REPORT

Data has been normalised in all the questions

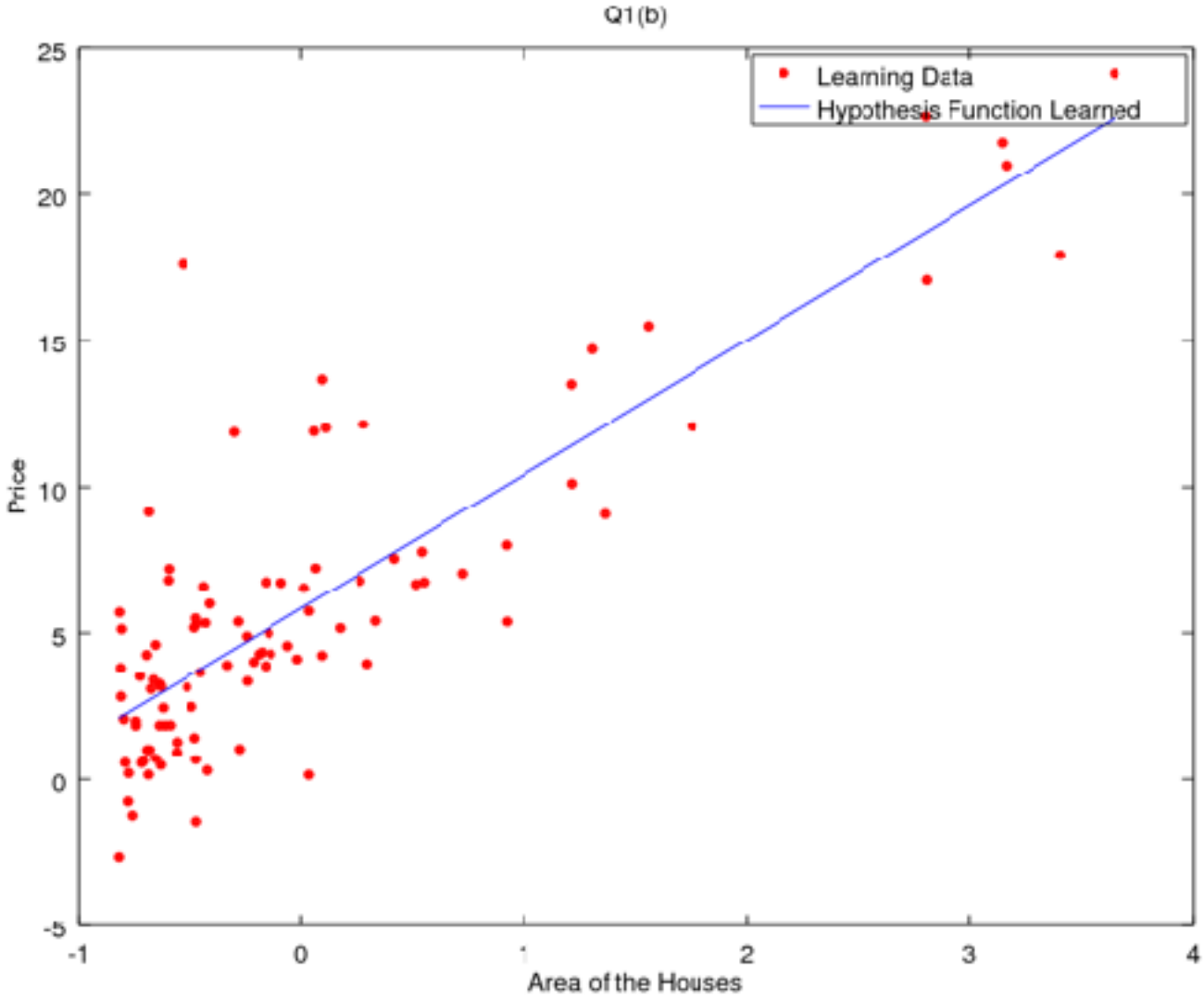
Q1.

Part (a) :

- Initialised the parameters matrix theta = zeros(n+1,1) while taking care of intercept term. Included x0 = 1 for all the examples. Made a function called getCost().
- **Stopping Criteria :** Now in the gradient descent stopping criteria was to check if in 2 consecutive iterations cost value changed less than by **0.0000001**.
- **Learning Rate :** Tried different learning rates in the increasing order as 0.03, **0.1**, 0.3, 1, 3. Chose **0.1** as my choice.
- Parameters/Theta vector learnt : **[5.8385 ; 4.5925]**. It took **211 iterations**.

