Recurrent Neural Networks

These slides draw on material from Geoff Hinton's course notes.
MODELING SEQUENCE DATA

THIS PART DRAWS HEAVILY (I.E. VERBATIM) FROM THE COURSE NOTES (CSC2535) OF GEOFF HINTON (2013)

• Memoryless models for sequences:
  ‣ Autoregressive models: Predict the next input in a sequence from a fixed number of previous inputs using “delay taps”.
  ‣ Feed-forward neural networks: Generalize autoregressive models by using non-linear hidden layers.
BEYOND MEMORYLESS MODELS

• If the model could possess a dynamic hidden state:
  ‣ Can store long term information.

• Dynamic Bayes networks (DBNs) can possess internal dynamic state
  ‣ Example 1: Linear Dynamic System with Gaussian noise model (Kalman Filter)
  ‣ Example 2: Discrete state, arbitrary observation type (Hidden Markov Model)
    - State is not observed, must be inferred.
    - Represent probability across N states with N numbers.
    - Efficient algorithms exist for HMM inference.
LIMITATIONS OF HMMS

• Consider modelling sentence production with an HMM.

• Generation procedure:
  ‣ At each time step $t$, sample state $s_t$ given state $s_{t-1}$.
  ‣ Everything important about the past outputs (output 0, ..., output $t-1$) is must be summarized in this choice of state.
  ‣ So with $N$ states, it can only remember $\log(N)$ bits of the past.

• Consider trying to generate the second half of the sentence with the first have already generated.
  ‣ Syntax needs to fit (number and tense agreement).
  ‣ Semantics need to fit.

• These aspects combined could be say 100 bits that need to be conveyed from the first half to the second. $2^{100}$ is big!
RECURRENT NEURAL NETWORKS

• RNNs are very powerful, because they combine two properties:
  ‣ **Distributed hidden state**: can efficiently store a lot of information about the past.
    - Note: real valued activations not 1-of-N
  ‣ **Non-linear dynamics**: can update their hidden state in complicated ways.
RECURRENT NEURAL NETWORKS

• Compare to: Feed Forward Neural Networks:
  ‣ Information is propagated from the inputs to the outputs
  ‣ No notion of “time” necessary
RECURRENT NEURAL NETWORKS

• RNNs can have arbitrary topology.
  ‣ no fixed direction of information flow

• Delays associated with specific connections
  ‣ Every directed cycle must contain a delay.

• Possesses an **internal dynamic state**.
RECURRENT NEURAL NETWORKS

• With directed cycles the network can do more than a feed-forward network, it has dynamics.
  ‣ Oscillate: Useful for pattern generation, eg. for walking, swimming, etc.
  ‣ Settle to a point attractor: Can represent semantics, like the meanings of words
  ‣ Can behave chaotically: Can think of this as skipping between stable oscillatory patterns. Potentially useful for memory encoding and retrieval.

• With internal state, the network can ‘remember’ things for a long time.
  ‣ Can decide to ignore input for a while if it wants to.
  ‣ BUT it is very hard to train an RNN to store information that’s not needed for a long time.
  ‣ In principal, the internal state can carry information about a potentially unbounded number of previous inputs.
RECURRENT NEURAL NETWORKS

• How to make sense of recurrent connections.
  ‣ Assume a unit time delay with each connection
  ‣ We can **unroll the RNN in time** to get a standard feedforward net that reuse the same weights at every layer.

RNN:

Unroll
**TRAINING RNNs**

- RNNs are usually trained using the backpropagation through time algorithm.

- **Backpropagation through time**: standard backprop through the unrolled RNN with the constraint that weights are shared.
BACKPROPAGATION THROUGH TIME

• **Backpropagation with weight constraints**: compute the gradients contributions for each weight at each time step.

  ‣ each time step = each layer in the unrolled feed-forward network.
  ‣ Let $E$ be the error and $w_1^{(i)}$ be weight $w_1$ at time $= i$.

