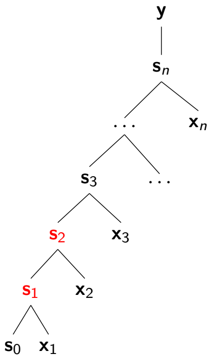
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



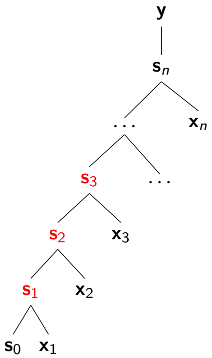
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



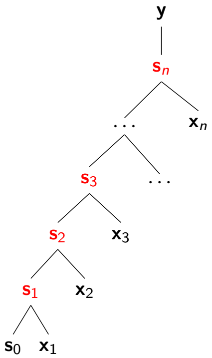
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



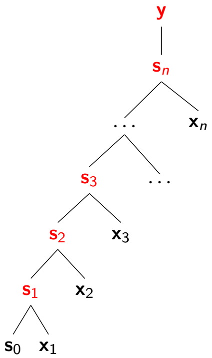
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



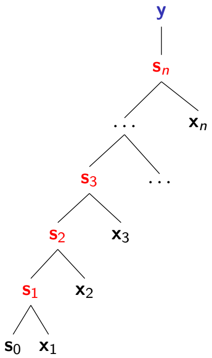
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



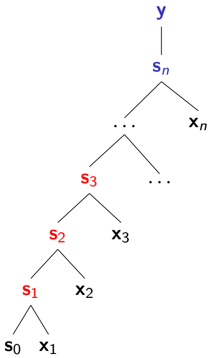
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



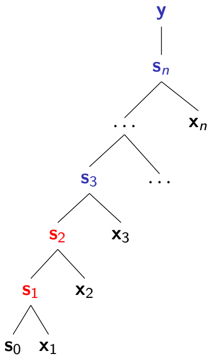
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



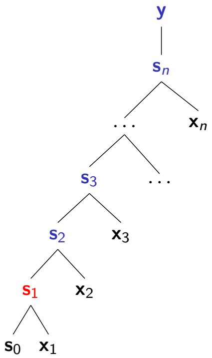
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



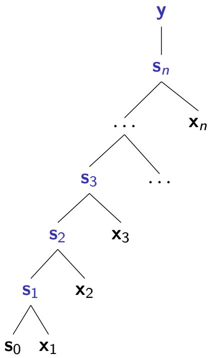
Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



Review: BPTT (Acceptor)

- Run forward propagation.
- Run backward propagation.
- Update all weights (shared)



Issues

- Can be inefficient, but batch/GPUs help.
- Model is much deeper than previous approaches.
  - This matters a lot, focus of next class.
- Variable-size model for each sentence.
  - Have to be a bit more clever in Torch.

Quiz

Consider a ReLU version of the Elman RNN with function R defined as

$$NN(\mathbf{x}, \mathbf{s}) = \text{ReLU}(\mathbf{s}\mathbf{W}^s + \mathbf{x}\mathbf{W}^x + \mathbf{b}).$$

We use this RNN with an acceptor architecture over the sequence $\mathbf{x}_1, \dots, \mathbf{x}_5$. Assume we have computed the gradient for the final layer

$$\frac{\partial L}{\partial \mathbf{s}_5}$$

What is the symbolic gradient of the previous state $\frac{\partial L}{\partial \mathbf{s}_4}$?

What is the symbolic gradient of the first state $\frac{\partial L}{\partial \mathbf{s}_1}$?

Answer

Chain rule, then relu cases, then to indicator notation

$$\begin{aligned}
\frac{\partial L}{\partial s_{4,i}} &= \sum_j \frac{\partial s_{5,j}}{\partial s_{4,i}} \frac{\partial L}{\partial s_{5,j}} \\
&= \sum_j \begin{cases} W^s_{i,j} \frac{\partial L}{\partial s_{5,j}} & s_{5,j} > 0 \\ 0 & o.w. \end{cases} \\
&= \sum_j \mathbf{1}(s_{5,j} > 0) W^s_{i,j} \frac{\partial L}{\partial s_{5,j}}
\end{aligned}$$

Answer

Multiple applications of Chain rule, combine relu cases.

$$
\begin{aligned}
\frac{\partial L}{\partial s_{1,i}} &= \sum_{j_2} \cdots \sum_{j_5} \frac{\partial s_{5,j_5}}{\partial s_{4,j_4}} \frac{\partial L}{\partial s_{5,j_5}} \\
&= \sum_{j_2} \cdots \sum_{j_5} \mathbf{1}(s_{2,j_2} > 0 \wedge \ldots \wedge s_{5,j_5} > 0) W^s_{i,j_2} \ldots W^s_{j_4,j_5} \frac{\partial L}{\partial s_{5,j}}
\end{aligned}
$$

The Promise of RNNs

- We hope to learn a model with memory.
- For acceptors this means long-range interaction.

  `How can you not see this movie?`

  `You should not see this movie.`
- Memory interaction here is at $\mathbf{s}_1$, but gradient signal is at $\mathbf{s}_n$

Vanishing Gradients

- Gradients at early layers go through many squashing layers.
- For instance consider quiz with hardtanh

$$\sum_{j_2} \ldots \sum_{j_5} \mathbf{1}((0 < s_{2,j_2} < 1) \wedge \ldots \wedge (0 < s_{5,j} < 1)) W^s_{i,j_2} \ldots W^s_{j_4,j_5} \frac{\partial L}{\partial s_{5,j}}$$

- The indicator term causes a tendency towards *vanishing gradients*.
- If this occurs, model cannot learn long-term dependencies.

Contents

Deep Networks

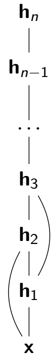
- This same issue occurs in deep MLPs.

$$NN_{layer}(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$$

$$\begin{array}{c} \mathbf{h}_n \\ | \\ \mathbf{h}_{n-1} \\ | \\ \cdots \\ | \\ \mathbf{h}_2 \\ | \\ \mathbf{h}_1 \\ | \\ \mathbf{x} \end{array}$$

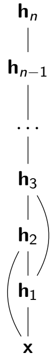
Thought Experiment: Additive Skip-Connections

$$NN_{sl1}(\mathbf{x}) = \frac{1}{2}\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + \frac{1}{2}\mathbf{x}$$



Thought Experiment: Dynamic Skip-Connections

$$
\begin{aligned}
NN_{sl1}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t) \\
\mathbf{W}^t &\in \mathbb{R}^{d_{\mathrm{in}}\times 1}
\end{aligned}
$$



Contents

LSTMs