Tutorial Assignment III

Instructions: Please read the below instructions carefully to complete this assignment.

1. This assignment has three parts (Part I, 2, and 3) and will be graded out of 25 points.
2. Please work on this assignment individually.
3. The boilerplate code of the assignment has been provided. Please submit all your code in one python (*.py) file; name it as <*your_mac_id*>_part1-2-3.py.
4. Please submit your report (*.pdf) file; name it as <*your_mac_id*>_report.py.
5. Submit these files on Avenue.
6. Please DO NOT upload ANY datasets. The datasets you upload will NOT be used for assignment evaluation.
7. Please import only the below functions in your code. The assignment will NOT be graded if any other function from any other library is used.

```
import numpy as np
import pandas as pd

from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score
from sklearn.utils import shuffle
```

8. The coding assignment will be evaluated on both **functionality** and **code quality**.
    a. Functionality refers to correctness of solution, sanity checks, stress tests, robustness, catching edge cases, comprehensiveness in code architecture, conciseness, and performance on the held-out test set (where applicable).
    b. Code quality refers to cleanliness/ readability, well-chosen data structures, program design for optimal efficiency, and clear and meaningful comments existing for *at least* 75% of the lines of code.

9. Please do not hard code any aspects of your model, except model parameters.

10. Please do not use AI to generate your code. For this class, it will count as academic integrity violation. While AI is good at generating text, it is still not good at producing quality code for building ML models. I have observed several students make fundamental mistakes on their ML models because they prompted AI to complete their task. *Most importantly, to know whether AI has given you the correct solution, you should be able to know what the correct solution is*. Using AI to learn how to build AI is a dystopian idea that will harm your learning and mastering AI.

Context: Breast Cancer prediction has been a widely studied problem in the medical community. We have discussed this problem several times in the class. Please refer to class notes or ISLP for more details on why this prediction task is particularly challenging.

Challenge: We want to predict breast cancer in patients using a small data set of ~600 samples.

Dataset: The dataset has been downloaded from this source and shared to you as csv file.

Part I (Max 8 points)

Goal: Build a Support Vector Machine (SVM) Classification model to reliably classify data samples corresponding to Malignant and Benign tumors. A preliminary boilerplate code has been provided to you. This was also shared with you during week 3 when we discussed SVM. Your goal is to augment that code using a regularization technique within SVM. We have discussed L1 and L2 regularization techniques in class. Another popular regularization technique is **early stopping**, which limits the number of iterations while training the model. Early stopping is implemented as follows:

1. Execute gradient descent for 1 iteration.
2. Compute loss using the model's loss function.
3. Compare loss for the current iteration with loss in the previous iteration.
4. If the difference is beyond a specified loss threshold, go to step 1, otherwise stop the gradient descent process.

In this part of the assignment, implement 1) early stopping for the given SVM classifier (lines 103 onwards in boilerplate), 2) along with SVM's loss function (lines 76-80 in boilerplate). The function `stochastic_gradient_descent()` must print the number of iterations your early stopping algorithm takes to achieve optimal performance. The **output** of this part should be 1) the loss function plotted for every $1/10^{th}$ of the epoch (e.g. if your epoch size is 5000, plot the loss after every 500^{th} iteration) for both training loss and validation loss, 2) the accuracy of your model performance on a **held-out validation set**. This time, you will select your own held-out validation set, which should be at least 20% the size of the complete dataset selected through random sampling (do not hard code this part). How to plot the validation loss? First, train the model for 1 iteration, and then compute the loss using the predicted values and actual values of the observations in held-out validation set for that model. Repeat the process for all epochs. Therefore, in summary, you validate the model as you train it.

A successful early-stopping algorithm will reach the lowest possible train error before it plateaus or validation loss increases. This means, your training function must also **plot** your train and validation loss, well beyond your stopping iterations to verify whether your early stopping algorithm is yielding appropriate performance. You must also carefully select the loss threshold that aligns with your loss function. This number is typically of the magnitude

of 10^{-2} or 10^{-3}, but it largely depends on how you are computing the loss. You may also increase the window of comparison to improve your algorithm, i.e. instead of comparing with the previous iteration, you may compare with 2nd previous iteration.

Part II (Max 8 points)

Goal: Build a Support Vector Machine (SVM) Classification model to reliably classify data samples corresponding to Malignant and Benign tumors using **mini-batch** stochastic gradient descent. Therefore, instead of training 1 sample at a time, you should compute gradient descent over a small batch size of n samples. You are free to select n for optimal performance. Remember that in mini-batch gradient descent, the loss is computed by averaging the loss for that batch of n samples; everything else remains the same (same loss function, and gradient function). To optimize the performance of your minibatch, you *may* re-configure the model parameters, if necessary.

Show the performance metrics for minibatch gradient descent **compared** with stochastic gradient descent, in terms of accuracy, precision, recall (you need to implement these at line 142), and plotting the train and validation losses. The **output** of this part should be 1) loss function plotted for every 1/10th of the epoch (e.g. if your epoch size is 5000, plot the loss after every 500th iteration) showing both training loss and validation loss for stochastic gradient descent (computed in Part 1) and minibatches (computed in this part), 2) the final accuracy, precision, recall of your model performance on a held-out validation set. Use the same held-out test set as Part 1.

Part III (Max 9 points)

Goal: Build a Support Vector Machine (SVM) Classification model that can reliably classify data samples corresponding to Malignant and Benign tumors using the **smallest number of samples.** In class, we have discussed how different data sampling strategies like uncertainty sampling can help us minimize the cost of training. The goal of data sampling strategies is to select instances strategically around the decision boundary that best represents the dataset. In this part, you will attempt to integrate an active learning strategy in Support of Vector Machines. This technique was first proposed in 2000 at ICML, a premier venue for state-of-the-art ML research. The sampling strategy you will implement, which is a simplified version of the originally proposed strategy, goes something like this:

1. Consider an SVM classifier that has just been trained on a few labeled data (n), where n is a subset of the full dataset N. You may select the first n samples randomly.
2. We assume that the rest of the N-n dataset is not labeled yet, and it is uncorrelated to the predicted labels. This means, you may use the existing labels and need not gather new labels. But instead of training the model on all data instances, you will train them on a small subset of data instances.
3. Using the initially trained classifier, find the sample for which we get the **smallest** SVM loss (note that you have already implemented this loss function)

4. Use this sample to train the classifier further. You may use both stochastic or mini-batch GD for this task.

Implement the sampling strategy at line 133. For the main algorithm for training in part_3( ), you can use the following steps:

1. Train an initial classifier with random samples.
2. Perform prediction on all remaining samples.
3. Select the next probable sample, using the above sampling strategy.
4. Train the classifier further.
5. Repeat steps 2 – 4, till you get a satisfactory performance.

The **output** of this part should be 1) loss function plotted for every 1/10th of the epoch (e.g. if your epoch size is 5000, plot the loss after every 500th iteration) showing both training loss and validation loss for gradient descent (computed after every training iteration), 2) the final accuracy, precision, recall of your model performance on a held-out validation set. Use the same held-out test set as Part 1, 3) the number of samples you used for getting the optimal performance. (You may also try flipping step 3, instead of the smallest, use the **largest** error sample to see what happens).

Report: Please provide a short summary (2-4 lines each part) of your results of all 3 parts. Submit the pdf report as per the given instructions.