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial w_1^{(3)}} + \frac{\partial E}{\partial w_1^{(2)}} + \frac{\partial E}{\partial w_1^{(1)}} + \frac{\partial E}{\partial w_1^{(0)}}$$
RNN INITIALIZATION

• Need to specify the initial activations of the hidden units and output units.

• Could initialize them to a fixed value (such as 0.5).

• Better to treat the initial state as learned parameters.

• Learn these the same way we do with other model parameters:
  ‣ Start off with random guesses of the initial state values.
  ‣ Backpropagate the prediction error through time all the way to the initial state values and compute the gradient of the error with respect to these initial state parameters.
  ‣ Update these parameters by following the negative gradient.
We can specify inputs in several ways:

- Specify the initial states of all units.
- Specify the initial states of a subset of units.
- Specify the desired activations for a subset of units for multiple time steps.
  - useful when modelling a sequential task.
INPUT SIGNALS FOR RNNs

- We can specify inputs in several ways:
  - Specify the initial states of all units.
  - Specify the initial states of a subset of units.
  - Specify the desired activations for a subset of units for multiple time steps.
    - useful when modelling a sequential task.
• We can specify the target for training an RNN in a few different (task dependent) ways:
  ‣ Specify the desired final activations of all units.
  ‣ Specify the desired activations for all units for multiple time steps.
    - useful when learning attractors.
  ‣ Specify the desired activity of a subset of units.
    - Other units are input or hidden units.
We can specify the target for training an RNN in a few different (task dependent) ways:

- Specify the desired final activations of all units.
- Specify the desired activations for all units for multiple time steps.
  - useful when learning attractors.
- Specify the desired activity of a subset of units.
  - Other units are input or hidden units.
EXAMPLE: BINARY ADDITION

• Can we train a feed-forward net to do binary addition?

• **Problems:**
  
  ‣ We must decide in advance the maximum number of digits in each number.
  
  ‣ The processing applied to the beginning of a long number does not generalize to the end of the long number because it uses different weights.

• Feed-forward nets do not generalize well on binary addition.
BINARY ADDITION AS A DYNAMIC PROCESS

- Moves from right to left over two input numbers.
- Finite state atomaton
  - Decides what transition to make by looking at the next column.
  - It prints after making the transition.
RNN FOR BINARY ADDITION

• Network has two input units and one output unit.
  ‣ Each input corresponds to a digit from a distinct binary number.

• Desired output at each time step is the output for the column that was provided as input two time steps ago.
  ‣ Takes one time step to update the hidden units based on the two input digits.
  ‣ Takes another time step for the hidden units to cause the output.
RNN FOR BINARY ADDITION

• RNN solution has 3 hidden units have all possible interconnections in all directions:
  ‣ Allows hidden activity pattern at one time step to vote for the hidden activity pattern at the next time step.

• Input units have feedforward connections that allow them to vote for the next hidden activity pattern.
RNN FOR BINARY ADDITION

What the network learns:

• Learns 4 distinct patterns of activity across the 3 hidden units.

• These patterns correspond to the nodes in the finite state automaton.
  ‣ Don't confuse hidden units with states in the finite state automaton.
  ‣ The automaton is restricted to be in exactly one state at each time. The hidden units are restricted to have exactly one vector of activity at each time.

• An RNN can emulate a finite state automaton, but it is exponentially more powerful.
  ‣ With N hidden neurons it has $2^N$ possible binary activity vectors in the hidden units.
  ‣ Important when the input stream has several separate things going on at once.
LOTS OF PROMISE, BUT ...

- Recurrent neural networks are an extremely powerful class of model.
- Unfortunately, it is very difficult to learn long-term dependencies in a recurrent network.
  - Exploding and vanishing gradients in backpropagation through time is a huge issue that remains somewhat unresolved.
EXPLODING / VANISHING GRADIENT

• Forward pass made stable by saturating nonlinearities, i.e. sigmoid activations
  ‣ Forward pass determines the slopes of the linear function used for backpropagation.

