The attached assignment is a group of files split according to their respective functionalities.

* Logistic regression using gradient descent
* Predicting ‘high’ for Auto dataset
* Function to split dataset into test & train
* Training algorithm on training set and calculating errors
* Stopping when the value of the objective function does not change by more than 1% of its initial value over the last 10 training steps
* Running logistic regression program for a fixed value of η and for a fixed stopping rule (producing reasonable results in your experiments so far) 100 times, for different values of the initial weights (produced as above, as independent random numbers in [−0.7,0.7])
* Modifying the training procedure as follows. Train F (X, λ) 4 times using gradient descent with different values of the initial weights and then choose the prediction rule with the best training objective

**Task 1:**

Requirement: Logistic regression using gradient descent

File: logisticRegressionModel.R

Formulae used:

1. Gradient descent using Sigmoid function

Text

Description automatically generated

1. Objective function Text, letter

   Description automatically generated

Pseudocode / algorithm:

initialize weights (vector length of number of features), bias, cost list (empty vector)

for number of iterations:

Z = weights . transpose (x) + bias

sigmoid = σ (Z)

cost = - Σ (y \* log (sigmoid) + (1 - y) \* log (1 – sigmoid))

dW = (sigmoid - y) \* x

dB = Σ (sigmoid - y)

updated weight = weight – (learning rate \* dW)

updated bias = bias – (learning rate \* dB)

append cost to list

return weight list, bias, predicted label list, cost list

**Task 2:**

Requirement: Predicting ‘high’ for Auto dataset

File: task2.R

Formulae used:

1. Gradient descent using Sigmoid function

Text

Description automatically generated

1. Objective function Text, letter

   Description automatically generated

Pseudocode / algorithm:

load dataset and show dimensions

initialize empty list of labels

for number of rows in dataset:

if mpg > 23:

high = 1

else:

high = 0

append high to list

add dummy variables for qualitative attributes (in this case, origin):

origin splits into 2 columns : origin1 and origin 2

origin1 : 1 for all rows with value of origin as 1 else 0

origin2 : 1 for all rows with value of origin as 2 else 0

add horsepower, weight, year, origin columns to new dataset

scale dataset

add list of labels to last column of dataset

initialize learning rate, no of iterations, list of test error rates

call function created in task 1 to compute weight list, bias, predicted label list, cost list

plot graph of no of iterations and acquired cost list

calculate predicted labels as 1 for values > 0.5 else 0

Observations:

Actual labels:

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0

[44] 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0

[87] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1

[130] 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1

[173] 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1

[216] 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0

[259] 0 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1

[302] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[345] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1

[388] 1 1 1 1 1

Predicted labels:

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0

[44] 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 1 0 0

[87] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1

[130] 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1

[173] 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1

[216] 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0

[259] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1

[302] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[345] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[388] 1 1 1 1 1

**Task 3:**

Requirement: Function to split dataset into test & train

File: task3.R

Pseudocode / algorithm:

send dataset to function

set seed for the pseudo-random number generator as 2110

sample size = dataset / 2 (50-50 for train & test)

calculate index of split using sample()

train set = [index, ]

test set = [-index,]

return train dataset, test dataset, train labels, test labels

Observations:

X\_train shape = 196 rows, 5 columns

X\_test shape = 196 rows, 5 columns

Y\_train shape = 1 row, 196 columns

Y\_test shape = 1 row, 196 columns

**Task 4:**

Requirement: Training algorithm on training set and calculating errors

File: task4.R

Pseudocode / algorithm:

Load and split dataset like in Task 2 and Task 3

compute 10 iterations and learning rates as:

10^-10 < learning rates < 10^-1

1000 < iterations < 10000

for each iteration and learning rate:

calculate weight list, bias, predicted label list, cost list from function in Task 1

calculate train and test error rates

calculate accuracy (1- error rate)

create matrix with final values

Observations:

Table

Description automatically generated

**Task 5:**

Requirement: Stop when the value of the objective function does not change by more than 1% of its initial value over the last 10 training steps

File: task5.R

Pseudocode / algorithm:

Load and split dataset like in Task 2 and Task 3

Initialize learning rate, weights and bias (-0.7 to 0.7)

while True:

calculate weight list, bias, predicted label list, cost list from function in Task 1

calculate train and test error rates

calculate accuracy (1- error rate)

if % increase of current element is less than 0.1% of (current - 10th ) element:

break

increment index of item in set

Observations:

Chart

Description automatically generated with medium confidence

**Task 6:**

Requirement: Run your logistic regression program for a fixed value of η and for a fixed stopping rule (producing reasonable results in your experiments so far) 100 times, for different values of the initial weights (produced as above, as independent random numbers in [−0.7,0.7])

File: task6.R

Pseudocode / algorithm:

Load and split dataset like in Task 2 and Task 3

Initialize learning rate, weights and bias (-0.7 to 0.7)

for number of test cases(100):

calculate weight list, bias, predicted label list, cost list from function in Task 1

calculate train and test error rates

calculate accuracy (1- error rate)

append test accuracy to list

plot test accuracy in box plot

Observations:

Test error rates for 100 cases with random weights between -0.7 to 0.7

Shape, rectangle

Description automatically generated

**Task 7:**

Requirement: Modify the training procedure as follows. Train F (X, λ) 4 times using gradient descent with different values of the initial weights and then choose the prediction rule with the best training objective.

File: task6.R

Pseudocode / algorithm:

Load and split dataset like in Task 2 and Task 3

Initialize learning rate, weights and bias (-0.7 to 0.7)

for number of test cases(4):

case A:

calculate weight list, bias, predicted label list, cost list from function in Task 1

calculate train error rates

case B:

calculate weight list, bias, predicted label list, cost list from function in Task 5

calculate train error rates

append accuracies, weights, biases to lists

calculate minimum train error rate for case A and case B

Observations:

[1] "Train error list for Case A:"

[1] 0.05825763 0.05837222 0.05823909 0.05823391

[1] "Train error list for Case B:"

[1] 0.06012960 0.06011646 0.06014351 0.05987700

[1] "Task 4 optimal prediction rules"

[1] "Error rate:"

[1] 0.05823391

[1] "Weights:"

[1] -1.57099988 -3.41073460 1.36782063 -0.61067584 0.07274591

[1] "Bias:"

[1] -1.148341

[1] ""

[1] "Task 5 optimal prediction rules"

[1] "Error rate:"

[1] 0.059877

[1] "Weights:"

[1] -1.18196356 -2.99507242 1.22643989 -0.59666238 0.03464358

[1] "Bias:"

[1] -0.8972854

[1] ""