Many slides adapted from Arun Mallya and Justin Johnson (and Stanford CS231n)
Outline

• Sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit (and how to train it)
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
• Recurrent network architectures
• Applications in (a bit) more detail
  • Language modeling
  • Image captioning
Sequential prediction example 1: Sentiment classification

• Goal: classify a text sequence (e.g., restaurant, movie or product review, Tweet) as having positive or negative sentiment
  
  • “The food was really good”
  • “The vacuum cleaner broke within two weeks”
  • “The movie had slow parts, but overall was worth watching”
Sequential prediction example 1: Sentiment classification

- Recurrent model:
Sequential prediction example 2: Text generation

- Sample from the distribution of a given text corpus (also known as language modeling)
Sequential prediction example 2: Text generation

- Sample from the distribution of a given text corpus – also known as *language modeling*
- Can be done one character or one word at a time:
Sequential prediction example 3: Image captioning

- A cat sitting on a suitcase on the floor
- A cat is sitting on a tree branch
- A dog is running in the grass with a frisbee
- A white teddy bear sitting in the grass
- Two people walking on the beach with surfboards
- A tennis player in action on the court
- Two giraffes standing in a grassy field
- A man riding a dirt bike on a dirt track

Source: J. Johnson
Captions generated using neuraltalk2
Sequential prediction example 3: Image captioning

“The”

“dog”

“is”

“hiding”

“STOP”

Classifier

Classifier

Classifier

Classifier

Classifier

CNN

“START”

“The”

“dog”

“is”

“hiding”
Example 4: Machine translation

https://translate.google.com/
Example 4: Machine translation

- Multiple input – multiple output (or sequence to sequence) scenario:

```
Correspondances
La
nature
Matches Nature is
```

![Diagram of machine translation process]

Input: "Correspondances" "La" "nature"
Output: "Matches" "Nature" "is"
Summary: Input-output scenarios

- **Single - Single**
  - Feedforward Network

- **Multiple - Single**
  - Sequence Classification

- **Single - Multiple**
  - Sequence generation, captioning

- **Multiple - Multiple**
  - Sequence generation, captioning

- **Multiple - Multiple**
  - Translation
Outline

• Sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
Recurrent unit
Recurrent unit

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

Output at time \( t \)

Hidden representation at time \( t \)

Input at time \( t \)

Classifier

Hidden layer

Recurrence:

\[
h_t = f_W(x_t, h_{t-1})
\]
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) = \tanh W \left( \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right) \]
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) = \tanh W \left( \begin{array}{c} x_t \\ h_{t-1} \end{array} \right) \]

\[ \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} = 2\sigma(2a) - 1 \]

[Image source]
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) = \tanh W \left( \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right) \]

\[ \frac{d}{da} \tanh(a) = 1 - \tanh^2(a) \]
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) \]
\[ = \tanh W \left( \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right) \]
\[ = \tanh (W_x x_t + W_h h_{t-1}) \]
RNN forward pass

\[ e_t = -\log(y_t(GT_t)) \]
\[ y_t = \text{softmax}(W_yh_t) \]
\[ h_t = \tanh W \left( \begin{array}{c} x_t \\ h_{t-1} \end{array} \right) \]

----- shared weights
RNN forward pass: Computation graph

\[ e_t = -\log(y_t(GT_t)) \]

\[ y_t = \text{softmax}(W_y h_t) \]

\[ h_t = \tanh(W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}) \]
Training: Backpropagation through time (BPTT)

- The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input.
- The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights.
Backpropagation through time

Forward through entire sequence to compute loss, then backward to compute gradient

Source: J. Johnson
Backpropagation through time

Problem: Takes a lot of memory for long sequences!

