Assignment 1

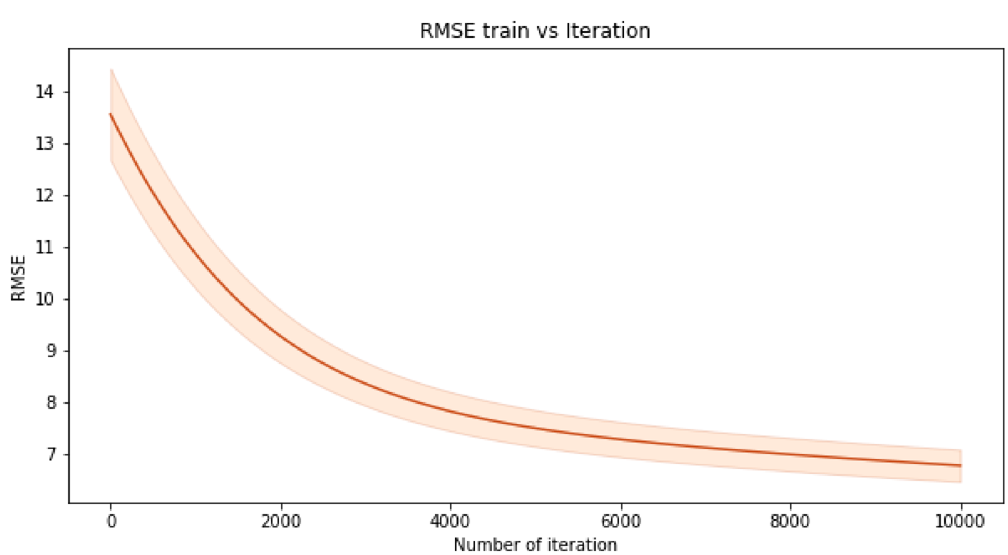
Question 1

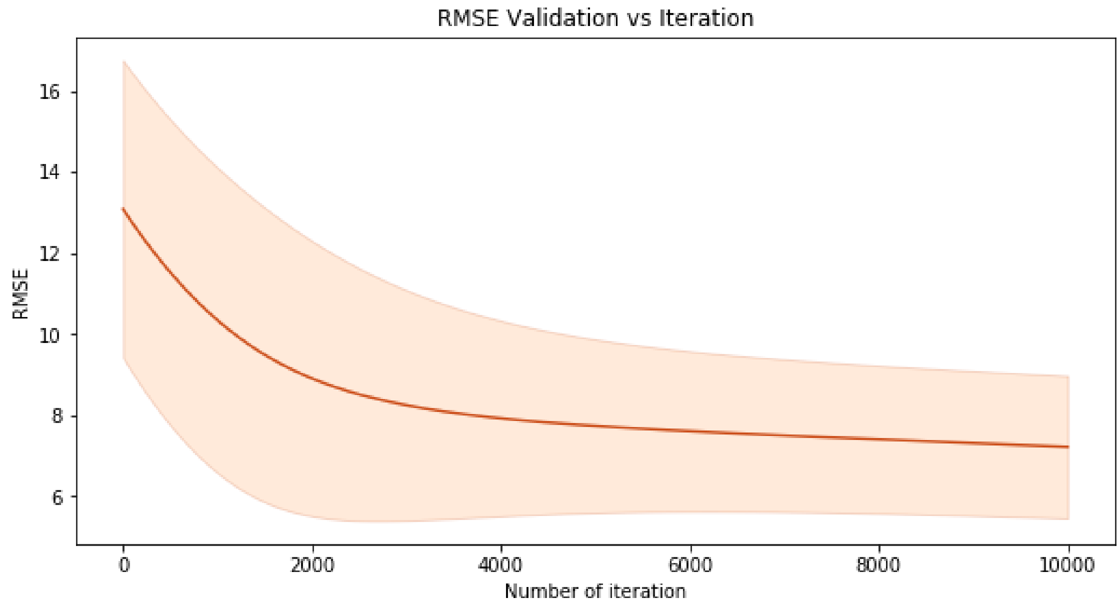
The solution of Question 1 is present in *Question_1.ipynb* Jupiter notebook. The code is well documented and a brief explanation is also provided below.

