Assignment 2: Gradient Descent for Linear and Logistic Regression

Loyola Marymount University *Professor Alex Wong*
CSMI-533: Data Science and Machine Learning

Implement logistic regression and linear regression using gradient descent
There are many ways to train logistic and linear regression models. For this assignment we will be using gradient descent

We will implement the `GradientDescentOptimizer` class using the Gradient Descent Algorithm. We will also implement the `LinearRegressionGradientDescent` class and the `LogisticRegressionGradientDescent` class -- both of which will use the `GradientDescentOptimizer` class to update their respective weights.

We will apply the logistic regression models to the Wisconsin breast cancer dataset and the digits dataset (for classifying 7s and 9s).

We will apply the linear regression models to the Boston housing price dataset and the diabetes dataset.

A skeleton of the classes is provided in assignment2.py

You will complete the following functions for the assignment:

In the `GradientDescentOptimizer` class:

1) `__compute_gradients` : Returns the gradient of the logistic, mean squared or half mean squared loss
2) `update` : Updates the weight vector based on logistic, mean squared or half mean squared loss

In the `LogisticRegressionGradientDescent` class:

1) `fit` : Fits the model to x and y by updating the weight vector using gradient descent
2) `predict` : Predicts a label for each feature vector x
3) `score` : Predicts labels based on feature vector x and computes the mean accuracy of the predictions

In the `LinearRegressionGradientDescent` class:

4) `fit` : Fits the model to x and y by updating the weight vector using gradient descent
5) `predict` : Predicts a real value for each feature vector x
6) `score` : Predicts labels based on feature vector x and computes the mean squared error

Here is the sample output:

```
Results on Wisconsin breast cancer dataset using scikit-learn Logistic
Regression model
Training set mean accuracy: 0.9590
Testing set mean accuracy: 0.9649
Results on digits 7 and 9 dataset using scikit-learn Logistic
Regression model
Training set mean accuracy: 1.0000
Testing set mean accuracy: 0.9722
Results on Boston housing price dataset using scikit-learn Linear
Regression model
Training set mean accuracy: 23.2335
Testing set mean accuracy: 10.8062
Results on diabetes dataset using scikit-learn Linear Regression model
Training set mean accuracy: 2991.9850
Testing set mean accuracy: 1735.9381
Results on Wisconsin breast cancer dataset using our Logistic
Regression model
Training set mean accuracy: 0.0000
Testing set mean accuracy: 0.0000
Results on digits 7 and 9 dataset using our Logistic Regression model
Training set mean accuracy: 0.0000
Testing set mean accuracy: 0.0000
Results on Boston housing price dataset using our Linear Regression
model
Training set mean accuracy: 617.1287
Testing set mean accuracy: 369.2698
Results on diabetes dataset using our Linear Regression model
Training set mean accuracy: 29088.9673
Testing set mean accuracy: 28946.6889
```

Submission:
You will submit the following to Bright Space

1) <last_name>_<first_name>_assignment2.py

Grading:
I will be executing the assignment using the following command:

```
python <last_name>_<first_name>_assignment2.py
```

Your code must run for me to assign points!

Your assignment will be graded on:

1) the correctness of your implementation of the `GradientDescentOptimizer` class
2) the correctness of your implementation of the `LogisticRegressionGradientDescent` class
3) the correctness of your implementation of the `LinearRegressionGradientDescent` class
4) results of your `LogisticRegressionGradientDescent` class on the Wisconsin breast cancer dataset and the digits dataset
5) results of your `LinearRegressionGradientDescent` class on the Boston housing price dataset and the diabetes dataset

Note: your results should deviate no more than 5%-7% of the performance given by scikit-learn

Late Policy:
For each day the assignment is late, 50% of its worth will be deducted, e.g. 100% on time, 50% 1 day late, 25% 2 days late, etc.