Announcements

• Pset 3 is out, due 10/20

• Psets 2 deadline was also extended to 10/20
9. Memory and sequence modeling

- CNNs for sequences
- RNNs
- LSTMs
- Sequence models
“What color is the chair?”
What color is the chair?
red
“What will the girl do next?”
Sequences

“An”, “evening”, “stroll”, “through”, “a”, “city”, “square”
Convolutions in time
“The Persistence of Memory”, Dali 1931
Rufus
Recurrent Neural Networks (RNNs)
Recurrent Neural Networks (RNNs)

Outputs $\hat{y}$

Hidden $h$

Inputs $x$

Time
Recurrent Neural Networks (RNNs)

\[ h_t = f(h_{t-1}, x_t) \]

\[ y_t = g(h_t) \]
Recurrent Neural Networks (RNNs)

\[
\begin{align*}
\mathbf{h}_t &= f(\mathbf{h}_{t-1}, \mathbf{x}_t) \\
\mathbf{y}_t &= g(\mathbf{h}_t)
\end{align*}
\]
Recurrent Neural Networks (RNNs)

\[
\begin{align*}
\text{Outputs } \hat{y} &= \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \\
\text{Hidden } h &= \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \\
\text{Inputs } x &= \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \quad \bigcirc \\
\end{align*}
\]

\[
\begin{align*}
a_t &= \mathbf{W}h_{t-1} + \mathbf{U}x_t + b \\
h_t &= \tanh(a_t) \\
o_t &= \mathbf{V}h_t + c \\
\hat{y}_t &= \text{softmax}(o_t)
\end{align*}
\]
Deep Recurrent Neural Networks (RNNs)

Outputs $\hat{y}$

Hidden

Inputs $x$

$W_L$, $U_L$, $W_1$, $U_1$, $W_2$, $U_2$, $V$
Backprop through time

Outputs $\hat{y}$

Hidden $h$

Inputs $x$

\[
\frac{\partial \hat{y}_t}{\partial x_0} = \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial h_0} \frac{\partial h_0}{\partial x_0}
\]
\[
\frac{\partial J}{\partial W} = \sum_{t=0}^{T} \frac{\partial \mathcal{L}_t(\hat{y}_t, y_t)}{\partial W}
\]
Parameter sharing

Parameter sharing —> sum gradients
\[
\frac{\partial J}{\partial W} = \sum_{t=0}^{T} \frac{\partial \mathcal{L}_t(\hat{y}_t, y_t)}{\partial W}
\]

\[
\frac{\partial \mathcal{L}_t}{\partial W} = \sum_i \frac{\partial \mathcal{L}_t}{\partial W^i}
\]
Recurrent linear layer

\[ a_t = Wh_{t-1} + Ux_t + b \]

\[ \frac{\partial L}{\partial h_{t-1}} = \frac{\partial L}{\partial a_t} W \]

\[ \frac{\partial L}{\partial x_t} = \frac{\partial L}{\partial a_t} U \]

\[ \frac{\partial J}{\partial W} = \sum_{t=0}^{T} \frac{\partial L(\hat{y}_t, y_t)}{\partial W} \]
The problem of long-range dependences

Why not remember everything?

- Memory size grows with $t$

- This kind of memory is **nonparametric**: there is no finite set of parameters we can use to model it

- RNNs make a Markov assumption — the future hidden state only depends on the immediately preceding hidden state

- By putting the right info in to the hidden state, RNNs can model dependences that are arbitrarily far apart
The problem of long-range dependences

Capturing long-range dependences requires propagating information through a long chain of dependences.

Old observations are forgotten

Stochastic gradients become high variance (noisy), and gradients may vanish or explode
LSTMs
Long Short Term Memory
[Hochreiter & Schmidhuber, 1997]

A special kind of RNN designed to avoid forgetting.

This way the default behavior is not to forget an old state. Instead of forgetting by default, the network has to learn to forget.
[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]
C_t = Cell state

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]
Decide what information to throw away from the cell state.

Each element of cell state is multiplied by $\sim 1$ (remember) or $\sim 0$ (forget).

