Declan Callisto
CSCI-467
Project 4 Report

Pipeline Description

For this project, I decided to implement a logistic regression classifier with gradient descent. I decided to use this model because after the in-class lectures about logistic regression and gradient descent earlier in the semester, I was pretty lost on how it worked. Because of this, I looked up online resources to help improve my understanding. In the process of trying to get a better grasp on the concepts, I ended up finding the material really interesting, so I decided to give it a try on my own.

While I wrote all of my own code for this assignment, I would be remiss if I did not cite Martín Pellarolo's implementation of a logistic regression classifier on the website Medium. His post gave me a much better understanding of exactly how gradient descent works. My gradient function to calculate the direction of greatest decrease in loss is analogous to his, with the one difference being that he doesn't round his probability matrix before subtracting the correct training labels. I'm not entirely sure why he does this, but I ran my model using his implementation as well, and got only slightly lower accuracies.

To pre-process the data, I appended all of the training and test features with a final column of zeros, so that a bias term could be included in the vector of weights.

Results

Below are accuracy results, with rows corresponding to the number of iterations and columns corresponding to the hyperparameter alpha. DR stands for "Didn't Run".

	0.001	0.01	0.025	0.04	0.07
125	0.6498	0.6706	DR	0.6862	0.6807
250	0.7027	0.7035	0.6992	0.6977	0.6957
500	DR	0.7605	0.7562	0.7543	DR
600	DR	0.7455	DR	0.7282	DR

One obstacle I ran into was trying to figure out the best number of iterations for my model. After discussing this problem with my classmate Alex Nagel, we realized I could calculate my model accuracy on the training data after each iteration in order to find the best weights across all iterations, rather than just using the weights at the end. This improved my accuracies across all combinations of alphas and iterations tested.

Using the best combination found of iterations and alpha (500 and .01), my class accuracies were:

Class 1:	.9494
Class 2:	.9809
Class 3:	.8914
Class 4:	.9749
Class 5:	.8889
Class 6:	.9930
Class 7:	.8940
Class 8:	.9813
Class 9:	.9747
Class 10:	.9805

My confusion matrix is attached as a separate CSV.