Homework Assignment 05

Due: Dec 07, 2023

1. Linear regression.

Open the provided .ipynb file and tackle the problems below. Before getting started, bring your normalize(X, mu=None, sigma=None) from Homework Assignment #3. Read the code blocks starting with “# Problem 1 supplements”. In this problem, you are provided and working with the Boston Housing dataset that were originally derived from information collected by the U.S. Census Service concerning housing in the Boston area. The dataset consists of 22 variables including:

Input attributes

CRIM: per capita crime rate by town

ZN: proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS: proportion of non-retail business acres per town.

CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX: nitric oxides concentration (parts per 10 million)

RM: average number of rooms per dwelling

AGE: proportion of owner-occupied units built prior to 1940

DIS: weighted distances to five Boston employment centres

RAD: index of accessibility to radial highways

TAX: full-value property-tax rate per $10,000 PTRATIO: pupil-teacher ratio by town

B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT: % lower status of the population

NOISE1-8: garbled values intentionally added for the purpose of the assignment

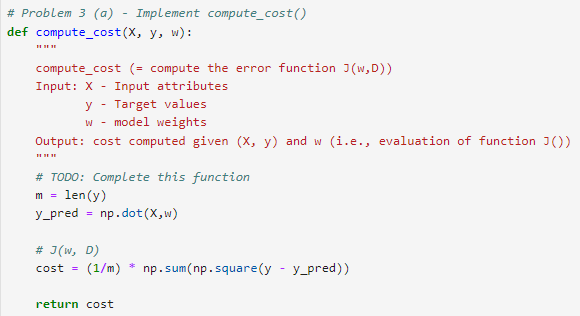
Target variable

MEDV: Median value of owner-occupied homes in $1000's

The goal of the dataset is to predict the price value of the houses using the given features.

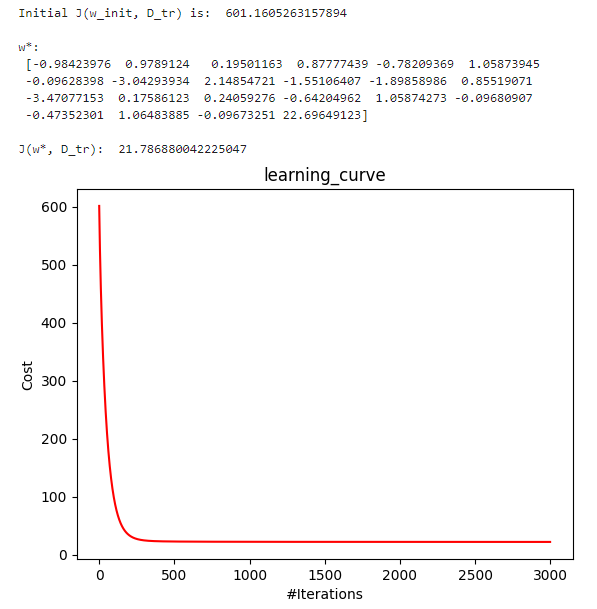
(a) (4 pt.) Implement compute\_cost(X, y, w) according to J(w, D) that we have discussed in class. Your complete function should work with the gradient\_descent() and train\_linreg() functions in the following code cell. Submit your complete function both in your report and in Jupyter notebook

Solution)



(b) (4 pt.) Find the code cell starting with “# Problem 1 (b) execute this code cell”, after you complete Problem (a). You will find a set of output that looks like below. Spend some time with train\_linreg() to make yourself understood to the learning (i.e., model fitting) process with the linear regression algorithm. Explain the numerical output and the learning curve displayed on the screen after running the code snippet (in your report). (You are highly encouraged to play with the training functions by modifying the learning rate and number of iterations. However, your results in the report should be from the model trained with learning rate (alpha) = 0.005 and number of iterations (n\_iters) = 3000.

