Problem 1: Linear Regression :: Gradient Descent

(a)

- Stopping Criteria used, stop when:

$$|J(\theta_{prev}) - J(\theta_{curr})| < tolerance$$

  where:

  - θ_{curr} and θ_{prev} are the θ learned in current and previous iterations respectively
  - J is the cost function

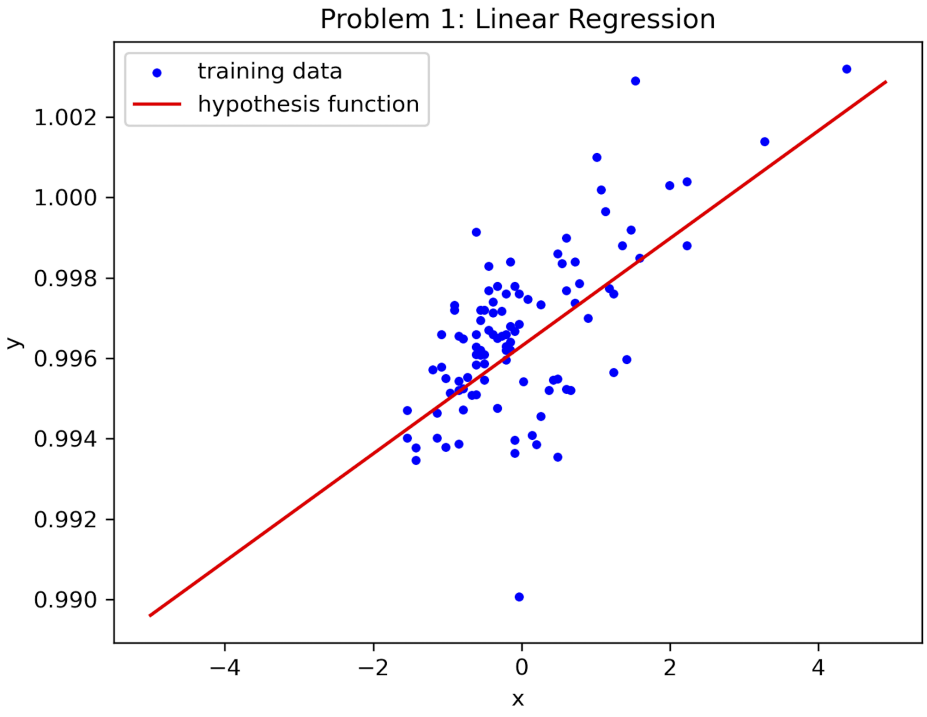
- Parameters chosen:

  - learning rate $\eta = 0.01$
  - tolerance = 10^{-9}

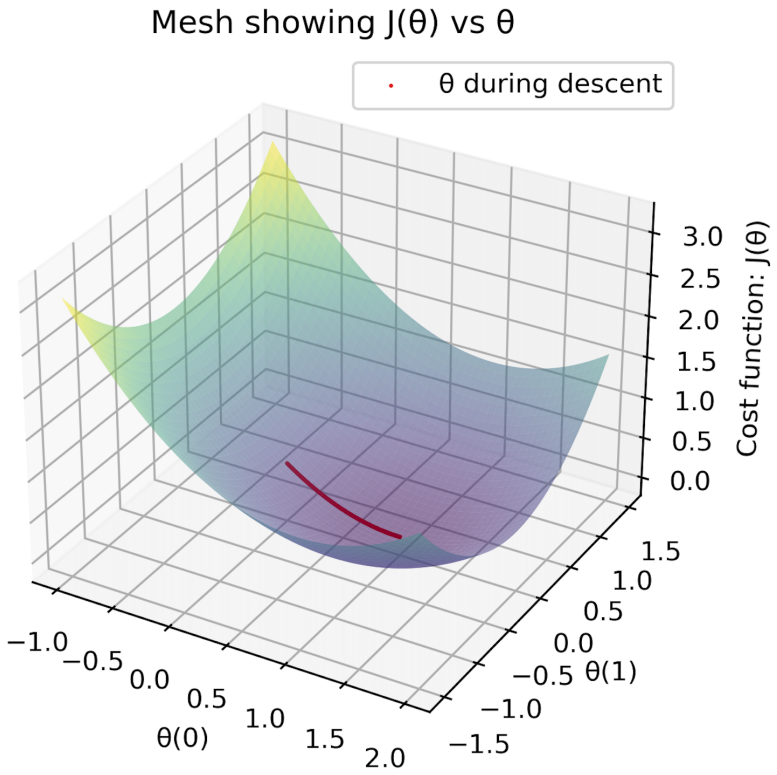
- Final values obtained:

  - $\theta_{learned}$: `[0.9963043  0.00133977]` (transposed)
  - $J(\theta_{learned})$: 1.24465 x 10^{-6}
  - Number of iterations: 8053
  - Running Time: 0.15697908401489258 seconds

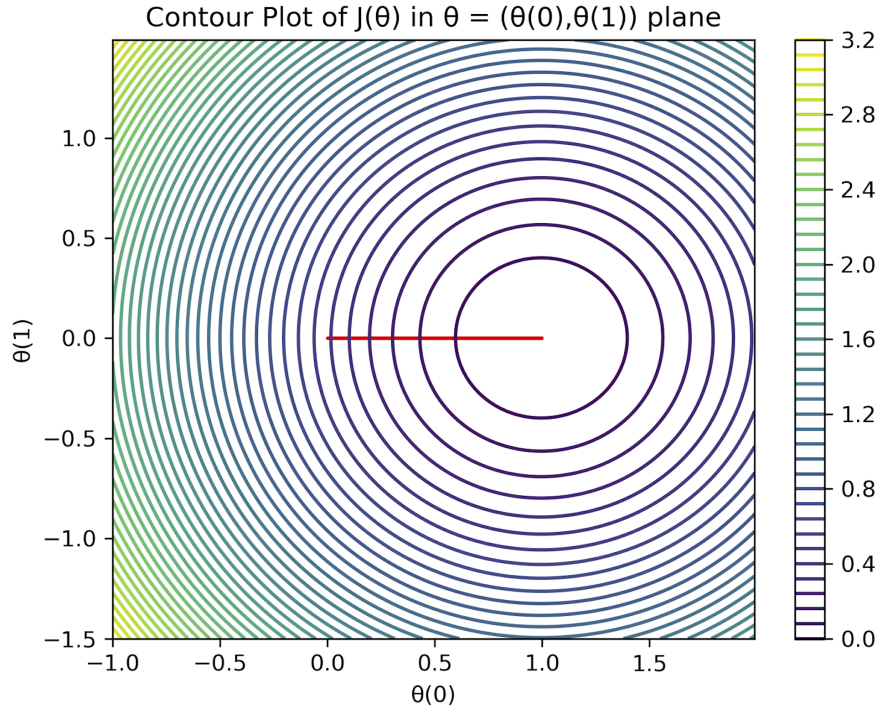
(b) Hypothesis function and training data:



