CS6375: Machine Learning (Spring '23)
Programming Assignment #1 Logistic Regression

Group Members
1. Vikas K Ainapur (VKA220000)
2. Swetha Vinay Argekar (SVA210001)

Problem: Implement the **Logistic Regression** algorithm. The code skeleton as well as data sets for this assignment can be found on e-Learning.

Data Set: The data set (in the folder ./data/) is obtained from the UCI Repository and are collectively the MONK's Problem. These problems were the basis of a first international comparison of learning algo- rithms[1]. The training and test files for are named monks-3.train and monks-3.test. There are six attributes/features (columns 2–7 in the raw files), and the class labels (column 1). There are 2 classes. Refer to the file ./data/monks.names for more details.

a. (**Reporting Error Rates**, 30 points) For the following learning rates {0.01, 0.1, 0.33} and iterations {10, 100, 1000, 10000} fit Logistic Regression Models using the MONKS-3 train set and compute the average training and test errors. Report what the best parameters are for the model.

Solution:

We obtained below average training and test errors.

Learning Rate	Number of Iterations	Average training error	Average test error
0.01	10	0.4918032786885246	0.5277777777777778
0.1	10	0.319672131147541	0.3611111111111111
0.33	10	0.22131147540983606	0.25462962962962965
0.01	100	0.3114754098360656	.3587962962962963
0.1	100	0.20491803278688525	0.23148148148148148
0.33	100	0.1885245901639344	0.2199074074074074
0.01	1000	0.20491803278688525	0.23148148148148148
0.1	1000	0.18032786885245902	0.2037037037037037
0.33	1000	0.20491803278688525	0.19907407407407407
0.01	10000	0.18032786885245902	0.2037037037037037
0.1	10000	0.1885245901639344	0.18518518518518517
0.33	10000	0.18032786885245902	0.18287037037037038

We observe that as the learning rate increases and number of iterations increases, the error value decreases. The best parameter is chosen to be the one with the least error.

Hence, the learning rate = 0.33, Num of iterations: 10000 is the best parameter for the model in the question.

b. (**Saving the Model**, 50 points) Retrain the models on the best parameters and save it as a pickle file in the format 'NETID_lr.obj'. The object file will be loaded for grading and the class functions will be individually tested.

Solution: The object file has been attached.

Code snippet:

```
iter = 10000
a = 0.33
lr =  SimpleLogisiticRegression()
lr.fit(Xtrn, ytrn, lr=a, iters=iter)
train_pred = [lr.predict_example(lr.w, x) for x in Xtrn]
train_error = lr.compute_error(ytrn, train_pred)
train_errors.append(train_error)
test_pred = [lr.predict_example(lr.w, x) for x in Xtst]
test_error = lr.compute_error(ytst, test_pred)
test_errors.append(test_error)
accuracy(train_pred,ytrn)
# best coefficients for the above model
print(lr.w)
```

```
0.819672131147541
[ 7.37882228  0.03728872 -2.06532278 -0.16645992 -0.12115999 -1.49183085
  0.55856625]
```

```
netid = 'vka220000'
file_pi = open('vka220000_model_1.obj', 'wb')  #Use your NETID
pickle.dump(lr, file_pi)
```

c. (**scikit-learn**, 10 points) For monks datasets, use scikit-learns's default Logistic Regression Algorithm2. Compute the train and test errors using sklearn's Logistic Regression Algorithm. Compare the results with the version you implemented and speculate on why there is a difference in performances between the two algorithms. **Do not change the default parameters**.

Solution:

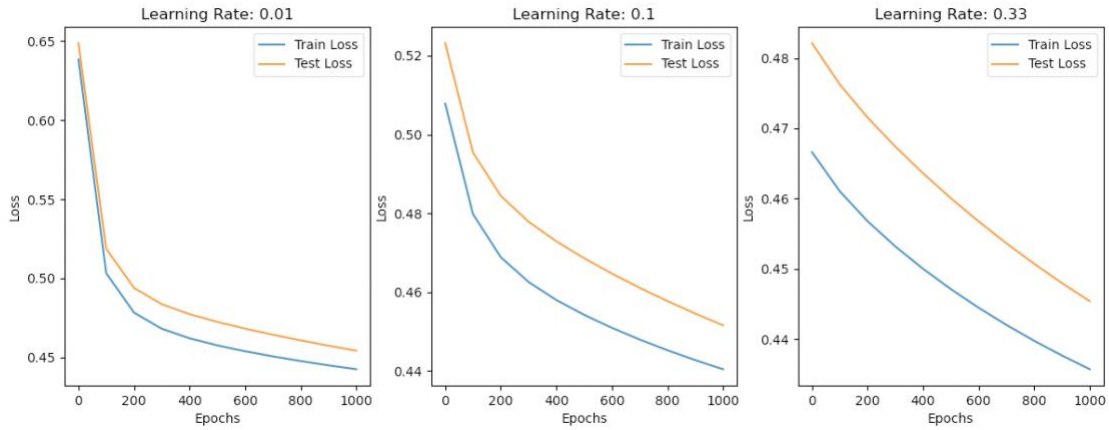
	Average training error	Average test error
Our values	0.18032786885245902	0.18287037037037038
Scikit values	0.16393442622950816	0.1782407407407407

Observations:

- We notice that as the Scikit-learn's logistic regression algorithm has lower error values for both training and test error. We conclude that the Scikit-learn's logistic regression has better performance than our custom class. This might be due to the variation in the manner in which features pre-processing and feature engineering is implemented. It might be due to the fact that the Scikit has a better optimization algorithm leading to better performance.

d. (**Plotting curves**, 10 points) For each of the learning rates, fit a Logistic Regression model that runs for 1000 iterations. Store the training and testing loss at every 100th iteration and create three plots with epoch number on the x-axis and loss on the *y*-axis.

Solution:



.