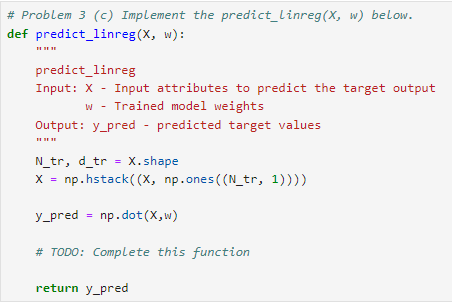
Solution)



The x-axis is the number of iterations (#Iterations), and the y-axis is the cost (Cost). The decrease in the graph indicates that the cost is decreasing as the learning progresses. It initially appears to be rapidly decreasing, and then converging after a certain point in time. w\_star is the final weight of the trained linear regression model. J(w\*, D\_tr) is the final cost of the training data, and a smaller value means that the model is predicting more accurately.

(c) (4 pt.) Find the code cell starting with “# Problem 1 (c)” and implement the predict\_linreg() function. Submit your complete function both in your report and in Jupyter notebook

Solution)



(d) (4 pt.) Execute the code cell starting with “# Problem 1 (d)”. Report and analyze the results in your report

Solution)

Prediction values were calculated using the given input (X) and the weight (w) of the trained model through the ‘predict\_linreg’ function. ‘y\_tr\_pred’ and ‘y\_ts\_pred’ represent the predicted values of the linear regression model for the training data and the test data, respectively. The ‘get\_rmse’ function was used to calculate the root mean square error (RMSE) between the predicted and actual values for the training and test data. The RMSE value is an indicator of the difference between the predicted and actual values, and it is judged that the smaller the value, the better the performance of the model. The 4.66763967930312 as the result of 'print(get\_rmse(y\_tr, y\_tr\_pred))' is the RMSE value for the training data and the RMSE value for the 5.0022230992847 test data as the result of 'print(get\_rmse(y\_ts, y\_pred))'.



If you look at the graph, the blue point is the value of training data, the orange point is the value of testing data, and the gray y=x straight line shows the ideal predictive performance. This graph shows the actual and predicted values of training and test data in a scatterplot. You can see from the graph that most of the training data values are clustered in the y=x straight line, but some are located far away in the y=x straight line. In comparison, the testing data values are closer to the y=x straight line than the training data. From the graph, the pattern of training data and testing data is similar, indicating that the model is being trained well.

(e) (2 pt.) What is the polynomial feature function? Write down the result of , where , and the degree is 2, in your report

Solution)

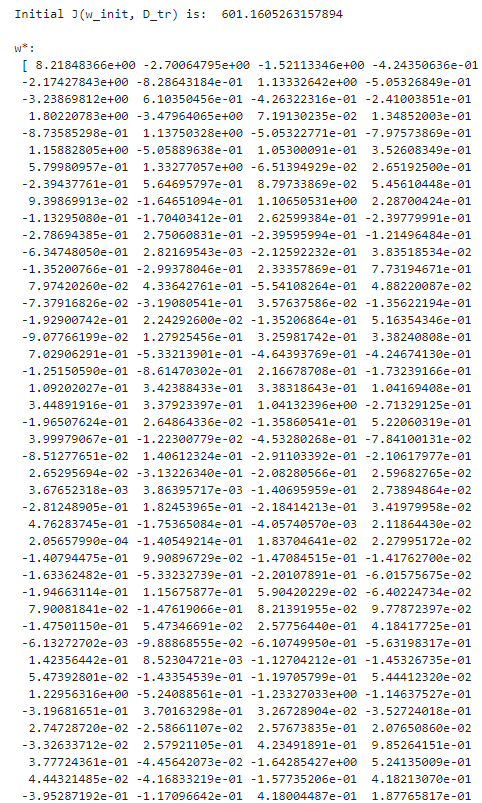
A polynomial feature function is a function that generates a polynomial of the underlying feature (original variable). This extends existing features to higher-order terms, allowing the model to learn more complex patterns.