(c) 3D mesh showing the error function $J(\theta)$ on z-axis on (θ_0, θ_1) plane

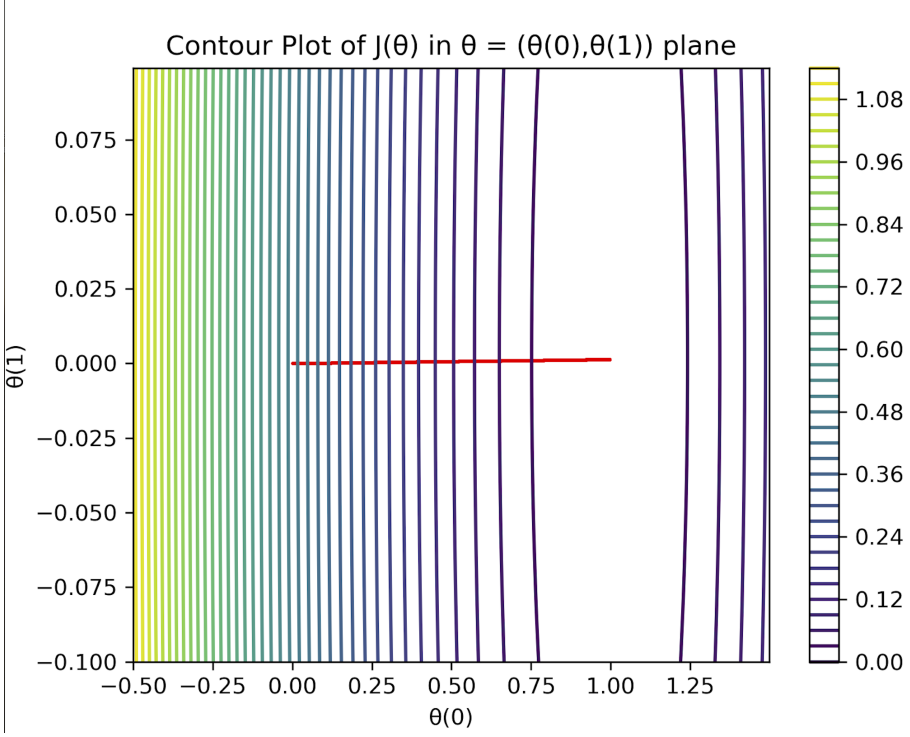


(d) Contours of the error function $(\eta = 0.01)$

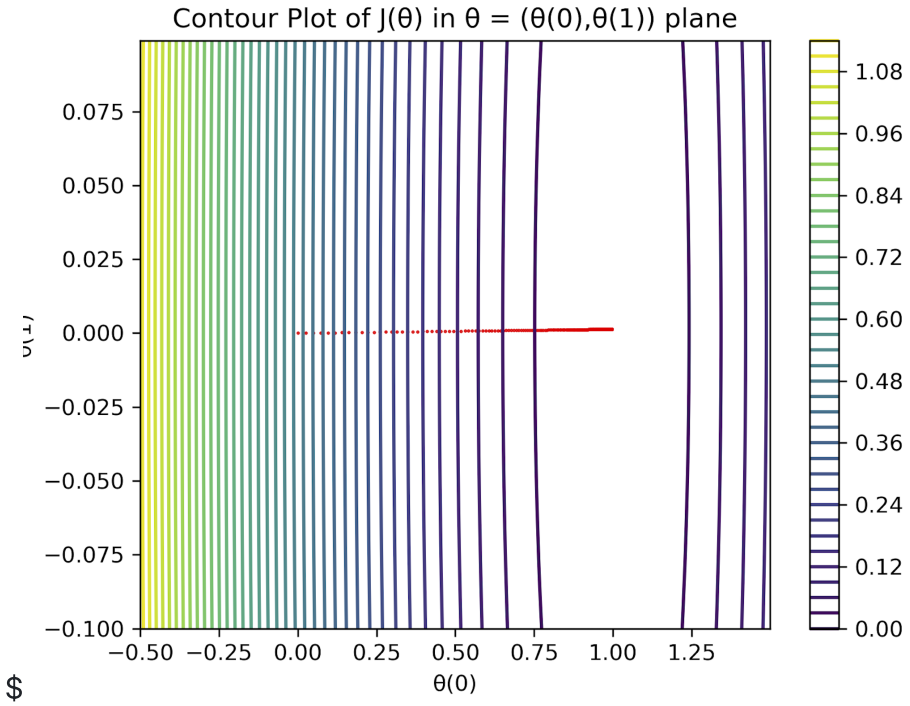


(e) Contour plots for different η values:

