Q1:

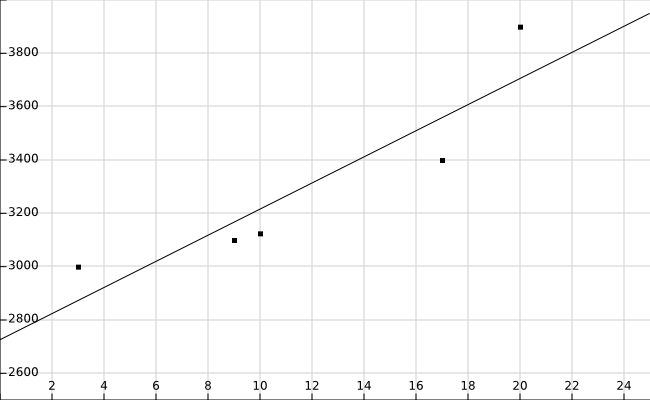
I have coded and used the python program added in the appendix A for this example. This program can be made more efficient using external libraries for matrix operations, however, it serves the purpose of this example.

The initial values of w0 and w1 were set to zeros, where w0 and w1 are the parameters of the hypothesis, and value of alpha was 0.01, where alpha is the learning rate.

The values of w0 and w1 after the first five iterations were:

|  |  |  |
| --- | --- | --- |
| iterations | w0 | w1 |
| 1 | 33.05 | 407.9 |
| 2 | 17.6373 | 94.8119 |
| 3 | 39.3231228 | 333.9513784 |
| 4 | 32.5736289208 | 150.124726682 |
| 5 | 47.5831748831 | 290.261768962 |

Additionally, I have used 10,000 iterations and the final values of w0 is 2726.94201042 and w1 is 48.9879651711. I have plotted them in the following graph as 2726.94 and 48.98 respectively and got the hypothesis shown by the straight line in the following figure. Here the black dots represent the given sets of (x,y).



Q2:

A problem can be predicting the resale value of a car.

The features I'd like to extract are:

1. Producing company

2. Model

3. Year of production

4. Last inspection date

5. Gasoline efficiency

6. Current mileage

7. Number of previous owners

8. Number of accidents

Appendix A:

#Gradient descent algorithm for the Exercise 1

x = [17, 20, 3, 10, 9]

y = [3400, 3900, 3000, 3125, 3100]

#y = [34, 39, 30, 31.25, 31]

def getCost(w0, w1, x, y):

N = len(x)

error = 0

for i in range(0, N):

error += ( w0 + w1 \* x[i] - y[i] ) \*\* 2

return error / (2 \* N)

def gradient(w0, w1, x, y, iterations, alpha):

N = len(x)

errorW0 = 0

errorW1 = 0

for i in range(0, N):

errorW0 += (w0 + w1 \* x[i] - y[i])

errorW1 += (w0 + w1 \* x[i] - y[i]) \* x[i]

return (w0 - (alpha/N) \* errorW0, w1 - (alpha/N) \* errorW1)

alpha = 0.01

iterations = 10000

#iterations = 5

w0 = 0

w1 = 0

k=0

while k < iterations:

cost = getCost(w0, w1, x, y)

(w0, w1) = gradient(w0, w1, x, y, iterations, alpha)

print str(k) + " : " +str(cost) + " : " + str(w0) + " : " + str(w1)

k = k + 1