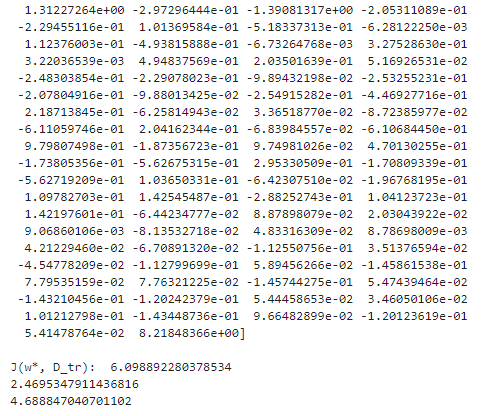
If s given, and the degree of the polynomial is 2. To convert to a quadratic polynomial using a polynomial feature function, you can calculate:

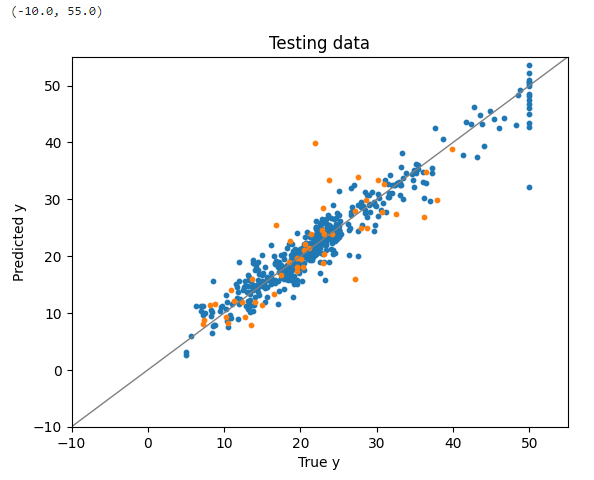
(f) (4 pt.) Complete the code cell starting with “# Problem 1 (e)” such that your complete code trains a linear regression model on the degree-2 polynomial features. Report and analyze the results in your report. How do you compare this model to the model obtained from Problem 3 (b)?

\* Note: You can use the polynomial feature function implemented in Scikit-Learn: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html>

Solution)

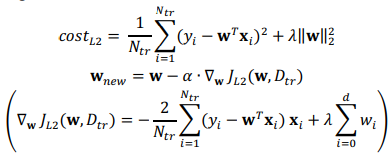




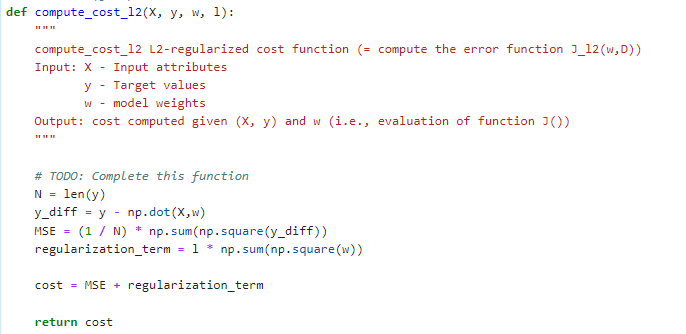


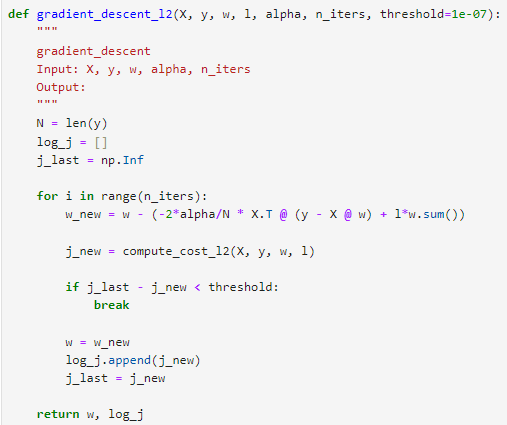
As a result of checking the rmse value and comparing it with the value in problem (d), the value of get\_rmse(y\_tr, y\_tr\_pred) decreased from 4.66763967930312 to 2.4695347911436816, while get\_rmse(y\_ts, y\_ts\_pred) also decreased from 5.00222309928477 to 4.68847040701102. Since a smaller value of RMSE means better performance of the model, we can see that the model with degree = 2 polynominal features performs better. The graph shows that the training data is located closer to y=x than the graph of problem(d). The testing data is also located closer to y=x than the graph of problem(d). Closer to the y=x graph, which represents the ideal predictive performance, shows that this model has improved predictive performance.

