Project 1.2 – LeToR using Linear Regression

Linear Regression

The parameters required to predict the target variable are being obtained using 2 methods:

1. Closed-Form Solution
2. Stochastic Gradient Descent (SGD)

1. Closed-Form Solution

$t = w^T \emptyset(x)$

$w = (\lambda I + \emptyset^T \emptyset)^{-1} \emptyset^T t$

Below, we will see the effect of different parameters on these weights and Root Mean Square Error.

Effect of Regularizer

To obtain better generalization and avoid over fitting, a regularization term is added to the error function. Encourages weight values toward 0 (but not exactly 0).

As can be seen from the graph and table below, as λ increases the E_{rms} increases. The reason is as λ increases the effect of weights decreases and slowly tends to zero with higher values and so the fitting decreases.

The values of other parameters: M = 10; n = 200

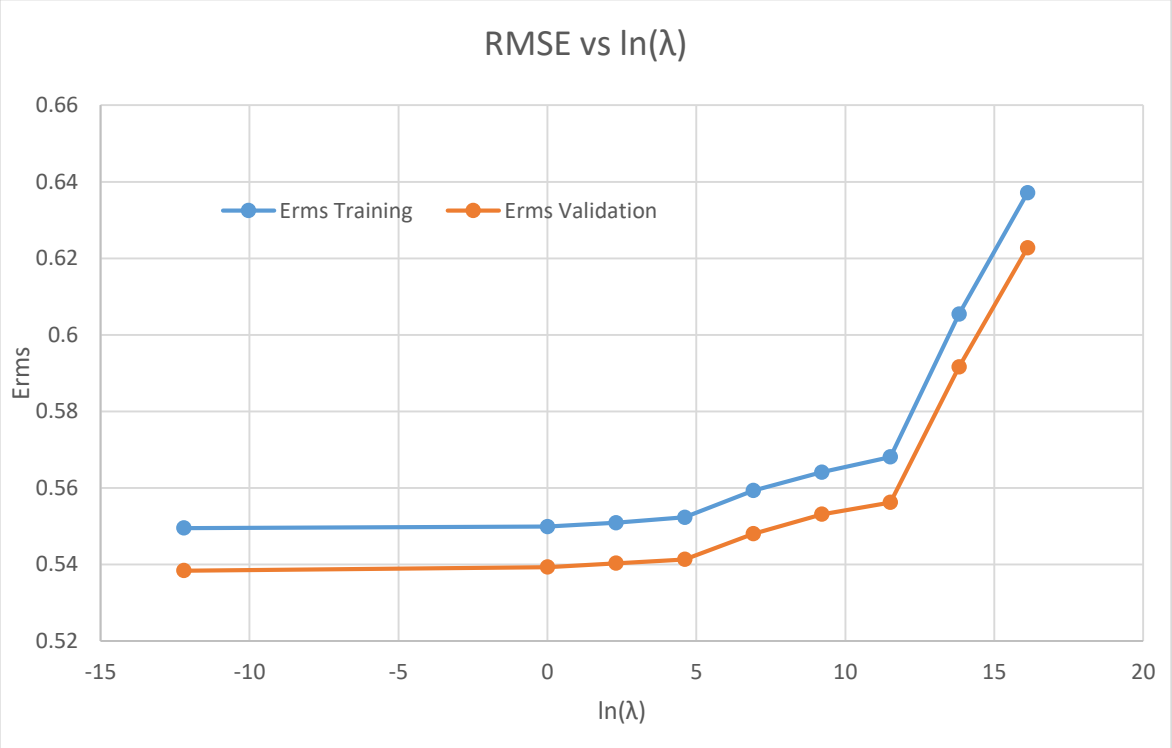

Fig 1: Root Mean Square Error for different values of λ

ln(λ)	Erms Training	Erms Validation	Erms Testing
-12.20607265	0.5495	0.5384	0.6278
0	0.5499	0.5393	0.6282
2.302585093	0.5509	0.5403	0.6283
4.605170186	0.5523	0.5413	0.6293
6.907755279	0.5593	0.548	0.6353
9.210340372	0.5641	0.5531	0.6401
11.51292546	0.5681	0.5562	0.6497
13.81551056	0.6054	0.5916	0.6994
16.11809565	0.6371	0.6227	0.7348

Table 1: Vales of Root Mean Square Error for different values of λ

Weights when λ=0 → [-9.86234965, 4.49028386, -0.37002739, -6.49858044, 0.41628716, 2.22489921, 0.70418777, 6.73699993, 2.11683928, 0.60969069]

Weights when $\lambda=10^6$ →[0.01046994, 0.01074463, 0.01026972, 0.01086795, 0.01033156, 0.01065768, 0.00960117, 0.01070166, 0.00994311, 0.01016279]
As we can see, the values of weights are approaching zero.

