Linnaeus University
Introduction to Machine learning, 2DV516
Amilcar Soares
`amilcar.soares@lnu.se`

Assignment 2

Introduction

In this Assignment, you will use Python to handle several exercises related to gradient descent, linear regression, logistic regression, etc. The datasets used in the assignment can be downloaded in Moodle. The assignment deliverables are further divided into lectures.

Submission Instructions

All exercises are individual. We expect you to submit a Jupyter Notebook (i.e., pre-organized and provided through Moodle) and the .py files with the classes' exercise implementations. Your submission should include all the datasets and files we need to run your programs (we will run your notebook). When grading your assignments, we will, in addition to functionality, also take into account code quality. We expect well-structured and efficient solutions. Finally, keep all your files in a single folder named as follows: `username_A2` (e.g., cb223pg_A2) and submit a zipped version of this folder through Moodle.

In this assignment, you must implement all models as subclasses of **MachineLearningModel**. Since the class **MachineLearningModel** provides the abstract methods *fit*, *predict*, and *evaluate*, your implementations should provide implementations for such methods. Please check the documentation of **MachineLearningModel** to understand what these methods should do, as well as what their input parameters are and what they should return as results. You must also implement the classes **DecisionBoundary**, **ROCAnalysis**, and **ForwardSelection** provided to you. Please check their documentation to understand what these methods should do, what their input parameters are, and what they should return as results. All your implementations of such classes will be used throughout this assignment.

Linear and Polynomial Regression (Lecture 2)

Part 1: Classes Implementations

1. Implement a class **RegressionModelNormalEquation** that implements the abstract class **MachineLearningModel**. All methods should be implemented and properly documented. This class must work for polynomials of any *degree* (i.e., an input parameter that must be captured in the class constructor).

2. Implement a class **RegressionModelGradientDescent** that implements the abstract class **MachineLearningModel**. All methods should be implemented and properly documented. This class must work for polynomials of any *degree* and receive other parameters such as the *learning rate* and *number of iterations*.

3. Both implementations should be vectorized. **When implementing these classes, your vector β should start with all values as 0.** In implementing the **fit()** method, ensure

you track how the cost function evolved over the number of iterations (i.e., store it in an array you can retrieve after the model is built). This will be needed later in the assignment.

Part 2: Using your Implementations for Multivariate Regression Model (degree = 1)

1. In this part, you will use a reduced version of the Boston Housing Dataset (**housing-boston.csv**). We will use the first two input variables as the features in this part of the assignment. The last variable is the value to predict.

   - **INDUS:** proportion of nonretail business acres per town.
   - **RM:** average number of rooms per dwelling.
   - **MEDV:** Median value of owner-occupied homes in $1,000s.

   Read the dataset and store the values as vectors in the variables X_e and y. For this part of the assignment, the degree of the polynomial for your models must be 1.

2. Plot the dataset. You must plot two figures side by side (e.g., use the subplot method), with the value to be predicted as the $y - axis$ and each variable on the $x - axis$.

3. Use your implementation of the regression model with the normal equation (RegressionModelNormalEquation) and report:

   - The values for β.
   - The cost.
   - The predicted value for an instance with values for INDUS and RM equals $2.31, 6.575$, respectively.

4. Now, normalize the input features, run the regression model with the normal equation, and report the same items. The predicted values for this experiment should be the same, but the β values change. Why?

5. Now, you will work with your implementation of the gradient descent for any degree polynomial. In this part, you must compare how the cost function evolves by using your model using a non-normalized and a normalized instance of your RegressionModelGradientDescen class.

   - You must plot two figures (e.g., use subplots) side by side to show how the cost evolves over 3000 iterations with a learning rate of 0.001 using and not using feature normalization.
   - Describe what is happening and why this happens (i.e., using or not normalization).

6. Finally, find and plot a figure with the hyperparameter's learning rate and the number of iterations (using the normalized version) such that you get within a difference of 1% of the final cost for the normal equation using this dataset.

Part 3: Using your Implementations for Multivariate Regression Model (degree > 1)

In this exercise, we will use the file `secret_polynomial.csv`. The data consists of 400 x, y points generated from a polynomial with some Gaussian noise added.

1. Start by creating a procedure to split the dataset into training and test sets. The proportion must be 80% for training and 20% for testing. Show your procedure working by plotting a figure with 3 subplots. The first plot must be the dataset with all data. The second must be the training set, and the third the test set.

2. Now fit and plot (e.g., using subplots) all polynomial models for degrees $d \in [1, 6]$. Observe your figure and decide which degree gives the best fit. Motivate your answer.

3. To increase the confidence of your answer, you must divide the data into training and test sets and make repeated runs with shuffled data (at least 20 runs). You must decide on the best way to make this decision. By using this approach, what is your decision and why?

