CSCI 447/547: Homework 3

Due: 10/21/2021

Please submit your responses to the following questions on Moodle. You may work with a partner, and are encouraged to develop solution strategies together, but your actual submitted responses must be entirely your own. Using the internet is okay, but try to solve these on your own first, and be sure not to plagiarize! The objective of these exercises is not to get the correct answer, but rather to demonstrate that you understand the answer and the question to which you are responding. As such, please use complete sentences and develop your arguments fully. In particular, don't leave it to me to try to divine your meaning!

Problem 1: Overparameterization

Consider a multi-class classification problem ($N = 3$) with a single feature (called x). Assuming that the logits are related to the feature via a linear model, we can write the conditional class probabilities as

$$P(y = k|x) = \frac{\exp\left(W_{0,k} + W_{1,k}x\right)}{\sum_j^N \exp\left(W_{0,j} + W_{1,j}x\right)}. \tag{1}$$

This model has $2N$ free parameters, two for each class. Show that this model is overparameterized; specifically, show that this model can be equivalently represented with $2(N-1)$ free parameters. HINT: With two classes, this procedure *exactly* recovers logistic regression.

Problem 2: Backpropagation

While the formulas for backpropagation that we've dealt with are generally for fairly large graphs, it's instructive to develop a little bit of intuition by performing it on very small graphs. Consider the following graph:

where x is an input, nodes with a σ apply the logistic function, nodes with an I apply the identity function (aka don't really do anything), and L is the squared error between a prediction and an observation y. Written out algebraically, this graph says

$$a^{(1)} = xw_1^{(1)} \tag{2}$$
$$z^{(1)} = \sigma(a_1) \tag{3}$$
$$[a_1^{(2)}, a_2^{(2)}] = z^{(1)} \times [w_1^{(2)}, w_2^{(2)}] \tag{4}$$
$$[z_1^{(2)}, z_2^{(2)}] = [\sigma(a_1^{(2)}), \sigma(a_2^{(2)})] \tag{5}$$
$$a^{(3)} = z_1^{(2)}w_1^{(3)} + z_2^{(2)}w_2^{(3)} \tag{6}$$
$$z^{(3)} = a^{(3)} \tag{7}$$
$$L = \frac{1}{2}(z^{(3)} - y)^2 \tag{8}$$
$$\tag{9}$$

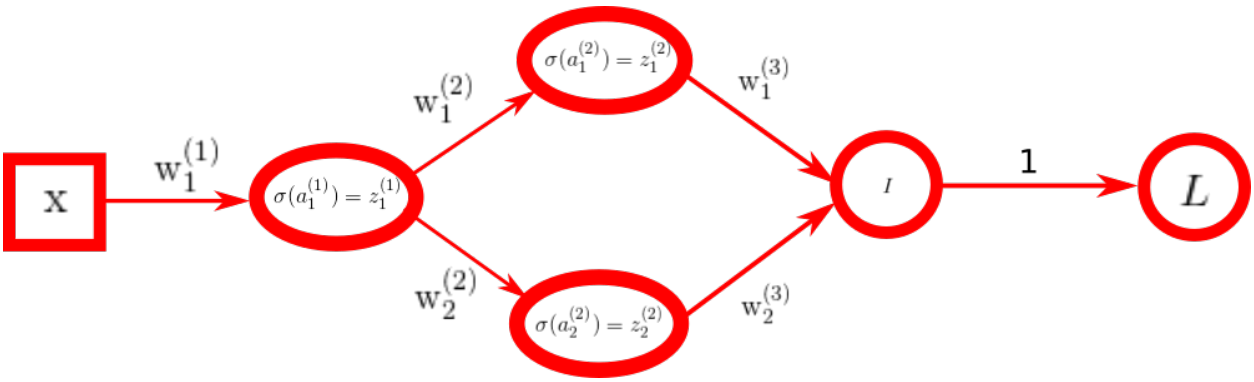

Figure 1: Graph of functional transformation corresponding to Problem 2

The chain rule can be used to take derivatives of L with respect to the various weights, even far back in the chain. For example, we could use the chain rule to take the derivative of L with respect to $w_1^{(3)}$ as follows:

$$\frac{\partial L}{w_1^{(3)}} = \frac{\partial L}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial w_1^{(3)}} = (z^{(3)} - y) \times 1 \times z_1^{(2)}$$

What we've done here is to trace the path from L to the parameter, taking the derivative between the outputs and inputs of each node as we work back through the graph, then evaluated each resulting derivative!

A

Derive and evaluate an expression for

$$\frac{\partial L}{\partial w_2^{(2)}}.$$

Create a python script that computes this derivative for the feature $x = 1.0$, data $y = 0.5$, and parameter values

$$w_1^{(1)} = 0.5 \tag{10}$$
$$w_1^{(2)} = 0.7 \tag{11}$$
$$w_2^{(2)} = -0.3 \tag{12}$$
$$w_1^{(3)} = 0.1 \tag{13}$$
$$w_2^{(3)} = -0.8 \tag{14}$$

As a check, the correct value is approximately 0.099.

B: GRAD STUDENTS

Implement the finite difference method and compare its result to the value you got through backpropagation (they should be close). Give two reasons why the finite difference method is a poor choice for computing derivatives in neural networks.

Problem 3: Robust Cross-Entropy

(From Bishop, Ch. 5) Consider a binary classification problem in which the target values are $y_{obs} \in 0, 1$, with a network output $y(x, w)$ that represents $P(y = 1|x)$ (i.e. logistic regression). Now suppose that there is a

probability ϵ that the class label on a training data point has been incorrectly set. Assuming independent and identically distributed data, write down the negative log likelihood. Verify that the cross-entropy error function (Bishop 5.21) is obtained when $\epsilon = 0$. Note that this error function makes the model robust to incorrectly labelled data, in contrast to the usual error function. HINT 1: Use the fact that

$$P(y_{obs}|x) = \underbrace{\sum_k P(y_{obs}, y = k|x)}_{\text{Sum Rule}} = \sum_k \underbrace{P(y_{obs}|y = k)P(y = k|x)}_{\text{Product Rule}}.$$

HINT 2: This expression will not be very pretty, and will involve the logarithm of a sum. NOTE: This problem is hard, and will be graded on a curve. Do your best to get as far as you can.