CSCE 421: Spring 2023 Homework 4

Assigned March 7, due on Tuesday, March 28, by 11:59 PM.

Submit your assignments (written+coding) seperately on gradescope. Please name your coding assignment as **'assignment4.py'**. Use the provided python template file, complete ONLY the functions (DO NOT edit function definitions, code outside the function, or use any other libraries).

A Few Notes:

- Coding assignments should be done only in Python.
- Please start early! This includes learning how to use Latex!
- You are required to fill out sections of the code marked "YOUR SOLUTION HERE".
- For solution please use the following template `https://www.overleaf.com/latex/templates/neurips-2022/kxymzbjpwsqx`.
- This is an individual assignment. While you are welcome to discuss general concepts together and on the discussion board your solutions must be yours and yours alone.
- **SHOW YOUR WORK.**

Problem 1: Implement Cross Entropy Cost

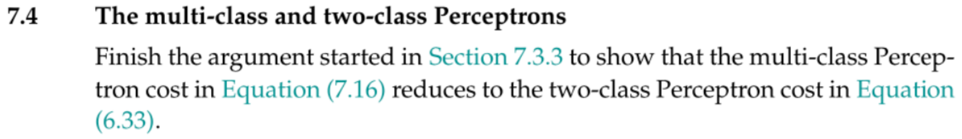
Use *2d_classification_data_entropy*.csv file provided in *Home work 4* on Canvas.

For this question, you will be implementing Cross Entropy Loss and model training based using the implemented loss function.

1. [Code] Fill in the function *read_classification_data*, which takes in the filename as string, and returns the data as 2 numpy arrays. Each with shape (number of rows in dataframe, 1) in the following order first row, second row.
2. [Code] Implement the *sigmoid* function and return the *sigmoid* of the given array s as a numpy number.
3. [Code] Fill in the function *cost_function* that returns the loss value(type: float) given the weight, bias, input, and target.
4. [Code] Fill in the function *cross_entropy_optimizer* to calculate the cross-entropy loss in each iteration and update weight and bias terms. Then the updated weight and updated bias terms as well as the costs saved in a list in the following order: updated_weight, updated_bias, costs.

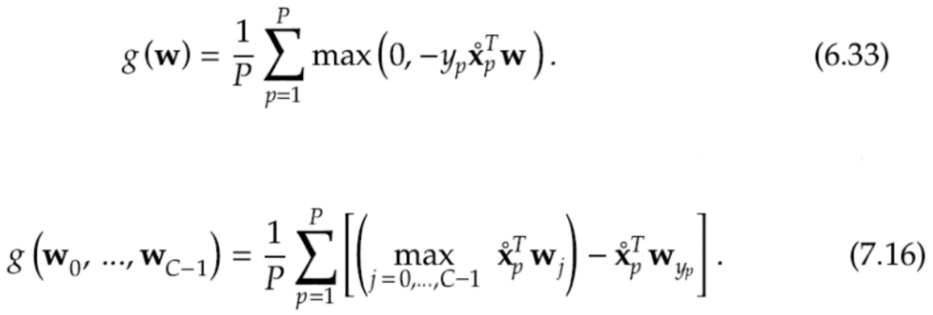
Problem 2: Multi-class perceptron

[Written] Complete question 7.4 from the textbook (Machine Learning Refined). (See Figure 1.)



