Assignment 2 - Objectives, Gradients, and Backpropagation

Atieh Armin
Student ID : 14658308

1 Theory

1. (10 points) Given $H = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ as an input, compute the gradients of the output with respect to this input for the following activation layers. Show your answer in **tensor form** by having a Jacobian matrix for each observation.

   (a) A ReLU layer

$$\frac{\partial g_0(z)}{\partial z_0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial g_1(z)}{\partial z_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   (b) A Softmax layer

$$\frac{\partial g_0(z)}{\partial z_0} = \begin{bmatrix} \frac{e^3+e^4}{(e^1+e^2+e^3)^2} & \frac{-e^3}{(e^1+e^2+e^3)^2} & \frac{-e^4}{(e^1+e^2+e^3)^2} \\ \frac{-e^3}{(e^1+e^2+e^3)^2} & \frac{e^3+e^5}{(e^1+e^2+e^3)^2} & \frac{-e^5}{(e^1+e^2+e^3)^2} \\ \frac{-e^4}{(e^1+e^2+e^3)^2} & \frac{-e^5}{(e^1+e^2+e^3)^2} & \frac{e^4+e^5}{(e^1+e^2+e^3)^2} \end{bmatrix} \approx \begin{bmatrix} 0.0819 & -0.0220 & -0.0599 \\ -0.0220 & 0.1848 & -0.1628 \\ -0.0599 & -0.1628 & 0.2227 \end{bmatrix}$$

$$\frac{\partial g_1(z)}{\partial z_1} = \begin{bmatrix} \frac{e^9+e^{10}}{(e^4+e^5+e^6)^2} & \frac{-e^9}{(e^4+e^5+e^6)^2} & \frac{-e^{10}}{(e^4+e^5+e^6)^2} \\ \frac{-e^9}{(e^4+e^5+e^6)^2} & \frac{e^9+e^{11}}{(e^4+e^5+e^6)^2} & \frac{-e^{11}}{(e^4+e^5+e^6)^2} \\ \frac{-e^{10}}{(e^4+e^5+e^6)^2} & \frac{-e^{11}}{(e^4+e^5+e^6)^2} & \frac{e^{10}+e^{11}}{(e^4+e^5+e^6)^2} \end{bmatrix} \approx \begin{bmatrix} 0.0819 & -0.0220 & -0.0599 \\ -0.0220 & 0.1848 & -0.1628 \\ -0.0599 & -0.1628 & 0.2227 \end{bmatrix}$$

   (c) A Logistic Sigmoid Layer

$$\frac{\partial g_0(z)}{\partial z_0} = \begin{bmatrix} \frac{e^{-1}}{(1+e^{-1})^2} & 0 & 0 \\ 0 & \frac{e^{-2}}{(1+e^{-2})^2} & 0 \\ 0 & 0 & \frac{e^{-3}}{(1+e^{-3})^2} \end{bmatrix} \approx \begin{bmatrix} 0.19661193 & 0 & 0 \\ 0 & 0.10499359 & 0 \\ 0 & 0 & 0.04517666 \end{bmatrix}$$

$$\frac{\partial g_1(z)}{\partial z_1} = \begin{bmatrix} \frac{e^{-4}}{(1+e^{-4})^2} & 0 & 0 \\ 0 & \frac{e^{-5}}{(1+e^{-5})^2} & 0 \\ 0 & 0 & \frac{e^{-6}}{(1+e^{-6})^2} \end{bmatrix} \approx \begin{bmatrix} 0.01766271 & 0 & 0 \\ 0 & 0.00664806 & 0 \\ 0 & 0 & 0.00246651 \end{bmatrix}$$

