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 $i \in \{1, \dots, n\}$

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

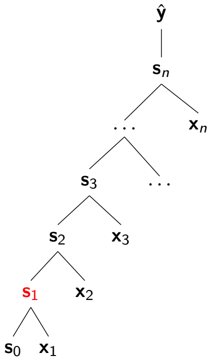
- θ is shared by all R

Example:

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

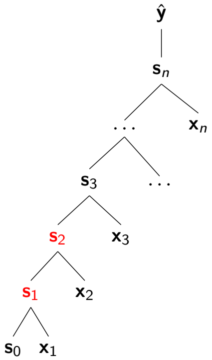
Review: BPTT (Acceptor)

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



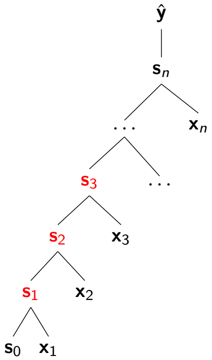
Review: BPTT (Acceptor)

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



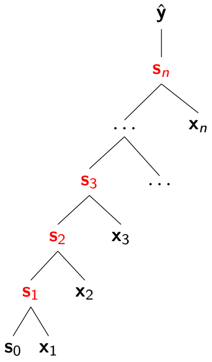
Review: BPTT (Acceptor)

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



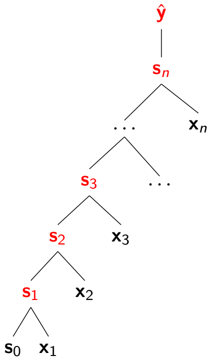
Review: BPTT (Acceptor)

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



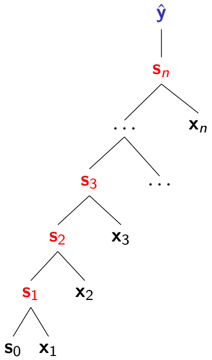
Review: BPTT (Acceptor)

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



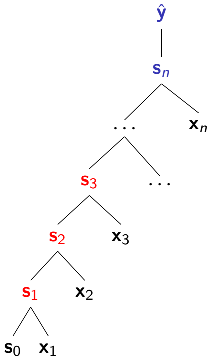
Review: BPTT (Acceptor)

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



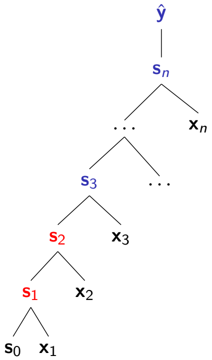
Review: BPTT (Acceptor)

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



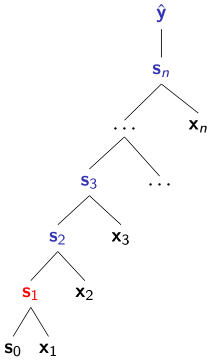
Review: BPTT (Acceptor)

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



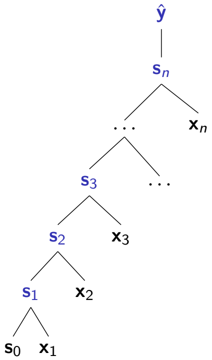
Review: BPTT (Acceptor)

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



Review: BPTT (Acceptor)

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



Review: 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 $\mathbf{x}_1, \ldots, \mathbf{x}_5$. 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

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

- Chain-Rule
- ReLU Gradient rule
- Indicator notation

Answer

$$
\begin{aligned}
\frac{\partial L}{\partial s_{1,j_1}} &= \sum_{j_2} \frac{\partial s_{2,j_2}}{\partial s_{1,j_1}} \ldots \sum_{j_5} \frac{\partial s_{5,j_5}}{\partial s_{4,j_4}} \frac{\partial L}{\partial s_{5,j_5}} \\
&= \sum_{j_2} \ldots \sum_{j_5} \mathbf{1}(s_{2,j_2} > 0) \ldots \mathbf{1}(s_{5,j_5} > 0) W^s_{j_1,j_2} \ldots W^s_{j_4,j_5} \frac{\partial L}{\partial s_{5,j_5}} \\
&= \sum_{j_2 \ldots j_5} \prod_{k=2}^{5} \mathbf{1}(s_{k,j_k} > 0) W^s_{j_{k-1},j_k} \frac{\partial L}{\partial s_{5,j_5}}
\end{aligned}
$$

- Multiple applications of Chain rule
- Combine and multiply.
- Product of weights.