Part 1

- I have used pandas to read the dataset from a csv file. Because values of each column are in different scale I have normalized every feature using the formula $x` = \frac{x-min}{max-min}$. Here max and min correspond to the minimum and maximum in the column of *x*.
- Function *rmse()* return root mean squared error for given input samples *X*, ground truth *Y*, and parameter *theta.*
- Function gradient_descent() performs a vectorized gradient descent using numpy operations and record train and validation RMSE for each iteration. Return new values of parameter *theta* and train and validation RMSE of every iteration.
- After this, I perform a linear regression on the given dataset with 5 fold cross-validation. RMSE history of each fold is recorded. Also, the slipt with least validation RMSE is recorded.
- Function plot_mean_rmse_iteration() takes in mean RMSE value for every iteration and standard deviation in RMSE value for every iteration and plots the required graphs.
- We perform gradient descent for 10,000 iterations with a learning rate of 0.0001

Part a





Part b

Train RMSE of 5 folds in order are:-

7.23952174, 6.68748314, 6.34400462, 6.60777458, 6.97458909

Train RMSE error = 6.77 ± 0.31

Validation RMSE of 5 folds in order are:-

4.59167776, 7.46397876, 10.04884293, 7.44348269, 6.58135478

Validation RMSE error = 7.23 ± 1.76

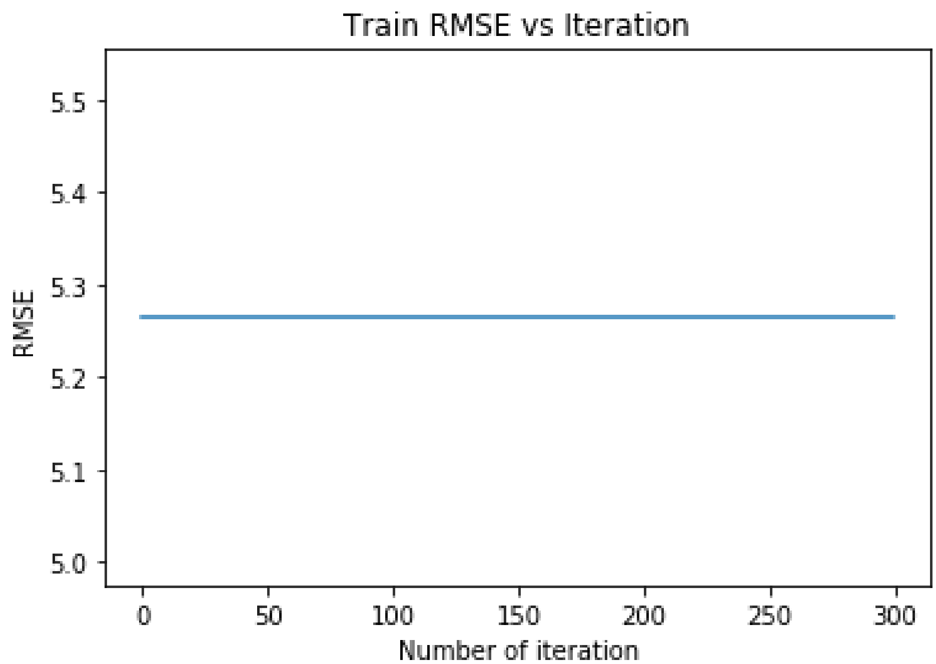
Part ii

- Fold with minimum validation RMSE from Part i was used as train and test split in this part. For my case, it was 1st fold.
- As instructed on backpack, sci-kit learn is used to perform L2 and regularized regression.

Part a

- GridSearchCV with cv=5 is used to find the best regularization parameters. Values of alpha checked were from 1.0 to 10.0.
- Best results were found with alpha = 3.0
- After getting the best value of alpha Ridge regression is used on the whole train set.