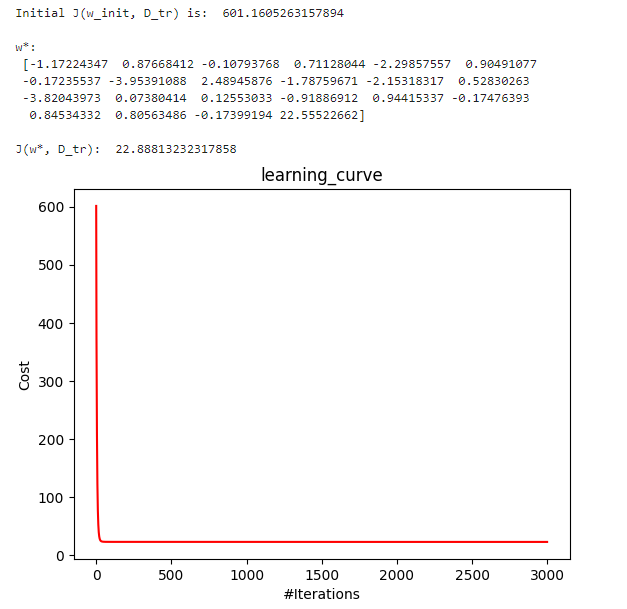
(g) (6 pt.) Find the code cell starting with “# Problem 1 (g, h)”. Complete functions compute\_cost\_l2() and gradient\_descent\_l2() by implementing how to compute variables cost and w\_new, respectively, for the L2-regularized (ridge) linear regression. Recall that

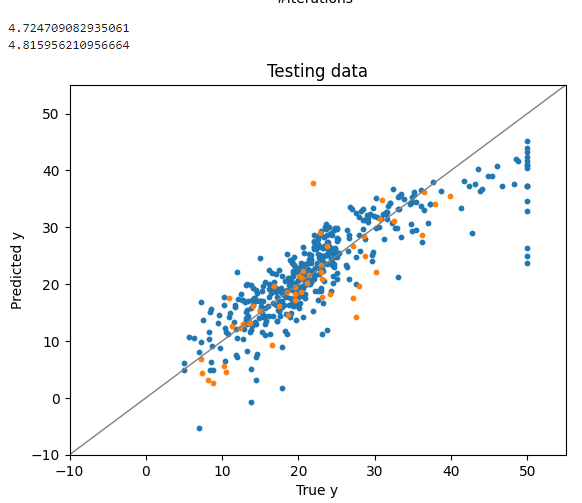


where represents the training dataset, denotes the number of training instances, denotes the input dimensionality, and is the regularization coefficient.









(h) (4 pt.) Execute the code cell that you have worked in Problem (g). Try the code cell with different combinations of parameter values l (lambda), alpha, and n\_iters to acquire better results. Report and analyze your best results in your report. How do you compare this model to the model obtained from Problem (b)?

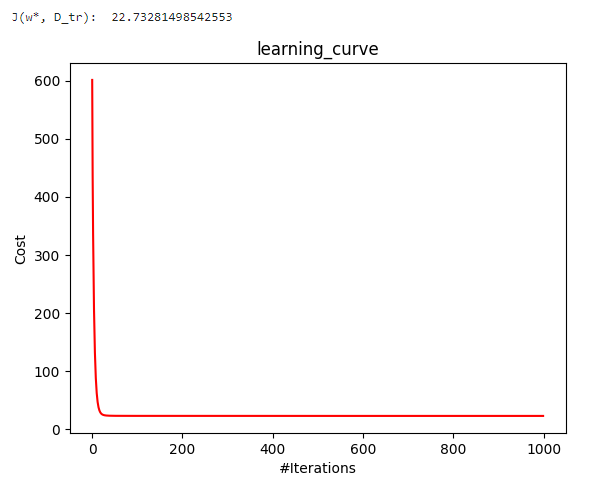
Problem (b)

RMSE(y\_tr 🡪 training data) : 4.66763967930312

RMSE(y\_ts 🡪 testing data) : 5.002223099284747



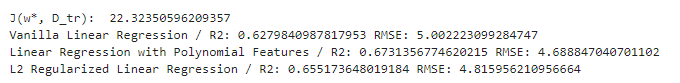






The value of l(lambda), alpha value, and n\_iter value were variously adjusted to for statement. When having the smallest RMSE value, the lambda value is 0.001, the alpha value is 0.0622, and the n\_iter value is 1000. At this time, the RMSE value is 4.832266321757755. The RMSE value of the testing data in problem (b) was 5.002223099284747. When the two values are compared, this model has better predictive performance because the RMSE value is lower when the lambda value is 0.001, the alpha value is 0.0622, and the n\_iter value is 1000.