The Promise of RNNs

- Hope: Learn the long-range interactions of language from data.
- For acceptors this means maintaining early state:
- **Example:** `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 $\mathbf{s}_n$

Long-Term Gradients

- Gradients go through multiplicative layers.
- Fine at end layers, but issues with early layers.
- For instance consider quiz with hardtanh

$$\sum_{j_2 \ldots j_5} \prod_{k=2}^{5} \mathbf{1}(1 > s_{k,j_k} > 0) W^s_{j_{k-1},j_k} \frac{\partial L}{\partial s_{5,j_5}}$$

- The multiplicative effect tends very large *exploding* or *vanishing* gradients.

Exploding Gradients

Easier case, can be handled heuristically,

- Occurs if there is no saturation but exponential blowup.

$$\sum_{j_2 \ldots j_n} \prod_{k=2}^{n} W^{s}_{j_{k-1},j_k}$$

- In these cases, there are reasonable short-term gradients, but bad long-term gradients.
- Two practical heuristics:
  - gradient clipping, i.e. bounding any gradient by a maximum value
  - gradient normalization, i.e. renormalizing the the RNN gradients if they are above a fixed norm value.

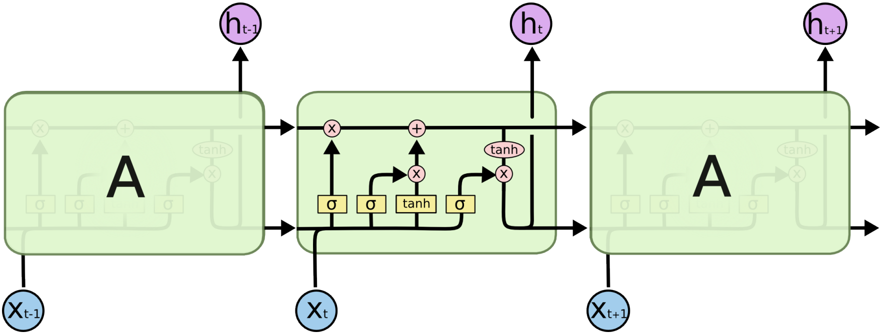
Vanishing Gradients

- Occurs when combining small weights (or from saturation)

$$\sum_{j_2 \ldots j_n} \prod_{k=2}^{n} W^s_{j_{k-1},j_k}$$

- Again, affects mainly long-term gradients.
- However, not as simple as boosting gradients.

LSTM (Hochreiter and Schmidhuber, 1997)



LSTM Formally

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{c}_i &= \mathbf{j} \odot \mathbf{i} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
\mathbf{h}_i &= \tanh(\mathbf{c}_i) \odot \mathbf{o} \\
\mathbf{i} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^{i}) \\
\mathbf{j} &= \sigma(\mathbf{x}\mathbf{W}^{xj} + \mathbf{h}_{i-1}\mathbf{W}^{hj} + \mathbf{b}^{j}) \\
\mathbf{f} &= \sigma(\mathbf{x}\mathbf{W}^{xf} + \mathbf{h}_{i-1}\mathbf{W}^{hf} + \mathbf{b}^{f}) \\
\mathbf{o} &= \tanh(\mathbf{x}\mathbf{W}^{xo} + \mathbf{h}_{i-1}\mathbf{W}^{ho} + \mathbf{b}^{o})
\end{aligned}
$$

- (eeks.)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Unreasonable Effectiveness of RNNs

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Proof. Omitted. $\square$

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. $\square$

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma **??**. $\square$

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

1. (1) $\mathcal{F}$ is an algebraic space over S.
2. (2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. $\square$

PANDARUS:

Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Music Composition

http://www.hexahedria.com/2015/08/03/
composing-music-with-recurrent-neural-networks/

Today's Lecture

- Build up to LSTMs
- Note: Development is ahistoric, later papers first.
- Main idea: Getting around the vanishing gradient issue.