$\eta = 0.01$

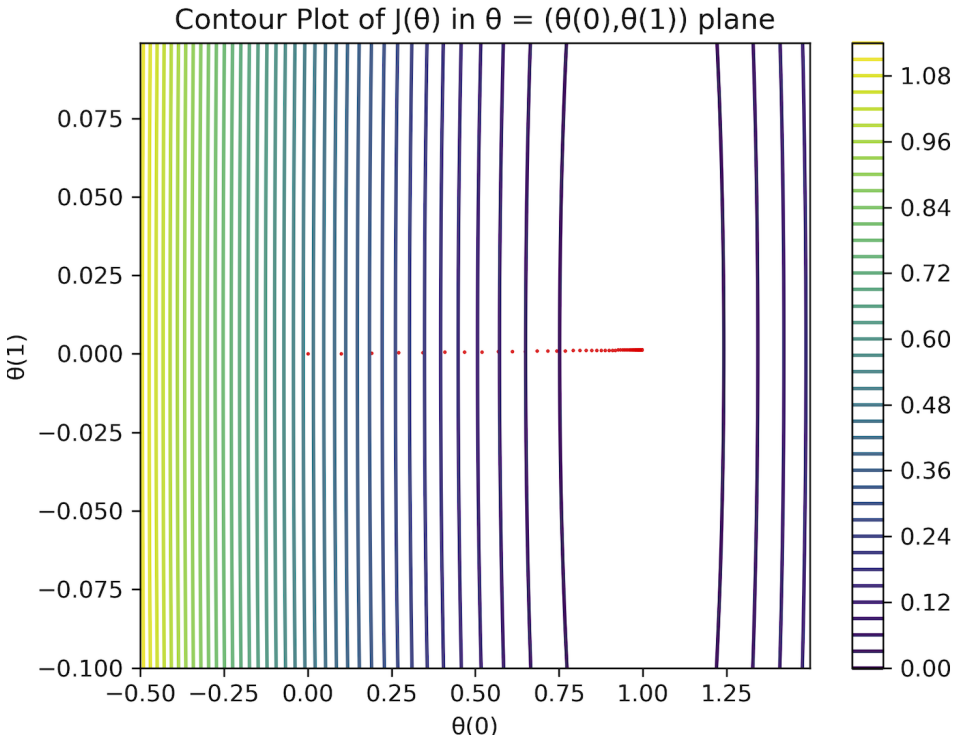


$\eta = 0.025$



$

$\eta = 0.1$



Observations:

Learning Rate	iterations	Time of Execution (seconds)
0.001	8053	0.1553962230682373
0.025	383	0.007498979568481445
0.1	100	0.0018768310546875
10	$diverges$	$diverges$

- For smaller values of learning rate (η), θ takes smaller steps in each iteration as the dots are less closesly spaced
- Smaller (η) takes larger number of iterations to convrge
- Larger (η), given it converges, hence also takes lesser time (difference though here is not significant due to small dataset)
- Too large (η) will diverge

Problem 2 :: Stochastic Gradient Descent

(a)

- Data sampling is done. Since it's randomly generated. Sample size = 1000000

(b)

- Suppose S (size = $|S|$) is the initial data, we partition it into r bathes, call them r_1 , r_2.....r_k$ where $r * k = |S|$

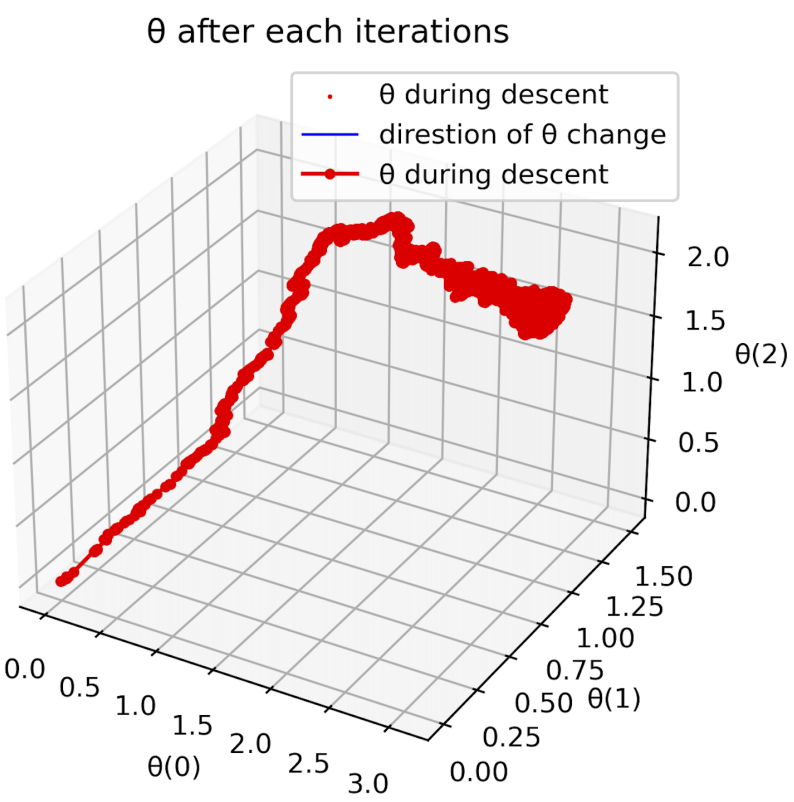
- One single $epoch$ is the `entire` traversal of the data, i.e. every batch once
- We iterate in a cyclic fashion $r_1 \rightarrow r_2 \ldots\ldots \rightarrow rk \rightarrow r1 \ldots$, for each batch r_i , we get the cost, call it J_i
- After one complete $epoch$ we take the average if $J_i, i \in [k]$,
- Let the mean J of the t^{th} complete epoch (as defined before, ran over all k batches once) be called J^t_{mean}
- Stopping criteria is:

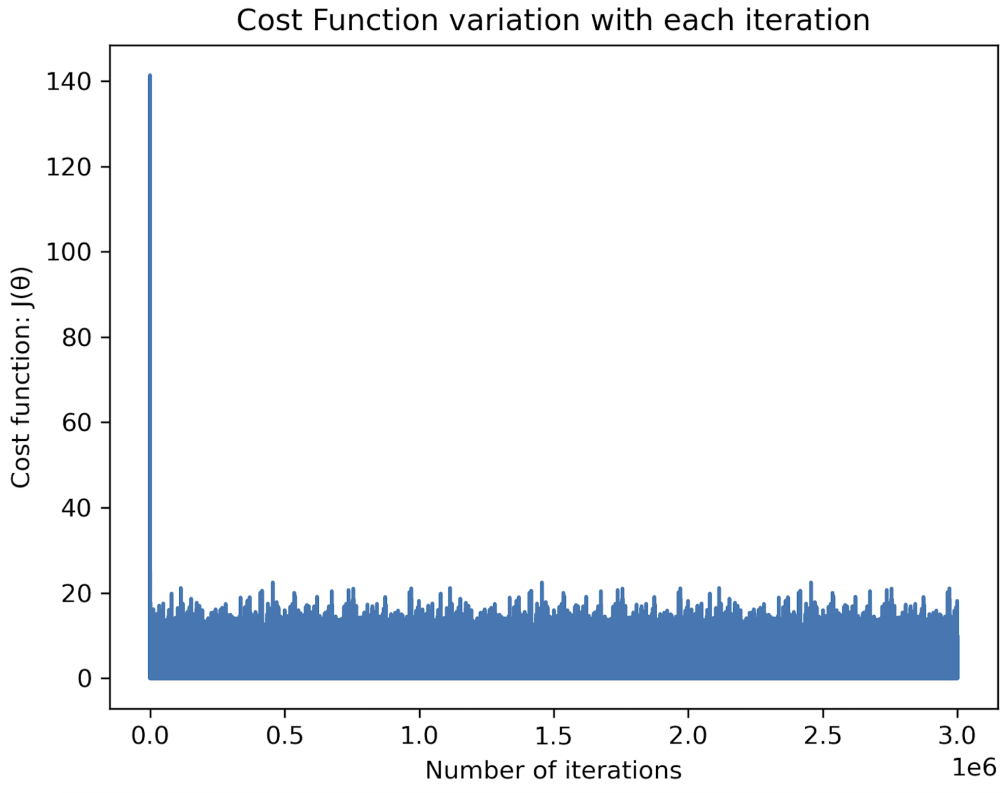
$$|J^t_{mean} - J^{t-1}_{mean}| < tolerance$$