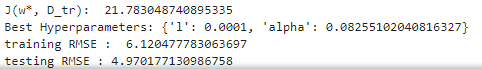
(i) (5 pt.) Find the code cell starting with “# Problem 1 (i)”. Complete functions compute\_r2() and compute\_rmse() by implementing the corresponding metrics as discussed in the class. Execute the cell and compare the results from the above executions (the vanilla linear regression, linear regression with the polynomial feature function, L2 regularized linear regression) in terms of R2 and RMSE. Which executions show the best and worst results? Draw out your conclusion using the evaluation metrics.



When vanilla linear regression is performed, the rmse value is 5.00222309928477, and the R2 value is 0.6279840987817953. In the case of linear regression with polarization features, R2 is 0.6731356774620215, and the RMSE value is 4.68847070701102. In the case of L2 Regularized Linear Regression, the R2 value is 0.655173648019184, and the RMSE value is 4.815956210956664. When comparing the RMSE values, it was the largest in the case of vanilla linear regression and the smallest in the case of linear regression with polinomic features. However, when comparing the values of r2, it was the largest in the case of linear regression with polinomic features and the smallest in the case of vanilla linear regression.

When the rmse value is small, the value of r2 is good for model performance when the value is large. When it comes to linear regression with polinomic features, the value of rmse is the best, and the value of r2 is the largest, so the performance is the best. Vanilla linear regression has the worst performance among the three because the value of rmse is the largest and the value of r2 is the smallest.

(j) (Extra 8 pt.) What is the best combination of l (lambda) and alpha for the L2-regularized (ridge) linear regression? Using a 5-fold cross-validation on (X\_tr, y\_tr), find the best combination of the hyperparameters. You may define the search ranges and intervals for l and alpha own your own. After both hyperparameter values are determined, train your final model with the entire training data and test its performance on (X\_ts, y\_ts). Note that the test dataset should be kept unseen during the cross-validation. Submit your implementation and results both in your report and in Jupyter notebook.

****

-> L2-regularized (ridge) linear regression’ result (training\_rmse, testing\_rmse)

When lambda is 1, and alpha is 0.08255102040816327, the average of the five rmse values calculated is the smallest, so we thought that it was the parameter value that performed the best. Using these parameter values, we trained the final model with the whole data and tested it with the test data. Looking at the results of the L2-regularized (bridge) linear regression above, the value of rmse is 4.8326632177755, and the value of rmse in the test data of the model using 5-fold cross-validation is almost the same as 4.970177130986758. However, looking at the rmse value of the training data, it can be seen that the rmse value of the testing data is smaller than that of very.

2. Linear discriminant analysis.

In this problem you will be working with a medical dataset, Breast Cancer Wisconsin1 ('breast-cancer-wisconsin.csv'). The dataset consists of 11 attributes:

Input attributes

Sample code number (id number)

Clump Thickness (range: 1-10)

Uniformity of Cell Size (range: 1-10)

Uniformity of Cell Shape (range: 1-10)

Marginal Adhesion (range: 1-10)

Single Epithelial Cell Size (range: 1-10)

Bare Nuclei (range: 1-10)

Bland Chromatin (range: 1-10)

Normal Nucleoli (range: 1-10)

Mitoses (range: 1-10)