Source: J. Johnson
Training: Backpropagation through time (BPTT)

• The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input
• The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights
• In practice, truncated BPTT is used: run the RNN forward $k_1$ time steps, propagate backward for $k_2$ time steps

https://machinelearningmastery.com/gentle-introduction-backpropagation-time/
Truncated backpropagation through time

Run forward and backward through chunks of the sequence instead of whole sequence

Source: J. Johnson
Truncated backpropagation through time

Carry hidden states forward in time farther, but only backpropagate for some smaller number of steps

Source: J. Johnson
Truncated backpropagation through time

Source: J. Johnson
RNN backward pass

\[
h_t = \tanh(W_x x_t + W_h h_{t-1})
\]

\[
\frac{\partial e}{\partial W_h} = \frac{\partial e}{\partial h_t} \odot (1 - \tanh^2(W_x x_t + W_h h_{t-1})) h_{t-1}^T
\]

\[
\frac{\partial e}{\partial W_x} = \frac{\partial e}{\partial h_t} \odot (1 - \tanh^2(W_x x_t + W_h h_{t-1})) x_t^T
\]

\[
\frac{\partial e}{\partial h_{t-1}} = W_h^T (1 - \tanh^2(W_x x_t + W_h h_{t-1})) \odot \frac{\partial e}{\partial h_t}
\]
Vanishing and exploding gradients

\[
\frac{\partial e}{\partial h_{t-1}} = W_h^T (1 - \tanh^2(W_x x_t + W_h h_{t-1})) \odot \frac{\partial e}{\partial h_t}
\]

Computing gradient for \( h_0 \)

involves many multiplications by \( W_h^T \)

and rescalings between 0 and 1

Gradients will vanish if largest singular value of \( W_h \) is less than 1

and explode if it’s greater than 1
Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit (and how to train it)
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
Long short-term memory (LSTM)

- Add a *memory cell* that is not subject to matrix multiplication or squishing, thereby avoiding gradient decay

The LSTM cell

\[ g_t = \tanh W_g (x_t, h_{t-1}) \]
The LSTM cell

\[ g_t = \tanh W_g (x_t, h_{t-1}) \]

\[ c_t = c_{t-1} + g_t \]

\[ h_t = \tanh c_t \]
The LSTM cell

\[ g_t = \text{tanh}(W_g x_t + W_{h_{t-1}} h_{t-1}) \]
The LSTM cell

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

\[ g_t = \tanh \left( W_g (x_t + h_{t-1}) \right) \]

\[ c_t = c_{t-1} + i_t \odot g_t \]

\[ h_t = \left( c_t \odot \overline{c_t} \right) + h_{t-1} \odot \overline{h_{t-1}} \]
The LSTM cell

\[ i_t = \sigma(W_i x_t + b_i), \]

\[ g_t = \tanh(W_g (x_t + h_{t-1})), \]

\[ c_t = c_{t-1} + i_t \oplus g_t, \]

\[ h_t = o_t \oplus \tanh(c_t), \]

\[ o_t = \sigma(W_o (x_t + h_{t-1} + b_o)), \]
The LSTM cell

\[
\begin{align*}
    i_t &= \sigma \left( W_i \left( x_t + h_{t-1} \right) \right) \\
    f_t &= \sigma \left( W_f \left( x_t + h_{t-1} \right) \right) \\
    g_t &= \tanh \left( W_g \left( x_t + h_{t-1} \right) \right) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
    o_t &= \sigma \left( W_o \left( x_t + h_{t-1} \right) \right) \\
    h_t &= o_t \odot \tanh c_t
\end{align*}
\]
LSTM forward pass summary

\[
\begin{align*}
\mathbf{g}_t &= \text{tanh} \left( W_g \mathbf{x}_t \right) \\
\mathbf{i}_t &= \sigma \left( W_i \mathbf{x}_t \right) \\
\mathbf{f}_t &= \sigma \left( W_f \mathbf{h}_{t-1} \right) \\
\mathbf{o}_t &= \sigma \left( W_o \mathbf{h}_{t-1} \right)
\end{align*}
\]

\[
\begin{align*}
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \text{tanh} \left( \mathbf{c}_t \right)
\end{align*}
\]
LSTM backward pass

Gradient flow from $c_t$ to $c_{t-1}$ only involves back-propagating through addition and elementwise multiplication, not matrix multiplication or tanh.

