172808
173705

P3: LINEAR AND LOGISTIC REGRESSION

LINEAR REGRESSION

Feature Normalization

First of all we load the linear regression dataset by running the multivariable.m script. After we have all the data in our workspace, we proceed to implement the feature normalization in featureNormalize.m following the formula:

$$x' = \frac{x - \bar{x}}{\sigma}$$

```
m = size(X , 1);
mu = mean(X);
sigma = std(X);

for i = 1 : m
    X_norm(i, :) = (X(i, :)-mu) ./ sigma;
end
```

To apply the formula, we have to iterate through each value of the columns and substracting the mean mu we previous stored in the proper variable and dividing it by the standard deviation of X.
This has to be done for a main reason: as we have the data variables differing a lot by orders of magnitude we want to have the data in an easier form which we know how to treat and for having the gradient descent method converge faster (find the global minimum).

Gradient Descent

In order to calculate the cost function in computeCostMulti.m we can reuse the code we previously did for the first practicum. So the formula we have to follow is:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

```
J = sum((X * theta - y) .^ 2) / (2*m);
```

Which we can compute with the previous snippet of code.

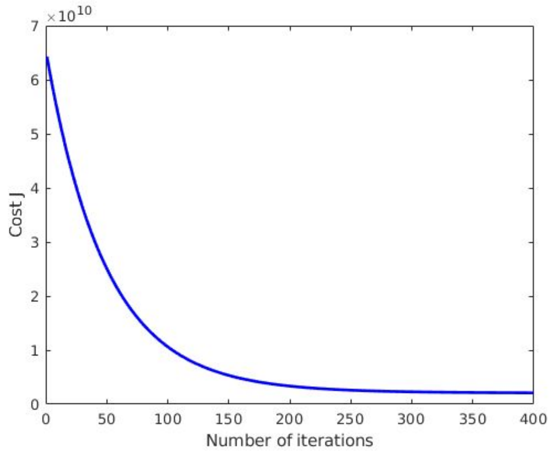
To run the gradient descent, we have to update the values of theta in a way which didn’t alter the data we want. To do that, we can reuse the practicum 1 code but adding the third theta variable and adapting to the current problem.

```
%Compute theta(1), theta(2) and theta(3)

temp_0 =(1/m)*sum((theta(1)+theta(2).*X(:,2) + theta(3).*X(:,3))-y); %derivative theta0
temp_1 =(1/m)*sum((((theta(1)+theta(2).*X(:,2) + theta(3).*X(:,3))-y)).*X(:,2)); %derivative theta1
temp_2 =(1/m)*sum((((theta(1)+theta(2).*X(:,2) + theta(3).*X(:,3))-y)).*X(:,3)); %derivative theta2
%Updating the values of theta
theta(1)=theta(1)-alpha*(temp_0);
theta(2)=theta(2)-alpha*(temp_1);
theta(3)=theta(3)-alpha*(temp_2);
```

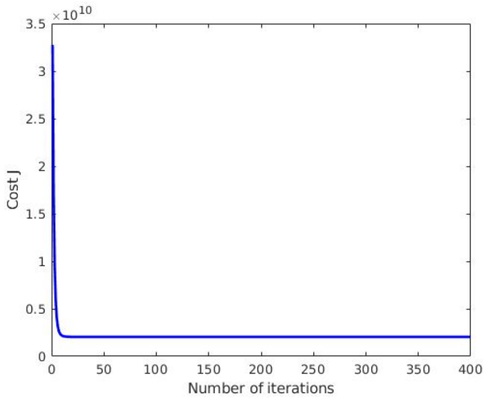
So, we obtain the following values and plot:

```
Theta computed from gradient descent:
 334302.063993
 100087.116006
 3673.548451
```

