CSE 40171: Artificial Intelligence

Artificial Neural Networks with Anatomical Fidelity: Recurrence in Artificial and Biological Networks
Homework #6 has been released
It is due at 11:59PM on 11/22
Project Updates are Due on 11/25 at 11:59PM

(See Course Website for Instructions)
Let’s turn our attention back to artificial neural networks for a moment...
How do we parse audio data?

https://www.youtube.com/watch?v=-zVgWpVXb64
Arms, and the man I sing, who, forc'd by fate,
And haughty Juno's unrelenting hate,
Expell'd and exil'd, left the Trojan shore.
Long labors, both by sea and land, he bore,
And in the doubtful war, before he won
The Latian realm, and built the destin'd town;
His banish'd gods restor'd to rites divine,
And settled sure succession in his line,
From whence the race of Alban fathers come,
And the long glories of majestic Rome.
O Muse! the causes and the crimes relate;
What goddess was provok'd, and whence her hate;
For what offense the Queen of Heav'n began
To persecute so brave, so just a man;
Involv'd his anxious life in endless cares,
Expos'd to wants, and hurried into wars!
Recurrent Networks

output layer
Architecture of a Traditional RNN
At each timestep $t$:

**Activation $a^{<t>}$:**

$$a^{<t>} = g_1(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$$

**Output $y^{<t>}$:**

$$y^{<t>} = g_2(W_{ya} a^{<t>} + b_y)$$
At each timestep $t$: 

$$y^{<t>} = \sigma(W_{aa} a^{<t-1>} + b_y + g_2)$$

$$a^t = \sigma(W_{ax} x^{<t>} + W_{ya} y^{<t>} + b_a + g_1)$$
Example Application: Music Generation

One-to-Many, $T_x = 1$, $T_y > 1$
Example Application: Named Entity Recognition

Many-to-Many, $T_x = T_y$

Image Credit: Afshine Amidi and Shervine Amidi Stanford CS 230
Training a Recurrent Net

Loss Function:
\[ \mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>}) \]

Backprop:
\[ \frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \left[ \frac{\partial \mathcal{L}^{(T)}}{\partial W} \right]_{(t)} \]

Weight Matrix
Advantages of RNNs

• Ability to process time series data of any length
• Model size does not increase with input size
• Takes into account history when processing
• Weights are shared across time
Disadvantages of RNNs

- Computation is slow
- Difficulty accessing information from many timesteps in the past
- Current state cannot consider any input from the future
Vanishing / Exploding Gradient

For many years, RNNs were interesting theoretically, but not practical to train

Reason: it is difficult to capture long term dependencies because of a multiplicative gradient that can be exponentially increasing / decreasing with respect to the number of layers in the network

Credit: Afshine Amidi and Shervine Amidi Stanford CS 230
RNN Strategy 1: Gradient Clipping

Cap the maximum value of the gradient:

Image Credit: Afshine Amidi and Shervine Amidi Stanford CS 230
RNN Strategy 2: Types of Gates

Specific gates that have a well-defined purpose can address the vanishing gradient problem.

General Form:
\[
\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b)
\]

<table>
<thead>
<tr>
<th>Type of gate</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update gate (\Gamma_u)</td>
<td>How much past should matter now?</td>
</tr>
<tr>
<td>Relevance gate (\Gamma_r)</td>
<td>Drop previous information?</td>
</tr>
<tr>
<td>Forget gate (\Gamma_f)</td>
<td>Erase a cell or not?</td>
</tr>
<tr>
<td>Output gate (\Gamma_o)</td>
<td>How much to reveal of a cell?</td>
</tr>
</tbody>
</table>

Image Credit: Afshine Amidi and Shervine Amidi Stanford CS 230
Standard Arch. That Works: LSTM

<table>
<thead>
<tr>
<th>Characterization</th>
<th>Long Short-Term Memory (LSTM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{c}^{&lt;t&gt;}$</td>
<td>$\tanh(W_c[\Gamma_r \star a^{&lt;t-1&gt;}, x^{&lt;t&gt;}]) + b_c$</td>
</tr>
<tr>
<td>$c^{&lt;t&gt;}$</td>
<td>$\Gamma_u \star \tilde{c}^{&lt;t&gt;} + \Gamma_f \star c^{&lt;t-1&gt;}$</td>
</tr>
<tr>
<td>$a^{&lt;t&gt;}$</td>
<td>$\Gamma_o \star c^{&lt;t&gt;}$</td>
</tr>
</tbody>
</table>

Dependencies

Image Credit: Afshine Amidi and Shervine Amidi Stanford CS 230
Recurrent Models of Brain Function
Two models of recurrence

Linear recurrent model:

$$\tau_r \frac{dv}{dt} = -v + F(h + M \cdot v)$$

Non-linear recurrent model:

$$F(h + M \cdot r) = [h + M \cdot r - \gamma]_+$$

Adds rectification
Tuning Curves

Hubel and Wiesel 1968
Linear Amplification

A

B

C

D

Image Credit: Dayan and Abbot 2001
Non-Linear Amplification

A

B

C

D

Image Credit: Dayan and Abbott 2001
Recurrent Model of Simple Cells in Primary Visual Cortex

Image Credit: http://www.lifesci.sussex.ac.uk/home/George_Mather/Linked%20Pages/Physiol/Cortex.html
Recurrent Model of Simple Cells in Primary Visual Cortex

Ben-Yishai, Bar-Or and Sompolinsky 2005

\[ \tau_r \frac{dv(\theta)}{dt} = -v(\theta) + \left[ h(\theta) + \int_{-\pi/2}^{\pi/2} \frac{d\theta'}{\pi} \left( -\lambda_0 + \lambda_1 \cos(2(\theta - \theta')) \right)v(\theta') \right] + \]

- Firing rate of neuron
- Orientation
- Clamped orientation angles
- Image contrast
- Overall amplitude

\[ h(\theta) = Ac(1 - \epsilon + \epsilon \cos(2\theta)) \]
Recurrent Model of Simple Cells in Primary Visual Cortex

Image Credit: Dayan and Abbott 2001
A Recurrent Model of Complex Cells in Primary Visual Cortex

Image Credit: http://www.lifesci.sussex.ac.uk/home/George_Mather/Linked%20Pages/Physiol/Cortex.html
A Recurrent Model of Complex Cells in Primary Visual Cortex

Chance, Nelson and Abbott 1999

Weight Function:

\[ M(\phi - \phi') = \frac{\lambda_1}{2\pi \rho_\phi} \]

control recurrence

Firing Rates Determined By:

\[ \tau_r \frac{dv(\phi)}{dt} = -v(\phi) + \left[ h(\phi) + \frac{\lambda_1}{2\pi} \int_{-\pi}^{\pi} d\phi' v(\phi') \right]_+ \]

feed-forward input
A Recurrent Model of Complex Cells in Primary Visual Cortex

A

B

Image Credit: Dayan and Abbott 2001