(d) A Tanh Layer

$$\frac{\partial g_0(z)}{\partial z_0} = \begin{bmatrix} \frac{4*e^1*e^{-1}}{(e^1+e^{-1})^2} & 0 & 0 \\ 0 & \frac{4*e^2*e^{-2}}{(e^2+e^{-2})^2} & 0 \\ 0 & 0 & \frac{4*e^3*e^{-3}}{(e^3+e^{-3})^2} \end{bmatrix} \approx \begin{bmatrix} 0.419974342 & 0 & 0 \\ 0 & 0.070650824 & 0 \\ 0 & 0 & 0.009866037 \end{bmatrix}$$

$$\frac{\partial g_1(z)}{\partial z_1} = \begin{bmatrix} \frac{4*e^4*e^{-4}}{(e^4+e^{-4})^2} & 0 & 0 \\ 0 & \frac{4*e^5*e^{-5}}{(e^5+e^{-5})^2} & 0 \\ 0 & 0 & \frac{4*e^6*e^{-6}}{(e^6+e^{-6})^2} \end{bmatrix} \approx \begin{bmatrix} 0.001340950 & 0 & 0 \\ 0 & 0.000181583 & 0 \\ 0 & 0 & 0.000024576 \end{bmatrix}$$

(e) A Linear Layer

$$\frac{\partial g_0(z)}{\partial z_0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial g_1(z)}{\partial z_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. (2 points) Given $H = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ as an input, compute the gradient of the output a fully connected layer with regards to this input if the fully connected layer has weights of $W = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$ as biases $b = \begin{bmatrix} -1 & 2 \end{bmatrix}$.

   Solution: $W^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$

3. (2 points) Given target values of $Y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and estimated values of $\hat{Y} = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$ compute the loss for:

   (a) A squared error objective function
   $J = mean((Y - \hat{Y}) * (Y - \hat{Y})) = 0.265$

   (b) A log loss (negative log likelihood) objective function)
   $J = mean(-(Y * \ln(\hat{Y}) + (1 - Y) * \ln(1 - \hat{Y}))) \approx 0.71356$

4. (1 point) Given target *distributions* of $Y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ and estimated distributions of $\hat{Y} = \begin{bmatrix} 0.2 & 0.2 & 0.6 \\ 0.2 & 0.7 & 0.1 \end{bmatrix}$ compute the cross entropy loss.

   Solution:

   $J = -mean(\sum_{K=1}^{K} Y_{:,K} * \log(\hat{Y}_{:,K})) = -mean(\sum_{K=1}^{K} \begin{bmatrix} \ln(0.2) & 0 & 0 \\ 0 & \ln(0.7) & 0 \end{bmatrix})$

   $= -mean(\begin{bmatrix} \ln(0.2) \\ \ln(0.7) \end{bmatrix}) \approx 0.983056$

5. (4 points) Given target values of $Y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and estimated values of $\hat{Y} = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$ compute the gradient of the following objective functions with regards to their input, $\hat{Y}$:

   (a) A squared error objective function

   $\frac{\partial J}{\partial \hat{Y}} = -2(Y - \hat{Y}) = \begin{bmatrix} 0.4 \\ -1.4 \end{bmatrix}$

   (b) A log loss (negative log likelihood) objective function)

   $\frac{\partial J}{\partial \hat{Y}} = -\frac{Y - \hat{Y}}{\hat{Y} * (1 - \hat{Y})} = \begin{bmatrix} 1.25 \\ -3.3333 \end{bmatrix}$

6. (1 point) Given target *distributions* of $Y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ and estimated distributions of $\hat{Y} = \begin{bmatrix} 0.2 & 0.2 & 0.6 \\ 0.2 & 0.7 & 0.1 \end{bmatrix}$ compute the gradient of the cross entropy loss function, with regard to the input distributions $\hat{Y}$.

   Solution: $\frac{\partial J}{\partial \hat{Y}} = -\frac{Y}{\hat{Y}} = \begin{bmatrix} -5 & 0 & 0 \\ 0 & -1.42857143 & 0 \end{bmatrix}$

Datasets

Kid Creative We will use this dataset for binary classification. This dataset consists of data for 673 people in a CSV file. This data for each person includes:

1. Observation Number (we'll want to omit this)
2. Buy (binary target value, Y)
3. Income
4. Is Female
5. Is Married
6. Has College
7. Is Professional
8. Is Retired
9. Unemployed
10. Residence Length
11. Dual Income
12. Minors
13. Own
14. House
15. White
16. English
17. Prev Child Mag
18. Prev Parent Mag

We'll omit the first column and use the second column for our binary target Y. The remaining 16 columns provide our feature data for our observation matrix X.