- Curve on right shows $J(\theta)$ after each iteration, notice that is `wobbles` but there's an average decrease in $J(\theta)$ after each `epoche` .
- This motivates the stopping criteria

Batch size: 1

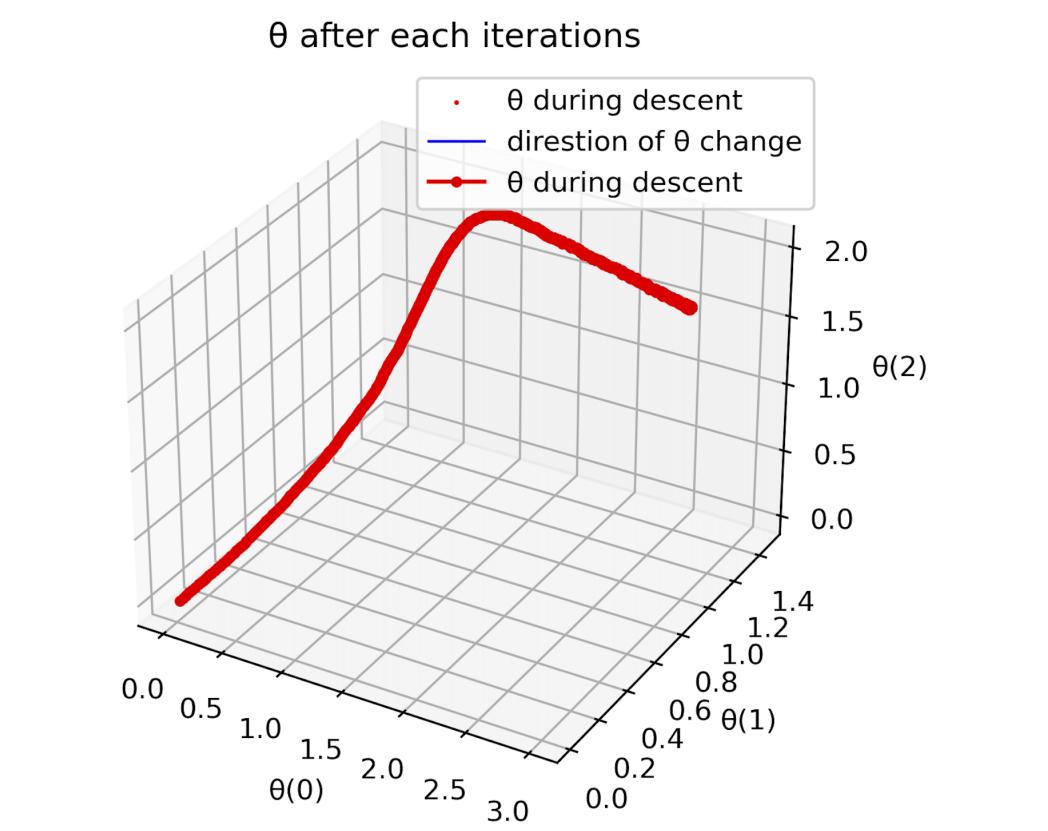
- Final θ value obtained: `[3.04305381 0.98018918 2.04124854]`

