CS 6320.002: Natural Language Processing
Fall 2020

Homework 5 — 45 points
Issued 17 Nov. 2021
Due 11:59 pm 06 Dec. 2021

Deliverables: Answers can be typed directly into Gradescope. LaTeX can be hand-typed or generated using Mathpix Snip. See the assignment guide for more details.

What does it mean to show your work? Write out the math step-by-step; we should be able to clearly follow your reasoning from one step to another. (You can combine obvious steps like simplifying fractions or doing basic arithmetic.) The point of showing your work is twofold: to get partial credit if your answer is incorrect, and to show us that you worked the problem yourself and understand it. We will deduct points if steps are missing.

1 Recurrent Neural Networks — 30 points

The questions in this section are based on material covered in Week 13.

1.1 Backpropagation Through Time

Suppose we have the following small recurrent neural network:

$$h^{(t)} = \text{RELU}(w_1 x_1^{(t)} + w_2 x_2^{(t)} + u h^{(t-1)} + b_h)$$
$$y^{(t)} = \sigma(v h^{(t)} + b_y)$$

(The superscript notation (t) is used to avoid confusion when we also want to use subscripts for our parameters; $h^{(t)}$ means the same thing as h_t in the slides.)

Suppose we have the following loss function:

$$L(y) = -\ln(y)$$

(For the problems in this section, it is not required to submit a picture of the computation graph, but it is highly recommended to draw one for yourself to aid in answering the questions.)

1.1.1

What is the partial derivative of the loss $L(y^{(1)})$ with respect to the parameter w_1, ie. $\frac{\partial L(y^{(1)})}{\partial w_1}$, in this network? Show your work and simplify the expression as much as possible. You can assume that all values are positive for RELU purposes.

(HINT: You may find it easier to introduce additional variables to represent the outputs of individual operations, ie. the nodes in the computation graph. If you do so, clearly state what each new variable represents.)

1.1.2

How many paths are there in the computation graph of this network from the parameter w_1 to the loss $L(y^{(2)})$? What are the partial derivatives $\frac{\partial L(y^{(2)})}{\partial w_1}$ along each of those paths? Show your work and simplify the expressions as much as possible.

(HINT: You may find it easier to introduce additional variables to represent the outputs of individual operations, ie. the nodes in the computation graph. If you do so, clearly state what each new variable represents.)

1.1.3

If we run this network for three time steps, calculating the loss $L(y^{(t)})$ and performing standard stochastic gradient descent at each time step $t = 1, 2, 3$, how many total updates does the parameter w_1 receive? Explain your answer (3-4 sentences).

1.2 Batched Recurrent Networks

Recall that training using *mini-batch gradient descent* usually gives a better training time/performance trade-off than stochastic and full-batch gradient descent. A mini-batch consists of b training inputs that are combined into a single input tensor. For a feedforward network, for example, if a single training input is a vector of length n, then a mini-batch is a matrix of shape $(b \times n)$.

Suppose we want to use mini-batch gradient descent to train a recurrent neural network. That is, we have b training sequences, where each sequence is a matrix of shape $(l_i \times m)$, where l_i is the length of the sequence and m is your word embedding size. There is no guarantee that the input sequences are all the same length. How can we combine them into a single input tensor? Describe a strategy to do this (2-3 sentences). Your answer should address the following issues:

- What is the shape of the mini-batch input tensor, and what does each dimension of the tensor mean?
- How do you assign values to the tensor's entries, and how does this solve the problem of different sequence lengths?
- How does your strategy affect the network's predictions and parameter updates?

1.3 Deep Recurrent Networks

Suppose we have the following single-layer recurrent neural network:

$$\mathbf{h}_1^{(t)} = f_1(\mathbf{W}_1\mathbf{x}^{(t)} + \mathbf{U}_1\mathbf{h}_1^{(t-1)} + \mathbf{b}_1)$$
$$\mathbf{y}^{(t)} = f_y(\mathbf{V}\mathbf{h}_1^{(t)})$$

Unfortunately, this single-layer RNN is not giving good performance, so we want to improve it by adding a second layer and a residual connection from the output of the first layer to the output of the second layer. What are the equations for this new, two-layer RNN? Keep your variable names and notation as close as possible to those in the original network.

1.4 Bidirectional Recurrent Networks

Suppose we have the following unidirectional recurrent neural network:

$$\overrightarrow{\mathbf{h}}_t = f_h(\mathbf{W}\mathbf{x}_t + \mathbf{U}\overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b})$$
$$\mathbf{y}_t = f_y(\mathbf{V}\overrightarrow{h}_t)$$

Unfortunately, this unidirectional, forward-only RNN is not giving good performance, so we want to improve it by adding a backward run of the same RNN to make the network bidirectional. What are the equations for this new, bidirectional recurrent network? Keep your variable names and notation as close as possible to those in the original network.

2 Attention Networks — 15 points

The questions in this section are based on material covered in Week 14.

2.1 Self Attention

Suppose we have the following RNN encoder-decoder model:

$$\mathbf{h}^{(i)} = \tanh(\mathbf{W}_h\mathbf{x}^{(i)} + \mathbf{U}_h\mathbf{h}^{(i-1)})$$
$$\mathbf{d}^{(0)} = \mathbf{h}^{(n)}$$
$$\mathbf{e}_{enc}^{(ij)} = \mathbf{v}_{enc} \cdot \tanh(\mathbf{V}_{enc1}\mathbf{h}^{(i)} + \mathbf{V}_{enc2}\mathbf{d}^{(j-1)})$$
$$\mathbf{a}_{enc}^{(j)} = \text{softmax}(\mathbf{e}_{enc}^{(j)})$$
$$\mathbf{c}_{enc}^{(j)} = \sum_i \mathbf{a}_{enc}^{(ij)}\mathbf{h}^{(i)}$$
$$\mathbf{d}^{(j)} = \tanh(\mathbf{W}_d\mathbf{y}^{(j-1)} + \mathbf{U}_d\mathbf{d}^{(j-1)} + \mathbf{V}_d\mathbf{c}_{enc}^{(j)})$$

Unfortunately, this encoder-decoder with attention only from the decoder to the encoder is not giving good performance, so we want to improve it by adding self-attention to the decoder. What are the equations for this new, self-attentive encoder-decoder? Keep your variable names and notation as close as possible to those in the original network.

(HINT: Look at the diagram depicting self attention in slide deck 25.)

2.2 Multi-Head Attention

Suppose we have a neural network with hidden size n that uses normal (single-head) attention. Unfortunately, this network is not giving good performance, so we want to improve it by switching to multi-head attention. How do we decide how many heads to use and how many dimensions each head should have? Are there any special considerations that we should be careful of when deciding these things? Briefly explain your answer (3-4 sentences).

2.3 Transformers and RNNs

Recall that while a transformer-based encoder can be trained efficiently using parallelization, a transformer-based decoder is very slow and can't be parallelized. One way to take advantage of the transformer's strong performance without sacrificing too much computation time is to use a transformer-based encoder and an RNN decoder. How could you build such an encoder-decoder network (1-2 sentences)? Your answer should address the following:

- How do you initialize the decoder hidden state?
- How do you allow the decoder to copy words from the input sequence if needed?
- How do you train the encoder-decoder end-to-end?