ML and Optimization Lec 25

- Seq2Seq model w/ attention
- Attention is all we need: seq2seq models w/o RNNs
- BERT: pretraining transformer models for NLP
**Seq2Seq model**

Input: \( x_1, \ldots, x_m \)

Output: \( x_1, \ldots, x_T \)

**Encoder**

\[
\begin{align*}
&x_1 \quad \rightarrow h_1 \\
&\vdots \quad \rightarrow h_{m-1} \\
&X_m \quad \rightarrow h_m
\end{align*}
\]

**Decoder**

\[
\begin{align*}
&S_0 \\
&\vdots \\
&\langle \text{START} \rangle \quad \rightarrow S_T
\end{align*}
\]

**Issue:** regardless of length of input sequence, all context/historical dependence of the output sequence is contained in a d-dimensional vector \( h_m \). This leads to poor performance (BLEU decreases w/ length of input sequence for Machine translation tasks).
**Seq2Seq with attention**

- Introduced in Bahdanau et al. 2015, Neural Machine Translation by Jointly Learning to Align and Translate

- Allow each output token to attend to each individual hidden state in the input sequence. Captures idea of focusing on relevant portion of input sequence. Avoids the need to capture everything relevant to all outputs $x_1, \ldots, x_T$ in the one vector $h_m$.
Seq2Seq w/ attention

Idea: replace our previous hidden state update for decoder

\[ s_{t-1} \rightarrow \begin{array}{c} \text{\_} \end{array} \rightarrow s_t \quad \text{i.e.} \quad s_t = g(s_{t-1}, x'_t) \]

w/ first finding which historical hidden states are most relevant:

\[ s_{t-1} \rightarrow x'_t \rightarrow [h_1 \ldots h_m] \rightarrow x'_t \text{ a prob distribution over } (h_1) \ldots (h_m) \]

then form a context vector

\[ c_t = \sum_{i=1}^{m} (x'_t)_i h_i \quad \text{and update with} \]

\[ s_t = g(s_{t-1}, c_t) \]

we removed dependence on \( x'_t \) because in the decoder \( x'_t \) isn't much more than \( s_{t-1} \)
In the context of decoding, in the decoder, use attention over the hidden states of the encoder.

$$\sum_{i=1}^{m} (d_{i})_{i} h_{i}$$
Options for computing the attention vectors $\alpha_t$

1) Option I (used in Bahdanau et al., 2015)

Take $\alpha_i = v^T \tanh (w) \begin{bmatrix} h_i \\ s_{t-1} \end{bmatrix}$ for $i = 1, \ldots, m$

Then $\alpha_t = \text{softmax} (\mathbf{z}) \in \mathbb{R}^m$

2) (Dot-product attention)

Idea: generate keys and queries as linear transforms of $h_i$ and $s_{t-1}$ respectively. Take logits $\alpha_t$ to be their inner-products.

$W_k h_i = k_i$

$W_q s_{t-1} = q_{t-1}$

$\alpha_i = \langle k_i, q_{t-1} \rangle$

$\alpha_t = \text{softmax} (\mathbf{z})$
Complexity of using attention

1) Simple ANN encoder models: $O(m + T)$

2) ANN + attention models $O(m \cdot T)$

because for each output token, most compute $\alpha_t \in \mathbb{R}^m$ by comparing to all inputs
Transformers: Attention is all we need

- 2017: Vaswani et al., Attention is all we need

- Insight: attention alone suffices to contain all historical information we need, so can eliminate recursive structure.

- Replace hidden representations in RNNs w/ contextual representations $C_t$ computed using self-attention (for encoder)

- Use combination of self-attention in decoder and general attention against the encoder contextual representations to get contextual representations to predict output tokens
Self-Attention (replace the hidden states learned in RNNs w/ contextual representations)

\[ c_i = \sum_{j=1}^{m} (\alpha_{ij}) \cdot v_j \]

where key \( k_i = W_k x_i \)
query \( q_i = W_q x_i \)
value \( v_i = W_v x_i \)

and \( (\alpha_{ij}) = \langle q_i, k_j \rangle \) and \( \alpha = \text{softmax} (\vec{\alpha}) \)
Multi-Head Self Attention

Just as in CNNs we benefit from learning multiple contextual representations:

\[(w_k)_i, (w_q)_i \rightarrow (w_v)_i \text{ for } i=1, \ldots, d\]
Contextual representations using multi-head self-attention

Self-Attention + Dense Layer