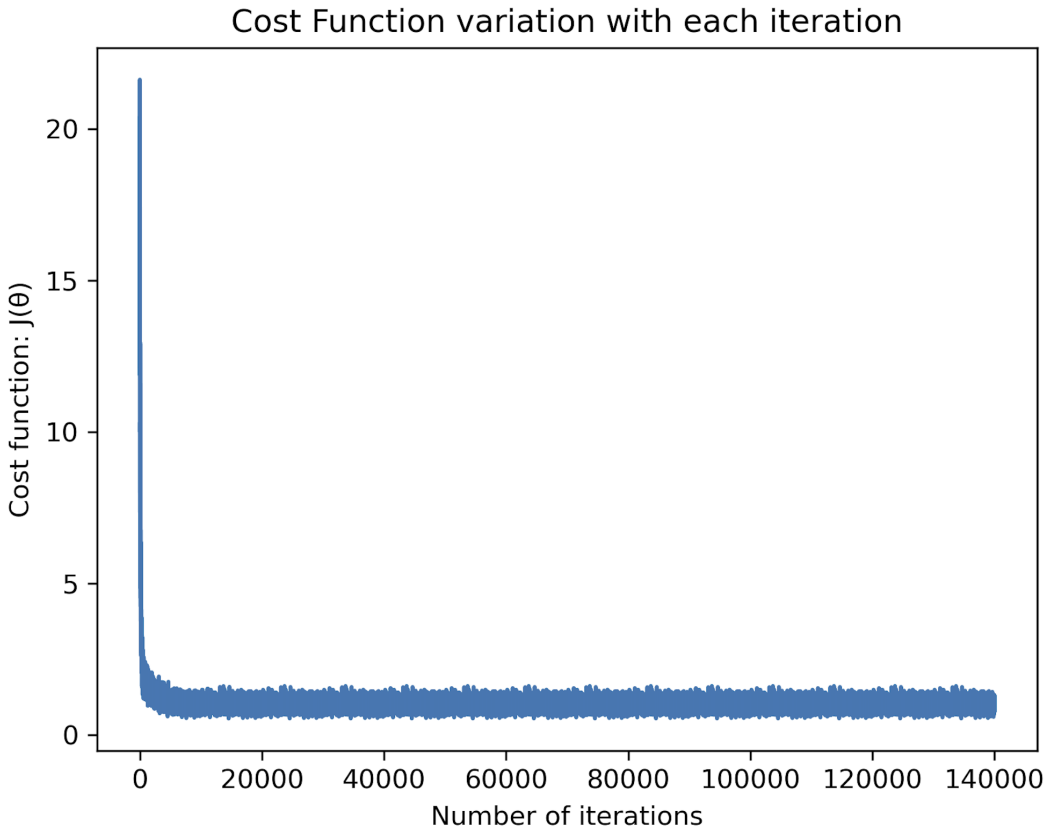




Batch size: 100

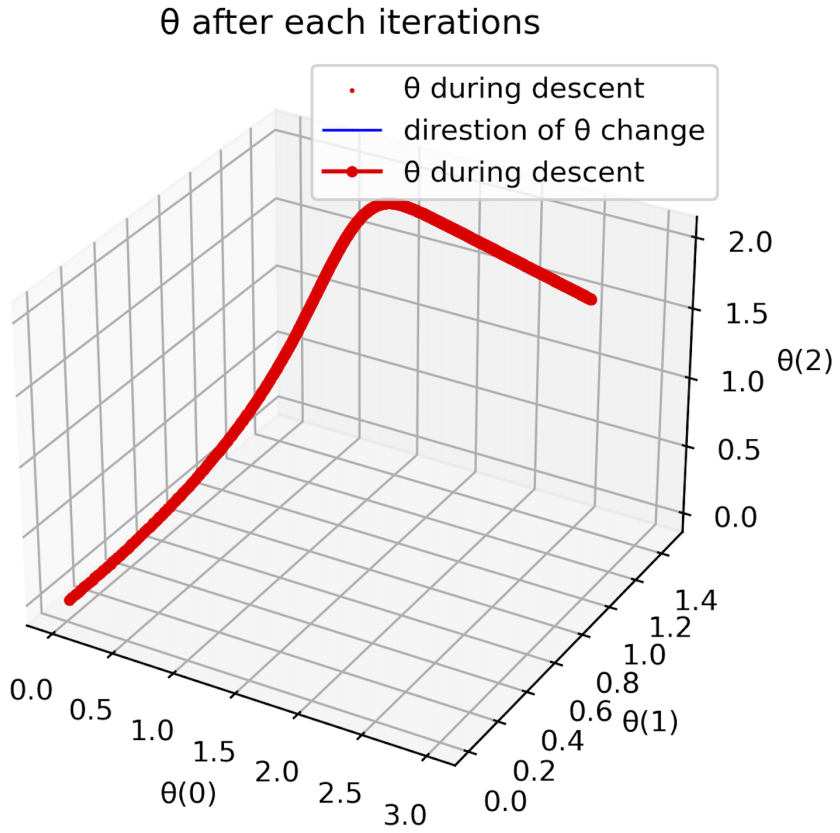
- Final θ value obtained: `[3.00324736 0.99868254 2.00283175]`