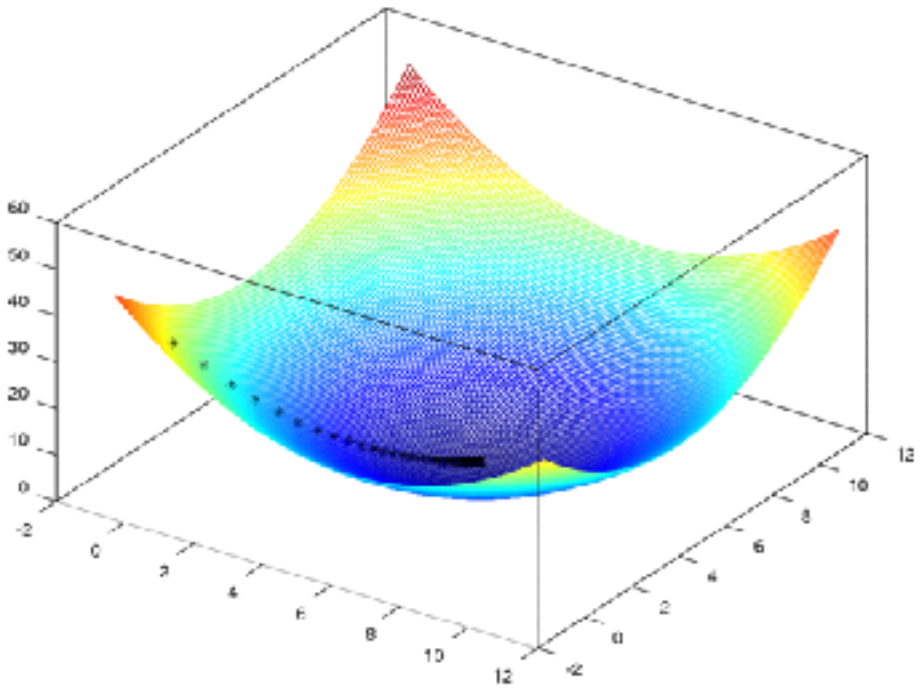
Part (b) :

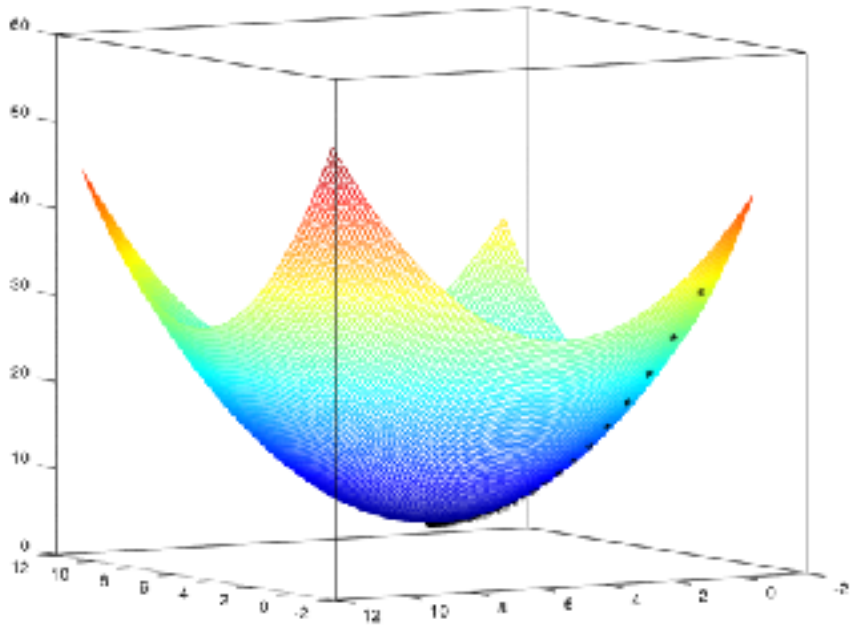
- Plotted the learning data first using x and y matrices and then plotted the hypothesis function learnt for the data given using the parameters learnt.



Part (c) :

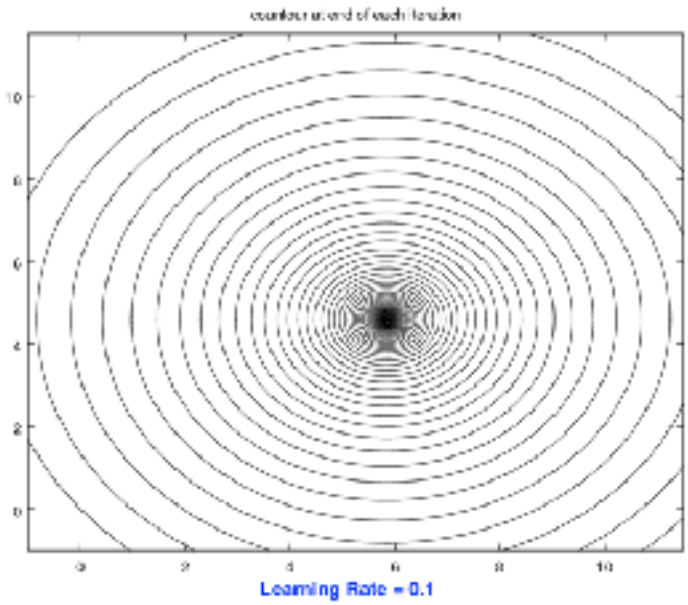
- First plotted the mesh of the error/cost function using mesh grid and evaluating the function on each point of that grid. Now in each iteration of gradient descent plotted the point in the 3D space using plot3() function for the current set of parameters. Also paused for 0.2 seconds after each iteration.

