L11: Recurrent Neural Networks

Spring 2021

(Some slides adapted from Chris Manning, Abigail See)
Announcements

• Midterm - Wednesday to Thursday (24 hours)
  • You can open the exam after Wed 12pm-Thu 9am (NOT later than that!!)
  • This lecture will be included in the midterm

• Open book: you can have access to our course materials (lecture slides, readings, videos) and calculators. Not allowed to access the internet otherwise. You can prepare a cheatsheet if you want.

• Please don’t use Ed during the exam period. If you have any questions, please write to cos484584.midterm@gmail.com (mention the problem ID in your email title)

• Please fill out your preference of exam time TODAY if you haven’t: https://forms.gle/E5dpMct7Y5v9AUbN8
Recurrent neural networks (RNNs)

How can we model sequences using neural networks?

- Recurrent neural networks = A class of neural networks used to model sequences, allowing to handle variable length inputs

- Very crucial in NLP problems (different from images) because sentences/paragraphs are variable-length, sequential inputs
Motivation: language models

n-gram language models

Sentence: “the cat sat on the mat”

\[ P(\text{the cat sat on the mat}) = P(\text{the}) \times P(\text{cat|the}) \times P(\text{sat|the cat}) \]
\[ \times P(\text{on|the cat sat}) \times P(\text{the|the cat sat on}) \]
\[ \times P(\text{mat|the cat sat on the}) \]

1st order
\[ P(\text{mat|the cat sat on the}) \approx P(\text{mat|the}) \]

2nd order
\[ P(\text{mat|the cat sat on the}) \approx P(\text{mat|on the}) \]

kth order Markov
\[ P(w_1w_2\ldots w_n) \approx \prod_i P(w_i|w_{i-k}\ldots w_{i-1}) \]

Q: How do we know what size of k is needed?
Motivation: language models

n-gram language models

the students opened their ______

as the proctor started the clock, the students opened their ______

Q: Why can’t we just keep a very large value of $k$?

Because it is too sparse to estimate the probabilities as $k$ increases:

$$P(w|\text{students opened their}) = \frac{\text{count(students opened their } w)}{\text{count(students opened their) }}$$
Motivation: language models

n-gram language models

Generate text with a 4-gram LM:

today the price of gold per ton, while production of shoe lasts and shoe industry, the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks, sept 30 end primary 76 cts a share.

Surprisingly grammatical!

...but incoherent. We need to consider more than three words at a time if we want to model language well.
Motivation: language models

Feedforward neural language model

- $P(\text{mat} \mid \text{the cat sat on the}) = ?$

Previous $n$ words = $k$-th order Markov assumption!

- Input layer ($n = 5$):
  $$x = [e_{\text{the}}; e_{\text{cat}}; e_{\text{sat}}; e_{\text{on}}; e_{\text{the}}] \in \mathbb{R}^{dn}$$

- Hidden layer
  $$h = \tanh(Wx + b) \in \mathbb{R}^{h}$$

- Output layer (softmax)
  $$z = Uh \in \mathbb{R}^{|V|}$$
  $$P(w = i \mid \text{the cat sat on the})$$
  $$= \text{softmax}_i(z) = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Q: Why is this model still not good enough?
Motivation: language models

Feedforward neural language model

- Input layer ($n = 5$):
  
  \[ x = [e_{\text{the}}; e_{\text{cat}}; e_{\text{sat}}; e_{\text{on}}; e_{\text{the}}] \in \mathbb{R}^{dn} \]

- Hidden layer
  
  \[ h = \tanh(Wx + b) \in \mathbb{R}^h \]

- $W \in \mathbb{R}^{h \times nd}$ scales with $n$

- The model learns separate patterns for the same item!