Target variable

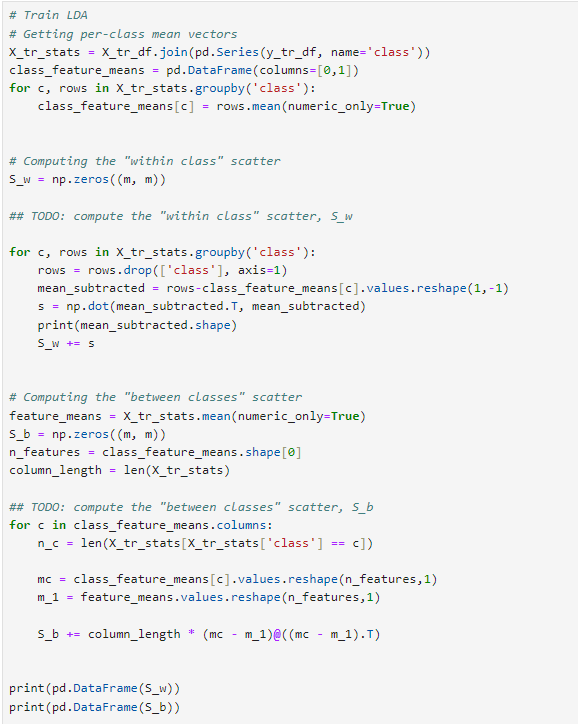
Class (2 for benign, 4 for malignant)

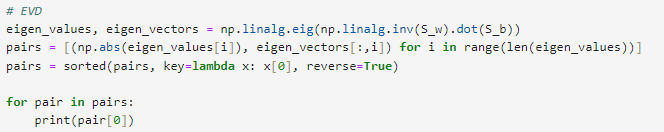
Read the provided code snippet for the preparation Steps 1 through 4 (“load raw data”, “remove rows with missing values”, “split data into X and y”, and “create a 0/1 label vector”) and make sense of the data preprocessing stage. Notice that the first attribute (Sample code number) is not used in the experiments.

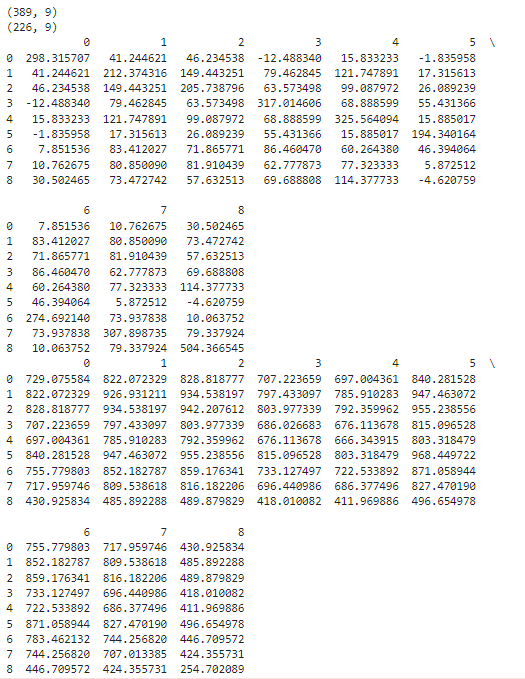
(a) (2 pt.) For a binary classification problem where the data instances are labeled with 0 and 1, write down the definition of and , in terms of individual observations ∶ n = 1, … , N and the sample mean of each class ( and )).

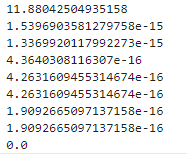
In linear discriminant analysis (LDA), and are covariance matrices used to maximize the separation between classes in binary classification problems. stands for variance within a class, and stands for variance between classes. and are sample means of class 0 and class 1, respectively. The goal of LDA is to maximize the ratio obtained by dividing the determinant of by the determinant of

(b) (4 pt.) Open the provided .ipynb code template. The template supplies normalize\_mv(), get\_accuracy(), and a code snippet that prepares data for this assignment. First try to get familiar with it. Go to Problem 1(b) in the template. Complete the code such that it computes `S\_w` and `S\_b` as discussed in class. Submit your complete code and its results both in your report and in Jupyter notebook.