Logistic Regression (Lecture 3)

Part 1: Classes Implementations

1. Implement a class **LogisticRegressionModel** that implements the abstract class **MachineLearningModel**. All methods should be implemented and properly documented. This class receives parameters such as the *learning rate* and *number of iterations.* This class should be implemented in a way that works for two classes only (i.e., 0 or 1).

2. Implement a class **NonLinearLogisticRegressionModel** that implements the abstract class **MachineLearningModel**. All methods should be implemented and properly documented. This class must work for polynomials of any *degree* and receive other parameters such as the *learning rate* and *number of iterations.* This class should work for only two input variables (e.g., X_1 and X_2, as discussed in class). This class should be implemented in a way that works for two classes only (i.e., 0 or 1).

3. Both implementations should be vectorized. When implementing these classes, your vector β should start with all values as 0. In your implementation of the evaluate function, ensure you keep track of how the cost function evolved over the number of iterations. This will be needed later in the assignment.

4. Remember that $log(0)$ = undefined. Therefore, you may add a term epsilon = 1e-15 to prevent this in using the *np.log()* function. Simply add this term inside the function, and you will avoid such errors.

Part 2: Using your Implementations for the LogisticRegressionModel and the NonLinearLogisticRegressionModel.

You will now try to classify bank notes as fake (0) or not (1). This dataset `banknote_authentication.csv` contains 1372 observations and has 2 features and (in column 3) binary labels of either fake (0) or not (1). Feature data were extracted using a Wavelet Transform tool from images of both fake and non-fake banknotes.

1. Read and normalize the data. Plot the 2 variables in the x and y-axis. Use different colors to plot the classes (i.e., 0 or 1). You should plot two series to obtain this figure.

2. Separate a validation set with 20% of the data. We will call the remaining 80% a sub-dataset.

3. Your task now is to decide on a learning rate and the number of iterations that would work well for your implementations of the LogisticRegression and your NonLinearLogisticRegression. The degree for the NonLinearLogisticRegression model must be 2. Create a

figure for each model showing the learning rate and number of iterations and plot the cost function $J(\beta)$ as a function over iterations. This approach must use the sub-dataset (the 80%) from step 2. Discuss your choice for an appropriate learning rate and the number of iterations.

4. Create a method that, given the sub-dataset, it will randomly split it into 80% for training and 20% for testing.

5. Repeat 20 times your experiments (i.e., using different seeds) with the decided learning rate and the number of iterations (step 2) using 20 different sub-datasets generated by your method from step 4. Report all accuracies (i.e., percentage of correct classifications) reported by each model in these 20 runs as a box-plot. Compare and discuss the two models. Are they qualitatively the same? Why?

6. Now plot the decision boundary using a similar code to the one provided in class. You must plot the decision boundaries for the normalized data, use both models (LinearLogisticRegression and NonLinearLogisticRegression) and your choice of hyperparameters (step 3), totaling two figures. You must fit your model on the subdataset, but plot the validation dataset only in the figure. The models that were fit are the ones to be used to create the decision boundary. Report also the accuracies for the two models. Discuss your results (e.g., similarities, differences, etc) for accuracy and the decision boundary plots.

Model Selection and Regularization (Lecture 4)

Part 1: Classes Implementations

1. Implement a class **ROCAnalysis** that calculates the metrics: TP-rate, FP-rate, precision, recall (i.e., same as tp-rate) and f-score.

2. Implement a class **ForwardSelection** that implements the feature forward selection algorithm seen in class. This process must use 80% (i.e., fitting the data) of the data for training the models and 20% (i.e., predicting in unseen data) for testing. This method should optimize your problem regarding the TP-rate metric. You must use your implementation of the ROCAnalysis class.

Part 2: Using your implementations of ROCAnalysis and ForwardSelection

For this exercise, you will use the `heart_disease_cleveland.csv` dataset. The dataset contains 13 numerical features, and the last feature is the target variable, which we have to predict. The value of 1 means the patient is suffering from heart disease, and 0 means the patient is normal.

1. Start by normalizing the data and separating a validation set with 20% of the data randomly selected. The remaining 80% will be called the sub-dataset.

2. Use your implementation of forward selection to estimate a reasonable classification model. You must use your implementation of Logistic Regression in this assignment. The decision to make a reasonable number of iterations and learning rate is up to you but must be justified. Optimize the model selection to produce the best f-score. You must use the sub-dataset in your forward selection process. Report the features selected by this process and discuss your results. Was the process successful when compared to using all features?

3. Report the performance of the best model in the validation set regarding all statistics available in your ROCAnalysis class. Discuss your results regarding these metrics and what you can conclude from this experiment.