Machine Learning - COL774
Assignment 1 - Report

CHAKSHU GOYAL
2015CS50280

1 Linear Regression

To predict density of wine based on its acidity, we try to model using linear regression. Batch gradient descent method is employed to optimise the cost function. Following are the observations.

1.1

$x^{(i)}s$ are first normalised to $(x^{(i)} - \mu)/\sigma$. Thereafter, initialised $\vec{\theta} = [-2, 2]$, keeping the learning rate $\eta = 0.0005$, and stopping criteria (θ difference in consecutive iterations) $= 10^{-11}$. The algorithm takes 289 iterations to converge and yields $y = h_\theta(x) = 0.001340x + 0.996619$.

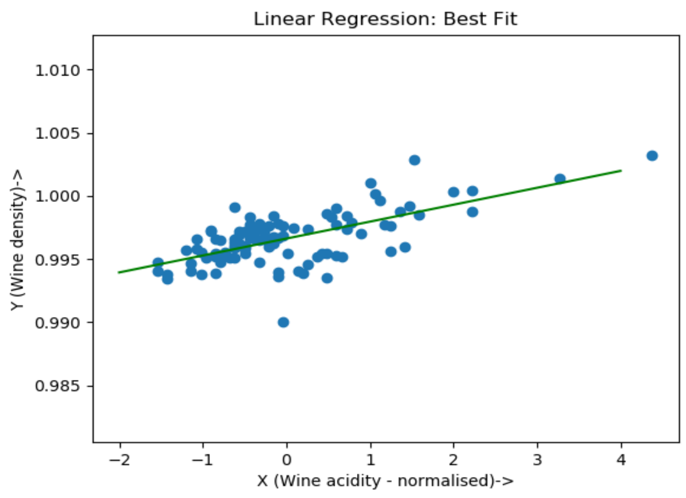

Figure 1: Hypothesis function

1.2

3D mesh plot with $J(\theta)$ on Z-axis and θs on the X-Y plane. We see as the iterations proceed cost tends to achieve the minimum value.

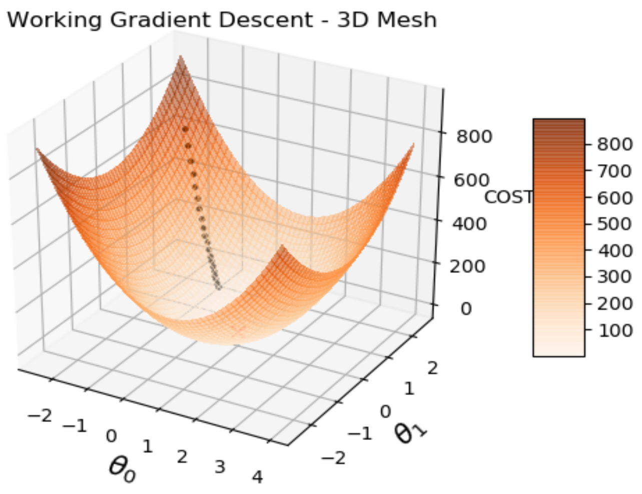

Figure 2: Cost in each iteration (*black dots*)

1.3

Contour plot for $J(\theta)$ vs θ. As iterations proceed, it tends to approach the red cross which is the minima for the function.

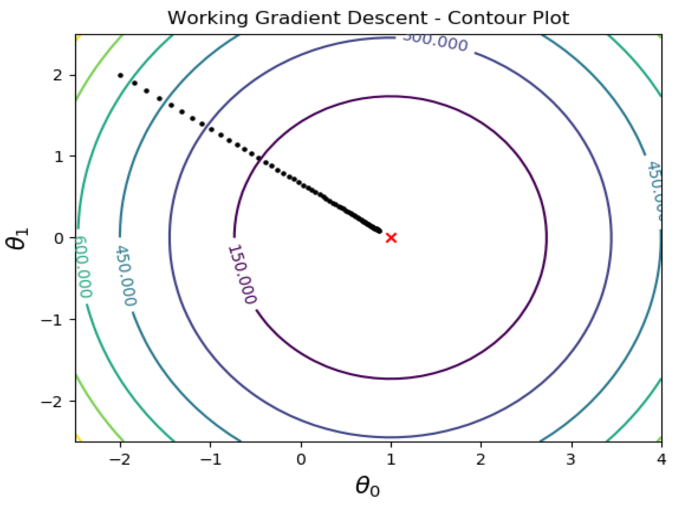

Figure 3: Cost in each iteration (*black dots*)