• Backprop gradient is unbounded.
  ‣ Backward pass is completely linear.
    ‣ If you double the error at the final layer, all the error derivatives will be double.
  ‣ Can either shrink (stable) or explode (unstable).
    ‣ If the weights are small, gradients shrink exponentially.
    ‣ If the weights are large, gradients grow exponentially.
• Understanding exploding / vanishing gradients:
  ‣ If we start a trajectory within an attractor, small changes in where we start make no difference to where we end up.
  ‣ But if we start close to the boundary, tiny changes can result in huge difference.
SOLUTIONS FOR TRAINING RNNS

• Sophisticated optimization:
  ‣ Deal with the vanishing gradients problem by using a fancy optimizer that can detect directions with a tiny gradient but even smaller curvature. (eg. Hessian-Free optimization).

• Echo state networks:
  ‣ Initialize the input ➔ hidden and hidden ➔ hidden connections very carefully so that the hidden state has a huge reservoir of weakly coupled oscillators which can be selectively driven by the input.
  ‣ Only learns the hidden ➔ output connections.

• Long short term memory (LSTM):
  ‣ Make the RNN out of modules that are designed to remember values for a long time.
  ‣ The LSTM has recently become the RNN model of choice for many applications.
LONG SHORT TERM MEMORY NETWORKS

• Hochreiter & Schmidhuber (1997) solved the problem of getting an RMM to remember things for a long time (like hundreds of time steps).

• They designed a memory cell using logistic and linear units with gating (or multiplicative) interactions:
  
  ‣ **Write (input) gate**: (on) information can get into the cell.
  
  ‣ **Keep (forget) gate**: (on) information stays in the cell.
  
  ‣ **Read (output) gate**: (on) information can be read by the cell.
LONG SHORT-TERM MEMORY NETWORKS

**LSTM cell equations:**

- $i_t = \sigma (W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$
- $f_t = \sigma (W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$
- $c_t = f_tC_{t-1} + i_t \tanh (W_{xc}x_t + W_{hc}h_{t-1} + b_c)$
- $o_t = \sigma (W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$
- $h_t = o_t \tanh (c_t)$
LONG SHORT-TERM MEMORY NETWORKS

• LSTM has been around for a while (>15 years)
  ‣ Long term memory properties have also been known for a while, but ...

• Recently, LSTM has emerged as the recurrent neural network of choice with state-of-the-art performance in:
  ‣ Speech recognition: (among others) Graves, Mohamed & Hinton (2013) showed state-of-the-art performance on TIMIT phoneme recognition task.
• When we last discussed speech recognition we saw that deep learning methods (i.e. neural networks) are now the dominate paradigm for the acoustic model.

• The dream is a full deep learning system complete with end-to-end training of the whole pipeline.

• For this we can to replace the HMM Language model with a model of word sequences: we can use LSTM RNN!
When we last discussed speech recognition we saw that deep learning methods (i.e. neural networks) are now the dominate paradigm for the acoustic model.

The dream is a full deep learning system complete with end-to-end training of the whole pipeline.

For this we can to replace the HMM Language model with a model of word sequences: we can use **LSTM RNN!**
SPEECH RECOGNITION

• When we last discussed speech recognition we saw that deep learning methods (i.e. neural networks) are now the dominate paradigm for the acoustic model.

• The dream is a full deep learning system complete with end-to-end training of the whole pipeline.
When we last discussed speech recognition we saw that deep learning methods (i.e. neural networks) are now the dominate paradigm for the acoustic model.

The dream is a full deep learning system complete with end-to-end training of the whole pipeline.

For this we can to replace the HMM Language model with a model of word sequences: we can use **LSTM RNN**!
• Graves, Mohamed & Hinton (2013) used a bidirectional LSTM to incorporate both previous and future contextual information to predict the sequence of phonemes from the sequence of utterances.
LSTM FOR SPEECH RECOGNITION

- Graves, Mohamed & Hinton (2013) used a bidirectional LSTM to incorporate both previous and future contextual information to predict the sequence of phonemes from the sequence of utterances.

