Assignment 3: CNN, Logistic Regression, Gradient Descent
Due Date: 11.59 PM 11/28/2023 Maximum Points: 120

Please follow the instructions. Failure to adhere to instructions will result in a penalty.

1 Question 1 (20 pts)

This is a log-linear game created by a professor of mine, Frank Ferraro. Make sure you are in Lesson 2.

It is a probabilistic model, that we have not covered in class. If in the dataset you have a total of 30 samples: **20** counts of object 1, and **10** counts of object 2, your objective is to match the actual counts with your predicted counts. For example, if your model predicts probability of object 1 is 0.7 and probability of object 2 is 0.3, and your dataset has 30 samples; You predict that you will have **0.7*30=21** counts of object 1 and **0.3*30 = 9** counts of object 2. Try to answer the following questions. The 'observed' counts are your actual counts of objects and 'expected' counts are your predicted counts of the same objects.

- When you increase the feature weight of 'circle', what happens to the expected counts of all the objects? (4 pts)
- When you click on the 'Show Gradient' under 'Hint', what do the red and blue bars under feature weights represent? (3 pts)
- When you click on 'Solve' under 'Hint', what are the values of all the features? Paste a screenshot. (2 pts)
- When you click on 'Solve' under 'Hint', what are the values of all the features, when regularization is selected as 'l1'? Paste a screenshot. (2 pts)
- Why are the values different? (3 pts)
- Let us say that the model is as follows: p(object) is the probability of the object. 'circle', 'striped', etc mean the feature weights of the corresponding features.
  $p(solidcircle) = \frac{e^{circle}+e^{solid}}{Z}$
  $p(stripedcircle) = \frac{e^{circle}+e^{striped}}{Z}$
  $p(solidtriangle) = \frac{e^{triangle}+e^{solid}}{Z}$
  $p(stripedtriangle) = \frac{e^{triangle}+e^{striped}}{Z}$
  Why do we need to divide these terms by Z? (2 pts) What would be an expression of Z, in order to ensure that it is a probability distribution? (Hint: Think about Softmax) (4 pts)

2 Question 2 (10 pts)

: For this question, consider the **Tensorboard Playground** For this question consider the dataset shown in Figure 1.

- Which features give you the best separation? Post a screenshot of your separation. (5 pts)
- What happens when you add a regularizer? Answer in terms of 'No of Epochs' and 'Training/Test Loss'. (5 pts)

Which features give you the best separation? Post a screenshot of your separation. (5 pts)

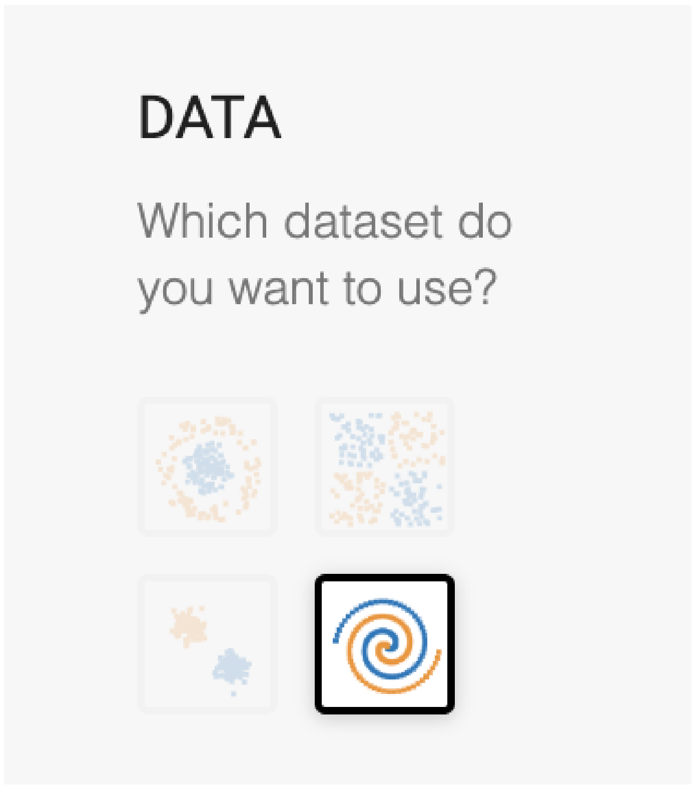

Figure 1: Use this dataset for Question 2

3 Queston 3: CNN Explainer (10 pts)

For this question, consider the **CNN Explainer**

- Observe the shape of each layer. By looking at these, tell me what is the kernel size, stride length, and maxpool window size? (5 pts)
- Click the '+' sign to add a new image. Upload the image of a **rattlesnake**, attached with the assignment. Note the output. Repeat this by attaching the image of a **coyote**. What is your observation? What can you say about the class 'Espresso'? (5 pts)

4 Question 4: Gradient Descent (20 pts)

We are going to implement gradient descent for a small dataset of 12 samples. It is defined in the data = [....].

