Author: Benjamin Smidt
Created: October 18th, 2022
Last Updated: October 18th, 2022

Assignment 4: RNNs, LSTM, and Attention with Image Captioning

Note to reader.

This is my work for assignment four (A4) of Michigan's course EECS 498: Deep Learning for Computer Vision. The majority of explanations and understanding are derived from Justin Johnson's Lectures and Stanford's CS 231N Lecture Notes. This document is meant to be used as a reference, explanation, and resource for the assignment, not necessarily a comprehensive overview of Neural Networks. If there's a typo or a correction needs to be made, feel free to email me at benjamin.smidt@utexas.edu so I can fix it. Thank you! I hope you find this document helpful.

Contents

1 Vanilla Recurrent Neural Networks

1.1 RNN Forward

Recurrent neural networks are very powerful in their ability to process and output variable length data. Said another way, RNNs can be fed different length inputs as well predict different length outputs making them a powerful and useful paradigm in many applications. We'll go into more depth as we go along but for now we'll start with vanilla recurrent neural networks.

$$h_t = f_w(h_{t-1}, x_t)$$

To achieve variable length inputs and outputs we need to change our neural network model a bit. Instead of having some predefined network size, we have a function that takes two inputs: the output of the previous computation (also known as the "hidden state") and some data input (usually interpreted as a sequence). See this picture for a visual. For vanilla neural networks (also known as "Elman RNNs") we use the following function.

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b) \tag{1}$$

where h_t is the current state, x_t is the input data, W_{hh} is our (reused) weight matrix for the hidden state input, and W_{xh} is our (reused) weight matrix for the current input x_t. We also add a bias b. To be clear, W_{hh} and W_{xh} do not change at all beween time steps. They are the same set of parameters throughout the neural network's computation. For out first function, *rnn-forward*, we simply write down Eq. (1) in code and store our needed variables in *cache* for backpropagation.

If you're wondering about initialization and how to know when to stop computing h_t, keep reading. I'll answer those and other questions as we go along.

1.2 RNN Backward

Let's look at backpropagating a given time step given our function. Recall that

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Thus, by quotient rule, our derivative is as follows

$$\frac{\partial\, tanh(z)}{\partial z} = \frac{(e^z + e^{-z})(e^z + e^{-z}) - (e^z - e^{-z})(e^z - e^{-z})}{(e^z + e^{-z})^2}$$

$$\frac{\partial\, tanh(z)}{\partial z} = \frac{(e^z + e^{-z})^2 - (e^z - e^{-z})^2}{(e^z + e^{-z})^2}$$

$$\frac{\partial\, tanh(z)}{\partial z} = \frac{(e^z + e^{-z})^2}{(e^z + e^{-z})^2} - \frac{(e^z - e^{-z})^2}{(e^z + e^{-z})^2}$$

$$\frac{\partial\, tanh(z)}{\partial z} = 1 - tanh^2(z)$$

If we set $z = W_{hh}h_{t-1} + W_{xh}x_t + b$, we get the first term in our backpropagation. Moving forward (or backward I guess) in our backpropagation, we'll next work on each of the variables inside the $tanh$ function starting with W_{hh} and W_{xh}.

$$\frac{\partial\, z}{\partial W_{hh}} = h_{t-1} \quad \text{and} \quad \frac{\partial\, z}{\partial W_{xh}} = x_t$$

Thus

$$\frac{\partial\, h_t}{\partial z}\frac{\partial\, z}{\partial W_{hh}} = h_{t-1}^T\, [1 - tanh^2(z)]$$

$$\frac{\partial\, h_t}{\partial z}\frac{\partial\, z}{\partial W_{xh}} = x_t^T [1 - tanh^2(z)]$$

where $z = W_{hh}h_{t-1} + W_{xh}x_t + b$ and the transposes are derived by the shape convention. Next we have h_{t-1} and x_t.

$$\frac{\partial\, z}{\partial h_{t-1}} = W_{hh} \quad \text{and} \quad \frac{\partial\, z}{\partial x_t} = W_{xh}$$

Thus

$$\frac{\partial\, h_t}{\partial z}\frac{\partial\, z}{\partial h_{t-1}} = [1 - tanh^2(z)]\, W_{hh}^T$$

$$\frac{\partial\, h_t}{\partial z}\frac{\partial\, z}{\partial x_t} = [1 - tanh^2(z)]\, W_{xh}^T$$

And finally for our bias

$$\frac{\partial\, z}{\partial b} = 1$$

$$\frac{\partial\, h_t}{\partial z}\frac{\partial\, z}{\partial b} = [1 - tanh^2(z)]$$

Of course, we'll have to manipulate the bias b more when we program the backpropagation since b is actually broadcast over the the outputs which needs to be accounted for in our backpropagation.