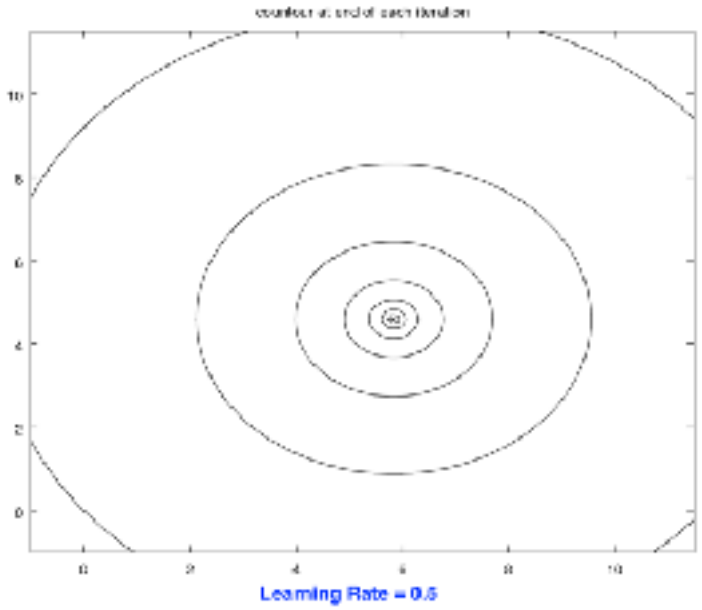


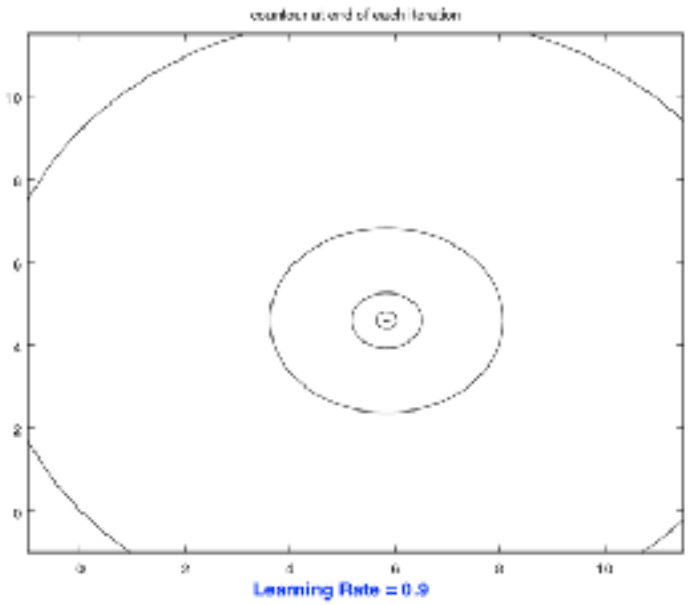


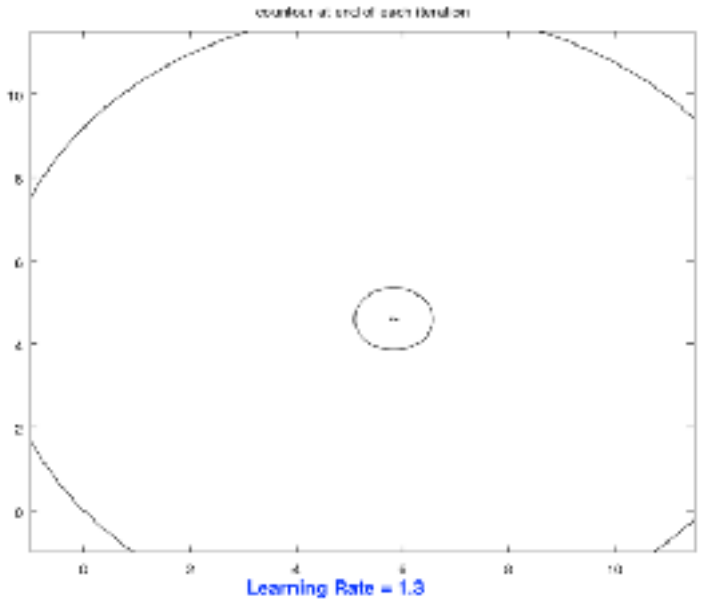
Part (d) :

- Here I used the values of function already evaluated on mesh grid in part (c). Ran gradient descent once again to plot the contours for each iteration with a gap of 0.2 sec each time.









