Flipped Classroom Questions on Recurrent Networks and Attention

Michael Collins

Question 1: Consider the equations for a recurrent model mapping an input source language sentence $x_1 \ldots x_n$ to an output target language sentences $y_1 \ldots y_m$:

Inputs: A sequence $x_1 \ldots x_n$ where each $x_j \in \mathbb{R}^d$.

Definitions: $\theta_{i,j,n} \in \mathbb{R}$ is a parameter for aligning source word i to target word j given source sentence length n

Computational Graph:

- **Encoding step:** use a bi-directional LSTM to map the input sentence $x_1 \ldots x_n$ to a new sequence $u^{(1)} \ldots u^{(n)}$.
- **Decoding:** For $j = 1 \ldots m$
  - Define
  $$a_{i,j} = \underbrace{\hspace{12em}}_{\text{COMPLETE CODE HERE}}$$
  - Set $c^{(j)} = \sum_{i=1}^{n} a_{i,j} u^{(i)}$
  - $\beta^{(j)} = g(W^{hc} c^{(j)} + b^h)$
  - $l^{(j)} = V \times \beta^{(j)} + \gamma$
  - $y_j = \arg\max_y l_y^{(j)}$
  - $j = j + 1$
  - **Until** $y_{j-1} =$ STOP
- Return $y_1 \ldots y_{j-1}$

Question 1a: How would you complete the code above to give similar behavior to IBM model 2? Recall that in IBM model 2 we have alignment parameters $q(i|j, m, n)$ and translation parameters $t(f|e)$.

Question 1b: Draw the computational graph for this model (you can take the inputs to this computation graph to be $u^{(1)} \ldots u^{(n)}$; there is no need to draw the computational graph for the encoding LSTM.

Question 1c: Suppose we have a training set consisting of *the dog/le chien the cat/le chat* and *the hospital/l'hôpital*. How would you set the $\theta_{i,j,n,m}$ parameters to model this data?

Question 1d: Suppose we now wish for each vector of alignment variables $a_j = \langle a_{1,j} \ldots a_{n,j} \rangle$ to depend on the previous alignment variables a_{j-1}, and the previous target-language word y_{j-1}. How would you alter the equations in the figure to achieve this?

Question 2: Consider the attention-based model for translation given in the lecture slides. The following pseudo-code can be used to calculate the distribution

$$p(y|y_1 \ldots y_{k-1}, x_1 \ldots x_n)$$

for any input sentence $x_1 \ldots x_n$ and target language prefix $y_1 \ldots y_{k-1}$:

Inputs: Source language sentence $x_1 \ldots x_n$, target language prefix $y_1 \ldots y_{k-1}$
Goal: Compute the conditional distribution

$$p(y|y_1 \ldots y_{k-1}, x_1 \ldots x_n)$$

- **Encoding step:** use an LSTM to map $x_1 \ldots x_n$ to $u^{(1)} \ldots u^{(n)}$
- **Decoding step:** For $j = 1 \ldots (k-1)$
  - For $i = 1 \ldots n$,
  $$s_{i,j} = A(\beta^{(j-1)}, u^{(i)}; \theta^A)$$
  - For $i = 1 \ldots n$,
  $$a_{i,j} = \frac{\exp\{s_{i,j}\}}{\sum_{i=1}^n \exp\{s_{i,j}\}}$$
  - Set $c^{(j)} = \sum_{i=1}^n a_{i,j} u^{(i)}$
  - $\beta^{(j)} = \text{LSTM}(\text{CONCAT}(y_{j-1}, c^{(j)}), \beta^{(j-1)}; \theta^D)$
  - $l^{(j)} = V \times \text{CONCAT}(\beta^{(j)}, y_{j-1}, c^{(j)}) + \gamma, \;\; q^{(j)} = \text{LS}(l^{(j)})$,
- **Output:** $q^{(j)}$ such that
$$q_y^{(j)} = \log p(y|y_1 \ldots y_{k-1}, x_1 \ldots x_n)$$

(Continued over page.)

Consider a beam-search algorithm for decoding with this model. Assume for simplicity that the input to the algorithm is an integer m specifying the length of the target language sentence. You can make use of the following primitives:

- BEAM(k) for $k = 0 \ldots m$ is a set of *items*. Each *item* is a pair $(y_1 \ldots y_k, \text{score})$ where $y_1 \ldots y_k$ is a partial translation, and
$$\text{score} = \sum_{j=1}^{k} \log p(y_j | y_1 \ldots y_{j-1}, x_1 \ldots x_n)$$
- The function ADD$(y_1 \ldots y_k, \text{score})$ adds a partial translation together with a score to BEAM(k). If the partial translation is not in the top 10 highest scoring partial translations in BEAM(k), it is not added. If after the addition there are more than 10 items in the beam, only the top 10 highest scoring items are retained in the beam. (The beam keeps the top 10 most likely translations at each point)
- INIT(BEAM) initializes BEAM(k) to be the empty set for $k \geq 1$, and initializes BEAM(0) to contain a single entry with partial translation equal to ϵ (the empty string), and score equal to 0.
- "foreach $(y_1 \ldots y_k, \text{score}) \in \text{BEAM}(k)$" initializes a *foreach* loop over the items in BEAM(k)
- We can use the computational graph in the pseudo-code above to calculate the distribution
$$\log p(y | y_1 \ldots y_{k-1}, x_1 \ldots x_n)$$
for any prefix $y_1 \ldots y_{k-1}$
- Finally, ARGMAX(BEAM) returns the highest scoring $y_1 \ldots y_k$, score pair in BEAM$(1) \cup$BEAM$(2) \ldots \cup$ BEAM(m) such that $y_k = \text{STOP}$

Question 2a: Write pseudo-code for a beam-search algorithm using the above primitives. The algorithm should take a source language sentence $x_1 \ldots x_n$, and an integer m specifying the maximum output length, as inputs.

Question 2b: If we naively calculate
$$p(y | y_1 \ldots y_{k-1}, x_1 \ldots x_n)$$
using the code given above, there will be a lot of repeated (wasted) computation. How would you make the algorithm more efficient, by caching some computation?

Question 3: Consider the function that maps a vector $s \in \mathbb{R}^d$ to a new vector $a \in [0,1]^d$ through the softmax function:

$$a_j = \frac{\exp\{s_j\}}{\sum_{i=1}^{d} \exp\{s_i\}}$$

This is the function frequently used in attention-based models.

For each $j = 1 \ldots d$, what is the value for

$$\frac{\partial a_j}{\partial s_j}$$

For each j, j' such that $j \neq j'$, what is the value for

$$\frac{\partial a_j}{\partial s_{j'}}$$

Hint: recall the quotient rule for differentiation. If $f(x) = g(x)/h(x)$, and $f'(x)/g'(x)/h'(x)$ is the derivative of $f(x)/g(x)/h(x)$ respectively, then

$$f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{(h(x))^2}$$