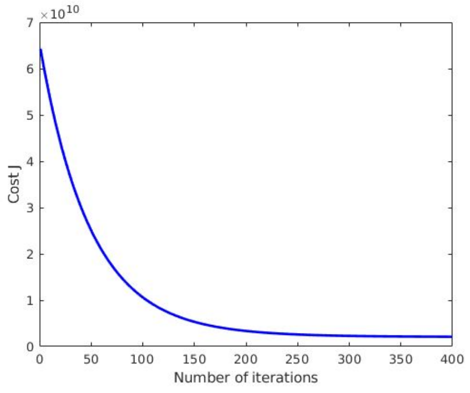


Selecting learning rates

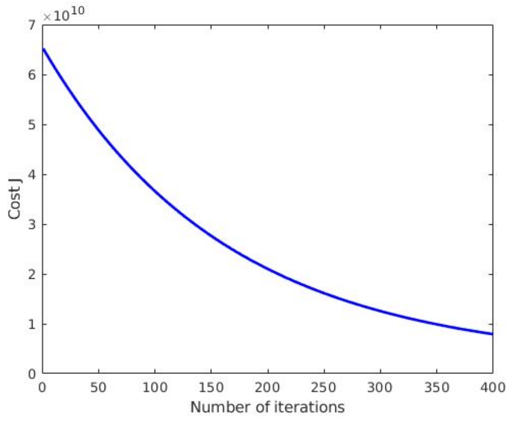
We have tried some different executions with different learning rates:



alpha = 0.3



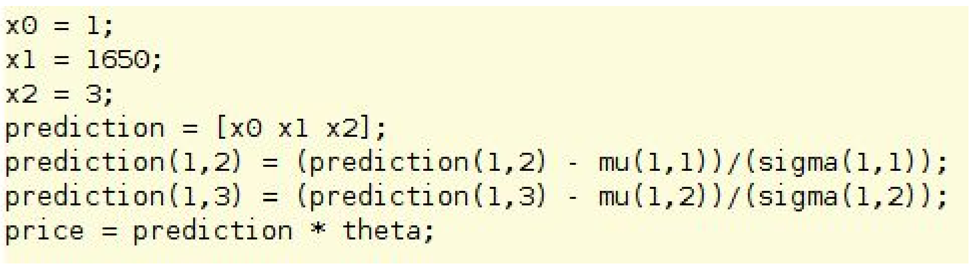
alpha = 0.01



alpha = 0.03

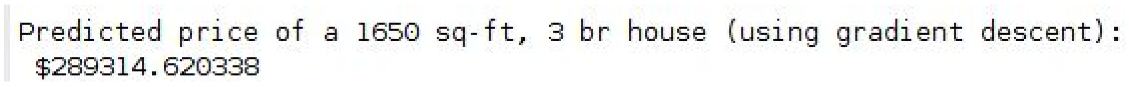
It’s clear that as we make alpha smaller, the model is taking more time to converge, whereas when we have larger values for alpha, the model seems to have a strange behaviour.

Finally we want to predict the value of a flat with 1650 square feet and 3 bedrooms. To do that we have to normalize the variables to predict accordingly to the reasoning we have taken through the practicum. We can do that as this:



```
x0 = 1;
x1 = 1650;
x2 = 3;
prediction = [x0 x1 x2];
prediction(1,2) = (prediction(1,2) - mu(1,1))/(sigma(1,1));
prediction(1,3) = (prediction(1,3) - mu(1,2))/(sigma(1,2));
price = prediction * theta;
```

And the resulting prediction is:



```
Predicted price of a 1650 sq-ft, 3 br house (using gradient descent):
 $289314.620338
```