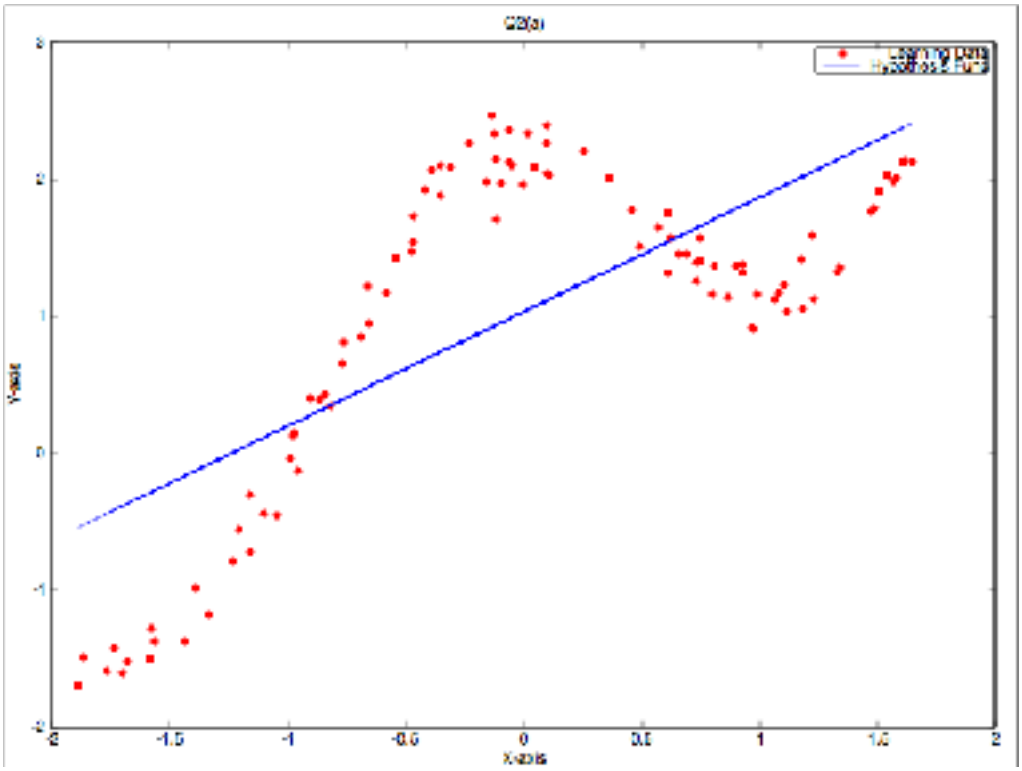
Part (e) : (Figures Above)

- As I increase the step size from 0.1 to 1.3, the distance between consecutive contours increases. The gradient descent takes less no. of iterations to converge, since we are taking larger steps in each iteration than before. But for step size 2.1 and 2.5 the contours start diverging instead of converging. It goes in an infinite loop, as it never satisfies the convergence criteria.
- This happens because in gradient descent we move towards the local minima (by calculating the gradient at that point) in each iteration, but while moving if we take a huge step we may end up far away on the other side of the minima, and if this new point has even larger magnitude of slope we would keep getting farther away from minima each time and never converge. Hence we need o take smaller steps for a successful convergence.
- Also while checking for different values of step size, can avoid going in an infinite loop by breaking from the loop if the magnitude of difference bw value of the function goes on increasing for some continuous iterations.