1.4

For $\eta = 0.021$ and 0.025, the algorithm does not converge. Other contours are plotted -

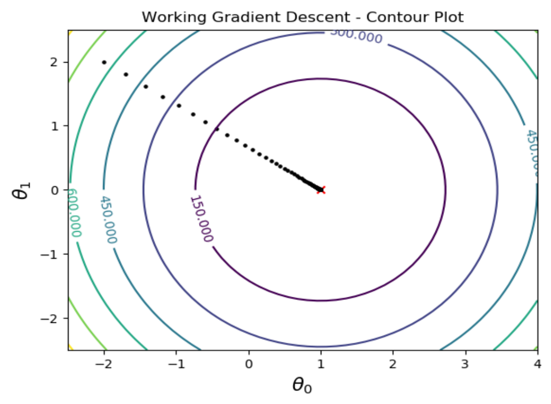

Figure 4: $\eta = 0.001$

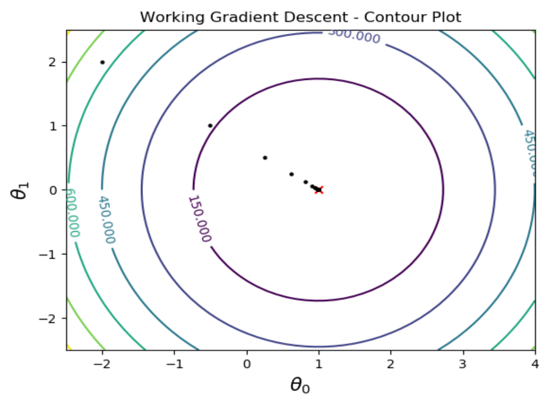

Figure 5: $\eta = 0.005$

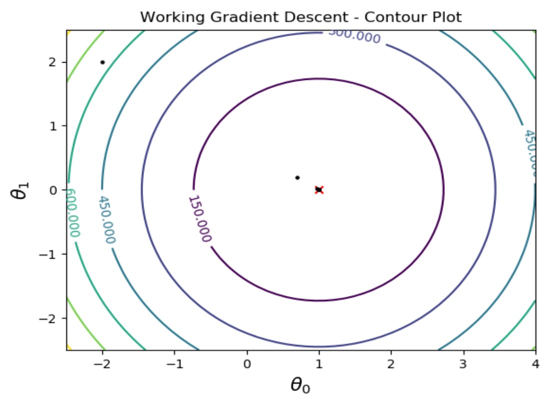

Figure 6: $\eta = 0.009$

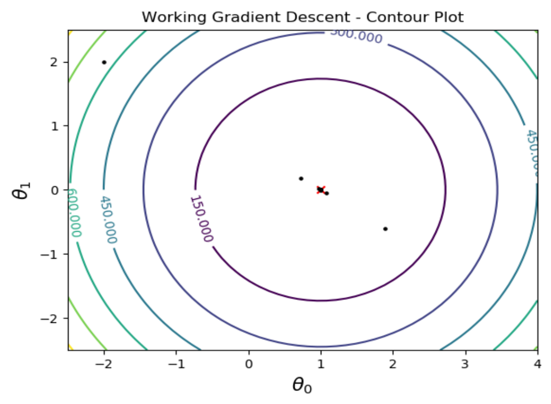

Figure 7: $\eta = 0.0013$

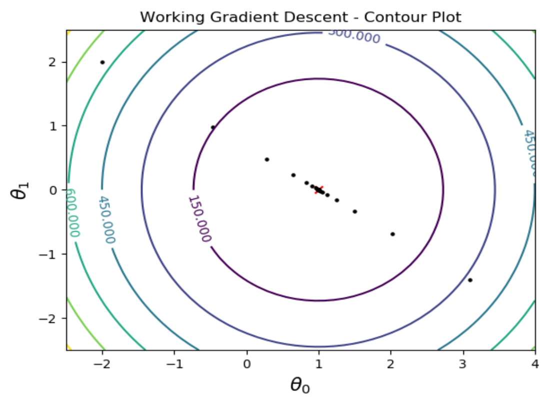

Figure 8: $\eta = 0.0017$

We observe that as we increase the learning rate, number of iterations taken to achieve convergence increases at first. This is true because now we take larger steps to go downhill along the curve. However, after a minima is achieved, in this case at $\eta = 0.009$, iterations begin to increase. We see that it overshoots the minima at times and oscillates before it finally attains its value. However, at very large values, convergence doesn't seem to occur because it keeps overshooting everytime it has to cross the minima.

2 Locally Weighted Linear Regression

Analytically, the solution for θ that minimises $J(\theta)$ is given by,

$$\theta = [(X^T X)^{-1} X^T]Y$$

Similarly, deriving to include local weights to weigh different training examples differently we obtain

$$\theta = [(X^T W X)^{-1} X^T W]Y$$

where W is a diagonal matrix consisting of weights for respective $x^{(i)}s$.

Using the unweighted equations (again after normalisation of X), we see that hypothesis for unweighted case is given by

$$y = 0.8351932x + 1.0312812$$

Further, we plot the general weighted model curves, varying bandwidth parameter τ, along with the unweighted model.