LOGISTIC REGRESSION

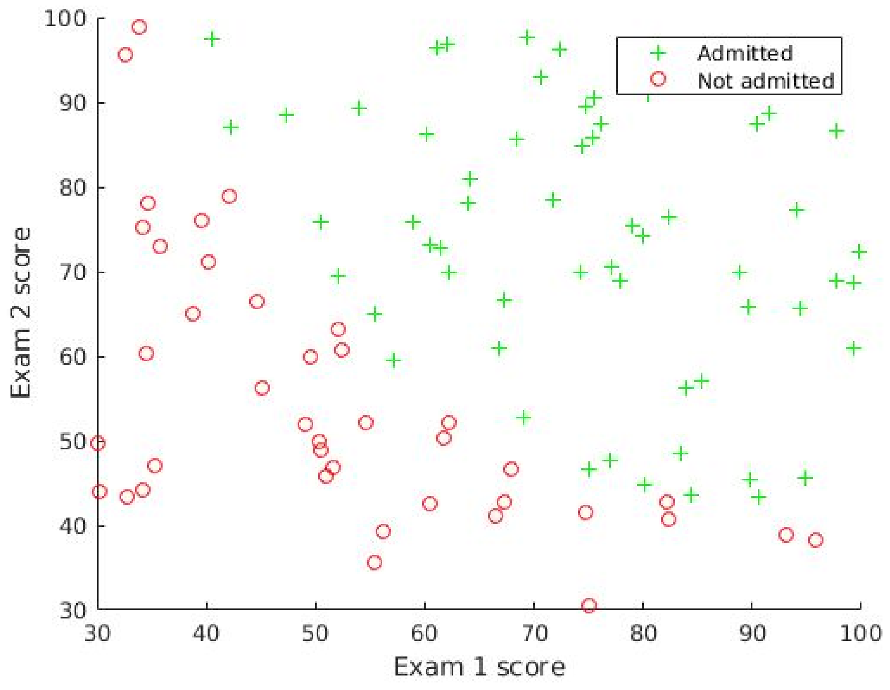
Visualizing the data

If we want to see the data in a proper way in order to better understand our dataset, we can add the following lines in plotData.m:

```
pos = find(y == 1);
neg = find(y == 0);

plot(X(pos , 1) , X(pos , 2) , 'g+');
plot(X(neg , 1) , X(neg , 2) , 'ro');
```

With the built-in function find(), we can know what are those variables that have a positive or negative output. In the plots, we are plotting first the positive responses for the two columns whereas in the second plot we are doing by the negative ones. This produces the following result:



Sigmoid function

The sigmoid function is given by the equation:

$$S(x) = \frac{1}{1 + e^{-x}}$$

Which can be implemented in the following way:

```
g = 1 / (1+exp(-z));
```

Learning parameters using fminunc

For using the fminunc Matlab function, we only need to calculate the cost function, to get our J(theta) (cost function), and the gradient of a given dataset. Then instead of having to define the steps, like we did in the previous assignment with gradient descent, fminunc is doing it all by itself.

For the cost function of a logistic regression we set the cost function:

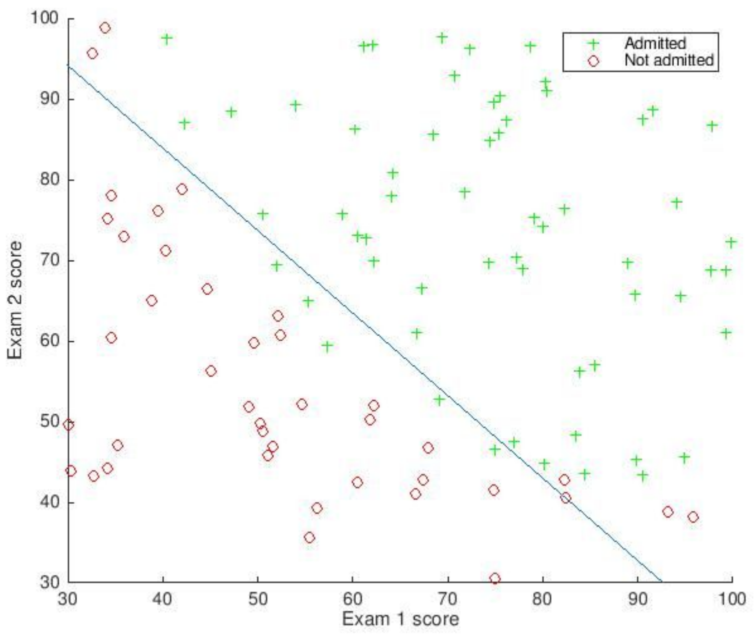
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1-h_\theta(x^{(i)}))]$$

where h(x):

