CSE574 Introduction to Machine Learning

Project 1.2

Learning to Rank Using Linear Regression

Project Report

Submitted By

Karan Manchandia

**Abstract**

This project aims to solve the learning to rank (LeToR) problem that arises in Information Retrieval by using machine learning. We solve this problem by using linear regression and we get a scaler output y for each input vector x. The LeToR data set has input features derived from the query document pair. The result would be target scaler values 0, 1, 2. The larger the value, the greater is the match between the query and the document.

**Introduction**

There are two ways we solve this problem:

1. By using Closed Form Solution
2. By using Stochastic Gradient Decent Solution

**Closed Form Solution:**

Generally speaking, the linear regression model helps us to find a hypothesis or trend that is linear in shape.

**Why we cannot use classification for this problem?**

We should use classification when we are finding buckets for our inputs, but in the LeToR problem, we are finding a particular value for our input, which can be done using linear regression.

Linear Regression establishes a relationship between two variables by fitting a linear equation to observed data. Our linear regression functio has the form:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$$

Here w is a weight vector, ∅ is a vector of M basis function. In closed form solution we are using Gaussian Radial Basis Function which is given by the equation:

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^\top \Sigma_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right)$$

Here ∑ is called covariance matrix.

**Why we multiply ∑ by 200 (in the code)?**

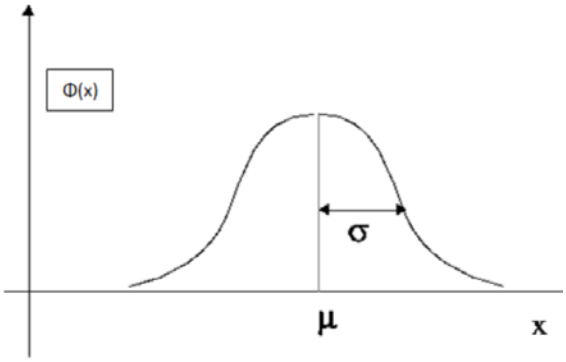
The Covarience matrix decides how broadly basis function spreads. So, to increase the width of basis function to get a better approximation and better results we multiply it by 200.

**Why we are using Basis Function?**

- The functions that help us to build some complex functions are called basis function. We are using basis function because whatever curve that can best fit our input data can be derived by using M number of basis function.
- Using basis function we can convert our problem to a linear regression problem.

**Why we are using Gaussian Basis Function and not any other basis function?**

We are using Gaussian Basis Function because we have control over its shape. We can change its shape so that it can suit our data. The figure below shows a gaussian basis function with mean μ and standard deviation ∂. We can change its shape by expanding or compressing it.




**On what factors number of Basis Function Depends?**

The number of basis function depends on the number of clusters we get for a given data set. It may also depend on how much the given data set is varying.

**The gaussian basis function we are using to solve the problem is not linear, so why it is called a linear regression problem?**

In order to solve a problem using simple linear regression, we do linear composition of variables. But for the problem, we are given here (LeToR) we have to do linear composition of functions ∅(x). That is why this problem is a linear regression problem.

The next step in the closed form solution would be to minimize the sum of square error for input vector X={x1,…,xn} and target vector t={t1,….,tn}, which is given by the equation:

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

Next, the close form solution is given by the equation:

$$\mathbf{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top \mathbf{t}$$

This equation is known as Moore Penrose pseudo inverse.

**Why we use Moore Penrose pseudo inverse?**

To get the solution to our problem we have to do inverse of Φ matrix, which is not a square matrix, hence its inverse is not possible. To solve this problem we multiply $\Phi^{-1}$ with an identity matrix, substitute $\Phi^T$ $(\Phi^T)^{-1}$ in place of the identity matrix, and hence we get Moore Penrose pseudo inverse equation.

The dimentions of all the matrices play an important role in the closed form solution to the LeToR problem. It is required to take care of dimentions such that the martices have a perfect fit in the equations.