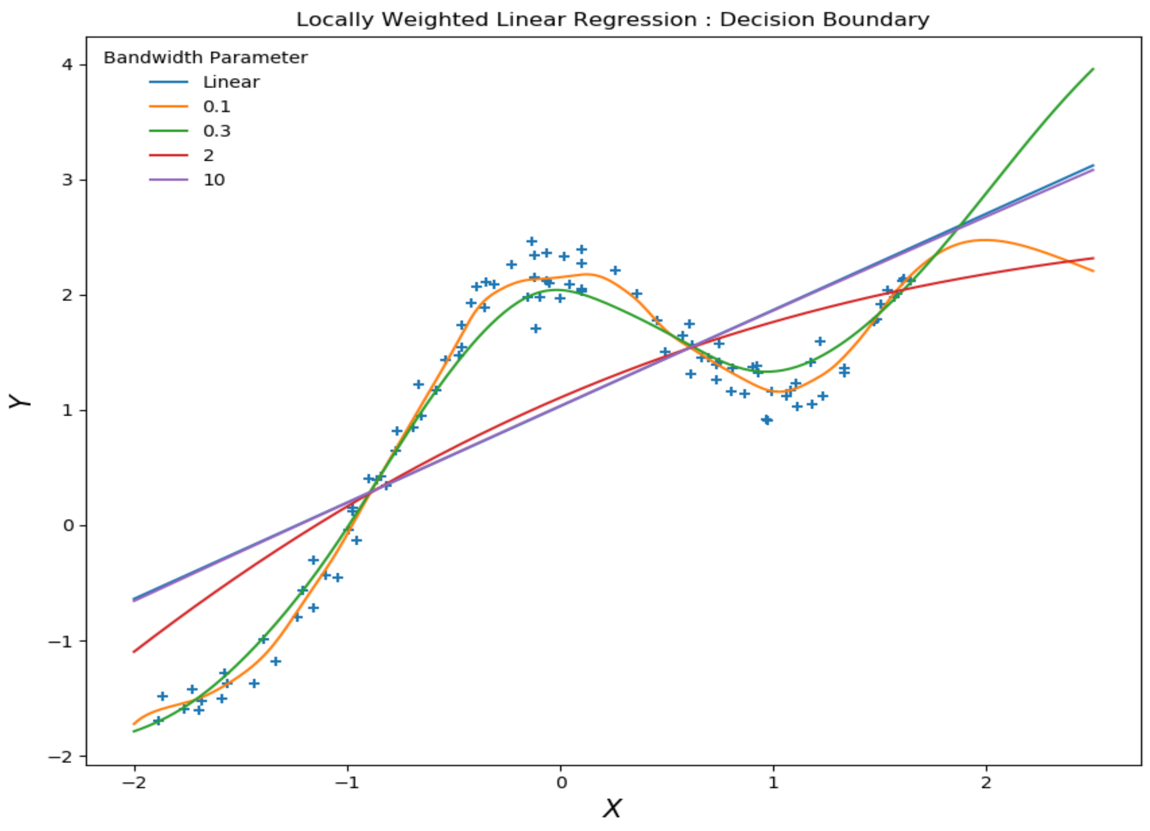

Figure 9: Hypothesis functions

Now, weights are given by, $w^{(i)} = exp(-\frac{(x-x^{(i)})^2}{2\tau^2})$. To have a strong fit means to put larger weights for the training datapoints in the neighbourhood of x, which would be prevented by the bandwidth parameter, τ. Hence when τ is too large, everybody gets equal weights, while when it is too small there is too much emphasis to neighbourhood and we get an overfit. The above plot very well justifies the scenario. (best $\tau = 0.3$)

3 Logistic Regression

Initialised $\vec{\theta} = [0, 0, 0]$ after normalising $x_1^{(i)}s$ and $x_2^{(i)}s$. Now, implementing Newton's method to maximise $L(\theta)$ where hypothesis $g(\theta^T x) = 1/(1 + exp(-\theta^T x))$, we obtain,