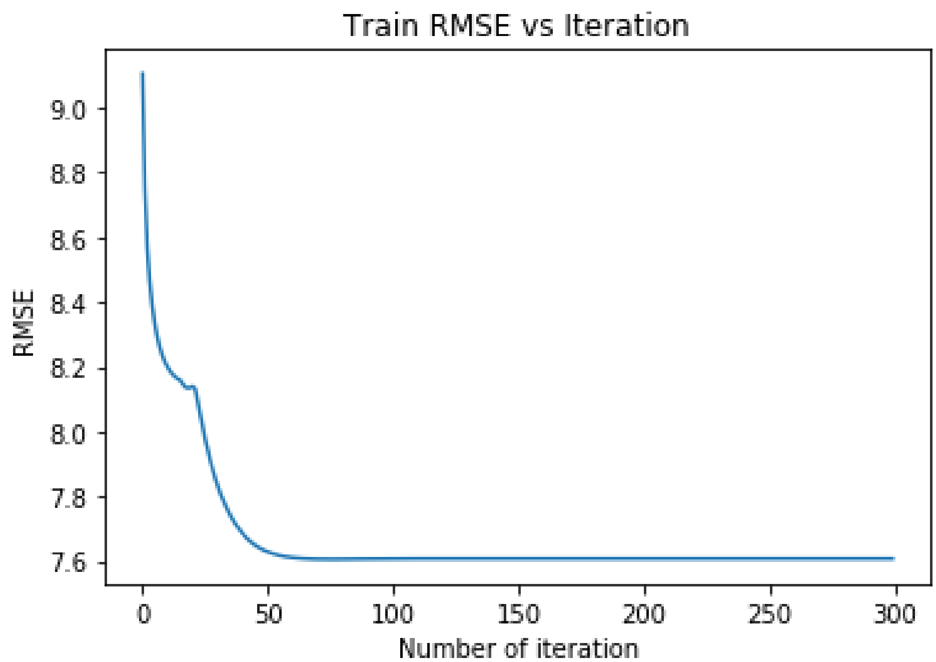
- Due to the way ridge regression is performed internally in sklearn, there is no way to get RMSE history for each iteration. However, because question needs us to plot a graph of RMSE vs iteration on train data I came up with a hack. Ridge regression has a parameter *max_iter* which is used to control the number of iteration, we can run ridge regression again and again for different iteration values and record last RMSE value of each time. I would like to mention that this is a very costly method and is not a very good programming practice, however, that will give the information required to plot the graph.
- Required graph plot is



- For some reason which I could not understand my train error for Ridge regression never changes.
- Train RMSE = 5.26457478718326
- Test RMSE = 3.143437256620382
- Here Test RMSE is less then Train RMSE which is not what usually happens. However, considering we have intentionally use a test train split in which test slip fit very well with the regression curve it is not surprising here.

Part b

- Steps same as Part a.
- Best value of apha is 1.0
- Required graph



- As shown in graph, model converges in around 70 iterations and there is no decrease in RMSE after that.
- Train RMSE = 7.607429679597901
- Test RMSE = 4.1099384763825535

Part iii

- RMSE of the train and test/validation splits are simillar in all cases and hence we can say none of the models are overfitting.
- L2 Regularization gets the leat RMSE value, then Normal regression and L1 regularized regression has the largest RMSE. This proves that regularization heps in better model fitting.
- There is also a significant time diffrence in the convergence of Regulrized version vs the normal version, however, I will not say it is completely due to regularization. We are using Sk learn for regularized models. Sklearn does a lot of optimizations which are out of the scope of this course and I think that is why those models are converging significantly faster.
- As no model has significantlt high RMSE, none of them is underfitting.

Question 2

- I have used Sci-kit learn inbuild LogisticRegression to perform this question.
- Solver 'saga' is used because it is faster than default 'libliner' and can be scaled to multiple threads.
- No preprocessing is being done and standard train test split of MNIST is used.

Part i

- Time taken to train model with L2 penalty was 6.205001592636108 minutes.
- Train accuracy = 92.6688220785281 %
- Test accuracy = 91.6658663465386 %
- Function get_accuracy_of_class() gives accuarcy for each class in ovne-vs-all fashion.
- The train accuracy of each class in order class number 0 to 9 are:-
  99.3917072195187 %,
  99.35004333044464 %,
  98.35344310379308 %,
  98.11845876941537 %,
  98.73841743883741 %,
  98.04346376908206 %,
  99.10672621825212 %,
  98.75508299446704 %,
  97.51183254449703 %,
  97.96846876874875 %
- The Test accuary of each class in order class number 0 to 9 are:-
  99.29971988795518 %,
  99.09963985594238 %,
  97.98919567827131 %,
  97.82913165266106 %,
  98.34933973589436 %,
  97.78911564625851 %,
  99.03961584633854 %,
  98.56942777110844 %,
  97.32893157262905 %,
  97.6390556222489 %

Part ii

- Time taken to train model with L1 penalty was 6.220364042123159 minutes.
- Train accuracy = 92.74381707886141 %
- Test accuracy = 91.29651860744298 %
- Function get_accuracy_of_class() gives accuarcy for each class in ovne-vs-all fashion.