Effect of Number of Clusters (M)

The number of clusters are determined by us and according to the M value the k-means creates M number of clusters. We can see from the graph below that as the number of clusters increases we are essentially nearing all the features of the original data and the E_{rms} decreases as the number of clusters increases i.e., the data will be fit more properly. The increase in number of clusters results in increase of number of features.

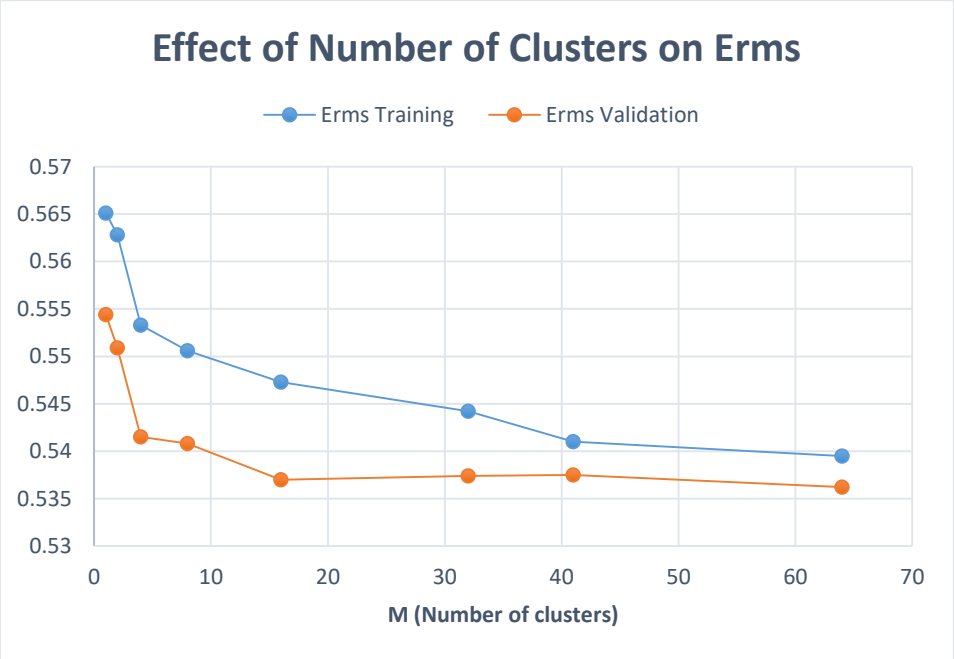

Fig 2: Effect of Number of Clusters (M) on Root Mean Square Error

M	Erms Training	Erms Validation	Erms Testing
1	0.5651	0.5544	0.6402
2	0.5628	0.5509	0.6384
4	0.5533	0.5415	0.6317
8	0.5506	0.5408	0.6292
16	0.5473	0.537	0.6272
32	0.5442	0.5374	0.6235
41	0.541	0.5375	0.6203
64	0.5395	0.5362	0.6194

Table 2: Root Mean Square Error values for different values of M

Effect of Scaling Factor of BigSigma

The variance values we get for the features are very small and as a result when these values are used to obtain the ϕ(x) and Weight parameters, they turn out be nearly similar and the output becomes uniform. In order to avoid this, we scale the variances by multiplying it with larger values {np.dot(n, BigSigma), where 'n' is the multiplication factor}

As we can see in the graph below, with increase in the multiplication factor the E_{rms} decreases for both Training and Validation data because with increase in these values the weights increase and the model will be able to predict the target variable more accurately and will be a better fit.

