CS165B: Intro. to Machine Learning, Fall 2020
Programming assignment # 2

Due: Sunday 11:59pm, Nov. 8th

IMPORTANT NOTE You can discuss the problems with your TAs and instructor, but all answers and codes written down and turned in must be your own work.

The programming assignment is for you to extend/modify the linear regressor in assignment #1 based on iterative stochastic gradient descent (SGD) with regularization into a binary logistic regressor and compare the performance of your logistic regressor with that of sklearn's SVM. You must use Python for this assignment (so autograder on gradescope can produce outputs and display you on the leaderboard), and you must implement the iterative solver yourself (not calling an existing package). The functionalities that your solver must support include the following:

- A 2D grid search with one dimension being the learning rate α and the other dimension being the regularization weight λ.
- For efficiency and robustness consideration, you are allowed to use a minibatch size different from 1 like in program #1. Commonly used minibatch sizes are 8, 16, 32, etc. (e.g., the default value in Tensorflow is 32). You are allowed to hard code this number in your program.
- You must perform n-fold validation with n equals three. That is, you divide the training data randomly into three equal parts, you use two parts for training and the remaining part for validation and perform such a learning process three times (each time with a different third component reserved for validation). After you are done, you pick the best parameters among the three trials and perform one last training process using all available data (with maybe 10% data reserved for validation) to give a final "polish" to the results.
- You should use the augmented feature vector $[x, 1]$ so that weights and bias are treated in a uniform way as $[w, w_o]$, where w_o replaces the bias. Like we discussed in the discussion session, the new encoding will give you the same results but now you only need to implement one equation for updating all the weights without having to implement a separate equation for updating the bias.

For comparison purpose you should run through the same data with an SVM. You do NOT need to implement SVM yourself, but can call the one in sklearn. The exercise is to compare the run time and performance of a logistic regressor based on gradient descent with that of the SVM based on maximizing margin.

We will use the sklearn's breast cancer data set which is relatively small (569 samples total, with 30 features and 2 classes - benign vs. malignant). `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html`. The restriction of dataset size is necessary as SVM training (with kernel) can take a long time to complete if there are too many training samples.

The input/output parameters of your logistic regression function and the phases of operation are similar to those in program#1, so please consult program #1 handout for more details. You should base your program #2 on your program #1 (but remember to make a copy first so you have a record of your program #1 just in case). Again, the program skeleton is provided for you.

It is highly recommended that you implement a graphing function that graphs the training and validation errors as functions of iterations. Your program should finish its training phase in a "reasonable" amount of time on the given training data. By "reasonable," we mean less than half an hour on Gradescope. The codes you turn in must NOT save any parameters (except minibatch size) as the Gradescope will test your program from scratch; that is, your logistic regressor must perform a full grid-search from a random starting search point for w and b.

Any other specifications not following the above are invalid. While it is possible to design the API differently, for consistency you must follow the specification given above.

Submission

- Only submit hw2.py to gradescope, no other files will be accepted and please don't modify the filename
- Please check both numpy array shape and value from model.testing
- Your program must be finished within 30 mins Otherwise it will be timeout
- Leave the auto-testing part as it is given. Only changes the parameters specified in skeleton code