**Dimensionality Table:**

| Matrix/ Vector | Dimentions<br>[Note: Dimentions may change for transposes] |
|---|---|
| Data | (69623,41) |
| Training Data | (55699,41) |
| Validation Data | (41,6962) |
| Validation Target Data | (6962,1) |
| Covarience Matrix (Big Sigma) | (41,41) |
| Mu Matrix | (10,41) |
| Phi Matrix | (55699,10) |
| Target Matrix | (55699,1) |
| Erms Matrix | (55699,1) |

I have tuned the values of hyperparameters in order to find the most optimal value for these hyperparameters, so as to get the most accurate results.

**Changing the Hyper-parameters to get the most optimum values for best accuracy**

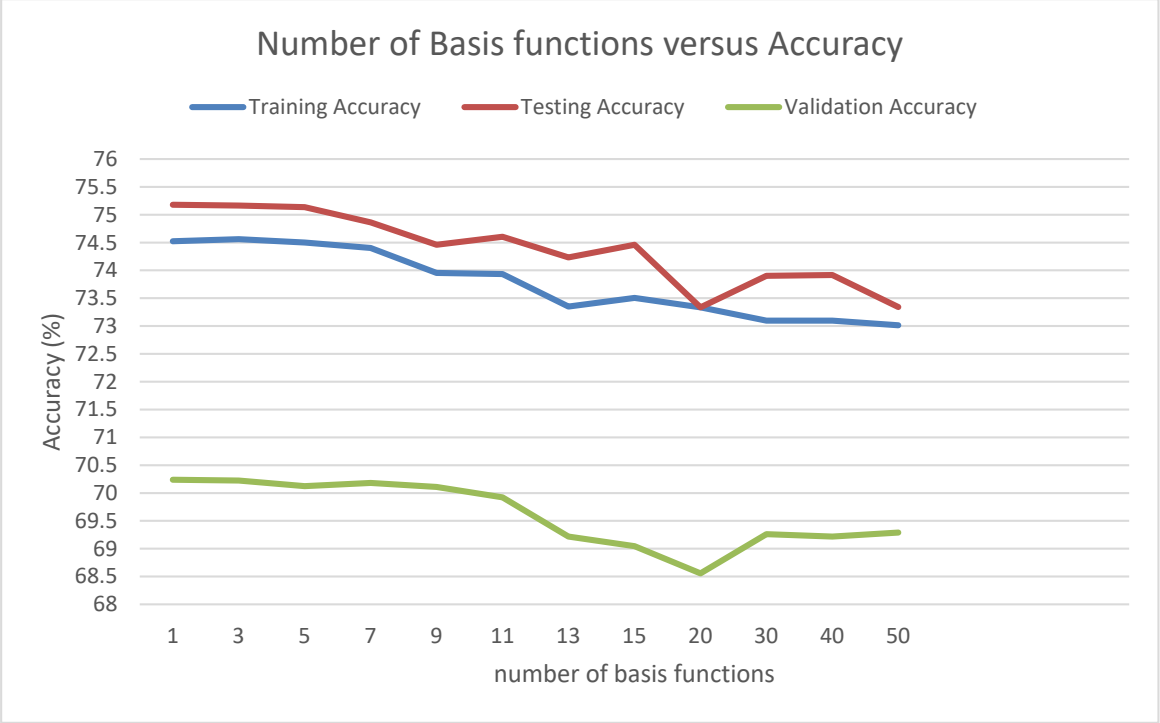
The variables which determine the network structure along with the variables which determine how the network training is done are called hyper-parameters.

Examples: regularizer

**Changing the number of basis function, keeping the regularizer cofficient constant and analyzing its effect on accuracy and Erms:**

The table below shows different values for Accuracy by keeping $\lambda$=0.03 and changing the number of basis function