- The train accuracy of each class in order class number 0 to 9 are:-
  99.39837344177055 %,
  99.37504166388907 %,
  98.36177588160789 %,
  98.12845810279315 %,
  98.7434171055263 %,
  98.05679621358576 %,
  99.12672488500767 %,
  98.7700819945337 %,
  97.54516365575628 %,
  97.98180121325245 %
- The test accuracy of each class in order class number 0 to 9 are:-
  99.28971588635455 %,
  99.04961984793917 %,
  97.95918367346939 %,
  97.76910764305722 %,
  98.32933173269308 %,
  97.75910364145658 %,
  99.0296118447379 %,
  98.52941176470589 %,
  97.29891956782714 %,
  97.57903161264506 %

Part iii

- Model is not overfitting, as we can see that accuracy numbers on both train and test set are very identical, this is a sign of model is fairly generalized. It can be proven futher by using k fold cross-validation, however, because the dataset is so large and refined we can confidently make this statment by only using the default train and test set.
- Considering accuracy number on both train and test set is fairly good we can not say that model is underfitting, however, one should know that even better performance can be obtained on this dataset using fine tunning or other types of classifiers.
- L2 regularization works slightly better than L1.
- Training time is also almost identical.

Question 3

3) To understand this we have to look into how decision boundary for simple logistic regression is made:-

we do

$$h_\theta(x) = g(\theta^T x)$$

where

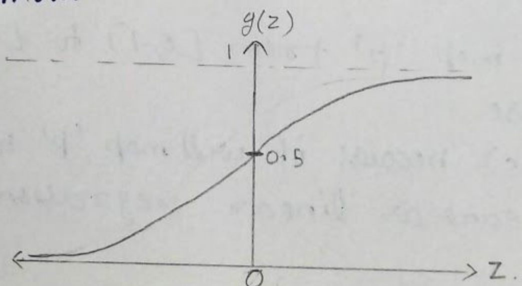
$$g(z) = \frac{1}{1+e^{-z}} \leftarrow \text{Sigmoid function.}$$

Now $g(z)$ will give us probability values.

$\therefore$ for $y=1$ we need $h_\theta(x) \geq 0.5$.

||ly for $y=0$ we need $h_\theta(x) \leq 0.5$

→ Plot sigmoid



→ if a sample has to be predicted as $y=1$ then:-

$h_\theta(x) \geq 0.5$

$\Rightarrow g(\theta^T x) \geq 0.5$

from plot of above sigmoid we can tell that to get $g(z) \geq 0.5$, z should be ≥ 0.

hence $\theta^T x \geq 0$

||ly for $y=0$, $\theta^T x < 0$

$\therefore$ prediction ~~value~~ class depends on value of $\theta^T x$ which is eqn of a line & hence logistic regression uses a linear separation.

Question 4

4) In logistic regression because Y is categorical we want to predict probability value 'p' for a class of Y

now $p \in [0,1]$ but linear regression the output will be $[-\infty, \infty]$
if we want to use

then one option is we use some invertible function of 'p' which goes $[-\infty, \infty]$ & this is logit transform.

→ odd is a invertiable of 'p'

$$\text{Odds} = \frac{p}{1-p} \quad \Big| \quad p = \frac{\text{odds}}{1+\text{odds}}$$

Now
odds map 'p' from $[0,1]$ to $[0,\infty]$.
we can use
log (odds) because it will map 'p' to $[-\infty, \infty]$
which is same as linear regression.

hence
to get 'p' we do linear regression on log (odds) & this is logit transform.

→ logistic regression Expression.

Logistic regression Expression → $\log(\text{odds}) = \theta^T X$

$$\log\left(\frac{p}{1-p}\right) = \theta^T X$$

say $z = \theta^T X$

$$\log\left(\frac{p}{1-p}\right) = z.$$

How to recove p →

$$\Rightarrow \frac{p}{1-p} = e^z$$

$$\Rightarrow p = (1-p)e^z$$

$$\Rightarrow p = e^z - pe^z.$$

$$\Rightarrow p + pe^z = e^z.$$

$$\Rightarrow p(1+e^z) = e^z$$

$$\Rightarrow p = \frac{e^z}{1+e^z}$$

$$p = \frac{1}{1+e^{-z}} \Rightarrow \text{Sigmoid function.}$$

Question 5

$$H(x) = -\int P(x) \ln (P(x))\, dx$$

$$= -\int N(x|\mu,\Sigma) \ln N(x|\mu,\Sigma)\, dx$$

$$= -\int N(x|\mu,\Sigma) \ln \left(\frac{1}{|2\pi\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \right) dx$$

$$= -\int \underbrace{N(x|\mu,\Sigma)}_{\text{integration is 1}} \underbrace{\left(-\frac{1}{2}\ln(|2\pi\Sigma|)\right)}_{\text{constant.}} dx + \frac{1}{2}\int N(x|\mu,\Sigma)\left((x-\mu)^T\Sigma^{-1}(x-\mu)\right) dx$$

$$= \frac{1}{2}\ln(|2\pi\Sigma|) + \frac{1}{2}\int Tr[\Sigma^{-1}(x-\mu)(x-\mu)^T]\, N(x|\mu,\Sigma)\, dx.$$