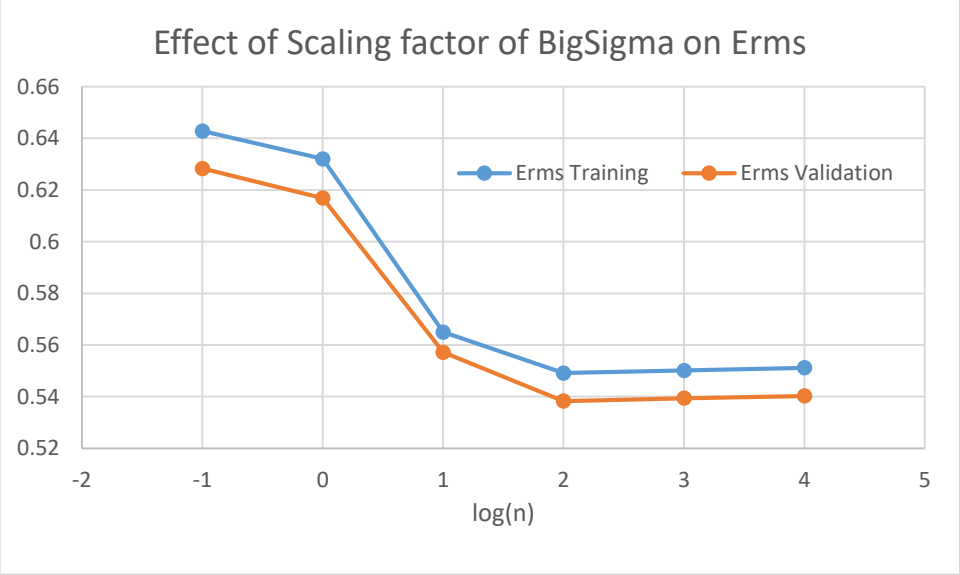

Fig 3: Effect of scaling factor (n) of BigSigma on Root Mean Square Error

n*BigSigma	log(n)	Erms Training	Erms Validation	Erms Testing
0.1	-1	0.6428	0.6283	0.7409
1	0	0.632	0.6168	0.7305
10	1	0.565	0.5572	0.6414
100	2	0.5491	0.5383	0.6271
1000	3	0.5501	0.5394	0.6288
10000	4	0.5512	0.5403	0.629

Table 3: Root Mean Square Error values for different values of n (scaling factor)

2. Stochastic Gradient Descent (SGD)

The stochastic gradient descent algorithm first takes a random initial value $w^{(0)}$ then it updates the value. This is done by taking the derivative (tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter 'learning rate'. Effect of different parameters is explained below.

Effect of Learning Rate

Initially when the learning rate is very low the function is not reaching its minimum within the 256 iterations and so the E_{rms} is very high. Same is the case when we compare the Erms for learning rates of 0.001 and 0.01. For 0.001 the E_{rms} is high when compared to 0.01. From 0.01, as the learning rate increases the Erms also increases. With higher values of learning rate, the function converges to minimum quickly but at the same time it might overshoot the minimum and diverge resulting in never reaching the minimum value. For this reason, when learning rate is 0.2, the E_{rms} shoots up to 87.33 as shown below.

In order to show the effect of learning rate, I fixed λ (La) = 10, x1=0, xn=256, M=10 and n = 200