| Number of basis function (M) | Training Accuracy | Validation Accuracy | Testing Accuracy |
|---|---|---|---|
| 1 | 74.52198423670085 | 75.17954610744039 | 70.23843723068084 |
| 3 | 74.55968688845401 | 75.16518241884516 | 70.2240735420856 |
| 5 | 74.50043986427045 | 75.1364550416547 | 70.12352772191899 |
| 7 | 74.40349018833372 | 74.8635449583453 | 70.18098247629992 |
| 9 | 73.95464909603405 | 74.46136167767882 | 70.10916403332375 |
| 11 | 73.93310472360365 | 74.60499856363114 | 69.92243608158576 |
| 13 | 73.34961130361407 | 74.23154266015513 | 69.21861534041942 |
| 15 | 73.50580800373436 | 74.46136167767882 | 69.04625107727665 |
| 20 | 73.33524838866047 | 73.33524838866047 | 68.55788566503878 |
| 30 | 73.09826029192625 | 73.9011778224648 | 69.2617064062051 |
| 40 | 73.09646492755705 | 73.91554151106004 | 69.21861534041942 |
| 50 | 73.0138781665739 | 73.3409939672508 | 69.29043378339557 |
| 100 | 73.11800929998743 | 73.51335823039356 | 68.85952312553864 |

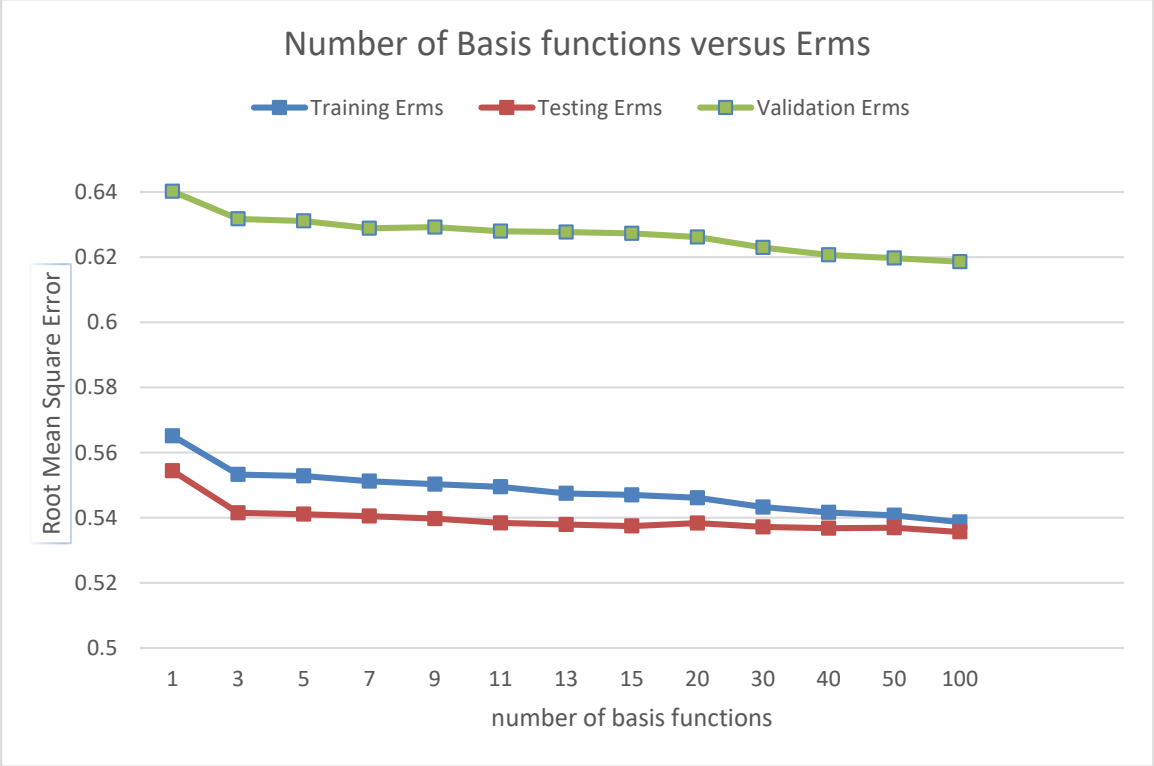



It can be clearly seen in the graph above that as we increase the number of basis function, the accuracy of the model decreases.