“all the” appears in different positions of two sliding windows
What are recurrent neural networks?
Recurrent neural networks (RNNs)

A family of neural networks allowing to handle **variable length inputs**

A function: \( y = \text{RNN}(x_1, x_2, \ldots, x_n) \in \mathbb{R}^h \) where \( x_1, \ldots, x_n \in \mathbb{R}^d \)

Core idea: apply the same weights repeatedly at different positions
Recurrent neural networks (RNNs)

Proven to be an highly effective approach to language modeling, sequence tagging as well as text classification tasks:

- **Language modeling**
- **Sequence tagging**
- **Text classification**
Recurrent neural networks (RNNs)

Form the basis for the modern approaches to machine translation, question answering and dialogue systems:

sequence-to-sequence models
Simple recurrent neural networks

A function: \( y = \text{RNN}(x_1, x_2, \ldots, x_n) \in \mathbb{R}^h \) where \( x_1, \ldots, x_n \in \mathbb{R}^d \)

\( h_0 \in \mathbb{R}^h \) is an initial state

\[ h_t = f(h_{t-1}, x_t) \in \mathbb{R}^h \]

\( h_t \) : hidden states which store information from \( x_1 \) to \( x_t \)

Simple RNNs:

\[ h_t = g(Wh_{t-1} + Ux_t + b) \in \mathbb{R}^h \]

\( g \): nonlinearity (e.g. tanh),

\( W \in \mathbb{R}^{h \times h}, U \in \mathbb{R}^{h \times d}, b \in \mathbb{R}^h \)

This model contains \( h \times (h + d + 1) \) parameters, and optionally \( h \) for \( h_0 \) (a common way is just to set \( h_0 \) as \( 0 \))
Simple recurrent neural networks

\[ h_t = g(Wh_{t-1} + Ux_t + b) \in \mathbb{R}^h \]

Key idea: apply the same weights \( W, U, b \) repeatedly
RNNs vs Feedforward NNs

Feed-Forward Neural Network

Recurrent Neural Network

Input layer 1

Hidden layer 1

Hidden layer 2

Output layer
Recurrent Neural Language Models (RNNLMs)

\[ P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \times \ldots \times P(w_n \mid w_1, w_2, \ldots, w_{n-1}) \]

\[ = P(w_1 \mid h_0) \times P(w_2 \mid h_1) \times P(w_3 \mid h_2) \times \ldots \times P(w_n \mid h_{n-1}) \]

- Denote \( \hat{y}_t = \text{softmax}(W_o h_t) \), \( W_o \in \mathbb{R}^{V \times h} \)

- Cross-entropy loss:

\[
L(\theta) = - \frac{1}{n} \sum_{t=1}^{n} \log \hat{y}_{t-1}(w_t)
\]

\[ \theta = \{ W, U, b, W_o, E \} \]
Progress on language models

On the Penn Treebank (PTB) dataset
Metric: perplexity

$$ppl(S) = 2^x$$ where
$$x = - \frac{1}{W} \sum_{i=1}^{n} \log_2 P(S^i)$$

<table>
<thead>
<tr>
<th>Model</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>KN5</td>
<td>141.2</td>
</tr>
<tr>
<td>KN5 + cache</td>
<td>125.7</td>
</tr>
<tr>
<td>Feedforward NNLM</td>
<td>140.2</td>
</tr>
<tr>
<td>Log-bilinear NNLM</td>
<td>144.5</td>
</tr>
<tr>
<td>Syntactical NNLM</td>
<td>131.3</td>
</tr>
<tr>
<td>Recurrent NNLM</td>
<td>124.7</td>
</tr>
<tr>
<td>RNN-LDA LM</td>
<td>113.7</td>
</tr>
</tbody>
</table>

(Mikolov and Zweig, 2012): Context dependent recurrent neural network language model
Progress on language models

