ENGR 421/DASC 521: Introduction to Machine Learning
Homework 2: Discrimination by Regression
Deadline: November 9, 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 clothing images of size 28 pixels $\times$ 28 pixels (i.e., 784 pixels). These images are from ten distinct classes, namely, t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. You are given two data files:

   a. `hw02_data_points.csv`: clothing images,

   b. `hw02_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])

[[-0.00927313 -0.02139474 -0.00617494 ... -0.0242896   0.00962816
```

```
  -0.01155617]
 [-0.01018683 -0.02156036 -0.00459622 ... -0.02461277  0.00911796
  -0.00997184]
 [-0.01051949 -0.02205936 -0.00510674 ... -0.02544551  0.00887556
  -0.01037016]
 ...
 [-0.00663916 -0.02030309 -0.01106297 ... -0.0220678   0.00963036
  -0.01157551]
 [-0.00909753 -0.02062855 -0.00619399 ... -0.02403018  0.00946359
  -0.01073183]
 [-0.00903548 -0.0225592  -0.00460063 ... -0.02534667  0.0101888
  -0.01130968]]
[[-0.01046386 -0.02157114 -0.00547472 -0.02174966 -0.01136625 -0.02936662
  -0.00954295 -0.02382486  0.01059064 -0.0100504 ]]
[74740.34186581 29850.23951448 29848.0611521  29845.81601964
 29843.50093399 29841.11250389 29838.64711247 29836.10089806
 29833.46973297 29830.74919999]
```

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)

[ 9 10  5 ...  9  9  8]
[1 2 3 ... 9 9 5]
```

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)

[[4867   64  175  390   45    0 1568    0   29    0]
 [  41 5554    7  107   58    0   21    0    7    2]
 [  76   91 3794   48  943    0 1015    0   88    0]
 [ 536  216   45 4976  381   13  304    0  108    2]
 [  42   41 1226  182 4054    0 1257    0   32    0]
 [ 269   26  458  182  242 4525  650  511  307  162]
 [  42    3  187   89  182    0  961    0   32    1]
 [   4    0    3    0    0 1056    3 4936  129  306]
 [ 120    4  105   24   95   58  220    6 5247    2]
 [   3    1    0    2    0  348    1  547   21 5525]]
```

```
[[791   9  34  49   3   0 271   0   3   1]
 [  7 928   1  23  10   0   5   0   1   0]
 [ 22  19 643  11 140   0 144   0  14   0]
 [ 90  29   7 844  53   2  55   0  20   0]
 [  4  10 195  28 712   0 220   0   5   0]
 [ 46   3  65  32  39 740 125  85  53  28]
 [  5   2  34  11  32   0 147   0   4   0]
 [  2   0   0   0   0 182   0 803  20  56]
 [ 33   0  21   2  11  13  33   0 877   0]
 [  0   0   0   0   0  63   0 112   3 915]]
```

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.