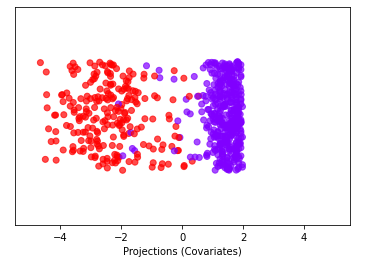






🡪 pair

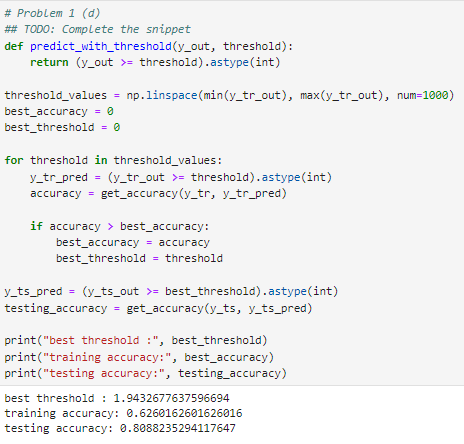
(C) (3 pt.) Upon the completion of Problem (b), execute the following code snippet and find the resulting figure. What does the figure represent? Explain in your own words (in your report).



Looking at the plot above, the x-axis appears as a projected value (y\_tr\_out), and classes are classified by color.

From the plot above, you can see how the data was projected due to the selected discriminant vector. If data points belonging to the same class are gathered close to each other, they can be considered well-differentiated. The data on the left is mostly red, and the data on the right is mostly purple. Red and purple data are mixed in the middle, but you can see that there is a good separation between classes.

(d) (4 pt.) Write a script that finds the best threshold value from the training data. And then perform a prediction on the testing data and report the accuracy. You may want to find the decision threshold value using y\_tr\_out, then report the accuracy on both y\_tr\_out and y\_ts\_out. Submit your complete code and its results both in your report and in Jupyter notebook



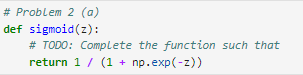
**3. Logistic Regression**

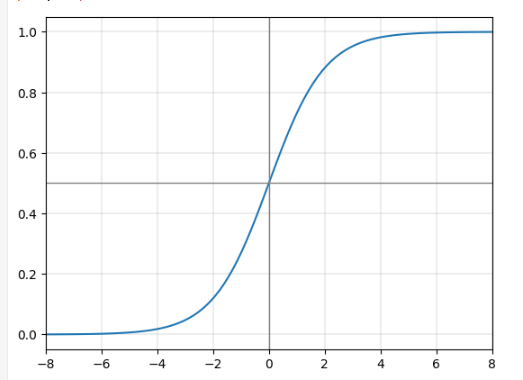
In this problem you are going to work with the logistic regression method. First, review the provided implementation of normalize\_mv(), get\_accuracy(Y\_true, Y\_pred), and the data preparation steps 1 through 6. In this problem our goal is to complete the experiment with logistic regression as given in the code cell starting with “# Experiment with LogReg” (Steps 7 through 9).

(a) (3 pt.) Implement sigmoid(x) that returns the sigmoid value for input x, in the code cell starting with “# Problem 2 (a)”. Recall that the sigmoid is defined as:

After completing the function, execute the code cell to visualize a sigmoid curve. Submit your complete function and the results of the code cell both in your report and in Jupyter notebook

Solution)



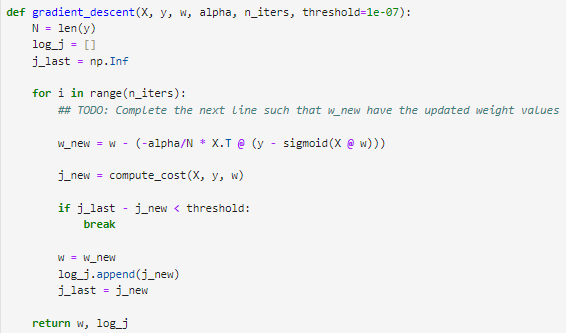


(b) (4 pt.) Write the objective function for logistic regression, when using the gradient descent algorithm. You may refer to the lecture slides and the provided compute\_cost() function.

Solution)

(c) (4 pt.) Complete the gradient\_descent() function such that it properly updates w each iteration. Submit your complete code snippet both in your report and in Jupyter notebook.