The table below shows different values for Erms by keeping $\lambda$=0.03 and changing the number of basis function:

| Number of basis function (M) | Training Erms | Validation Erms | Testing Erms |
|---|---|---|---|
| 1 | 0.565137868842015 | 0.5543879020301472 | 0.640197043941923 |
| 3 | 0.5532657321656597 | 0.5414959724453651 | 0.6317551102769534 |
| 5 | 0.5528324950745139 | 0.541076887367104 | 0.6311034211888344 |
| 7 | 0.551201819357068 | 0.5404920143550035 | 0.6288723788814049 |
| 9 | 0.5502989201579978 | 0.5397327164878437 | 0.6292082884082184 |
| 11 | 0.5494771324161077 | 0.5384049388896263 | 0.6279580380267824 |
| 13 | 0.5474660225183823 | 0.5379079208130623 | 0.6276971401283503 |
| 15 | 0.546993051788288 | 0.537447938693596 | 0.6273104029684066 |
| 20 | 0.5461233210909084 | 0.538337426902244 | 0.6261722176363763 |
| 30 | 0.5432653244902734 | 0.5371612531927624 | 0.6229395783688194 |
| 40 | 0.5416041034111791 | 0.5367791494791946 | 0.6206950526697718 |
| 50 | 0.5407355725302192 | 0.5369173812186422 | 0.6197019977884616 |
| 100 | 0.5386691233396258 | 0.5356313787746063 | 0.6185971503264592 |

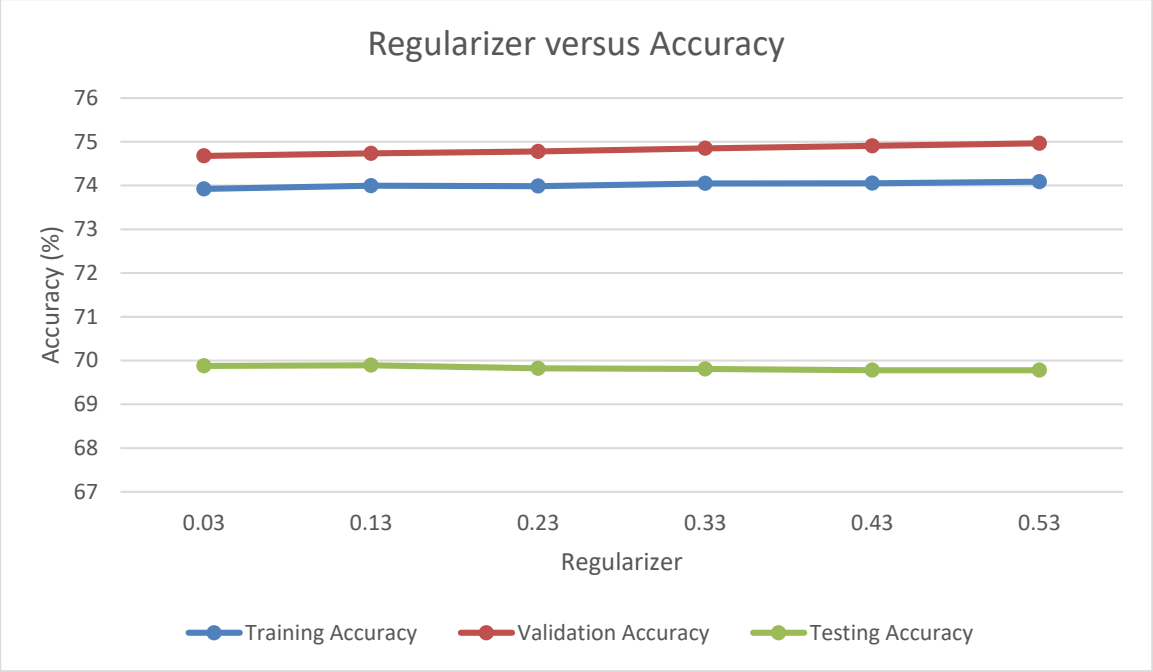
The data shown in the table above is plotted in the graph below:




It can be clearly seen in the graph above that as we increase the number of basis function, the root mean square error of the model decreases.