On the Penn Treebank (PTB) dataset
Metric: perplexity

<table>
<thead>
<tr>
<th>Model</th>
<th>#Param</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mikolov &amp; Zweig (2012) – RNN-LDA + KN-5 + cache</td>
<td>9M‡</td>
<td>-</td>
<td>92.0</td>
</tr>
<tr>
<td>Zaremba et al. (2014) – LSTM</td>
<td>20M</td>
<td>86.2</td>
<td>82.7</td>
</tr>
<tr>
<td>Gal &amp; Ghahramani (2016) – Variational LSTM (MC)</td>
<td>20M</td>
<td>-</td>
<td>78.6</td>
</tr>
<tr>
<td>Kim et al. (2016) – CharCNN</td>
<td>19M</td>
<td>-</td>
<td>78.9</td>
</tr>
<tr>
<td>Merity et al. (2016) – Pointer Sentinel-LSTM</td>
<td>21M</td>
<td>72.4</td>
<td>70.9</td>
</tr>
<tr>
<td>Grave et al. (2016) – LSTM + continuous cache pointer†</td>
<td>-</td>
<td>-</td>
<td>72.1</td>
</tr>
<tr>
<td>Inan et al. (2016) – Tied Variational LSTM + augmented loss</td>
<td>24M</td>
<td>75.7</td>
<td>73.2</td>
</tr>
<tr>
<td>Zilly et al. (2016) – Variational RHN</td>
<td>23M</td>
<td>67.9</td>
<td>65.4</td>
</tr>
<tr>
<td>Zoph &amp; Le (2016) – NAS Cell</td>
<td>25M</td>
<td>-</td>
<td>64.0</td>
</tr>
<tr>
<td>Melis et al. (2017) – 2-layer skip connection LSTM</td>
<td>24M</td>
<td>60.9</td>
<td>58.3</td>
</tr>
<tr>
<td>Merity et al. (2017) – AWD-LSTM w/o finetune</td>
<td>24M</td>
<td>60.7</td>
<td>58.8</td>
</tr>
<tr>
<td>Merity et al. (2017) – AWD-LSTM</td>
<td>24M</td>
<td>60.0</td>
<td>57.3</td>
</tr>
<tr>
<td>Ours – AWD-LSTM-MoS w/o finetune</td>
<td>22M</td>
<td>58.08</td>
<td>55.97</td>
</tr>
<tr>
<td>Ours – AWD-LSTM-MoS</td>
<td>22M</td>
<td>\textbf{56.54}</td>
<td>\textbf{54.44}</td>
</tr>
<tr>
<td>Merity et al. (2017) – AWD-LSTM + continuous cache pointer†</td>
<td>24M</td>
<td>53.9</td>
<td>52.8</td>
</tr>
<tr>
<td>Krause et al. (2017) – AWD-LSTM + dynamic evaluation†</td>
<td>24M</td>
<td>51.6</td>
<td>51.1</td>
</tr>
<tr>
<td>Ours – AWD-LSTM-MoS + dynamic evaluation†</td>
<td>22M</td>
<td>\textbf{48.33}</td>
<td>\textbf{47.69}</td>
</tr>
</tbody>
</table>

*(Yang et al, 2018): Breaking the Softmax Bottleneck: A High-Rank RNN Language Model*
RNNs: pros and cons

**Advantages:**

- Can process any length input
- Computation for step $t$ can (in theory) use information from many steps back
- Model size doesn’t increase for longer input context

**Disadvantages:**

- Recurrent computation is **slow** (can’t parallelize)
- In practice, difficult to access information from many steps back (optimization issue)

We will learn Transformer networks!

We will see some advanced RNNs (e.g., LSTMs, GRUs)
Training RNNLMs

• Forward pass + backward pass (compute gradients)

• Forward pass:

\[ L = 0 \quad h_0 = 0 \]

For \( t = 1, 2, \ldots, n \)

\[ y = - \log \text{softmax}(W_o h_{t-1})(w_i) \]

\[ x_t = e(w_t) \]

\[ h_t = g(W h_{t-1} + U x_t + b) \]

\[ L = L + \frac{1}{n} y \]

\[ L = L + \frac{1}{n} y \]
What is the running time of a forward pass?

(a) $O(h \times (d + h + |V|))$
(b) $O(n \times h \times (d + h + |V|))$
(c) $O(n \times (d + h + |V|))$
(d) $O(n \times h \times (d + h))$

The answer is (b).