2 Update Your Codebase

In this assignment you'll add gradient and backwards methods to your existing fully-connected layer and activation functions, and implement your objective functions. **Again, make sure these work for a single observation and multiple observations (both stored as matrices). We will be unit testing these.**

Adding Gradient Methods

Implement *gradient* methods for your fully connected layer, and all of your activation layers. When applicable, you may decide whether a class' gradient method returns a matrix or a tensor. The prototype of these methods should be:

```
#Input: None
#Output:  Either an N by D matrix or an N by (D by D) tensor
def gradient(self):
    #TODO
```

Adding Backwards Methods

Add the *backward* method to our activation and fully-connected layers! You might want to consider having a default version in the abstract class *Layer*, although we'll leave those design decisions to you. In general, the *backward* methods should takes as inputs the backcoming gradient, and returns the updated gradient to be backpropagated. The methods' prototype should look like:

```
def backward(self,gradIn):
    #TODO
```

Adding Objective Layers

Now let's implement a module for each of our objective functions. These modules should again each be in their own file with the same filename as the class/module, and implement (at least) two methods:

- *eval* - This method takes two explicit parameters, the target values and the incoming/estimated values, and computes and returns the loss (as a single float value) according to the module's objective function. This should work both for a single observation, and a set of observations.
- *gradient* - This method takes the same two explicit parameters as the *eval* method and computes and returns the gradient of the objective function using those parameters.

Implement these for the following objective functions:

- Squared Error as *SquaredError*
- Log Loss (negative log likelihood) as *LogLoss*
- Cross Entropy as *CrossEntropy*

Your public interface is:

```
class XXX():
  #Input: Y is an N by K matrix of target values.
  #Input: Yhat is an N by K matrix of estimated values.
  # Where N can be any integer>=1
  #Output: A single floating point value.
  def eval(self,Y, Yhat):
    #TODO

  #Input: Y is an N by K matrix of target values.
  #Input: Yhat is an N by K matrix of estimated values.
  #Output: An N by K matrix.
  def gradient(self,Y, Yhat):
    #TODO
```

3 Forwards-Backwards Propagate a Dataset

In HW1 you implemented forwards propagation for the Kid Creative dataset with the following architecture (note that I have added on a LogLoss layer):

Input→FC (1 output)→Logistic Sigmoid→LogLoss

Now let's do forwards-backwards propagation. Using the code shown in the *Objectives and Gradients* slides, perform one forwards-backwards pass. As you go backwards through each layer, report the *mean* gradient, as averaged over the observations. For this particular architecture you should report the mean gradient coming backwards out of:

1. Log Loss

   The mean gradient over the observations: 1.25697004

2. Logistic Sigmoid Layer

   The mean gradient over the observations: 0.31424251

3. Fully-Connected Layer

   The mean gradient over the observations:

   $[-2.94982167e-05, -2.52023890e-05, 3.15693841e-07, 9.83215445e-06,$
   $4.57662518e-06, 1.36992071e-05, 1.25418764e-06, 2.36893760e-05,$
   $-1.32203669e-05, 8.05898963e-06, -1.01927440e-05, 1.91218270e-05,$
   $-4.40532476e-06, -5.91548007e-06, -1.88423270e-05, -1.99929625e-05]$

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Source Code
3. readme.txt file

The readme.txt file should contain information on how to run your code to reproduce results for each part of the assignment.

The PDF document should contain the following:

1. Part 1: Your solutions to the theory question
2. Part 2: Nothing. We will unit test these, but again we encourage you do so yourself, particularly using the examples from the theory questions.
3. Part 3: The mean gradients coming backwards out of the four modules.