Q2

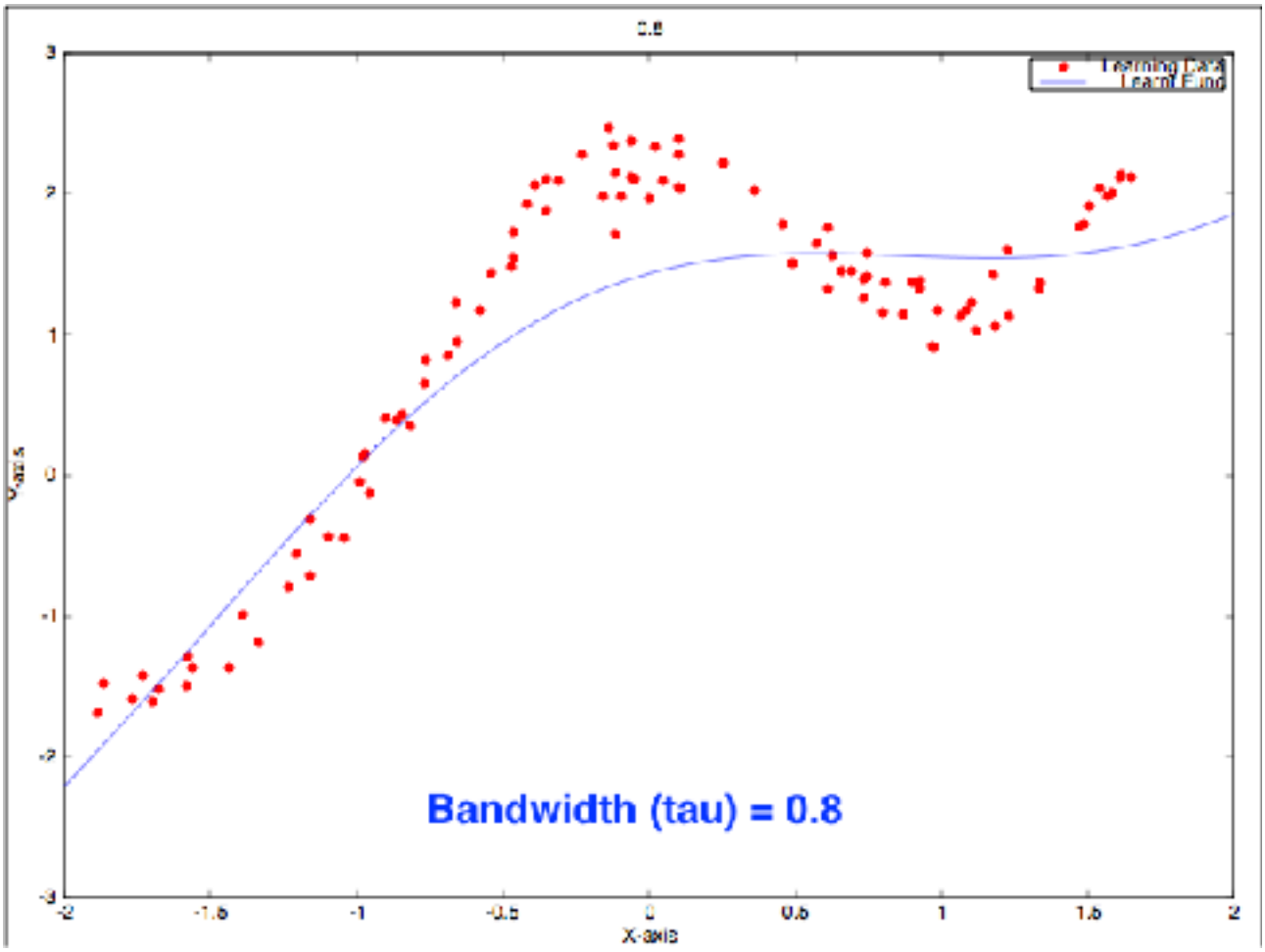
Part (a) :

- First used normal equation to calculate theta, while taking care of the intercept term. Then plotted the data points followed by the straight line resulting from theta vector.



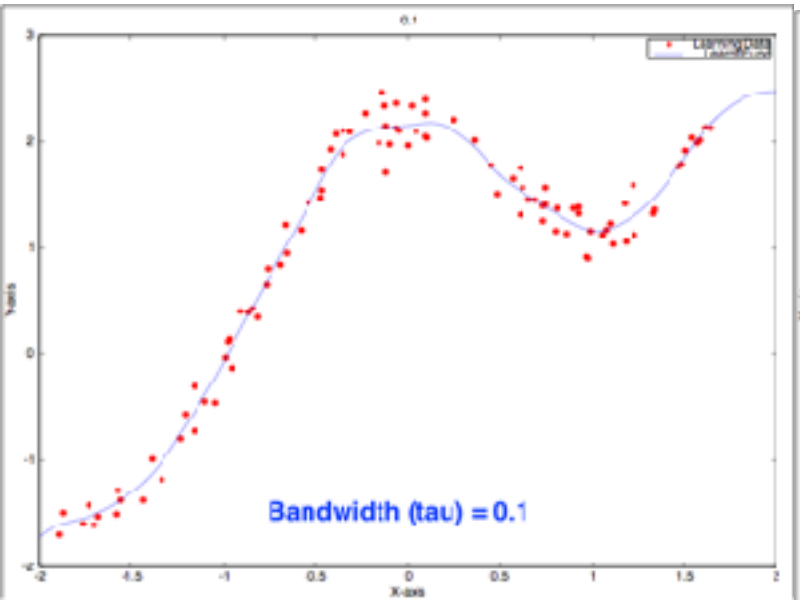
Part (b) :

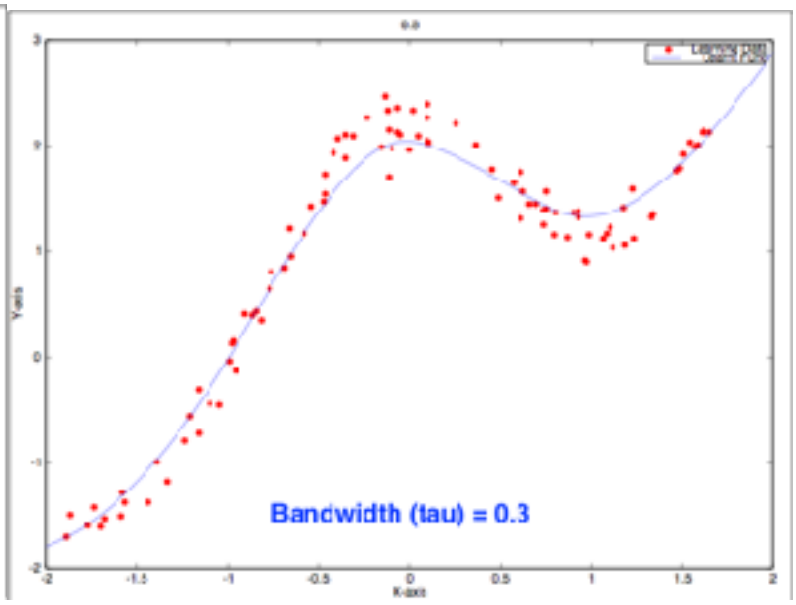
- Since had to plot for many values of bandwidth (tau). I made a function plot4me(), in which first I make a row vector containing points (range of points appropriately taken from the input data) on which I will calculate the weight function while plotting graph. Now in a loop on all those points I calculate diagonal matrix W and hence **theta = (inv(X'*W*X))*X'*W*y**. Finally I plot the points learnt from this weighted normal equation.