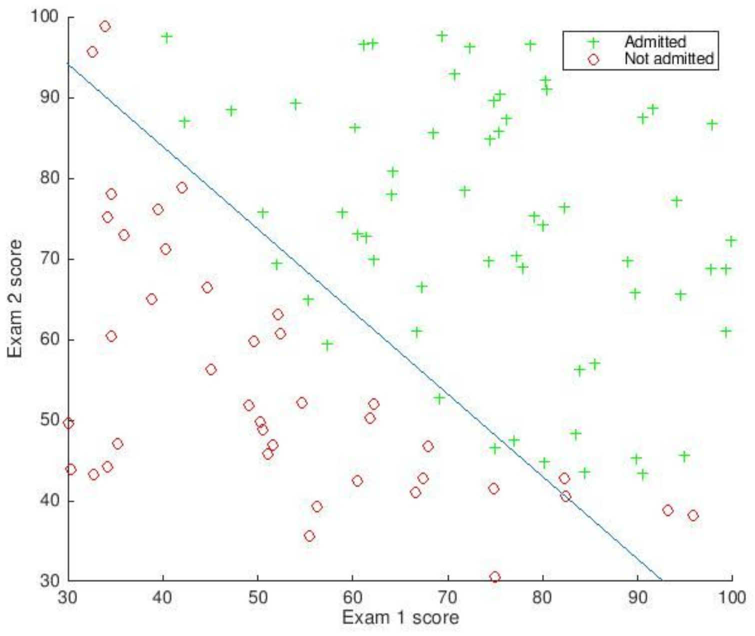
$$h_\theta(x) = g(\theta^T x)$$
$$g(z) = \frac{1}{1+e^{-z}}$$

When we get this values fminunc iterates over it until it converges to a minimum or it reach the maximum number of iterations.

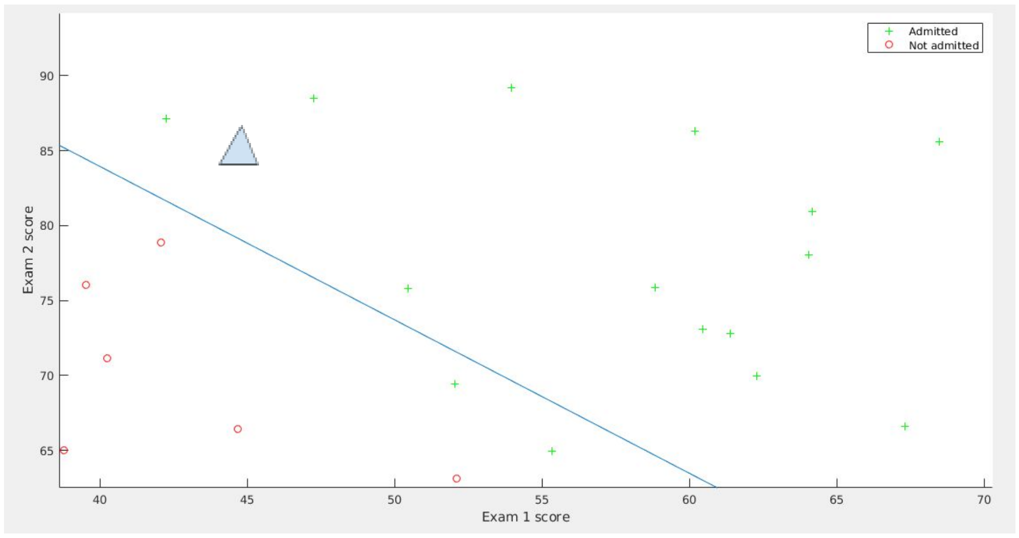
After running it with our dataset, fminunc has given us the cost of 0.203498, with it we have been able to plot a decision boundary through our dataset:



Evaluating logistic regression



After having learnt the parameters, we can use the resulting model to predict if a student with the following marks: Exam 1, 45 and Exam 2, 85, will be admitted.



So, according to the model, this student will be admitted since he is upper the regression line for both exam's scores.