Worksheet 1

CS 6421
January 30, 2020

Deep Learning: Problem Formulation

1. You have a single hidden-layer neural network for a binary classification task. The input is $X \in \mathbb{R}^{n \times m}$, output $\hat{y} \in \mathbb{R}^{1 \times m}$ and true label $y \in \mathbb{R}^{1 \times m}$. The squared-error loss function is $\mathcal{J}$. The forward propagation equations are:

$$\begin{aligned} z &= WX + B \\ \hat{y} &= \sigma(z) \\ \mathcal{J} &= (\hat{y} - y)^2 \end{aligned}$$

   (a) We want to compute how to change the weights W based on errors in the loss function $\mathcal{J}$. Define a partial derivative D to compute this.

   (b) Express D in terms of a sequence of partial derivatives in the network, i.e., including $\partial z / \partial X$.

   (c) Compute a closed-form expression for D.

   (d) Draw the computation graph corresponding to this network.

   (e) Write out the TensorFlow code for forward inference in the network.

2. You are solving the binary classification task of classifying images as cat vs. non-cat. You design a CNN with a single output neuron. Let the output of this neuron be z. The final output of your network, $\hat{y}$ is given by: $\hat{y} = \sigma(ReLU(z))$.

   You classify all inputs with a final value $\hat{y} \geq 0.5$ as cat images. What problem are you going to encounter?

3. You train a simple network in which the final output of your network, $\hat{y}$ is given by a sigmoid activation function: $\hat{y} = \sigma(Wx + b)$.

   (a) If you initialise the weights in W, b to be large numbers, show analytically that the network will not learn for input $x \geq 0$.

   (b) If you initialise only the weights in b to be large numbers, will you have the same problem? Again, show why or why not.

4. You are given the following piece of code for forward propagation through a single hidden layer in a neural network. This layer uses the sigmoid activation. Identify and correct the error.

```python
import numpy as np
def forward_prop(W, a_prev, b):
    z = W*a_prev + b
    a = 1/(1+np.exp(-z)) #sigmoid
return a
```