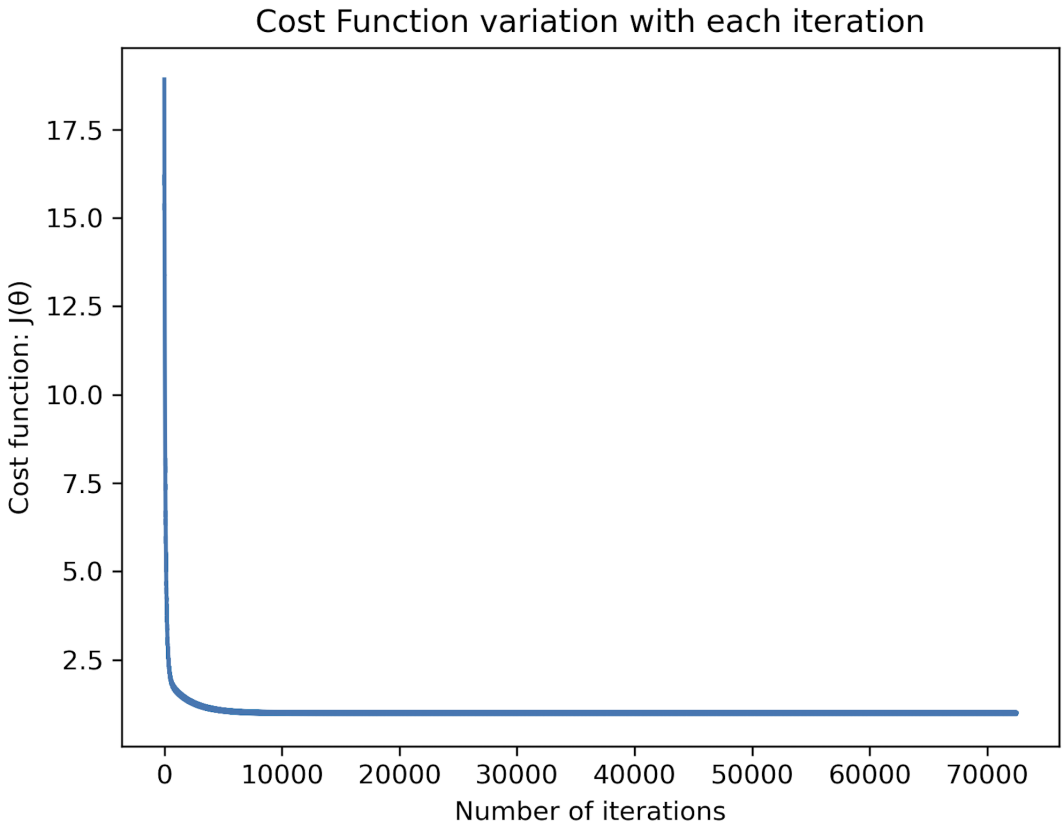




Batch size: 100000

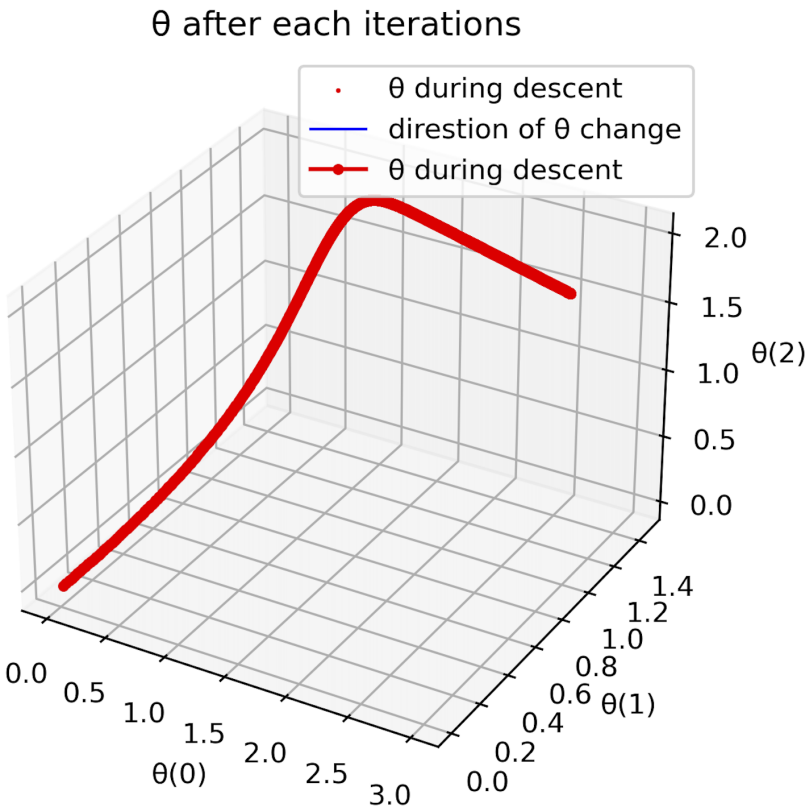
- Final θ value obtained: `[3.00205064 0.99928366 2.00016621]`

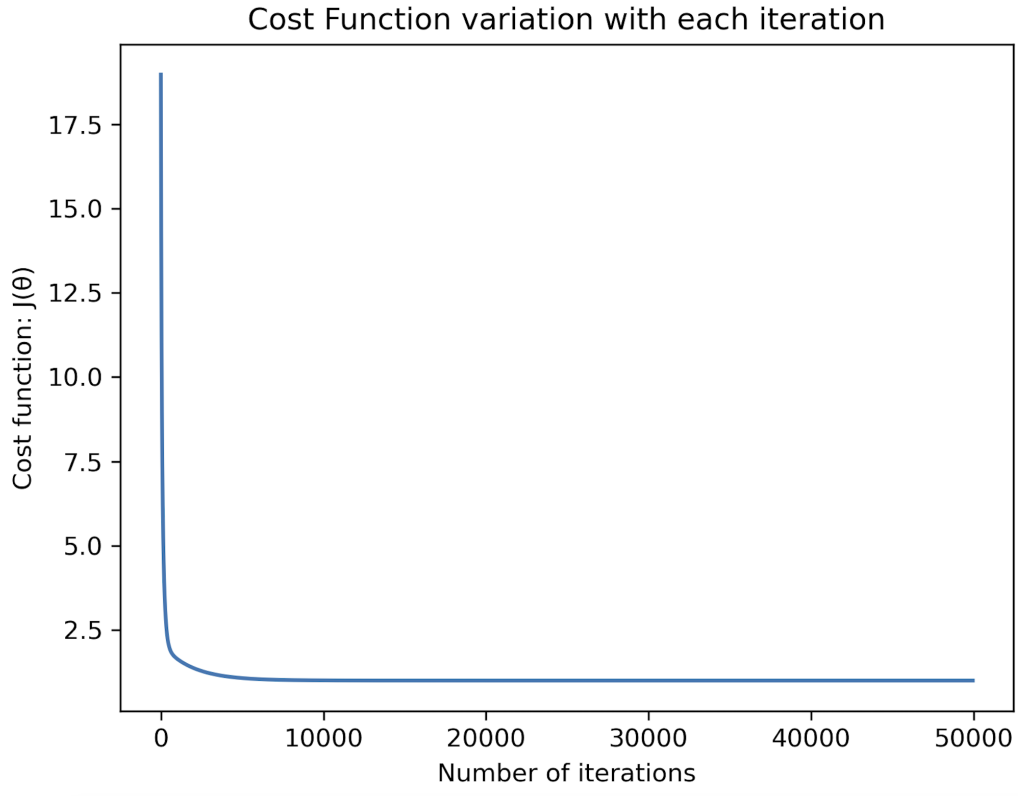




Batch size: 1000000

- Final θ value obtained: `[3.00208445 0.99947265 2.00003254]`





- shuffling after each iteration we should shuffle it in each Epoch to learn on possible variance from the data. The idea is that, a small bach, when iterated once, is hidden from the information of the `rest` data. Hence, if we keep shuffling after each epoch, `diffent sets` of information will be `bundled together` after each epoch, different from the bundle, present at that batch in the previous index, that will be be helpful in learning θ.

Batch Size	Number of Batches	Time of Execution (seconds)	Epoches	Iterations	J_{mean}
1	1000000	36.28994202613830	3	3000000	1.0117977074062936

Batch Size	Number of Batches	Time of Execution (seconds)	Epoches	Iterations	J_{mean}
100	10000	1.6672818660736084	13	130000	1.0022151554291192
10000	100	4.327163934707642	716	71600	1.002131704752294
1000000	1	244.99610996246338	49630	49630	1.0021296408169293

Observations:

- Different batch sizes, converse to alomst the same θ value, also close to $\theta_{original}$
- The number of $epoches$ monotonically increases with batch size
- Overall, the lowest batch size shows the lowest number of times the `entire data` has to be seen (epoches) which was infact the main intuition behind making batches.
- The number of iterations monotonically decreases with batch size
- Quantitatively, since in one iteration, B(1) sees only a small set of data (a lot of information is hidden in 1 iteration), it will `wobble` by larger amounts in each iteration, hence requiring a large number of iterations, analogously we can see why B(1000000) needs least iterations
- The same `wobbling` effect can be seen the $J(\theta)$ vs iteration graph and explained by the above argument
- But it should be noted that a single iteration for different batch sizes is not of the same computatinal effort, it's proportional to the batch size
- Hence, we see that the overall running time is least for 100 and 10000 sized batches (the middle ones). They give the fastest (absolute time) convergence.

