Nate Boldt
COMP479
Professor Dligach
Nov 14, 2021

Homework 4 Report

For this homework, I decided to use a brand new data set regarding sonar data points in determining if the obstacle ahead is a rock or an explosive mine. The data set was manually split by me into an 80% training set and a 20% test set(s).

For model performance metrics on this data, the F1 score is a better representation than the accuracy. Because the data set is fairly imbalanced (more rocks than mines) and the 'cost' of a false "rock positive" is very high (don't want to run into a mine!). A logistic regression classifier was used as the main model for both cases in the homework.

First, the data is ingested into the python script, and we pre-process our data set. In this case, the pre-processing was very easy and only required swapping out binary values for our two classes of labels. Next, the process changes quite drastically between the two model implementations between the n-fold cross validation and the grid search procedure.

For n-fold cross validation, some additional pre-processing occurs by determining what the folds look like for the input 'n' parameter. We can figure out how many rows 'X' go into a single fold by taking the quotient of the number of rows in our training set and our 'n' value. In the case where 'n' is 10 and our dataset has 159 rows, we populate a fold with 15 rows and 9 rows go unused. Then, we start the first n-fold by randomizing/shuffling our training data and selecting the top 'X' rows in the set – and removing them from ever being selected again. Then, we can train on the non-'X' testing set within the training set and test on those 'X' values. Looping through the many possibilities of training vs test sets until each row has a turn being in the test set gives us 10 model trainings and evaluations. Our best f1 score was 0.93, and our worst was 0.67, with all values averaging out to 0.72. These values will differ on each run because of the induced randomization of the data sets.

For the grid search implementation, we dig into two different hyperparameters, 'C' and the 'tolerance' of the classifier. The grid search has a fairly straight forward process – simply run the model through a series of combinations of those two hyper parameters and evaluate the performance. For simplicity, I used the same value sets for both parameters of [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, and 100.0]. The best performing model according to the f1 score is a 'C' value of 100 and a tolerance of 0.1. These numbers surprised me a bit – I had thought the C value would be lower and tolerance would be higher. To examine this a bit further, I calculated the accuracy in the grid as well, and that combination also had the highest accuracy. Some models did not even converge – most of these were due to low C values. The worst performing models that did actually converge also trend lower on the C values. Overall, it is fair to conclude that in this data set, a higher 'C' value is beneficial to performance.

In order to evaluate this best model against data it hasn't seen before, we also ran it on the test set. The f1 score was lower than the training data – 0.66. The accuracy was much lower at 49%. Across both experiments in n-fold and grid search, we can see that we would not want to use this model in a real application if we're trying to avoid mines!