Contents

Review: Deep ReLU Networks

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

- Given a input $\mathbf{x}$ can build arbitrarily deep fully-connected networks.
- Deep Neural Network (DNN) :

$$NN_{layer}(NN_{layer}(NN_{layer}(\ldots NN_{layer}(\mathbf{x}))))$$

Deep Networks

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

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

- Can have similar issues with vanishing gradients.

$$\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = \sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}$$

Deep Networks

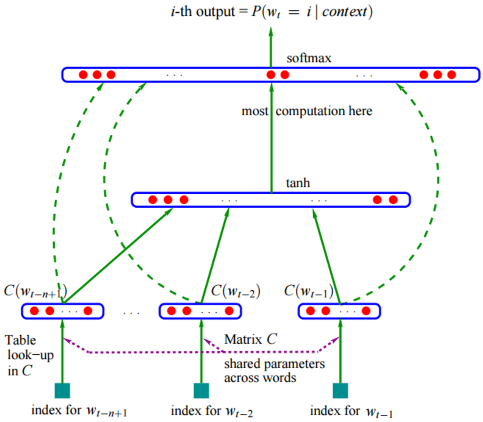
$$NN_{layer}(\mathbf{x}) = \mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$$

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

- Can have similar issues with vanishing gradients.

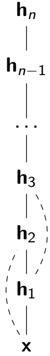
$$\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = \sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}$$

Review: NLM Skip Connections



Thought Experiment: Additive Skip-Connections

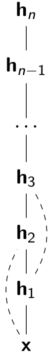
$$NN_{sl1}(\mathbf{x}) = \frac{1}{2}\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + \frac{1}{2}\mathbf{x}$$



Exercise: Gradients

Exercise: What is the gradient of $\frac{\partial L}{\mathbf{h}_{n-1}}$ with skip-connections?

Inductively what happens?



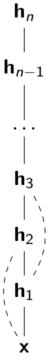
Exercise

We now have the average of two terms. One with no saturation condition or multiplicative term.

$$\begin{aligned}\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = & \quad \tfrac{1}{2} \quad (\sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}) + \\ & \quad \tfrac{1}{2} \quad (h_{n-1,j_{n-1}} \frac{\partial L}{\partial h_{n,j_{n-1}}})\end{aligned}$$

Thought Experiment 2: Dynamic Skip-Connections

$$
\begin{aligned}
NN_{sl2}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t) \\
\mathbf{W}^1 &\in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{hid}}} \\
\mathbf{W}^t &\in \mathbb{R}^{d_{\mathrm{hid}} \times 1}
\end{aligned}
$$



Dynamic Skip-Connections: intuition

$$
\begin{aligned}
NN_{sl2}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t)
\end{aligned}
$$

- No longer directly computing the next layer $\mathbf{h}_i$.
- Instead computing $\Delta\mathbf{h}$ and dynamic update proportion t.
- Seems like a small change from DNN. Why does this matter?

Dynamic Skip-Connections Gradient

$$
\begin{aligned}
NN_{sl2}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t)
\end{aligned}
$$

- The t values are saved on the forward pass.
- t allows for a direct flow of gradients to earlier layers.

$$
\begin{aligned}
\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = \quad & (1-t) \quad (\sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}) \\
+ \quad & t \quad (h_{n-1,j_{n-1}} \frac{\partial L}{\partial h_{n,j_{n-1}}}) + \dots
\end{aligned}
$$

- (What is the ...?)

Dynamic Skip-Connections

$$
\begin{aligned}
NN_{sl2}(\mathbf{x}) &= (1-t)\,\mathrm{ReLU}(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) + t\mathbf{x} \\
t &= \sigma(\mathbf{x}\mathbf{W}^t + b^t)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = & \quad (1-t) \quad (\sum_{j_n} \mathbf{1}(h_{n,j_n} > 0) W_{j_{n-1},j_n} \frac{\partial L}{\partial h_{n,j_n}}) \\
+ & \quad\quad t \quad\quad (h_{n-1,j_{n-1}} \frac{\partial L}{\partial h_{n,j_{n-1}}}) + \ldots
\end{aligned}
$$

- Note: $\mathbf{h}$ and $\mathbf{W}^t$ are also receiving gradients through the sigmoid!
- (Backprop is fun.)

