Today’s Lecture

so far
- we discussed RNNs as encoder and decoder
- we discussed some architecture variants:
  - RNN vs. GRU vs. LSTM
  - attention mechanisms

today
- some important components of neural MT architectures:
  - dropout
  - layer normalization
  - deep networks
- non-recurrent architectures:
  - convolutional networks
  - self-attentional networks

General Architecture Variants

NMT with Convolutional Neural Networks

NMT with Self-Attention

Dropout

wacky idea: randomly set hidden states to 0 during training
motivation: prevent "co-adaptation" of hidden units
→ better generalization, less overfitting

(a) Standard Neural Net
(b) After applying dropout

[Srivastava et al., 2014]
Dropout

(a) Standard Neural Net
(b) After applying dropout.

- implementation:
  - for training, multiply layer with "dropout mask"
  - randomly sample new mask for each layer and training example
  - hyperparameter $p$: probability that state is retained
    (some tools use $p$ as probability that state is dropped)
  - at test time, don’t apply dropout,
    but re-scale layer with $p$ to ensure expected output is the same
  - (you can also re-scale by $\frac{1}{p}$ at training time instead)

Dropout and RNNs

for recurrent connections, applying dropout at every time step blocks information flow

solution 1: only apply dropout to feedforward connections

Figure 2: Regularized multilayer RNN. The dashed arrows indicate connections where dropout is applied, and the solid lines indicate connections where dropout is not applied.

Layer Normalization

- if input distribution to NN layer changes, parameters need to adapt to this covariate shift
- especially bad: RNN state grows/shrinks as we go through sequence
- normalization of layers reduces shift, and improves training stability
- re-center and re-scale each layer $a$ (with $H$ units)
- two bias parameters, $g$ and $b$, restore original representation power

$$ \mu = \frac{1}{H} \sum_{i=1}^{H} a_i $$
$$ \sigma = \sqrt{\frac{1}{H} \sum_{i=1}^{H} (a_i - \mu)^2 } $$
$$ h = \left( \frac{g}{\sigma} \odot (a - \mu) + b \right) $$
Deep Networks

Increasing model depth often increases model performance. An example of a stack RNN is shown below:

\[
\begin{align*}
    h_{t,1} &= g(U_1 h_{t-1,1} + W_1 x_t) \\
    h_{t,2} &= g(U_2 h_{t-1,2} + W_2 h_{t,1}) \\
    h_{t,3} &= g(U_3 h_{t-1,3} + W_3 h_{t,2}) \\
\end{align*}
\]

Deep Networks

Often necessary to combat vanishing gradient: residual connections between layers:

\[
\begin{align*}
    h_{t,1} &= g(U_1 h_{t-1,1} + W_1 x_t) \\
    h_{t,2} &= g(U_2 h_{t-1,2} + W_2 h_{t,1}) + h_{t,1} \\
    h_{t,3} &= g(U_3 h_{t-1,3} + W_3 h_{t,2}) + h_{t,2} \\
\end{align*}
\]

Convolutional Networks

Core idea: rather than using fully connected matrix between two layers, repeatedly compute dot product with small filter (or kernel).

2d convolution with 3x3 kernel

General Architecture Variants

NMT with Convolutional Neural Networks

NMT with Self-Attention
Convolutional Neural Machine Translation

Convolutional encoder actually predates RNN encoder

Convolutional Neural Machine Translation with Attention

- to keep representation size constant, use padding
  \( \rightarrow \) similar variable-size representation as RNN encoder
- kernel can be applied to all windows in parallel
Self-Attention

Convolution Self-Attention

- use your favourite attention mechanism to obtain input context
- in decoder, information from future tokens is masked during training
- effective context window depends on network depth and kernel size

Equation 13.78 on page 51.

The attention mechanism is essentially unchanged from the canonical neural translation model. Recall that it is based on an association between the word representations computed by the encoder $h_j$ and the previous state of the decoder $s_{i-1}$ (refer back to Equation 13.78 on page 51).

same criticisms of recurrent architecture:
- recurrent computations cannot be parallelized
- core idea: instead of fixed-width convolutional filter, use attention
- there are different flavours of self-attention
  - here: attend over previous layer of deep network

General Architecture Variants

NMT with Convolutional Neural Networks

NMT with Self-Attention

Convolution

Self-Attention

Attention Is All You Need

### Transformer architecture
- stack of $N$ self-attention layers
- self-attention in decoder is *masked*
- decoder also attends to encoder states
- *Add & Norm*: residual connection and layer normalization

### Multi-Head Attention
- **basic attention mechanisms**: Scaled Dot-Product Attention
  \[
  \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V
  \]
- query $Q$ is decoder/encoder state (for attention/self-attention)
- key $K$ and value $V$ are encoder hidden states
- multi-head attention: use $h$ parallel attention mechanisms with low-dimensional, learned projections of $Q$, $K$, and $V$

### Multi-Head Attention
- motivation for multi-head attention:
  different heads can attend to different states

### Comparison
- **empirical comparison difficult**
  - some components could be mix-and-matched
    - choice of attention mechanism
    - choice of positional encoding
    - hyperparameters and training tricks
  - different test sets and/or evaluation scripts
### Comparison

**SOCKEYE [Hieber et al., 2017] (EN-DE; newstest2017)**

<table>
<thead>
<tr>
<th>System</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>deep LSTM</td>
<td>25.6</td>
</tr>
<tr>
<td>Convolutional</td>
<td>24.6</td>
</tr>
<tr>
<td>Transformer</td>
<td><strong>27.5</strong></td>
</tr>
</tbody>
</table>

**Marian (EN-DE; newstest2016)**

<table>
<thead>
<tr>
<th>System</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>deep LSTM</td>
<td>32.6</td>
</tr>
<tr>
<td>Transformer</td>
<td><strong>33.4</strong></td>
</tr>
</tbody>
</table>

---

### Empiricism vs. Theory

- Our theoretical understanding of neural networks lags behind empirical progress.
- There are some theoretical arguments why architectures work well... (e.g. self-attention reduces distance in network between words)
- ...but these are very speculative.

---

### Further Reading

- Required reading: Koehn, 13.7
- Consider original literature cited on relevant slides.

---

### Bibliography I

Decoding with Large-Scale Neural Language Models Improves Translation.

Recurrent Neural Network Regularization.
CoRR, abs/1409.2329.