$L = 0 \quad \mathbf{h}_0 = \mathbf{0}$

For $t = 1, 2, \ldots, n$

$y = - \log \text{softmax}(\mathbf{W}_o \mathbf{h}_{t-1})(w_t)$

$x_t = e(w_t)$

$\mathbf{h}_t = g(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}x_t + \mathbf{b})$

$L = L + \frac{1}{n}y$
Training RNNLMs

- Backward pass:
  - Backpropagation? Yes, but not that simple!

\[ P(y|\mathbf{x}) = \text{softmax}(W g(V f(\mathbf{x}))) \]

- The algorithm is called Backpropagation Through Time (BPTT).
Backpropagation through time [advanced]

\[ h_1 = g(W h_0 + U x_1 + b) \]
\[ h_2 = g(W h_1 + U x_2 + b) \]
\[ h_3 = g(W h_2 + U x_3 + b) \]
\[ L_3 = - \log \hat{y}_3(w_4) \]

You should know how to compute: \( \frac{\partial L_3}{\partial h_3} \)

\[ \frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial h_3} \frac{\partial h_3}{\partial W} + \frac{\partial L_3}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_3}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W} \]

\[ \frac{\partial L}{\partial W} = - \frac{1}{n} \sum_{t=1}^{n} \sum_{k=1}^{t} \frac{\partial L_t}{\partial h_t} \left( \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W} \]

If \( k \) and \( t \) are far away, the gradients are very easy to grow/shrink exponentially (called the gradient exploding or gradient vanishing problem)
What will happen if the gradients become too large or too small?

(a) If too large, the model will become difficult to converge
(b) If too small, the model can’t capture long-term dependencies
(c) If too small, the model may capture a wrong recent dependency
(d) None of the above

All of these are correct (a) (b) (c) 😊
Backpropagation through time

One solution for gradient exploding is called gradient clipping — if the norm of the gradient is greater than some threshold, scale it down before applying SGD update.

\[
\text{Algorithm 1 Pseudo-code for norm clipping}
\]

\[
\hat{g} \leftarrow \frac{\partial \mathcal{L}}{\partial \theta}
\]

\[
\text{if } \|\hat{g}\| \geq \text{threshold} \text{ then}
\]

\[
\hat{g} \leftarrow \frac{\text{threshold}}{\|g\|} \hat{g}
\]

\[
\text{end if}
\]
Backpropagation through time

Gradient vanishing is a harder problem to solve:

When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _______

The writer of the books is/are (planning a sequel)

**Syntactic recency**: The *writer of the books* is (correct)

**Sequential recency**: The *writer of the books* are (incorrect)
Truncated backpropagation through time

- Backpropagation is very expensive if you handle long sequences
- Run forward and backward through chunks of the sequence instead of whole sequence
- Carry hidden states forward in time forever, but only back-propagate for some smaller number of steps
Applications and Variants
You can generate text by repeated sampling. Sampled output is next step's input.
Let’s have some fun

You can train an RNN-LM on any kind of text, then generate text in that style.

Good afternoon. God bless you.

The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretch of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.

Thank you very much. God bless you, and God bless the United States of America.
Let’s have some fun

You can train an RNN-LM on any kind of text, then generate text in that style.

“Sorry,” Harry shouted, panicking — “I’ll leave those brooms in London, are they?”

“No idea,” said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry’s shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn’t felt it seemed. He reached the teams too.

“You believe if we’ve got friendly to come down and out of the library. I think I’ve found out Potter, I asked you he had . . . me. I think he’s not telling Dobby if yeh get with our Hogwarts . . .”