(c) Errors obtained on Test data:

- for $\theta_{original}$, $J_\theta = 0.9829469215$
- here J_{mean} is J on mean θ of last epoch

Batch Size	J_{mean}
1	1.0842688473998006
100	0.983112794136199
10000	0.9829835949055766
1000000	0.9829773734489335

(d)

- The plots are shown above
- Let r be the size of a single batch
- From the diagrams, it's clear, that as r decreases, there is more `wobbling` in the value of θ after each iteration increases

- These movements were also analysed in `(c)`
- For smaller batch sizes, fluctuations are more, so they need more iterations to converge (as they're seeing a small part of the data in 1 iteration)
- For larger sizes, the consecutive values, in some sense are `consistent` and converge in lesser iterations

Problem 3: Logistic Regression

(a)

Update rule:

- Let there be n features (including intercept term) and m training examples $(x^{(i)}, y^{(i)})$, then H is an n x n matrix derived to be:

$$H_{pq} = \frac{1}{m}\sum_{i=1}^{m} g_\theta(x^{(i)})(1 - g_\theta(x^{(i)}))x_p^{(i)}x_q^{(i)}$$

  where

$$g_\theta(x^{(i)} = \frac{1}{1 + e^{-\theta^T x}}$$

- $l(\theta)$ is average log likelihood
- $\nabla_\theta l(\theta)$ is a size n vector with j^{th} component be derived to be:

$$(\nabla_\theta l(\theta))_j = \frac{1}{m}\sum_{i=1}^{m}(y - g_\theta(x))x_j$$

Stopping Criteria used, stop when:

$$|J(\theta_{prev}) - J(\theta_{curr}| < tolerance$$

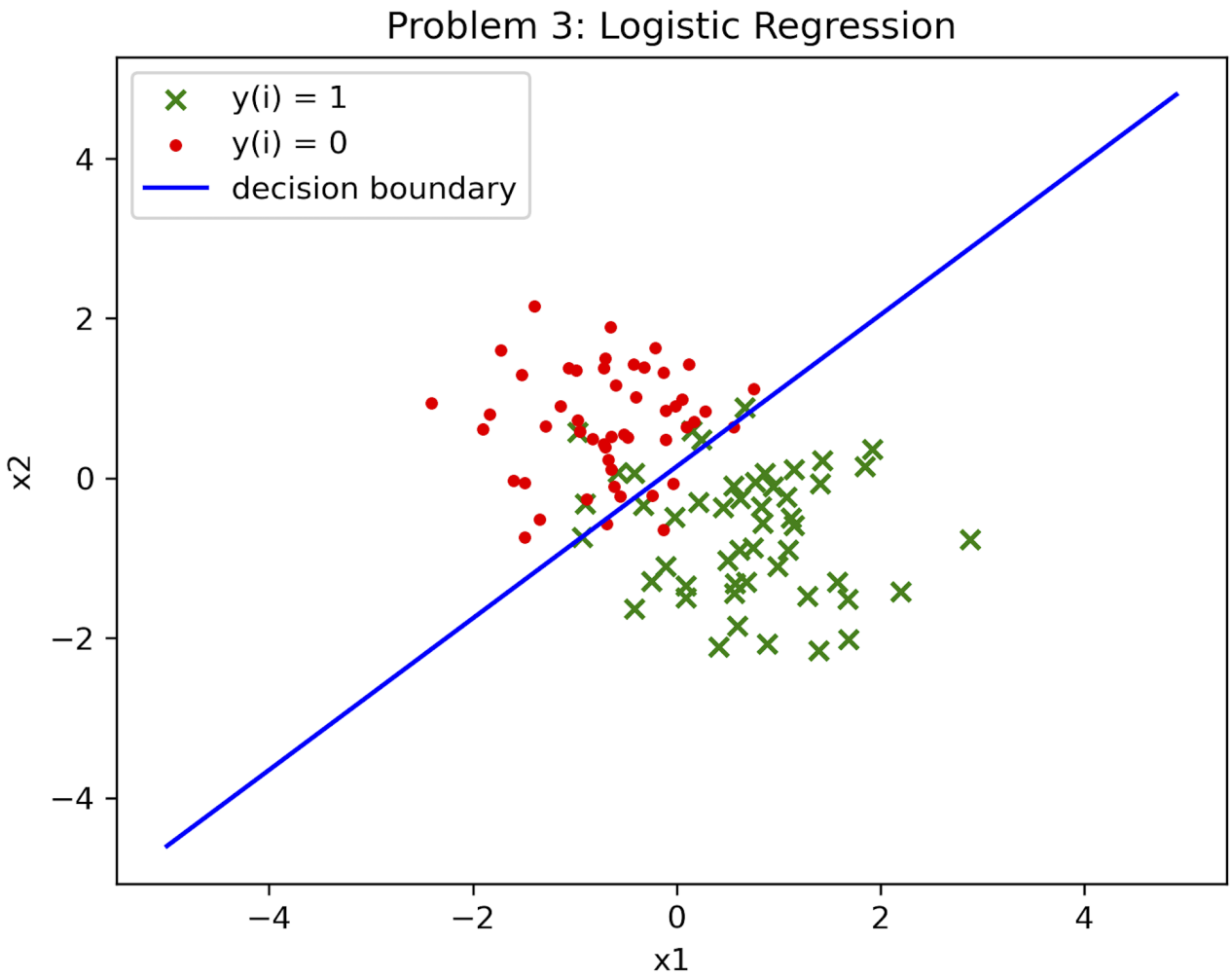
where:

- θ_{curr} and θ_{prev} are the θ learned in current and previous iterations respectively
- J is the cost function
- Finally, the update rule is:

$$\theta^{k+1} = \theta^k + H^{-1}\nabla_\theta l(\theta)$$

- Parameters chosen:
  - tolerance = 10^{-15}
- Final values obtained:
  - $\theta_{learned}$: `[ 0.40125316  2.5885477  -2.72558849]` (transposed)
  - $J(\theta_{learned})$: 0.035545843888358
  - Number of iterations: 9
  - Running time: 0.00534820556640625 seconds