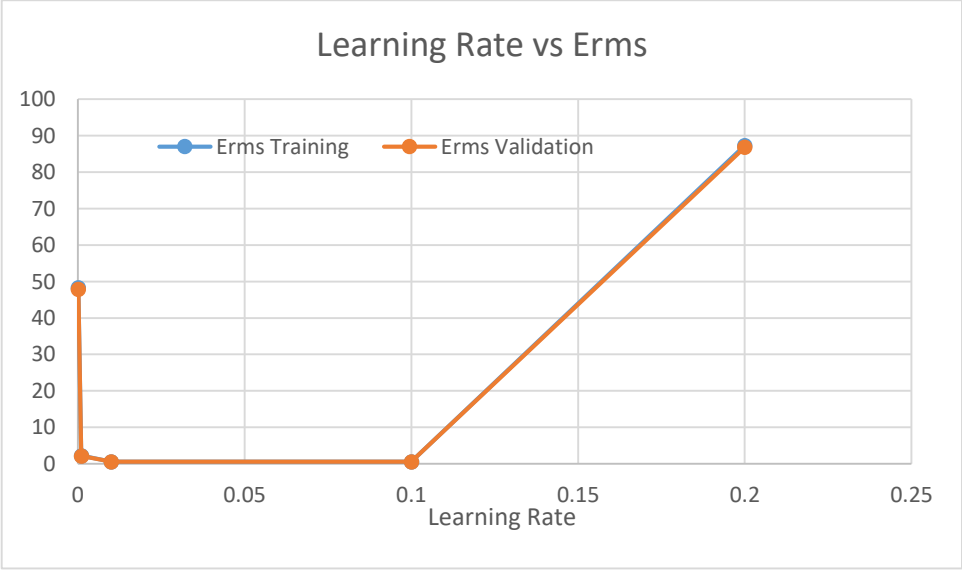

Fig 5: Effect of Learning rate on Root Mean Square error

Learning Rate	Erms Training	Erms Validation	Erms Testing
0.0001	48.30687	47.84882	48.19055
0.001	2.2068	2.1378	2.19842
0.01	0.55213	0.54017	0.63556
0.1	0.5649	0.55396	0.6356
0.2	87.32226	86.82268	87.33852

Table 5: Root Mean Square Error values for different values of Learning rate

Effect of Regularizer

As we can see from the graphs below, when as the λ increases the E_{rms} also increases but due to the data at hand, the E_{rms} for lower values of λ is high.

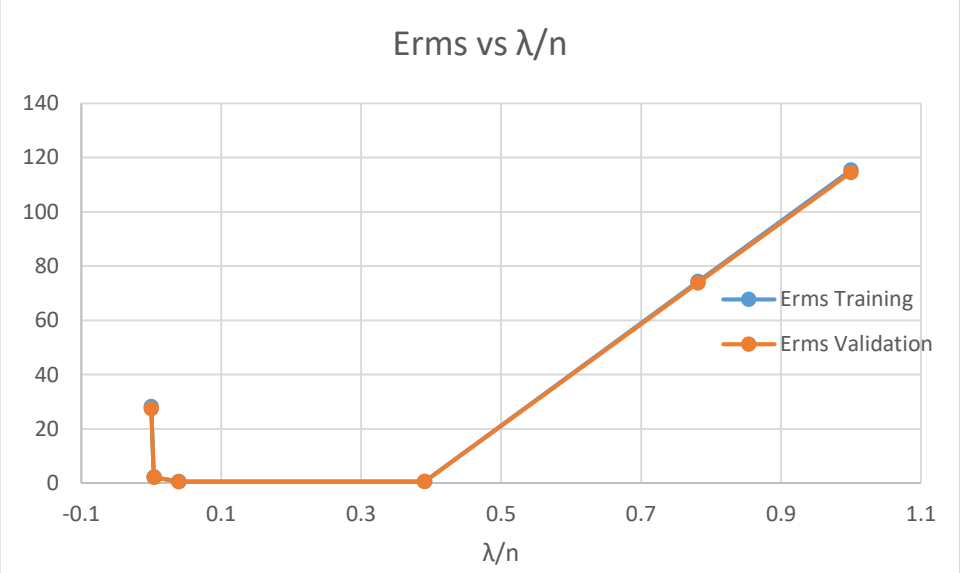

Fig 4: Effect of λ/n on Root mean square error

λ	ln(λ)	λ/n	Erms Training	Erms Validation	Erms Testing
0	0	0	28.22539	27.32931	27.96208
1	0	0.00390625	2.19797	2.12854	2.18931
10	2.302585	0.0390625	0.55213	0.54017	0.63556
100	4.60517	0.390625	0.58549	0.5723	0.67527
200	5.298317	0.78125	74.32758	73.7743	74.33497
256	5.545177	1	115.38454	114.5251	115.35865

Table 4: Root Mean Square Error values for different values of λ

Effect of Number of Clusters

As we can see from the graph and table below, as the number of clusters increase the Erms of training reduces because an increase in M results in an increase in number of features. The E_{rms} of Validation also follows the same trend and remains nearly constant at higher values of M.

Exception: The increase in Erms from 2 clusters to 4 clusters may be due to the number of samples that I have considered which is only 128. If I consider more number of samples the E_{rms} is decreasing continually as expected but for simplicity, I have gone with 128 samples.