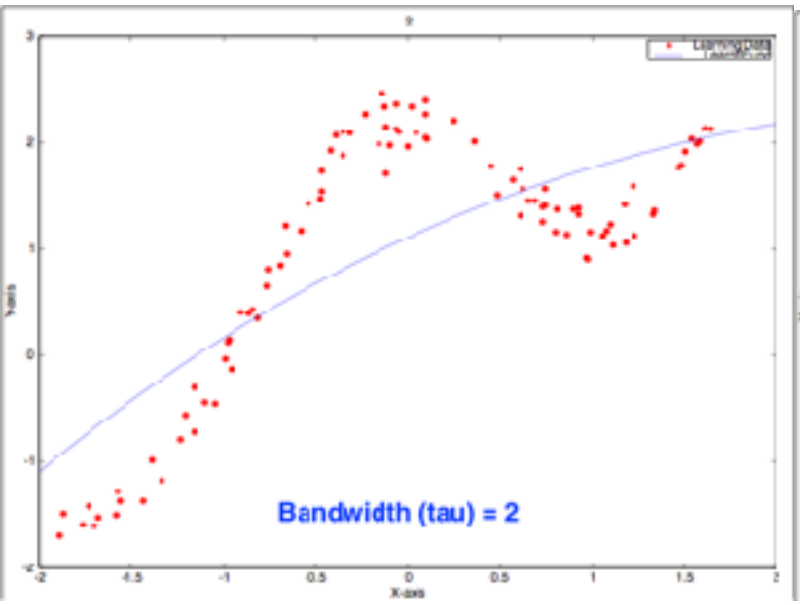


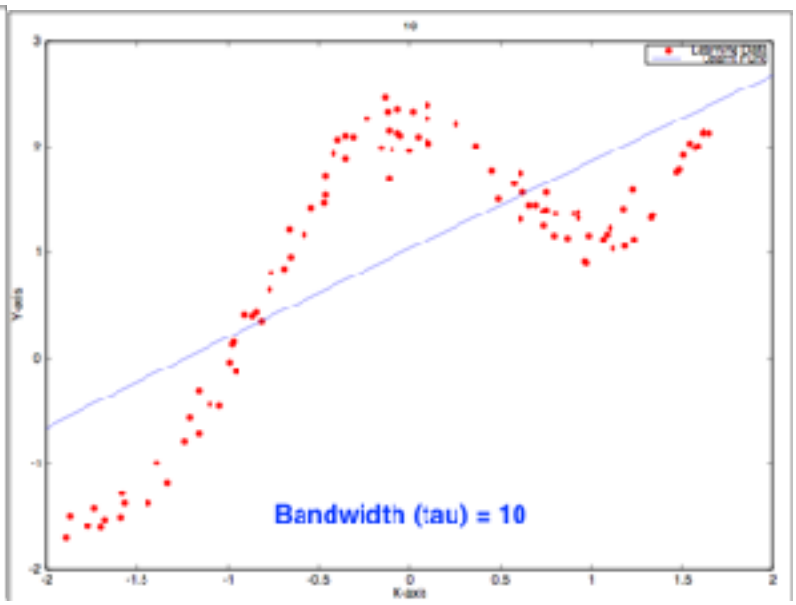
Part (c) :

- After trying for tau = 0.1, 0.3, 2, 10. I feel 0.3 one is the best fit for the given training data. In other cases I feel 0.1 does little bit overfitting and 2 and 10 cases do under-fitting.
- Basically what happens is that when tau is large, we give almost equal weight to all the points in the training data, hence diverging from the locally weighted case to the normal case as in part(a).
- Whereas when tau becomes more and more smaller, we are giving more and more weight to the closer points in the training data, hence when tau is extremely small we tend to overfit the data by chasing the closest points very precisely, and we get a zig zag curve instead of a smooth fit over the data.
- Hence we need a balanced value of the bandwidth parameter to ensure a good locally weighted linear regression.
- Plots for all 4 cases are below.