(b) Decision boundary $(h_\theta(x) = 0.5)$ learned and training data:



Problem 4: Gaussian Discriminant Analysis

(a)

- Alaska is classified as 0 and Canada as 1

$$\phi = 0.5$$

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

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

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

(d)

$$\phi = 0.5$$

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

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

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

(e) Equation of the Quadratic Boundary is:

- We assume:

$$y - -Bernoulli(\varphi)$$

$$x|y = 0 - -N(\mu_0, \Sigma_0)$$

$$x|y = 1 - -N(\mu_1, \Sigma_1)$$

- In case of 2 classes, this is obtaines by equating

$$P(y = 0|x) = P(y = 1|x)$$

i.e.

$$\frac{P(x|y = 1)P(y = 1)}{P(x)} = \frac{P(x|y = 0)P(y = 0)}{P(x)}$$

which solves to the decision boundary $(P(y = 1|x) = P(y = 0|x) = 0.5)$

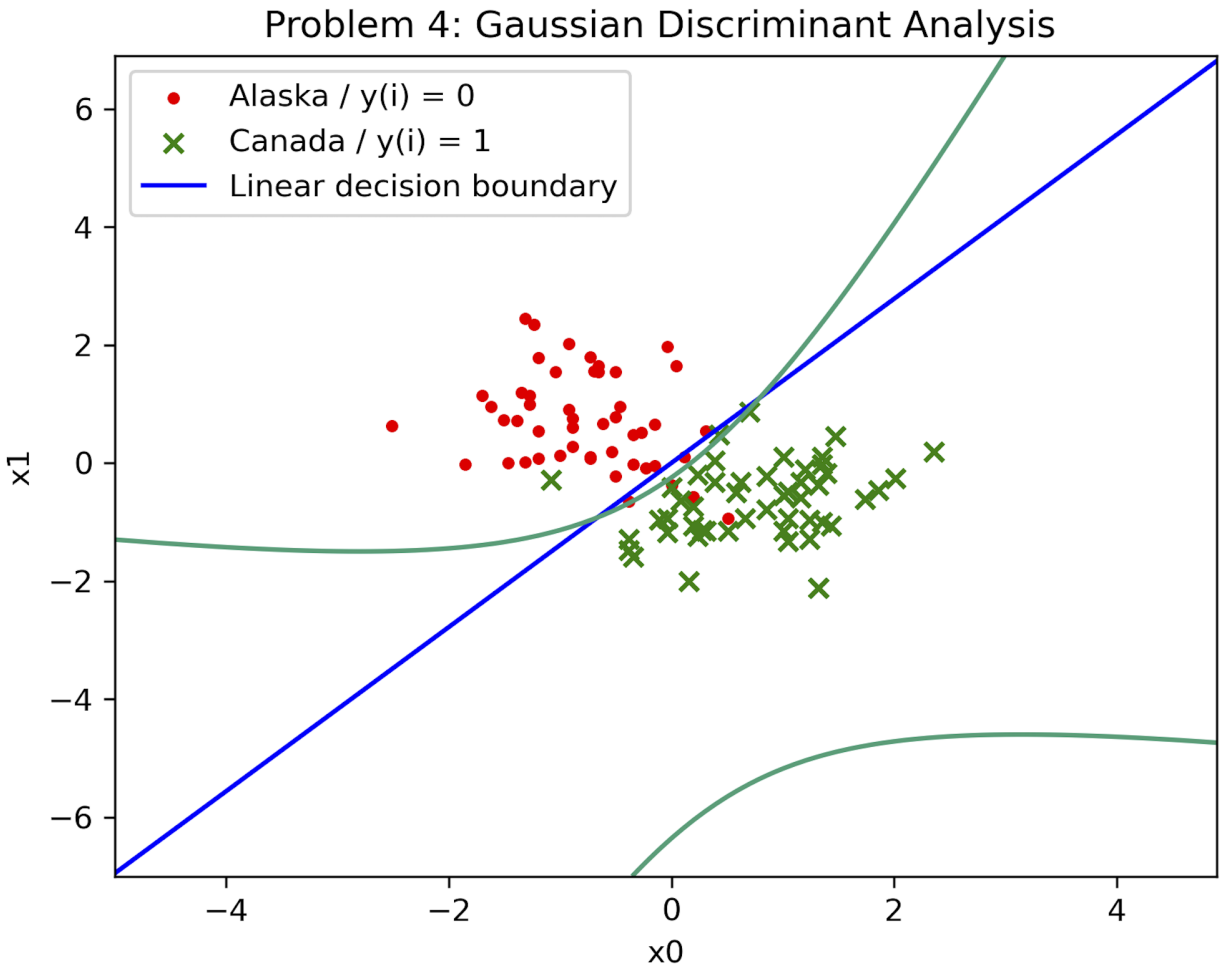
$$\frac{1}{2}x^T(\Sigma_1^{-1} - \Sigma_0^{-1})x - (\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1})x + \frac{1}{2}(\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0) + ln(\frac{1-\phi}{\phi}\sqrt{\frac{|\Sigma_1|}{|\Sigma_0|}}) = 0$$

(c) Setting $\Sigma_1 = \Sigma_2 = \Sigma$ in the above equation gives equation of the linear boundary:

$$-(\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1})x + \frac{1}{2}(\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0) + ln(\frac{1-\phi}{\phi}) = 0$$

(b) Plots of the Sample set, Linear Decision Boundary, and Quadratic Decision Boundary:

- The `green curve` repreents the quadratic boundary



(f)

- In general a Quadratic Boundary, may have the tendency to overfit while a linear boundary may have the tendency to underfi$ t (as we constrained both classes to have the same $\Sigma). In our diagram, the Quadratic Boundary, includes a few more `training points` of alaska region into within the expected region.
- Also notice, the Quadratic Boundary also has a branch in the bottom right region, suggestion that points in the region of `large x0` and `small x1` are predicted to be `Alaska` by the Quadratic Boundary and `Canada` by the linear boundary. It can be blamed on the fact, that we don't have data in this region and this also points to some overfitting by the Quadratic Boundary