$$\theta^{t+1} = \theta^t - H^{-1}\Delta_\theta l(\theta)$$

where H is the Hessian matrix and which comes out to be,

$$H_{ij} = -\sum_{i=1}^{m}(1 - g(\theta^T x^{(i)}))(g(\theta^T x^{(i)}))x_j^{(i)}x_k^{(i)}$$

and Δ matrix is given by,

$$\Delta_\theta l(\theta)_j = \sum_{i=1}^{m}[y^{(i)} - g(\theta^T x^{(i)})]x_j^{(i)}$$

3.1

With a stopping criteria of (θ difference in consecutive iterations) 10^{-7}, the algorithm terminates in 9 iterations and we get the following decision boundary.

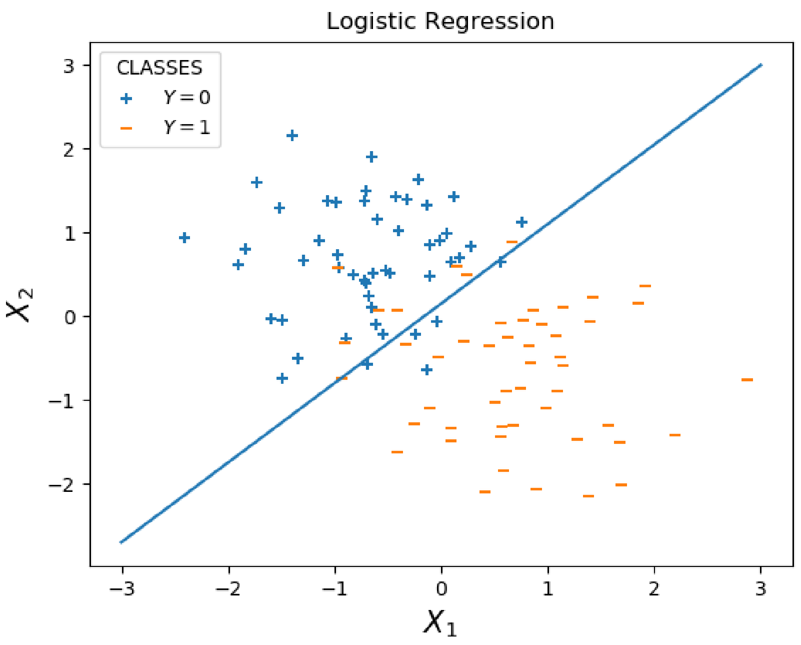

Figure 10: Decision Boundary

The equation of which is given by,

$$-2.725585X_2 + 2.5885477X_1 + 0.4012532 = 0$$

4 Gaussian Discriminant Analysis

Assigned Alaska as 0 and Canada as 1 for the purpose of classification, also normalised the data just as done previously.