Q3

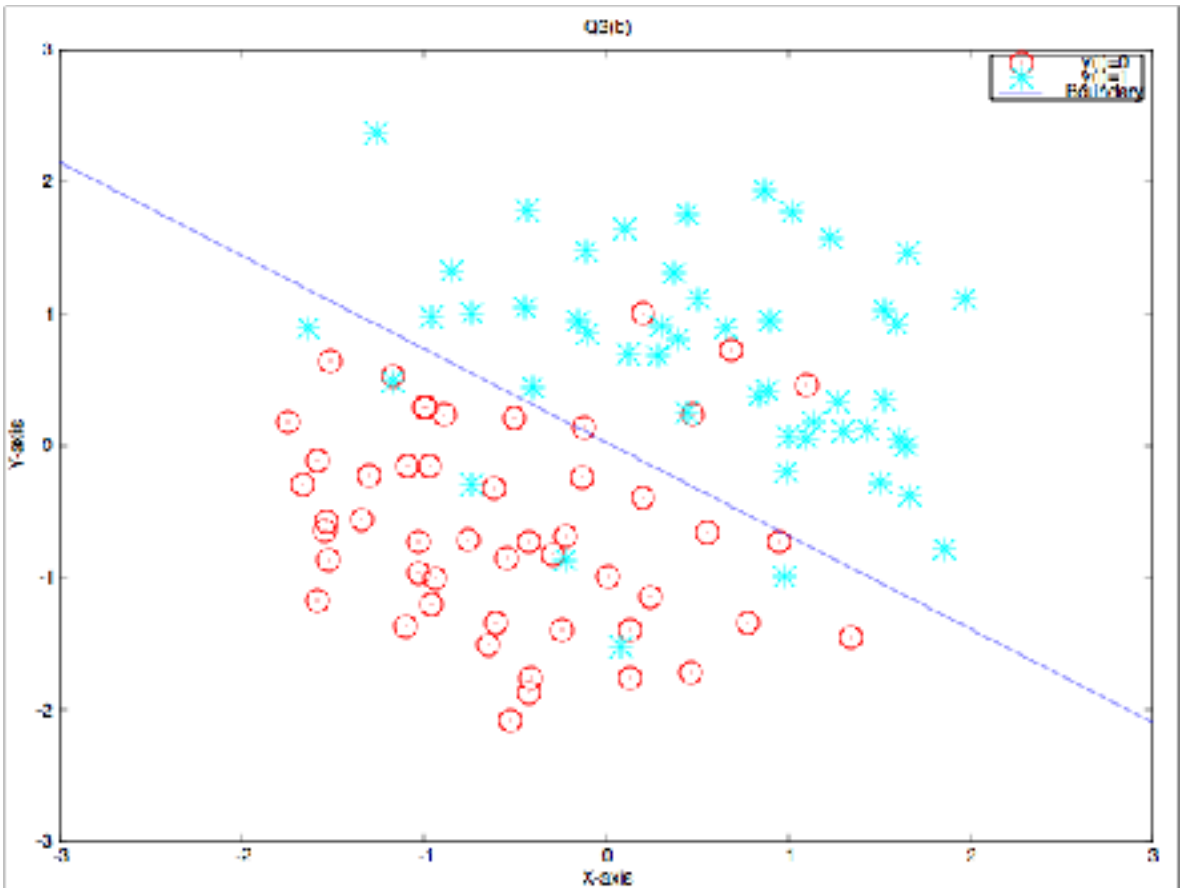
Part (a) :

- Firstly initialised theta appropriately while taking care of intercept term. Now our aim is to maximise function L, hence we need to find the parameters where the derivative of function L becomes zero. Applying Newton's method, I checked convergence of L' to zero by checking if the L2 norm (of difference bw the vectors theta of 2 consecutive iterations) is less than **0.000001**.
- Theta vector I obtained is

[-0.046697;
1.458049;
2.062351]

Part (b) :

- Made 2 different pairs of vectors to plot the 2 different set of points by different symbols. Finally just wrote the feature on the y axis in terms of the feature on x axis while using the coefficients learned, to plot the decision boundary.



Q4

Part (a) :

- $\mu 0 = \begin{matrix} 0.75529 \\ -0.68509 \end{matrix}$

- $\mu 1 = \begin{matrix} -0.75529 \\ 0.68509 \end{matrix}$

- $\Sigma = \begin{matrix} 0.429530 & -0.022472 \\ -0.022472 & 0.530646 \end{matrix}$