For complete details: Illustrated LSTM Forward and Backward Pass
LSTM variant: Gated recurrent unit (GRU)

- Get rid of separate cell state
- Merge “forget” and “output” gates into “update” gate

\[
\begin{align*}
    r_t &= \sigma \left( W_r \left( x_t h_{t-1} \right) + b_t \right) \\
    h_t' &= \tanh \left( r_t \odot h_{t-1} \right) \\
    z_t &= \sigma \left( W_z \left( x_t h_{t-1} \right) + b_z \right) \\
    h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot h_t'
\end{align*}
\]

Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit (and how to train it)
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
• Recurrent network architectures
Recall: Input-output scenarios

- **Multiple - Single**: Sequence generation, captioning
- **Single - Multiple**: Sequence generation, captioning
- **Multiple - Multiple**: Sequence generation, captioning
- **Multiple - Multiple**: Translation
RNN architectures

- Most general configuration:
Multi-layer RNNs

• We can of course design RNNs with multiple hidden layers

\[
\begin{align*}
&y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\
&x_1 & x_2 & x_3 & x_4 & x_5 & x_6
\end{align*}
\]

• Anything goes: skip connections across layers, across time, …
Multi-layer RNNs

• We can of course design RNNs with multiple hidden layers

• Anything goes: skip connections across layers, across time, …
Multi-layer RNNs

- We can of course design RNNs with multiple hidden layers

- Anything goes: skip connections across layers, across time, …
Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction (common in speech recognition)
Google Neural Machine Translation (GNMT)

Outline

- Examples of sequential prediction tasks
- Common recurrent units
  - Vanilla RNN unit
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)
- Recurrent network architectures
- Applications in (a bit) more detail
  - Language modeling
  - Image captioning
  - Machine translation
Language modeling: Character RNN

Output symbol $y_i$

Output layer (linear + softmax)

Hidden state $h_i$

One-hot encoding $x_i$

Input symbol

$$p(y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1})$$

$$\approx \prod_{i=1}^{n} P_W(y_i | h_i)$$

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Language modeling: Character RNN

100th iteration

tynd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoaee rraranbyne 'nhthnee e plia tklrgd t o idoe ns,smtt h ne etie h,hexrtrs nigite,aoaenns lng

train more

300th iteration

"Tmont thithey" fomesserliund Keushey. Thom here sheulke, ammerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome coaniogenc Phe lism thond hon at. MeDiromotion in ther thize."

train more

700th iteration

Aftair fall unsuch that the hall for Prince Velzonshi's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

train more

2000th iteration

"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had ofthen him. Pierre aking his soul came to the packs and drove up his father-in-law women.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Searching for interpretable hidden units

“...you mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties,” warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Searching for interpretable hidden units

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Searching for interpretable hidden units

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

if statement cell
Searching for interpretable hidden units

Searching for interpretable hidden units

Searching for interpretable hidden units

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Recurrent models: Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
• Recurrent network architectures
• Applications in (a bit) more detail
  • Language modeling
  • Image captioning
Image caption generation

Training time

- Maximize likelihood of reference captions

O. Vinyals, A. Toshev, S. Bengio, D. Erhan, **Show and Tell: A Neural Image Caption Generator**, CVPR 2015
Image caption generation: Test time

- How do we produce a caption given a test image?
  - How about always choosing the highest-likelihood word?
Image caption generation: Beam search

- Maintain $k$ (*beam width*) top-scoring candidate sentences according to sum of per-word log-likelihoods (or some other score)
- At each step, generate all their successors and keep the best $k$
Image caption generation: Beam search

The dog is hiding on the
A yellow cat is sitting on
One dog is sitting on the

Classifier
Classifier
Classifier
Classifier
Classifier

"START"

CNN

Image: A scene with a dog and a yellow cat sitting on a surface.
Image caption generation: Example outputs
How to evaluate image captioning?

Reference sentences (written by human annotators):

• “A dog hides underneath a bed with its face peeking out of the bed skirt”
• “The small white dog is peeking out from under the bed”
• “A dog is peeking its head out from underneath a bed skirt”
• “A dog peeking out from under a bed”
• “A dog that is under a bed on the floor”

Generated sentence:

• “A dog is hiding”
BLEU: Bilingual Evaluation Understudy

- **N-gram precision**: count the number of n-gram matches between candidate and reference translation, divide by total number of n-grams in candidate translation
  - Clip counts by the maximum number of times an n-gram occurs in any reference translation
  - Multiply by *brevity penalty* to penalize short translations

- Most commonly used measure for image captioning and machine translation despite multiple **shortcomings**

---

Original (French): J’ai mangé la pomme.

Reference translation: I ate the apple.

Based on BLEU, these are all “equally bad” output sentences.

I consumed the apple.

I ate an apple.