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

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]
Decide what new information to add to the cell state.

which indices to write to

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

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

what to write to those indices

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]
Forget selected old information, write selected new information.

\[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \]
After having updated the cell state’s information, decide what to output.

\[
o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)
\]
\[
h_t = o_t \ast \tanh(C_t)
\]

[Slide derived from Chris Olah: http://colah.github.io/posts/2015-08-Understanding-LSTMs/]
Sequence models
Once upon ___ → Predictor → time

Once ___ a time → Predictor → Upon
Once upon a time
There and back, again
The slow brown, fox
To be or not to, be

\[
\begin{pmatrix}
  x_1, \ldots, x_{n-1} \\
  x_n
\end{pmatrix}
\]

\[\xrightarrow{\text{Learner}}\]

Predictor

\[
\begin{pmatrix}
  x_1, \ldots, x_{n-1} \\
  x_n
\end{pmatrix}
\]

\[\xrightarrow{\text{Predictor}}\]

\[\hat{x}_n\]

Colorless green ideas sleep

\[\xrightarrow{\text{Predictor}}\]

\[\text{furiously}\]
Autoregressive probability model

\[ p(\mathbf{X}) = p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}) p(\mathbf{x}_{n-1} | \mathbf{x}_1, \ldots, \mathbf{x}_{n-2}) \ldots p(\mathbf{x}_2 | \mathbf{x}_1) p(\mathbf{x}_1) \]

\[ p(\mathbf{X}) = \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}) \]
Modeling a sequence of words

How to model $p(\text{time}|\text{Once, upon, a})$?

Just treat it as a next word classifier!

Once upon a $f$

year

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

0 1
How to represent words as numbers?

We can represent words as 1-hot-vectors of size $K$, where $K$ is the size of the vocabulary (e.g., $K=100,000$).
How to represent words as numbers?

\[ f_\theta : X \rightarrow \mathbb{R}^K \]

Once upon a $f$

Or, represent each character as a class (e.g., $K=26$ for English letters), and represent words as a sequence of characters.
A mild stimulant that enhances cognitive ability.

"Molecule-2-text"
A mild stimulant that enhances cognitive ability.
Training

Targets: $y$, A, mild, stimulant, that, enhances, cognitive, ability, END

Outputs $p_\theta(\cdot)$:

Hidden:
- LSTM
- LSTM
- LSTM
- LSTM
- LSTM
- LSTM
- LSTM
- LSTM

Input:

Max-likelihood objective: maximize probability the model assigns to each target word: $\arg\max_{\theta} \log p_\theta(y)$
Max-likelihood objective:
minimize cross-entropy between model outputs and one-hot encoded targets.

\[ f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} H(y_i, \hat{y}_i) \]
A mild stimulant that enhances cognitive ability.

Condition each next word prediction on the ground-truth preceding word.
Sample from predicted distribution over words.
Alternatively, sample most likely word.
Beam search

Sample multiple sequences (top-k greedy completions on each step), then pick the sequence with highest score.

Score could be model’s confidence: $p_{\theta}(y_1, \ldots, y_T|x)$
The problem of long-range dependences

Why not remember everything?

- Memory size grows with \( t \)

- This kind of memory is **nonparametric**: there is no finite set of parameters we can use to model it

- RNNs make a Markov assumption — the future hidden state only depends on the immediately preceding hidden state

- By putting the right info in to the hidden state, RNNs can model dependences that are arbitrarily far apart
The problem of long-range dependences

Other methods exist that do directly link old “memories” (observations or hidden states) to future predictions:

- Temporal convolutions
- Attention / Transformers (see https://arxiv.org/abs/1706.03762)
- Memory networks (see https://arxiv.org/abs/1410.3916)
Modeling arbitrarily long sequences

- **Recurrence** — recurrent weights are shared across time

- **Convolution** — conv weights are shared across time

- **Attention** — weights are dynamically determined as a function of the data (conv kernel with attention weights is shown on the right)
How do we model sequences?

**one to one**
- Input: No sequence
- Output: No sequence
- Example: “standard” classification / regression problems

**one to many**
- Input: No sequence
- Output: Sequence
- Example: Im2Caption

**many to one**
- Input: Sequence
- Output: No sequence
- Example: sentence classification, multiple-choice question answering

**many to many**
- Input: Sequence
- Output: Sequence
- Example: machine translation, video captioning, open-ended question answering, video question answering

Credit: Dhruv Batra, Andrej Karpathy

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Memory fast and slow

Parameters are "slow memory"

Data \[ \{x^{(i)}, y^{(i)}\}_{i=1}^{N} \rightarrow \text{Learner} \]

Data \[ x^{(i)} \rightarrow \text{Neural Net} \]

Activations are "fast memory"
Fast weights? Slow activations?

- **Hypernets** are nets that output weights of another net — these weights are a “fast memory” of the input to the hypernet.

- **Code books** use tensors of activations that are learned (backprop to activations). These activations are “slow memory” of the dataset you are learning.
9. Memory and sequence modeling

- CNNs for sequences
- RNNs
- LSTMs
- Sequence models