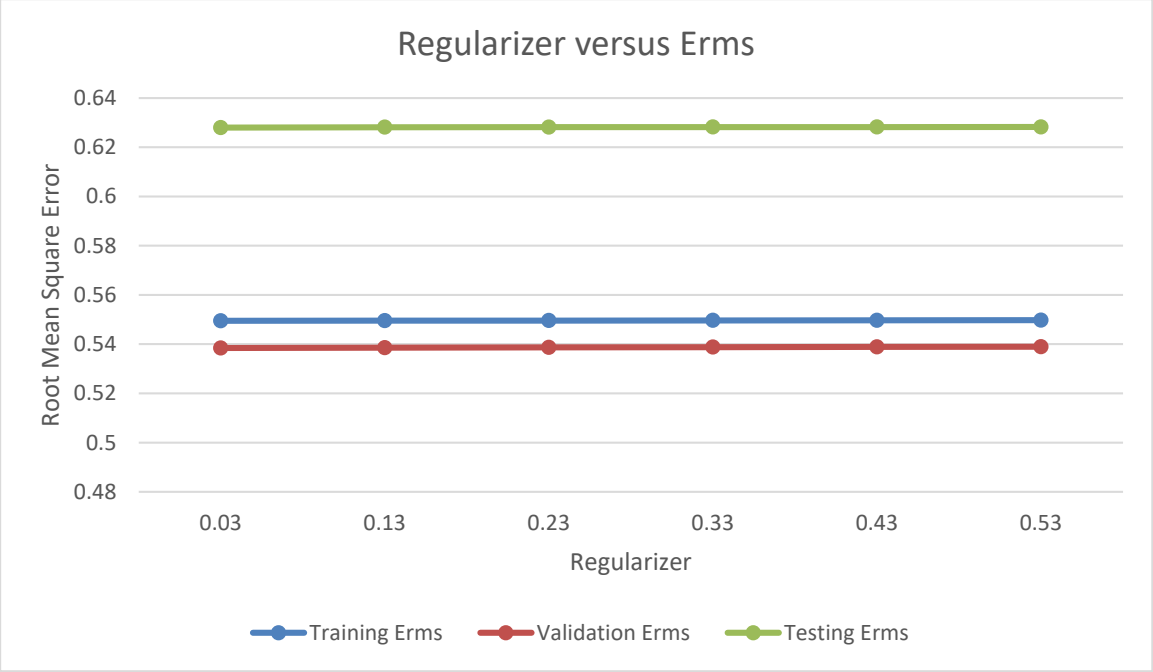
**Effect of changing the regularizer on Accuracy:**

| Regularizer | Training Accuracy | Validation Accuracy | Testing Accuracy |
|---|---|---|---|
| **0.03** | 73.92233253738846 | 74.6768170066073 | 69.87934501580006 |
| **0.13** | 73.99594247652561 | 74.73427176098822 | 69.8937087043953 |
| **0.23** | 73.98696565467962 | 74.77736282677391 | 69.82189026141913 |
| **0.33** | 74.04800804323237 | 74.84918126975008 | 69.8075265728239 |
| **0.43** | 74.05159877197077 | 74.906636024131 | 69.77879919563344 |
| **0.53** | 74.08750605935475 | 74.96409077851192 | 69.77879919563344 |




**Effect of changing the regularizer on Erms:**

| Regularizer | Training Erms | Validation Erms | Testing Erms |
|---|---|---|---|
| **0.03** | 0.5494694067137841 | 0.5384115249839229 | 0.6279559183688018 |
| **0.13** | 0.5495309173206506 | 0.5385396075248634 | 0.6281205974432619 |
| **0.23** | 0.5495837727485522 | 0.5386680276164826 | 0.6281724357625742 |
| **0.33** | 0.5496327695618876 | 0.5387787787466136 | 0.6281949354159674 |
| **0.43** | 0.5496787768415273 | 0.5388740973942274 | 0.6282050466735672 |
| **0.53** | 0.549722098417954 | 0.5389570258227554 | 0.62820888706331 |




**Stochastic Gradient Decent Solution:**

Stochastic Gradient Decent is defined as an iterative method for finding weights to solve a particular problem (LeToR in this case). The stochastic gradient decent solution first takes a random initial value of weights and then iteratively update the weights. This model goes in the opposite direction of the gradient of error, that is why it is called gradient decent. This model updates the weights using the equation:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)}$$

Where ,

Because of the linearity of differentiation we have:

$$\nabla E = \nabla E_D + \lambda \nabla E_W$$

Here,

$$\nabla E_D = -(t_n - \mathbf{w}^{(\tau)\top}\phi(\mathbf{x}_n))\phi(\mathbf{x}_n)$$
$$\nabla E_W = \mathbf{w}^{(\tau)}$$

Finally, we calculate the Erms value, which is given by the formula, as shown . The root mena square error (Erms) is used to measure the differences between predicted values by a model and the sample values.