I ate the potato.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIDEr-D</th>
<th>Meteor</th>
<th>ROUGE-L</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>m-RNN (Baidu/ UCLA)</td>
<td>0.886</td>
<td>0.238</td>
<td>0.524</td>
<td>0.72</td>
<td>0.553</td>
<td>0.41</td>
<td>0.302</td>
</tr>
<tr>
<td>m-RNN</td>
<td>0.851</td>
<td>0.263</td>
<td>0.508</td>
<td>0.698</td>
<td>0.542</td>
<td>0.392</td>
<td>0.299</td>
</tr>
<tr>
<td>MSR Captive</td>
<td>0.828</td>
<td>0.243</td>
<td>0.518</td>
<td>0.693</td>
<td>0.533</td>
<td>0.368</td>
<td>0.282</td>
</tr>
<tr>
<td>Google</td>
<td>0.843</td>
<td>0.245</td>
<td>0.514</td>
<td>0.699</td>
<td>0.545</td>
<td>0.387</td>
<td>0.299</td>
</tr>
<tr>
<td>Berkeley LR</td>
<td>0.782</td>
<td>0.205</td>
<td>0.482</td>
<td>0.654</td>
<td>0.481</td>
<td>0.345</td>
<td>0.249</td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td>0.805</td>
<td>0.239</td>
<td>0.512</td>
<td>0.688</td>
<td>0.515</td>
<td>0.372</td>
<td>0.268</td>
</tr>
<tr>
<td>MSR</td>
<td>0.837</td>
<td>0.247</td>
<td>0.516</td>
<td>0.689</td>
<td>0.515</td>
<td>0.372</td>
<td>0.268</td>
</tr>
<tr>
<td>Human[12]</td>
<td>0.854</td>
<td>0.219</td>
<td>0.499</td>
<td>0.666</td>
<td>0.498</td>
<td>0.362</td>
<td>0.26</td>
</tr>
</tbody>
</table>

**Metrics**

- **CIDEr-D**: CIDEr: Consensus-based Image Description Evaluation
- **METEOR**: Meteor Universal: Language Specific Translation Evaluation for Any Target Language
- **ROUGE-L**: ROUGE: A Package for Automatic Evaluation of Summaries
- **BLEU**: BLEU: a Method for Automatic Evaluation of Machine Translation

http://mscoco.org/dataset/#captions-leaderboard
### 2015 Captioning Challenge

Last update: June 8, 2015. Visit CodaLab for the latest results.

<table>
<thead>
<tr>
<th>Method</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human[^5]</td>
<td>0.638</td>
<td>0.675</td>
<td>4.836</td>
<td>3.428</td>
<td>0.352</td>
</tr>
<tr>
<td>Google[^4]</td>
<td>0.503</td>
<td>0.247</td>
<td>4.087</td>
<td>3.746</td>
<td>0.203</td>
</tr>
<tr>
<td>Montreal[^2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSR Cap[^8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average correctness of the captions on a scale 1-5 (incorrect - correct).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average amount of detail of the captions on a scale 1-5 (lack of details - very detailed).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage of captions that are similar to human description.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nearest Neighbor[^11]</td>
<td>0.216</td>
<td>0.255</td>
<td>3.801</td>
<td>2.716</td>
<td>0.196</td>
</tr>
<tr>
<td>PicSOM[^13]</td>
<td>0.202</td>
<td>0.250</td>
<td>3.965</td>
<td>2.552</td>
<td>0.182</td>
</tr>
<tr>
<td>Brno University[^3]</td>
<td>0.194</td>
<td>0.213</td>
<td>3.079</td>
<td>3.482</td>
<td>0.154</td>
</tr>
<tr>
<td>m-RNN (Baidu/ UCLA)[^16]</td>
<td>0.190</td>
<td>0.241</td>
<td>3.831</td>
<td>2.548</td>
<td>0.195</td>
</tr>
<tr>
<td>MIL[^6]</td>
<td>0.168</td>
<td>0.197</td>
<td>3.349</td>
<td>2.915</td>
<td>0.159</td>
</tr>
<tr>
<td>MLBL[^7]</td>
<td>0.167</td>
<td>0.196</td>
<td>3.659</td>
<td>2.420</td>
<td>0.156</td>
</tr>
</tbody>
</table>