7.4 The multi-class and two-class Perceptrons

Finish the argument started in Section 7.3.3 to show that the multi-class Perceptron cost in Equation (7.16) reduces to the two-class Perceptron cost in Equation (6.33).



$$g(\mathbf{w}) = \frac{1}{P}\sum_{p=1}^{P} \max\left(0, -y_p \mathring{\mathbf{x}}_p^T \mathbf{w}\right). \tag{6.33}$$

$$g\left(\mathbf{w}_0, \ldots, \mathbf{w}_{C-1}\right) = \frac{1}{P}\sum_{p=1}^{P}\left[\left(\max_{j=0,\ldots,C-1} \mathring{\mathbf{x}}_p^T \mathbf{w}_j\right) - \mathring{\mathbf{x}}_p^T \mathbf{w}_{y_p}\right]. \tag{7.16}$$

Figure 1: multi-class and two-class Perceptron cost

Problem 3: Complete ML Pipeline

Consider the binary classification that consists of distinguishing class 6 from the rest of the data points. For this question, you will be using SVMs combined with polynomial kernels to solve this classification problem. Use setimage train and test .csv files provided in Homework 4 on Canvas.

Problem 3-a: Data preprocessing

In this section, you will be preparing the dataset for the SVM classifier.

1. [Code] Fill in the function *read_sat_image_data*, which takes in the filename as a string, and returns the data as a pandas dataframe.

2. [Code] Fill in the function *remove_nan*, Remove nan values from the dataframe and return it.

3. [Code] Fill in the function *normalize_data* to normalize the dataframe and return the normalized dataframe as the output. (You can use StandardScaler from sklearn for Z_score normalization)

4. [Code] Fill in the function *labels_to_binary* to convert the labels into binary format. Change labels 1, 2, 3, 4, 5 to 0 and label 6 to 1. Return the updated dataframe.

Problem 3-b: Hyperparameter Tuning

In this section, you will be tuning the hyperparameters of the SVM classifier. Use SVMs combined with polynomial kernels to solve this classification problem. For each value of the polynomial degree, d = 1, 2, 3, 4, plot the average 5-fold cross-validation error plus or minus one standard deviation as a function of C (let the other parameters of the polynomial kernels be equal to their default values) ON THE TRAINING DATA. Report the best value of the trade-off constant C measured on the training internal cross-validation.

1. [Code] Fill in the function *cross_validate_c_vals*, which takes features dataframe (X), label dataframe (Y) as well as the number of folds and SVM hyperparameters (c_vals and d_vals). The function returns two numpy arrays for average and standard deviation of error values (order: (ERRAVGdc, ERRSTDdc))

2. [Written + Code] Fill in the function *plot_cross_val_err_vs_c*, to generate the graphs as described in the question. Please include the plots to the report document.

Problem 3-c: (Model Training and Testing)

Let (C*; d*) be the best pair found previously in the 5-fold internal cross-validation. Build a model for each pair on the full training data using the best c. Then test the model on the TEST DATA and plot the test errors for each model, as a function of d.

1. [Code] Fill in the function *evaluate_c_d_pairs*, which takes train features dataframe (X_train), train label dataframe (y_train), test features dataframe (X_test), test label dataframe (y_test) as well as the number of folds and SVM hyperparameters (c_vals and d_vals). The function returns four numpy arrays for Average MSE values, Average Support Vectors array, Average Number of Support Vectors that Violate the Margin array and Average Value of Hyperplane Margins array SuppVect (order: (ERRAVGdcTEST, SuppVect, vmd, MarginT))

2. [Written + Code] Fill in the function *plot_test_errors* to plot the error for each model as a function of d.

Problem 3-d: (Results Evaluation)

1. [Written + Code] Fill in the function *plot_avg_support_vec* to plot the average number of support vectors as a function of d. Add the plot to the report document.

2. [Written + Code] Fill in the function *plot_avg_violating_support_vec* to plot the average number of violating support vectors as a function of d. How many of the support vectors lie on the margin hyperplanes? Please provide your reasoning in the report document.

Problem 3-e: (Conceptual)

1. [Written + Code] Fill in the function *plot_avg_hyperplane_margins* Explain how the parameter d influences the model fit (margin size and number of support vectors).

2. [Written] Assume you were using an RBF kernel instead of a polynomial kernel, what would the parameter influence in terms of the model fit (margin size and number of support vectors. Explain your reasoning in the report document.

Note: function definitions and comments for each function provide a description of the problems the functions are supposed to address.