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>WEIGHTS</th>
<th>EPOCHS</th>
<th>PER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTC-3L-500H-tanh</td>
<td>3.7M</td>
<td>107</td>
<td>37.6%</td>
</tr>
<tr>
<td>CTC-1L-250H</td>
<td>0.8M</td>
<td>82</td>
<td>23.9%</td>
</tr>
<tr>
<td>CTC-1L-622H</td>
<td>3.8M</td>
<td>87</td>
<td>23.0%</td>
</tr>
<tr>
<td>CTC-2L-250H</td>
<td>2.3M</td>
<td>55</td>
<td>21.0%</td>
</tr>
<tr>
<td>CTC-3L-421H-uni</td>
<td>3.8M</td>
<td>115</td>
<td>19.6%</td>
</tr>
<tr>
<td>CTC-3L-250H</td>
<td>3.8M</td>
<td>124</td>
<td>18.6%</td>
</tr>
<tr>
<td>CTC-5L-250H</td>
<td>6.8M</td>
<td>150</td>
<td>18.4%</td>
</tr>
<tr>
<td>TRANS-3L-250H</td>
<td>4.3M</td>
<td>112</td>
<td>18.3%</td>
</tr>
<tr>
<td>PRETRANS-3L-250H</td>
<td>4.3M</td>
<td>144</td>
<td>17.7%</td>
</tr>
</tbody>
</table>
• Input sensitivity of a deep CTC RNN. The heatmap (top) shows the derivatives of the ‘ah’ and ‘p’ outputs printed in red with respect to the filterbank inputs (bottom).

• Note that the sensitivity extends to surrounding segments; this may be because CTC (which lacks an explicit language model) attempts to learn linguistic dependencies from the acoustic data.
Deep Learning for Speech Recognition

THIS SECTION DRAWS ON THE TUTORIALS OF VINCENT VANHOUCKE (ICML 2013) AND LI DENG (ICML 2014)
SPEECH RECOGNITION

• **GOAL**: Transform speech (audio) input to text output
  ‣ This is a structured output problem!

• Speech recognition task combines an acoustic model with a language model:

\[
\text{argmax}_W p(W \mid O) = \text{argmax}_W p(O \mid W) p(W)
\]

Audio waveform \hspace{2cm} Language model

Word sequence \hspace{2cm} Acoustic model
SPEECH RECOGNITION

• 1990s-2009: The traditional pipeline

  • Feature Extraction:
    ‣ Mel-frequency cepstral coefficients (MFCCs), Perceptual linear prediction (PLPs), bottleneck features

  • Acoustic Model:
    ‣ Gaussian Mixture models (diagonal covariances)

  • Language Model:
    ‣ Hidden Markov models
SPEECH RECOGNITION
SPEECH RECOGNITION

• 1990s-2009: The traditional pipeline

DSP → Feature Extraction → Acoustic Model → Language Model
SPEECH RECOGNITION

- **1990s-2009**: The traditional pipeline
  - DSP → Feature Extraction → Acoustic Model → Language Model

- **2009-2014**: The deep neural network pipeline.
  - DSP → Deep Neural Network Acoustic Model → Language Model
SPEECH RECOGNITION

• 1990s-2009: The traditional pipeline

  DSP → Feature Extraction → Acoustic Model → Language Model

• 2009-2014: The deep neural network pipeline.

  DSP → Deep Neural Network Acoustic Model → Language Model

• >2014 Full DNN end-to-end system?
DL FOR SPEECH RECOGNITION

NIST Evaluations of Automatic Speech Recognition

image from Li Deng, 2014
no improvement for 10+ years of research.
no improvement for 10+ years of research.

Use of deep learning (DBNs) quickly reduced the error from ~23% to <15%
no improvement for 10+ years of research.

Use of deep learning (DBNs) quickly reduced the error from ~23% to <15%

Early results used DBNs with greedy layer-wise unsupervised pretraining.
no improvement for 10+ years of research.

Use of deep learning (DBNs) quickly reduced the error from ~23% to <15%

Early results used DBNs with greedy layer-wise unsupervised pretraining.