Highway Network (Srivastava et al., 2015)

- Now add a combination at each dimension.
- $\odot$ is point-wise multiplication.

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

- $\tilde{\mathbf{h}}$; *transform* (e.g. standard MLP layer)
- $\mathbf{t}$; *carry* (dimension-specific dynamic skipping)

Highway Gradients

- The $\mathbf{t}$ values are saved on the forward pass.
- t_j determines the update of dimension j.

$$
\begin{aligned}
\frac{\partial L}{\partial h_{n-1,j_{n-1}}} = & \quad (\sum_{j_n}(1 - t_{j_n})\mathbf{1}(h_{n,j_n} > 0)W_{j_{n-1},j_n}\frac{\partial L}{\partial h_{n,j_n}}) \\
+ & \quad t_{j_{n-1}} \quad (h_{n-1,j_{n-1}}\frac{\partial L}{\partial h_{n,j_{n-1}}}) + \dots
\end{aligned}
$$

Intuition: Gating

- This is known as the *gating* operation

$$\mathbf{t} \odot \mathbf{x}$$

- Allows vector $\mathbf{t}$ to mask or gate $\mathbf{x}$.
- True gating would have $\mathbf{t} \in \{0, 1\}^{d_{\mathrm{hid}}}$
- Approximate with the sigmoid,

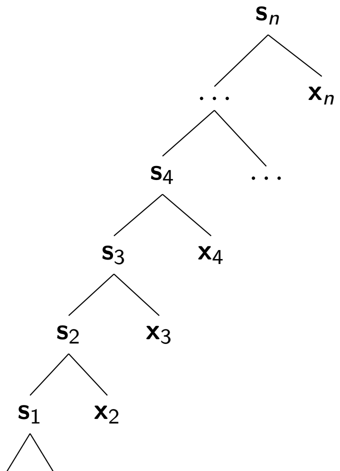
$$\mathbf{t} = \sigma(\mathbf{W}^t \mathbf{x} + \mathbf{b})$$

Contents

Back To Acceptor RNNs

- Acceptor RNNs are deep networks with shared weights.
- Elman tanh layers

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

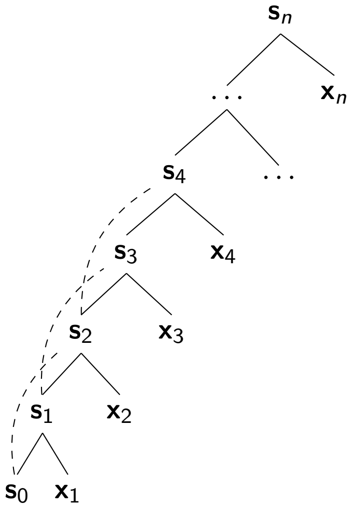


RNN with Skip-Connections

- Can replace layer with modified highway layer.

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= (1 - \mathbf{t}) \odot \tilde{\mathbf{h}} + \mathbf{t} \odot \mathbf{s}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^{x} + \mathbf{s}_{i-1}\mathbf{W}^{s} + \mathbf{b}) \\
\mathbf{t} &= \sigma(\mathbf{x}\mathbf{W}^{xt} + \mathbf{s}_{i-1}\mathbf{W}^{st} + \mathbf{b}^{t}) \\
\mathbf{W}^{xt}, \mathbf{W}^{x} &\in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{hid}}} \\
\mathbf{W}^{st}, \mathbf{W}^{s} &\in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{hid}}} \\
\mathbf{b}^{t}, \mathbf{b} &\in \mathbb{R}^{1 \times d_{\mathrm{hid}}}
\end{aligned}
$$

RNN with Skip-Connections



Final Idea: Stopping flow

- For many tasks, it is useful to halt propagation.
- Can do this by applying a reset gate **r**.

$$\begin{aligned}\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^{x} + (\mathbf{r} \odot \mathbf{s}_{i-1})\mathbf{W}^{s} + \mathbf{b}) \\ \mathbf{r} &= \sigma(\mathbf{x}\mathbf{W}^{xr} + \mathbf{s}_{i-1}\mathbf{W}^{sr} + \mathbf{b}^{r})\end{aligned}$$