$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^*)/N_{\mathrm{V}}}$$

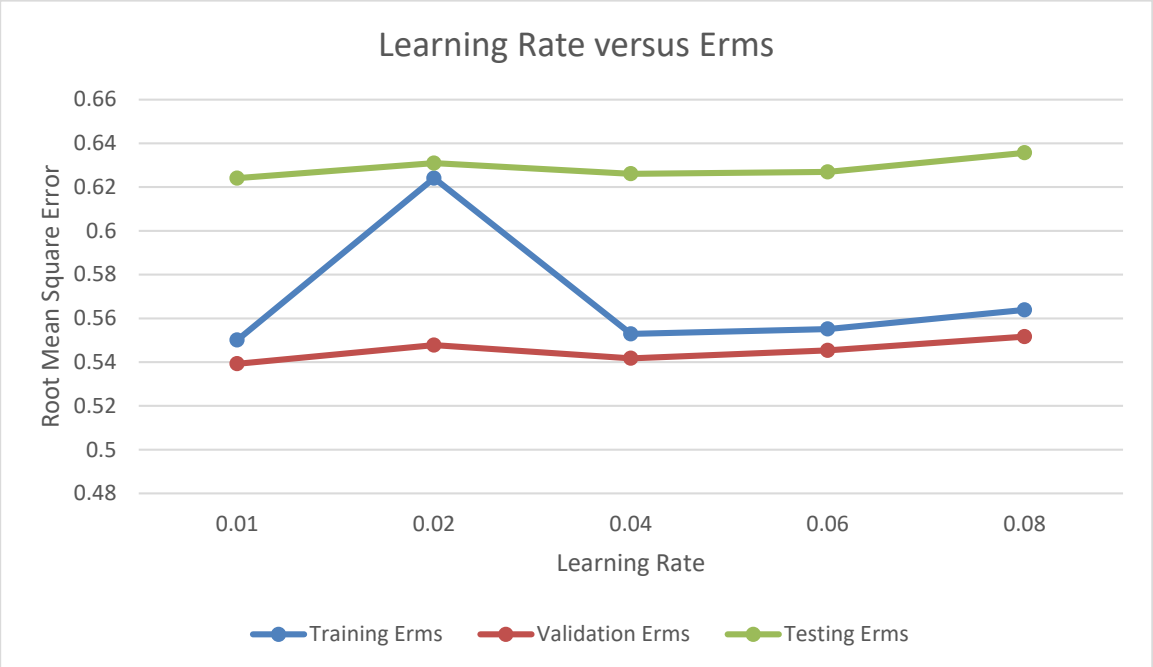
**What is the difference between Stochastic Gradient Decent and simple Gradient Decent?**

The simple gradient decent solution might find a local minimum rather then a global minimum, while the stochastic gradient decent is more efficient for finding the global minimum.

**Changing the learning rate, keeping the regularizer cofficient constant and analyzing its effect on Erms:**

As the name suggests, Learning Rate defines, how quickly a model updates its parameters.

| Learning Rate | Training Erms | Validation Erms | Testing Erms |
|---|---|---|---|
| 0.01 | 0.55009 | 0.53925 | 0.62406 |
| 0.02 | 0.62406 | 0.54776 | 0.63095 |
| 0.04 | 0.55291 | 0.54172 | 0.62607 |
| 0.06 | 0.55513 | 0.54536 | 0.62694 |
| 0.08 | 0.56386 | 0.55164 | 0.63567 |