Solution)



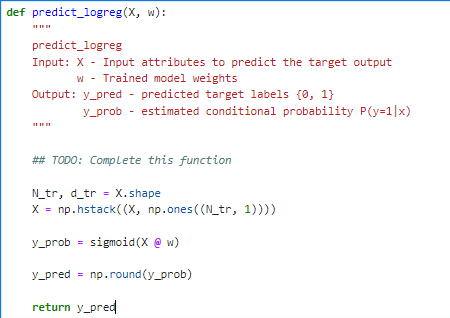
(d) (4 pt.) Given a trained weight parameter w for logistic regression, explain how you make prediction for an unseen test instance x’.

Solution)

To make a prediction about test instance x', you must go through the same preprocessing steps as the training data. Then calculate the test instance x' internally with the weight vector w. Enter this internal value into the logistic function. This means that applying the sigmoid function gives test instance x' the probability that it belongs to class 1. Usually, it is classified as class 1 if the probability is 0.5 or greater, and class 0 if it is less. As a result, the test instance is predicted as a label of 0 or 1 in the binary classification.

(e) (4 pt.) Implement predict\_logreg(X, w) that takes an input matrix (X) and a weight vector (w). The function returns the 0/1 predictions and estimated probability P(y = 1|x) for the input instances.

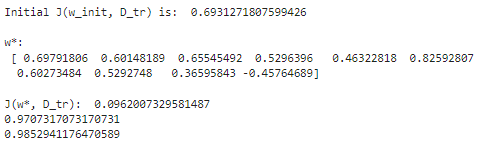
Solution)



(f) (3 pt.) Execute Steps 7-9, train, predict, and evaluate your logistic regression model. Read and analyze the results. What is your obtained w vector after training? What is your accuracy on the training data and that of the testing data?

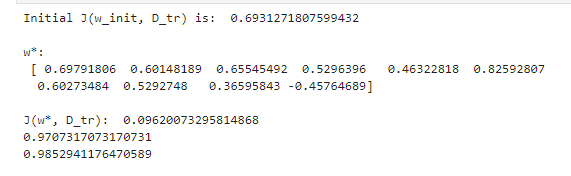
Solution)

The w vector after training is shown in the picture below.

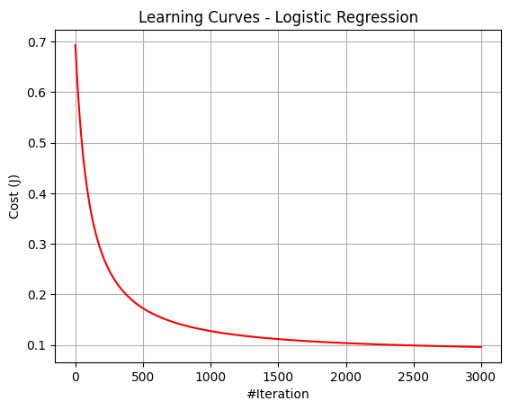


The accuracy of training data is 0.9707317073170731, and the accuracy of testing data is 0.9852941176470589.

정답



(g) (3 pt.) Using function plt.plot(), draw the learning curves from the above experiment. That is, variable log\_j\_logreg contains the loss evaluated on each training iteration. Visualizing this variable highlights the progression of training. Plot the changes recorded in log\_j\_logreg; that is, x-axis is the iteration number, and y-axis is the output of the J function. Include your answer in your report.



(h) (Extra up to 15 pt.) Implement the L2-regularized logistic regression and repeat the above experiment. After finishing implementation, adjust `lambda`, that determines the influence of the regularizer, and repeat the experiments to acquire the best test result. You may also adjust `learning rate` and `n\_iters` (#epochs) to improve the results. Submit your codework and results in Jupyter notebook. Submit the results and analysis in your report.

The results were checked by changing lambda, learning rate, and n\_iters. When lambda is 1e-05, when the learning rate is 0.01, and when n\_iters is 4000, the highest acuity value of 0.9852941176470589, came out. The graph is the learning curves when the parameter values above have been obtained. The trained model had 94.47% training accuracy and 98.53% test accuracy, and the final model weight (w\*, D\_tr)) and the lowest cost value (J(w\*, D\_tr)) were also optimized. Looking at the graph below, it can be seen that the cost is continuously decreasing repeatedly.