Gated Recurrent Unit (GRU) (Cho et al 2014)

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= (1 - \mathbf{t}) \odot \tilde{\mathbf{h}} + \mathbf{t} \odot \mathbf{s}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^x + (\mathbf{r} \odot \mathbf{s}_{i-1})\mathbf{W}^s + \mathbf{b}) \\
\mathbf{r} &= \sigma(\mathbf{x}\mathbf{W}^{xr} + \mathbf{s}_{i-1}\mathbf{W}^{sr} + \mathbf{b}^r) \\
\mathbf{t} &= \sigma(\mathbf{x}\mathbf{W}^{xt} + \mathbf{s}_{i-1}\mathbf{W}^{st} + \mathbf{b}^t) \\
\mathbf{W}^{xt}, \mathbf{W}^{xr}, \mathbf{W}^x &\in \mathbb{R}^{d_{\text{in}} \times d_{\text{hid}}} \\
\mathbf{W}^{st}, \mathbf{W}^{sr}, \mathbf{W}^s &\in \mathbb{R}^{d_{\text{hid}} \times d_{\text{hid}}} \\
\mathbf{b}^t, \mathbf{b} &\in \mathbb{R}^{1 \times d_{\text{hid}}}
\end{aligned}
$$

- $\mathbf{t}$; dynamic skip-connections
- $\mathbf{r}$; reset gating
- $\mathbf{s}$; hidden state

Example: Language Modeling

- In non-Markovian language modeling, treat corpus as a sequence
- **r** allows resetting the state after a sequence.

  consumers may want to move their telephones a little closer to the tv set </s> <unk> <unk> watching abc 's monday night football can now vote during <unk> for the greatest play in N years from among four or five <unk> <unk> </s> two weeks ago viewers of several nbc <unk> consumer segments started calling a N number for advice on various <unk> issues </s> and the new syndicated reality show hard copy records viewers ' opinions for possible airing on the next day 's show </s>

Contents

LSTM

- Long Short-Term Memory network uses the same idea.
- Model has three gates, input, output, forget.
- Developed first, but a bit harder to follow.
- Seems to be better at LM, comparable to GRU on other tasks.

LSTMs State

The state $\mathbf{s}_i$ is made of 2 components :

- $\mathbf{c}_i$; cell
- $\mathbf{h}_i$; hidden

$$R(\mathbf{s}_{i-1}, \mathbf{x}_i) \quad = \quad [\mathbf{c}_i, \mathbf{h}_i]$$

LSTM Development: Input and Forget Gates

The cell is updated with $\Delta\mathbf{c}$, not a convex combination (two gates).

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{c}_i &= \mathbf{j} \odot \tilde{\mathbf{h}} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^i) \\
\mathbf{j} &= \sigma(\mathbf{x}\mathbf{W}^{xj} + \mathbf{h}_{i-1}\mathbf{W}^{hj} + \mathbf{b}^j) \\
\mathbf{f} &= \sigma(\mathbf{x}\mathbf{W}^{xf} + \mathbf{h}_{i-1}\mathbf{W}^{hf} + \mathbf{b}^f)
\end{aligned}
$$

- $\mathbf{c}_i$; cell
- $\mathbf{j}$; input gate
- $\mathbf{f}$; forget gate

LSTM Development: Hidden

Hidden $\mathbf{h}_i$ squashes $\mathbf{c}_i$

$$
\begin{aligned}
R(\mathbf{c}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{h}_i &= \tanh(\mathbf{c}_i) \\
\mathbf{c}_i &= \mathbf{j} \odot \tilde{\mathbf{h}} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
\tilde{\mathbf{h}} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^{i}) \\
\mathbf{j} &= \sigma(\mathbf{x}\mathbf{W}^{xj} + \mathbf{h}_{i-1}\mathbf{W}^{hj} + \mathbf{b}^{j}) \\
\mathbf{f} &= \sigma(\mathbf{x}\mathbf{W}^{xf} + \mathbf{h}_{i-1}\mathbf{W}^{hf} + \mathbf{b}^{f})
\end{aligned}
$$

- $\mathbf{c}_i$; cell
- $\mathbf{h}_i$; hidden
- $\mathbf{j}$; input gate
- $\mathbf{f}$; forget gate