This is validas integation goest to each term of matrix

$$= \frac{1}{2}\ln(|2\pi\Sigma|) + \frac{1}{2} Tr\left\{\Sigma^{-1} \underbrace{\int (x-\mu)(x-\mu)^T N(x|\mu,\Sigma)\, dx}_{\Sigma}\right\}$$

$$= \frac{1}{2}\ln(|2\pi\Sigma|) + \frac{1}{2} Tr[\Sigma^{-1}\Sigma]$$

$$= \frac{1}{2}\ln(|2\pi\Sigma|) + \frac{D}{2}.$$

$$= \frac{D}{2}\ln(2\pi\Sigma) + \frac{D}{2} = \frac{D}{2}\ln(2\pi) + \frac{D}{2}\ln(\Sigma) + \frac{D}{2}$$

$$H(x) = \frac{D}{2}(1+\ln(2\pi)) + \frac{1}{2}\ln(|\Sigma|)$$

Hence proved.

Question 6

6) given $Y = (1/\lambda) R x + B$

where $B = [a\ b]^T$ & $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Now cost function = RMSE.

$$L = \sqrt{\sum_{i=1}^{n} \left(y_i - \left(\left(\frac{1}{\lambda}\right) R x_i + B \right) \right)^2}$$

to minimize L, we can say $\frac{\partial L}{\partial(\text{parameter})} = 0$

$\therefore$ to find λ.

$$\frac{\partial L}{\partial \lambda} = 0.$$

$$\frac{\partial}{\partial \lambda} \sum_{i=1}^{n} \left(y_i - \frac{1}{\lambda} R X_i - B \right)^2 = 0.$$

$$\sum_{i=1}^{n} 2\left(y_i - \frac{1}{\lambda} R(X_i) - B \right) \left(0 + \frac{R X_i}{\lambda^2} - 0 \right) = 0.$$

Now term 2 = 0 will not give any Real solution

$$\therefore \left(y_i - \frac{R(x_i)}{\lambda} - B = 0. \right.$$

$$y_i \lambda - R X_i - B\lambda = 0.$$

$$\lambda(y_i - B) = R X_i$$

$$\lambda = \frac{R X_i}{y_i - B}.$$

$$\therefore \lambda = \sum_{i=1}^{n} \frac{R(X_i)}{y_i - B}.$$

Illy we do $\frac{\partial L}{\partial R} = 0.$

$$\frac{\partial}{\partial R} \sum_{i=i}^{n} \left(y_i - \frac{1}{\lambda} R x_i - B \right)^2 = 0.$$

$$\sum_{i=1}^{n} 2\left(y_i - \frac{1}{\lambda} R x_i - B \right) \left(-\frac{X}{\lambda} \right) = 0.$$

to solve for R.

$$Y - \frac{R X}{\lambda} - B = 0.$$

$$R = \sum_{i=1}^{n} \frac{(Y_i - B)\lambda}{X_i}$$

Now here.

Y_i is a 2×1 vector.

B is a 2×1 vector

and X_i ________________

$\therefore$ $(Y_i - B)\lambda$ will be a 2×1 vector

and $(Y_i - B)\lambda * X_i^{-1}$ $\therefore$ $(2\times1) * (1\times2)$ vector

Hence R.H.S will be a 2×2 matrix.

we can do a element wise comparsion on L.H.S & RH.S to find $\underline{\theta}$.

Illy

$$B = \sum_{i=1}^{n} \left(y_i - \frac{R X_i}{\lambda} \right)$$

here R is (2×2) & X_i is (2×1)

$\therefore$ $R X_i = 2\times1$

hence RH.S is 2×1 which is same as dimension of B.

do a element wise comparision to find a & b.

To find optimal value using gradient descent we will have 3 update rules.

$$\lambda^{i+1} = \lambda^{i} - \alpha\left(\sum_{i=1}^{n} \frac{R(x_i)}{y_i - B}\right).$$

$$R^{i+1} = R^{i} - \alpha\left(\sum_{i=1}^{n} \frac{(y_i - B)\lambda}{x_i}\right)$$

$$B^{i+1} = B^{i} - \alpha\left(\sum_{i=1}^{n} \left(y_i - \frac{R x_i}{\lambda}\right)\right).$$