Recently, unsupervised pretraining appears to be irrelevant.
An industry-wide revolution:

**Microsoft:** Li Deng, Frank Seide, Dong Yu, ...

**IBM:** Brian Kingsbury, Tara Sainath, ...

**Google:** Vincent Vanhoucke, Andrew Senior, Georg Heigold, ...

**U of Toronto:** Goeff Hinton, George Dahl, Abdel-Rahman Mohamed, Navdeep Jaitly
To combine with language model, need a model of $p(\text{frame} \mid \text{state})$.
DEEP LEARNING RECIPE: HYBRID SYSTEM

- To combine with language model, need a model of \( p(\text{frame} \mid \text{state}) \)

- Can exploit Baye’s rule: 
  \[
  p(\text{frame} \mid \text{state}) \propto \frac{p(\text{state} \mid \text{frame})}{p(\text{state})}
  \]
DEEP LEARNING RECIPE: HYBRID SYSTEM

- To combine with language model, need a model of $p(\text{frame} \mid \text{state})$
- Can exploit Baye’s rule: $p(\text{frame} \mid \text{state}) \propto \frac{p(\text{state} \mid \text{frame})}{p(\text{state})}$
- Plug any discriminative classifier into $p(\text{state} \mid \text{frame})$
To combine with language model, need a model of $p(\text{frame} \mid \text{state})$

Can exploit Baye’s rule: $p(\text{frame} \mid \text{state}) \propto \frac{p(\text{state} \mid \text{frame})}{p(\text{state})}$

Plug any discriminative classifier into $p(\text{state} \mid \text{frame})$

**Idea**: Put a Deep Neural Network here
• **Input:** Simple scaled spectrogram features

• **Model details:**
  ‣ 4-10 fully-connected layers, 1000-3000 units / layer.
  ‣ Linear rectified activations / maxout, and softmax output.

• **Frame-by-frame training of DNN:**
  ‣ With minibatch SGD. Currently the **dominate training paradigm**.
  ‣ Trained simply to classify state values from frame-level audio data.

• **Sequence based discriminative training:**
  ‣ Define a smooth loss that takes into account work, phoneme or state-level errors. (Kingsbury, 2009)
SPEAKER ADAPTATION

  
  ‣ Improvements in generalization due to DL seems to overwhelm gains due to speaker adaptation

  ‣ Adaptation still works well with small networks but vanish as the networks grow.

<table>
<thead>
<tr>
<th>Number of parameters</th>
<th>Relative WER improvement</th>
<th>Sources (ICASSP 2013)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10M</td>
<td>32%, 10%</td>
<td>H. Liao, O. Abdel-Hamid</td>
</tr>
<tr>
<td>31M</td>
<td>15%</td>
<td>D. Yu</td>
</tr>
<tr>
<td>45M</td>
<td>7%</td>
<td>D. Yu</td>
</tr>
<tr>
<td>60M</td>
<td>5%</td>
<td>H. Liao</td>
</tr>
</tbody>
</table>
MULTILINGUAL SPEECH RECOGNITION

• Transfer learning and multitask learning work very well.

**STANDARD APPROACH**

- Language 1 text
- Language 1 audio
- Language 4 text
- Language 2 audio

*Input Layer*: window of acoustic feature frames
MULTILINGUAL SPEECH RECOGNITION

• Transfer learning and multitask learning work very well.

MULTILINGUAL SPEECH RECOGNITION

• Transfer learning and multitask learning work very well.

Chinese speech recognition character-error-rate when combined with transfer from European languages.

Target language: zh-CN
CONVOLUTIONAL NETS FOR SPEECH


  ‣ **Insight exploited:** on a Mel scale, a pitch change is mostly a shift in the frequency axis.

  ‣ Convolutions in frequency seem like a natural way to represent this invariance.

  ‣ Report significant improvement over basic DNN!
SPEECH RECOGNITION

• 1990s-2009: The traditional pipeline

• 2009-2014: The deep neural network pipeline.

• >2014 Full DNN end-to-end system?