4.1

Assuming identical covariance matrices i.e. $\Sigma_0 = \Sigma_1 = \Sigma$, and using the close form expressions for parameters, we obtain from our algorithm

$$\phi = 0.5$$

,

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

$$\Sigma_0 = \Sigma_1 = \Sigma = \begin{bmatrix} 0.42953048 & -0.02247228 \\ -0.02247228 & 0.53064579 \end{bmatrix}$$

Now, solving for decision boundary equating probabilities for both the classes we obtain a linear equation of the form $AX = B$ where,

$$A = (\mu_0^T - \mu_1^T)(\Sigma^{-1} + (\Sigma^{-1})^T)$$

and,

$$B = \mu_0^T \Sigma^{-1} \mu_0 - \mu_1^T \Sigma^{-1} \mu_1 + 2\log(\frac{\phi}{1-\phi})$$

which on plugging the above values by our algorithm yields the following equation -

$$4.877168 X_2 - 6.778509 X_1 = 0$$

4.2

Without making any assumptions about the covariance matrices whatsoever, we obtain the following parameters,

$$\phi = 0.5$$

,

$$\mu_0 = \begin{bmatrix} -0.75529433 \\ 0.68509431 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 0.75529433 \\ -0.68509431 \end{bmatrix}$$

$$\Sigma_0 = \begin{bmatrix} 0.38158978 & -0.15486516 \\ -0.15486516 & 0.64773717 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 0.47747117 & 0.1099206 \\ 0.1099206 & 0.41355441 \end{bmatrix}$$

Now, solving for decision boundary equating probabilities for both the classes we obtain a quadratic of the form $X^T AX + BX = G$ where,

$$A = \Sigma_1^{-1} - \Sigma_0^{-1}$$

$$B = \mu_0^T(\Sigma_0^{-1} + (\Sigma_0^{-1})^T) - \mu_1^T(\Sigma_1^{-1} + (\Sigma_1^{-1})^T)$$

and,

$$G = \mu_0^T \Sigma^{-1}\mu_0 - \mu_1^T \Sigma^{-1}\mu_1 + 2\log(\frac{\phi}{1-\phi}) - \log(\frac{|\Sigma_1|}{|\Sigma_0|})$$

which on plugging the above values by our algorithm yields the following equation :-

$$-0.6713478X_1^2 - 2.5736727X_1X_2 + 0.865932X_2^2 - 7.6157064X_1 + 5.7193461X_1 + 1.3557867 = 0$$

4.3

Plotting the data-points along with the decision boundary curves as derived above,