Our model makes predictions in the following way: $y_p = \phi_0 + \phi_1 * x + \phi_2 * x^2$ This implementation is present in the starter code (**Gradient_Descent_skeleton.ipynb**). We can calculate the gradient using a trick known as **finite differences**. If we evaluate the function and then change one of the parameters by a very small amount and normalize by that amount, we get an approximation to the gradient, so:

$$\frac{\partial L}{\partial \phi_0} \approx \frac{L[\phi_0 + \delta, \phi_1, \phi_2] - L[\phi_0, \phi_1, \phi_2]}{\delta} \tag{1}$$

$$\frac{\partial L}{\partial \phi_1} \approx \frac{L[\phi_0, \phi_1 + \delta, \phi_2] - L[\phi_0, \phi_1, \phi_2]}{\delta} \tag{2}$$

$$\frac{\partial L}{\partial \phi_2} \approx \frac{L[\phi_0, \phi_1, \phi_2 + \delta] - L[\phi_0, \phi_1, \phi_2]}{\delta} \tag{3}$$

In the above equation, consider $\delta = 0.005$ (defined in the skeleton code) Recall that parameters are updated in the following way

$$\phi_i = \phi_i - learningrate(lr) * \frac{\partial L}{\partial \phi_i} \tag{4}$$

The skeleton code contains the implementation of the Loss, Data, and Model.You will have to implement 2 functions

- compute_gradient: Returns the gradient for each of the parameters ϕ_0, ϕ_1, ϕ_2
- change_in_loss: Prints the loss when parameters were not updated, and the loss after updating the parameters. Consider learning rate (lr) to be 0.01 (defined in the skeletal code)

5 Question 5: CNN (30 pts)

Inside the lab3_dataset/q5/ folder, **unzip** the following files:

- 1. The input training images are in : cat_dog_data/train.zip
- 3. The test data is: cat_dog_data/test.zip

After unzip, you will get cat and dog folders inside train and test. load_images function is provided.

Complete CNN model in the given **q5_starter.ipynb** and submit it. Please check the comments inside the code for instructions. Write the code where you find the comment "# write your code here."

6 Question 6: Logistic Regression (30 pts)

Inside the lab3_dataset/q6/ folder, you will find the following files:

- 1. The input training data is : X_train_q6.csv
- 2. The input training label is : y_train_q6.csv
- 3. The test data is: X_test_q6.csv

Given the above dataset of graduate admissions, which includes various features like GRE score, TOEFL score, university rating, statement of purpose strength, letter of recommendation strength, undergraduate GPA, research experience, and the corresponding labels (labels are either 0 or 1) indicating 'Chance of Admit':
Instructions:

- Complete the **q6_starter.py** file.
- logistic_regression_equation(): Use the the equation

$$\hat{y} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n)}} \tag{5}$$

  to calculate the 'Chance of Admit' . Write the code to calculate the log loss (log_loss function is given) value for the training set for $w_0 = w_1 = w_2 = .. = 0.001$.

$$LogLoss = \frac{1}{m} \sum_i -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \tag{6}$$

- logistic_regression_library(): Train a logistic regression model (**using a library like "sklearn.linear_model.LogisticRegression"**) on the training data. Predict the 'Chance of Admit' for the provided test dataset. Note that the test labels are not provided, and you have to make predictions based on the logistic regression model trained. So no accuracy or loss need to be calculated for this function.
- Save and submit your predictions for X_test_q6.csv in a file named **y_predict_q6.csv** and submit your code file **q6_starter.py**.

Extra Instructions:

- Write your name, email, student id as comment inside each code and pdf/photo files.
- For question 1,2,3 submit pdf or image file. Do not submit any doc files.
- Do not submit files with different names. Use the mentioned name for the code and output file.

- Do not submit any given dataset files.
- Do not submit multiple files of the same code or the same output CSV.
- Please check your submitted CSV files. prediction CSV files should have only one column with a header.
- If you use google colab, upload the dataset folder with the same name. Do not change the names of folders or files. Example code:

  ```
  from google.colab import drive
  drive.mount('/content/drive')
  import pandas as pd
  X_train = pd.read_csv('/content/drive/My Drive/lab3_dataset/q6/X_train_q6.csv')
  # Before submission change back the path to 'lab3_dataset/q6/X_train_q6.csv'
  # X_train = pd.read_csv('lab3_dataset/q6/X_train_q6.csv')
  ```