Let’s have some fun

You can train an RNN-LM on any kind of text, then generate text in that style.

\begin{proof}
We may assume that $\mathcal{I}$ is an abelian sheaf on $\mathcal{C}$.
\item Given a morphism $\Delta : \mathcal{F} \to \mathcal{I}$ is an injective and let $\mathfrak{q}$ be an abelian sheaf on $X$.
Let $\mathcal{F}$ be a fibered complex. Let $\mathcal{F}$ be a category.
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\begin{label}{lemma-characterize-quasi-finite}
Let $\mathcal{F}$ be an abelian quasi-coherent sheaf on $\mathcal{C}$.
Let $\mathcal{F}$ be a coherent $\mathcal{O}_X$-module. Then
$\mathcal{F}$ is an abelian catenary over $\mathcal{C}$.
\item The following are equivalent
\begin{enumerate}
\item $\mathcal{F}$ is an $\mathcal{O}_X$-module.
\end{enumerate}
\end{label}
\end{enumerate}
\end{proof}

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

Input: a sentence of $n$ words: $x_1, \ldots, x_n$
Output: $y_1, \ldots, y_n, y_i \in \{1, \ldots, C\}$

$$P(y_i = k) = \text{softmax}_k(W_o h_i) \quad W_o \in \mathbb{R}^{C \times h}$$

$$L = -\frac{1}{n} \sum_{i=1}^{n} \log P(y_i = k)$$

Q: How do we decode $y_i$ at testing time?
Application: Sequence Tagging

Input: a sentence of $n$ words: $x_1, \ldots, x_n$

Output: $y_1, \ldots, y_n, y_i \in \{1, \ldots, C\}$

The model doesn’t model dependencies between output labels!

[advanced]

- We can still model the joint probabilities over $\{y_1, y_2, \ldots, y_n\}$ and use beam search at decoding time

- The main difference compared to MEMMs - you don’t need to define manual features and the RNNs can derive features automatically!

(Lample et al, 2016): Neural Architectures for Named Entity Recognition
Application: Text Classification

Input: a sentence of $n$ words

Output: $y \in \{1, 2, \ldots, C\}$

\[
P(y = k) = \text{softmax}_k(W_o h_n) \quad W_o \in \mathbb{R}^{C \times h}
\]

\[
L = - \log P(y = c)
\]
Multi-layer RNNs

- RNNs are already “deep” on one dimension (unroll over time steps)
- We can also make them “deep” in another dimension by applying multiple RNNs
- Multi-layer RNNs are also called **stacked RNNs**.
Multi-layer RNNs

The hidden states from RNN layer $i$ are the inputs to RNN layer $i + 1$

- In practice, using 2 to 4 layers is common (usually better than 1 layer)
- Transformer networks can be up to 24 layers with lots of skip-connections.
Bidirectional RNNs

• Bidirectionality is important in language representations:

```plaintext
terribly:
• left context “the movie was”
• right context “exciting !”
```
Bidirectional RNNs

\[ h_t = f(h_{t-1}, x_t) \in \mathbb{R}^h \]

\[ \vec{h}_t = f_1(\vec{h}_{t-1}, x_t), t = 1, 2, \ldots n \]

\[ \hat{h}_t = f_2(\hat{h}_{t+1}, x_t), t = n, n-1, \ldots 1 \]

\[ h_t = [\vec{h}_t, \hat{h}_t] \in \mathbb{R}^{2h} \]
Can we use bidirectional RNNs in the following tasks?
(1) text classification, (2) sequence tagging, (3) text generation

(a) Yes, Yes, Yes
(b) Yes, No, Yes
(c) Yes, Yes, No
(d) No, Yes, No

The answer is (c).
Bidirectional RNNs

- Sequence tagging: Yes! (esp. important)
Bidirectional RNNs

- Sequence tagging: Yes!

- Text classification: Yes!
  - Common practice: concatenate the last hidden vectors in two directions or take the mean/max over all the hidden vectors

- Text generation: No. Because we can’t see the future to predict the next word.
A note on terminology

- Simple RNNs are also called vanilla RNNs

- Sometimes vanilla RNNs don’t work that well, so we need to use some advanced RNN variants such as LSTMs or GRUs (next lecture)

- In practice, we always use multi-layer RNNs

... together with fancy ingredients such as residual connections with self-attention, variational dropout..

Slide credit: Abigail See (with modifications)
• Advanced RNN variants: LSTMs vs GRUs

\[ h_t = f(h_{t-1}, x_t) \in \mathbb{R}^h \]

• PyTorch/final project

Good luck with the midterm!