- This is the case of $\Sigma 0 = \Sigma 1 = \Sigma$

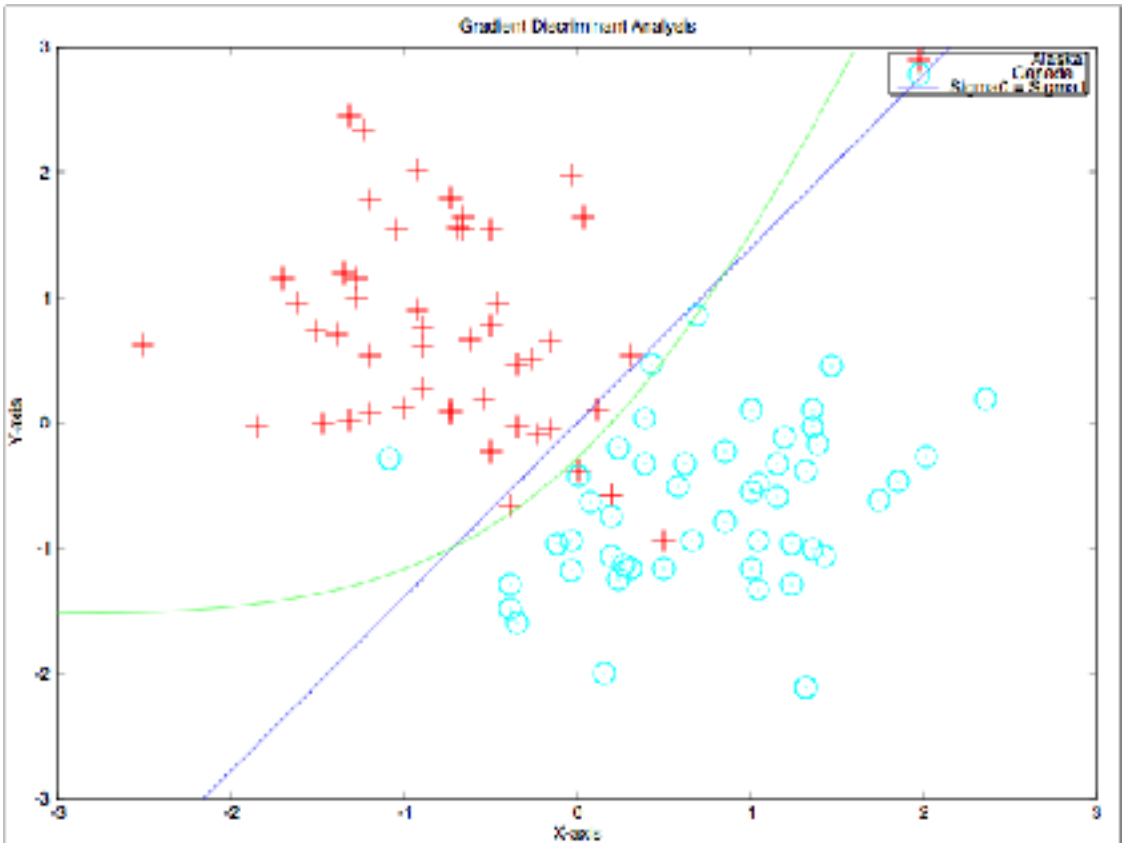
- Here Alaska is to 1 and Canada is to 0.

Part (b & c) :

- Simply plotted the training data first. Now the equation for the decision boundary is obtained by equating the probability of y = 1 given x, to probability of y = 0 given x and parameterised by $\mu 0$, $\mu 1$, $\sum$, ø. We get equation as

$$x' * inv(\sum) * (\mu 0 - \mu 1) + (\mu 0' - \mu 1') * inv(\sum) * x = 2*log(ø/1-ø).$$

- Here ø is the Bernoulli parameter for the distribution of y.
- I write x as [x1; x2], then find out the coefficient terms from the above equation and finally write x2 in terms of x1, so that I can plot the decision boundary easily.



Part (d) :

- $\mu 0$ and $\mu 1$ are same as before.

- $\sum 0$ = 0.47747 0.10992
  0.10992 0.41355

- $\sum 1$ = 0.38159 -0.15487
  -0.15487 0.64774

- Calculated using the formula mentioned in the question statement.

Part (e) :

- Again equating the probabilities as done in part (c), but this time $\sum 0 \neq \sum 1$, hence the quadratic terms don't cancel out and we get the equation as

$$(x-\mu 1)' * inv(\sum 1) * (x-\mu 1) - (x-\mu 0)' * inv(\sum 0) * (x-\mu 0) = 2*\log(ø/1-ø) + \log(det(\sum 1)/det(\sum 0))$$

- Now I took LHS as the function f and evaluated this function on all the points of the mesh grid. Chose range of the mesh grid appropriately from the max and min of the two features data given to us. Evaluated the value v as RHS of the equation above and used contour function to plot the equation above on the same graph as part (b&c).

Part (f) :

- Since we assumed the two variances to be equal in the first case, we saw that the quadratic terms get cancelled while deriving the equation hence we get a straight line as the decision boundary. Intuitively this is expected as on the same distance from the the mean points of the 2 data distributions we will get the probability values of the features equal, resulting in a straight line.
- In the second case we get the quadratic terms in the equation hence the decision boundary is curved one. Also its curved towards the feature points for y = 1 or Alaska. This can be explained as : Since these points (+ ones) have low variance, the probability values will decrease faster as we go farther from the mean point in the graph as compared to the probability of the other feature. Hence the points where the probabilities are equal from both sides will lie closer to the Alaska distribution as we move farther away in the graph. Hence curved towards + points.

- I feel since the + points have come little bit more inside the blue region than the blue ones in the red region when we compare from the linear line, the quadratic one captures these little points well.