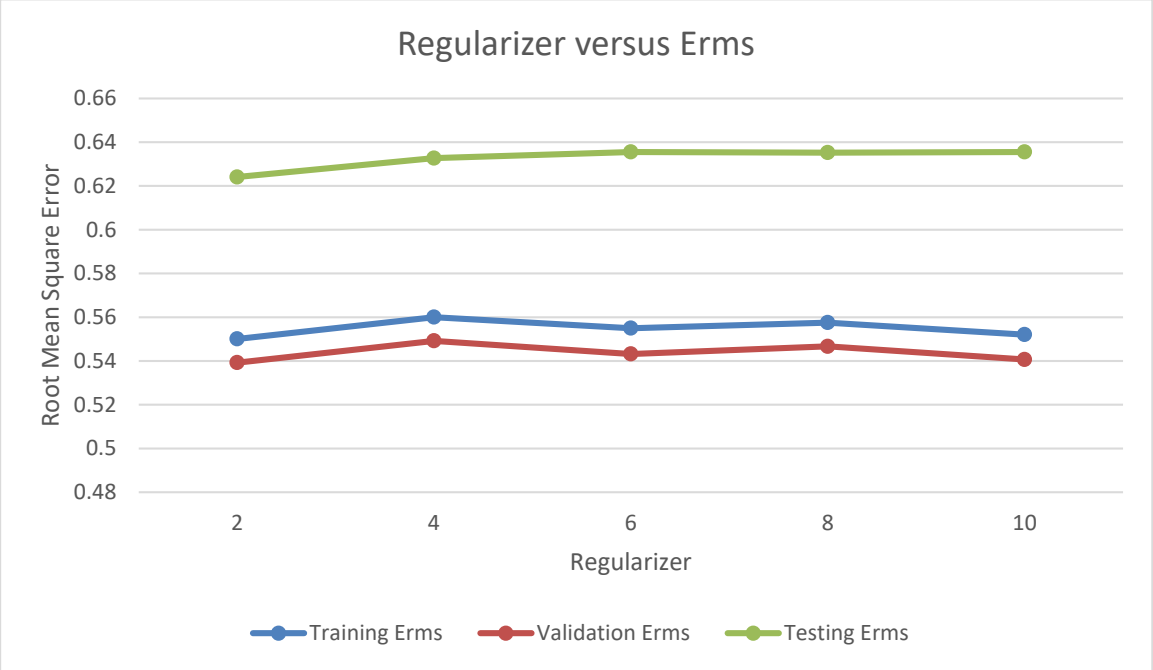


It can be clearly seen in the graph above that as we increase learning rate, the root mean square error of the model increases.

**Changing the regularizer cofficient by keeping learning rate constant and analyzing its effect on Erms:**

| Regularizer | Training Erms | Validation Erms | Testing Erms |
|---|---|---|---|
| 2 | 0.55009 | 0.53925 | 0.62406 |
| 4 | 0.56001 | 0.5492 | 0.63269 |
| 6 | 0.55496 | 0.5432 | 0.63551 |
| 8 | 0.55755 | 0.54668 | 0.63522 |
| 10 | 0.55206 | 0.54066 | 0.63553 |

The data shown in the table above is plotter in the graph below:




**Selection of the most optimal Hyperparameters:**

As presented in the above paragraphs and the graphs, the following are the most optimal values for the hyperparameters.

For Closed Form Solution:

Lambda = 0.03

No. of Basis Function (M) = 10

For Stochastic Gradient Decent Solution:

Lambda = 2

Learning Rate = 0.01

**Conclusion**

Disadvantages of closed form solution:

- The closed form solution does not work on super large data sets.

Advantages of Stochastic Gradient Decent:

- Stochastic Gradient Decent is quite effective and simple on machine learning tasks.
- Stochastic gradient decent is very scalable.

Disadvantages of Stochastic Gradient Decent:

- This stochastic gradient decent model only applies to differentiable (smooth) functions.

**References**


- https://www.cs.cmu.edu/~mgormley/courses/10701-f16/slides/lecture4.pdf
- https://www.cs.cmu.edu/~mgormley/courses/10601-s17/slides/lecture7-opt.pdf
- www.fon.hum.uva.nl/praat/manual/epoch.html
- https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a