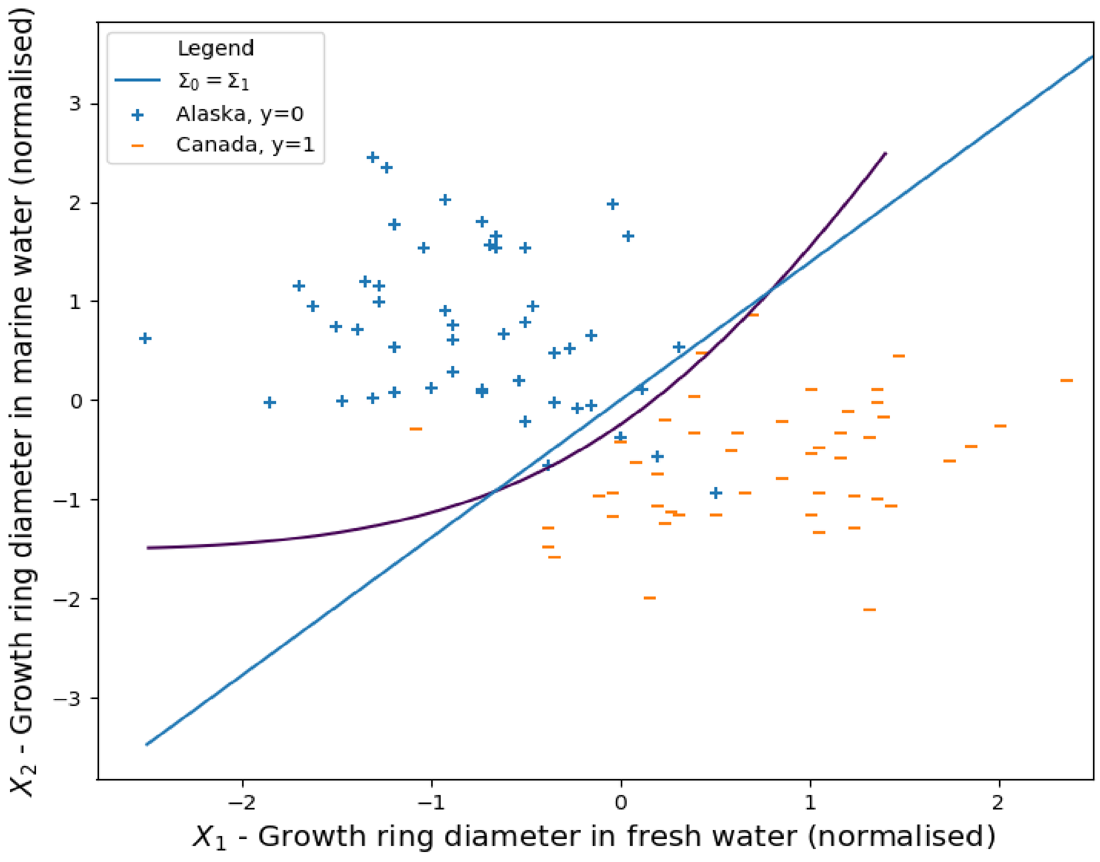

Figure 11: Boundary Equations

The quadratic equation is actually the equation of a hyperbola plotted as below,

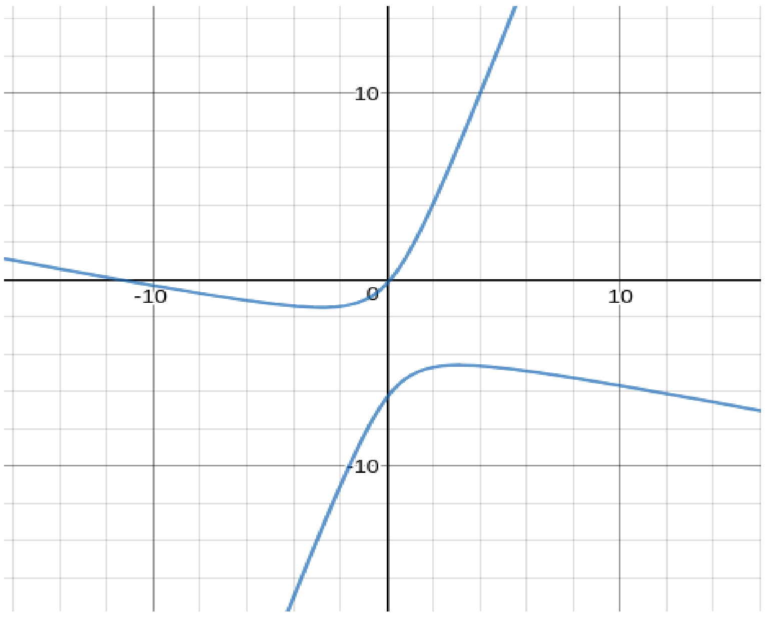

Figure 12: GDA decision boundary

As we observe from the plots that linear curve is a very good approximation and it produces erroneous results only for a few points very close to boundary. However, the tendency to accept errors for points lying in the ambiguous zone could be subjective and for that matter general GDA does a pretty decent job (only 2 training points are classified wrongly). Also to be noted that in this case $\phi = 0.5$. If in case this is not so, or in the training data there are a lot more $y = 0$ points than $y = 1$, linear could deviate very well.