Final Project: Gradient Descent CPS 580

Zackary McClamma

August 6, 2020

1 Code Operation Steps

1. Run GradientDescent.py in IDE or in command prompt (I used PyCharm for running and testing this program)
2. When running the program will first prompt the user to input a 1 to train the model or a 2 to test the model NOTE: If the model has not been trained and the user attempts to test the model it will prompt the user to train the model first
3. Training The Model
   (a) After selecting to train the model there will be a pop up window that displays the user's file directory, from there the user should navigate to where the training data file is stored and select that file
   (b) Once the input data file is selected the user will be prompted to input a convergence value. The convergence value should be less than one and also greater than zero, it is recommended that the user choose a value no smaller than 0.0001, because any value smaller takes a long time to process.
   (c) The model will now print out the number of iterations it took to reach convregence and the amount of time it took to run the *gradientDescent* function and the resulting theta values.
   (d) The user will then be prompted to input 0 to continue running the program or 1 to exit, selecting 0 will bring the user back to step 1. NOTE: If the user exits the program the model will have to be trained again before testing
4. Testing the model
   (a) If the user selects the option to test the model and the model has been trained then the program will produce a pop up window of the user's file directory system where they should navigate to and select the file containing the testing data.
   (b) After giving the testing data file the program will use the model to predict the labels according to the inputs of the testing data file, after running the model the program will display the accuracy with which it was able to predict the label.
   (c) The user will again be prompted to input 0 to continue or 1 to exit, selecting 0 will bring the user back to step 1

2 Learned Parameters

The learned parameters of this program are highly dependent on the convergence value given, I ran the program with the following convergence values and produced the results shown below.

- Convergence = 0.001
  - Iterations to convergence = 6919
  - Time to run gradient descent = 774.9931812286377 ms
  - $\theta_0 = -19.28208496385247$
  - $\theta_1 = -2.681445452271275$
  - $\theta_2 = 29.610017093508773$
  - Prediction accuracy with this model = 78.92061955469507%
- Convergence = 0.0001
  - Iterations to convergence = 478375
  - Time to run gradient descent = 285145.0538635254 ms
  - $\theta_0 = -7.281305624101816$
  - $\theta_1 = -81.16790636611564$
  - $\theta_2 = 45.378536269181566$
  - Prediction accuracy with this model = 89.43772990015765%

3 Additional Libraries Used

- **numpy**: From this library I use the array, log, and dot product functionality. The array is used to store the input data because it makes the operations on the data more concise and optimal in the code. The log function is used to calculate the theta values, and the dot product is used to quickly run the hypothesis calculation for all of th input values($\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$).
- **time**: This library is used to calculate the timing of training the model.
- **copy**: This is used to make a copy of varibles since the assignment operator in python passes by reference. This was needed to store the previous theta values in gradient descent so that the new values could be tested for convergence.
- **tkinter(Tk, filedialog)**: This library was used to open the pop up window to make a file selection