ENGR 421/DASC 521: Introduction to Machine Learning
Homework 3: Discrimination by Regression
Deadline: April 13, 2023, 11:59 PM

In this homework, you will implement a discrimination by regression algorithm for multiclass classification using Python. Here are the steps you need to follow:

1. Your discrimination by regression algorithm will be developed using the following modifications to the linear discrimination algorithm with the softmax function we discussed in the lectures.

    a. Instead of the softmax function, you are going to use K sigmoid functions to generate $\hat{y}_{ic}$ values. Please note that, in such a case, the summation of $\hat{y}_{ic}$ values is not guaranteed to be 1. However, you are going to pick the largest value to predict the class label.

    b. Instead of minimizing the negative log-likelihood (i.e., $-\sum_{i=1}^{N}\sum_{c=1}^{K} y_{ic}\log(\hat{y}_{ic})$), you are going to use the sum squared errors as the error function to minimize (i.e., $0.5\sum_{i=1}^{N}\sum_{c=1}^{K}(y_{ic}-\hat{y}_{ic})^2$). Please note that you need to find the correct update equations for this modified model.

2. You are given a multivariate classification data set, which contains 70000 handwritten digit images of size 28 pixels × 28 pixels (i.e., 784 pixels). You are given two data files:

    a. `hw03_data_points.csv`: handwritten digit images,

    b. `hw03_class_labels.csv`: corresponding class labels.

3. Divide the data set into two parts by assigning the first 60000 images to the training set and the remaining 10000 images to the test set. (10 points)

4. Implement a sigmoid function that calculates K sigmoid outputs for the given data points using the model parameters. (10 points)

5. Implement a one-hot encoding function that converts the given class labels into binary vectors of size K. (10 points)

6. Implement two gradient functions that calculates the partial derivatives of the error function with respect to the model parameters. (20 points)

7. Implement the discrimination by regression algorithm using the implemented sigmoid and gradient functions. (30 points)

```
W, w0, objective_values = discrimination_by_regression(X_train, Y_train,
                                                       W_initial, w0_initial)
print(W)
print(w0)
print(objective_values[0:10])
```

```
[[ 0.00365564  0.00606622 -0.0088745  ...  0.00485218 -0.00185243
  -0.00740226]
 [-0.00540071  0.00437433  0.00688369 ...  0.00157982 -0.00679959
```

```
   0.00848463]
 [-0.00817134 -0.00092595  0.00148658 ... -0.00699237 -0.00908637
   0.00426942]
 ...
 [-0.00946079 -0.0015333  -0.00975894 ...  0.00032909 -0.00258434
  -0.00681034]
 [-0.00492893  0.00776927 -0.00530092 ... -0.00368244  0.00115995
   0.00444868]
 [ 0.00567408 -0.006186    0.00702124 ... -0.006862    0.00447864
  -0.00451847]]
[[-0.06684353 -0.42498154 -0.44906758 -0.23974295 -0.1976174  -0.34234379
  -0.17061777 -0.71142747 -0.46719659 -0.3503129 ]]
[75843.6033738  27960.77145189 27710.32303216 27395.03035014
 26994.9705725  26492.71564956 25888.05578632 25205.98811306
 24458.53927375 23622.59096461]
```

8. Calculate the predicted class labels for the data points in your training and test sets using the learned model parameters. (10 points)

```
y_hat_train = calculate_predicted_class_labels(X_train, W, w0)
print(y_hat_train)

y_hat_test = calculate_predicted_class_labels(X_test, W, w0)
print(y_hat_test)
```

```
[ 1  1  1 ... 10 10 10]
[ 1  1  1 ... 10 10 10]
```

9. Calculate the confusion matrices for the data points in your training and test sets using the true and predicted class labels. (10 points)

```
confusion_train = calculate_confusion_matrix(y_train, y_hat_train)
print(confusion_train)

confusion_test = calculate_confusion_matrix(y_test, y_hat_test)
print(confusion_test)
```

```
[[6511  101   50   38   91   38  115  207   39    0]
 [  35 5040  191   37   46   55   98   86   52   22]
 [  18   95 5233    8  307    6   28  240  120   15]
 [   5  133   11 5232  131   43  103   61  280   15]
 [  50    8  203    7 4123   96    7  266   48   10]
 [  13  156   64   85  158 5583    6   60    7   76]
 [  12  138   91   13   40    3 5633   44  253    7]
 [  85  183  155   76  251   41   23 4679   63   84]
 [  12   48   89  331  114    2  201  149 5028    6]
 [   1   56   44   15  160   51   51   59   59 5688]]
[[1106   12    2    6    6    3   21   13    9    0]
 [   2  872   21    5    6    4   26   13    9    2]
 [   4   22  892    0   50    1    4   34   13    2]
[   1   18    1  888   26   11   11   14   52    0]
[   1    1   29    1  691   15    0   41   15    0]
[   4   20    8   14   24  903    4   15    3   13]
[   1   24   17    1   13    1  922   17   30    1]
[  16   42   20    9   44    4    4  800    6    7]
[   0   10   14   57   12    0   33   16  856    0]
[   0   11    6    1   20   16    3   11   16  955]]
```

What to submit: You need to submit your source code in a single file (`.py` file). You are provided with a template file named as `0099999.py`, where `99999` should be replaced with your 5-digit student number. You are allowed to change the template file between the following lines.

```
# your implementation starts below

# your implementation ends above
```

How to submit: Submit the file you edited to Blackboard by following the exact style mentioned. Submissions that do not follow these guidelines will not be graded.

Late submission policy: Late submissions will not be graded.

Cheating policy: Very similar submissions will not be graded.