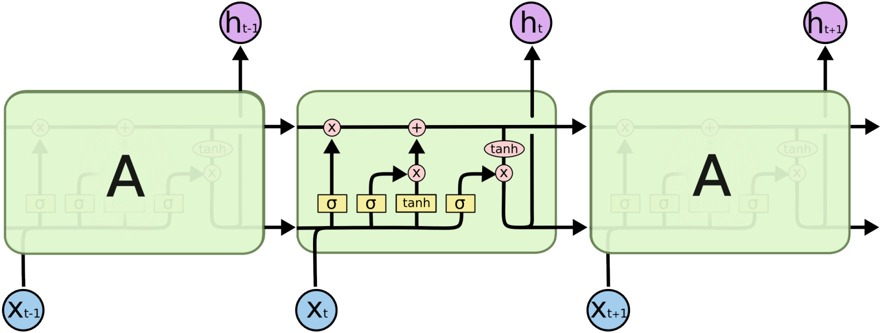
Long Short-Term Memory

Finally, another output gate is applied to **h**

$$
\begin{aligned}
R(\mathbf{s}_{i-1}, \mathbf{x}_i) &= [\mathbf{c}_i, \mathbf{h}_i] \\
\mathbf{c}_i &= \mathbf{j} \odot \mathbf{i} + \mathbf{f} \odot \mathbf{c}_{i-1} \\
\mathbf{h}_i &= \tanh(\mathbf{c}_i) \odot \mathbf{o} \\
\mathbf{i} &= \tanh(\mathbf{x}\mathbf{W}^{xi} + \mathbf{h}_{i-1}\mathbf{W}^{hi} + \mathbf{b}^{i}) \\
\mathbf{j} &= \sigma(\mathbf{x}\mathbf{W}^{xj} + \mathbf{h}_{i-1}\mathbf{W}^{hj} + \mathbf{b}^{j}) \\
\mathbf{f} &= \sigma(\mathbf{x}\mathbf{W}^{xf} + \mathbf{h}_{i-1}\mathbf{W}^{hf} + \mathbf{b}^{f}) \\
\mathbf{o} &= \tanh(\mathbf{x}\mathbf{W}^{xo} + \mathbf{h}_{i-1}\mathbf{W}^{ho} + \mathbf{b}^{o})
\end{aligned}
$$

Output Gate?

- The output gate feels the most ad-hoc (why tanh?)
- Luckily Jozefowicz et al (2015) find output not too important.



h t-1
h t
h t+1
A
σ
σ
tanh
σ
tanh
A
x t-1
x t
x t+1

Accuracy Results (Jozefowicz et al, 2015)

Arch.	Arith.	XML	PTB
Tanh	0.29493	0.32050	0.08782
LSTM	0.89228	0.42470	0.08912
LSTM-f	0.29292	0.23356	0.08808
LSTM-i	0.75109	0.41371	0.08662
LSTM-o	0.86747	0.42117	0.08933
LSTM-b	0.90163	0.44434	0.08952
GRU	0.89565	0.45963	0.09069
MUT1	**0.92135**	**0.47483**	0.08968
MUT2	0.89735	**0.47324**	0.09036
MUT3	0.90728	0.46478	**0.09161**

Accuracy of RNN models on three different tasks. LSTM models include versions without the forget, input, and output gates.

English LM Results (Jozefowicz et al, 2015)

Arch.	5M-tst	10M-v	20M-v	20M-tst
Tanh	4.811	4.729	4.635	4.582 (97.7)
LSTM	4.699	4.511	4.437	4.399 (81.4)
LSTM-f	4.785	4.752	4.658	4.606 (100.8)
LSTM-i	4.755	4.558	4.480	4.444 (85.1)
LSTM-o	4.708	4.496	4.447	4.411 (82.3)
LSTM-b	4.698	4.437	4.423	**4.380 (79.83)**
GRU	4.684	4.554	4.559	4.519 (91.7)
MUT1	4.699	4.605	4.594	4.550 (94.6)
MUT2	4.707	4.539	4.538	4.503 (90.2)
MUT3	4.692	4.523	4.530	4.494 (89.47)

NLL (Perplexity)