To achieve the plot below, the other parameters are set to the following values: λ=10; x1 = 0; xn = 128; n = 200; learning rate = 0.01

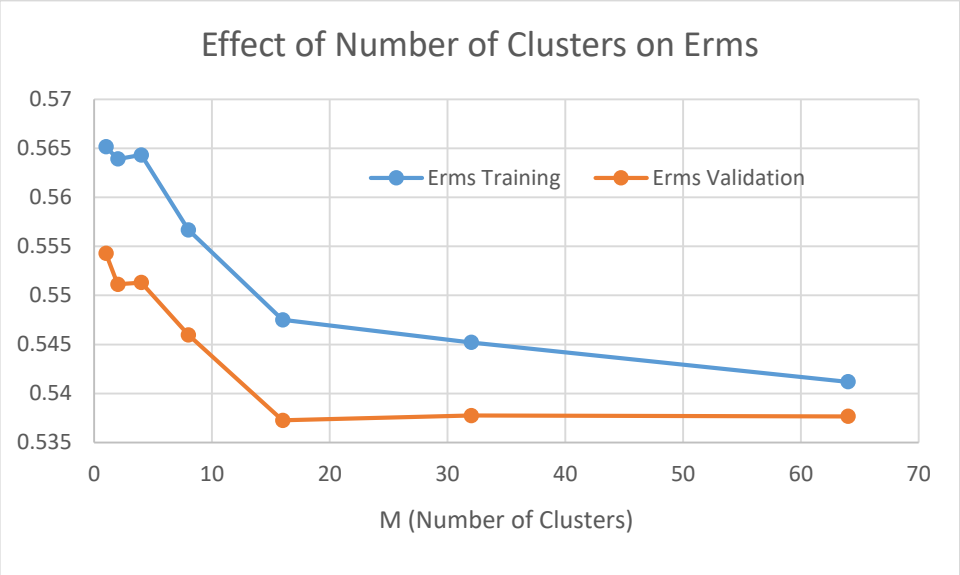

Fig 6: Effect of number of clusters (M) on Root Mean Square error

M	Erms Training	Erms Validation	Erms Testing
1	0.56517	0.5543	0.63592
2	0.56392	0.55113	0.63793
4	0.56432	0.55131	0.65096
8	0.55669	0.54598	0.64424
16	0.5475	0.53726	0.62426
32	0.5452	0.53774	0.6211
64	0.54119	0.53766	0.61844

Table 6: Root Mean Square Error values for different values of M

Effect of Scaling Factor of BigSigma

As we can see in the graph below, with increase in the multiplication factor the Erms decreases for both Training and Validation data because with increase in these values the weights increase and the model will be able to predict the target variable more accurately.

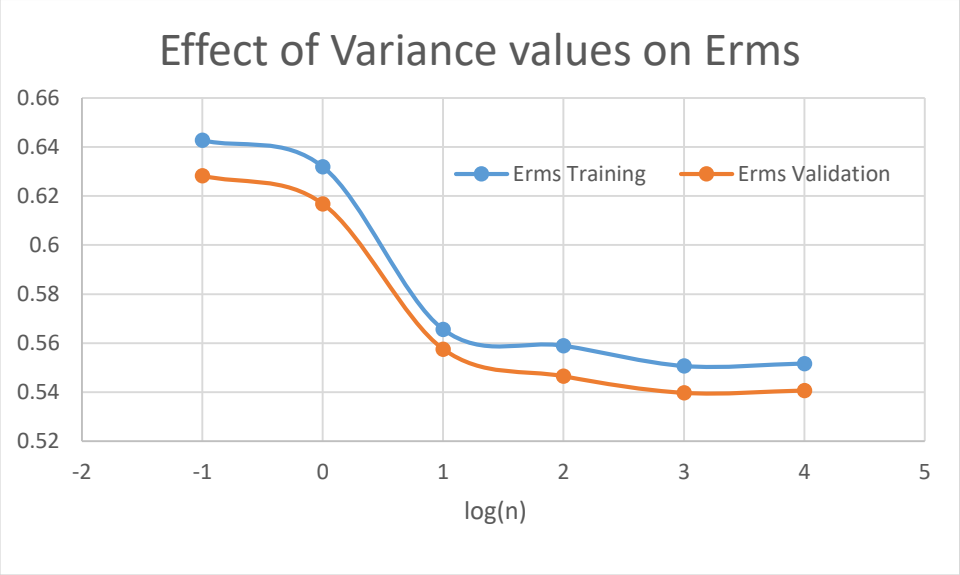

Fig 7: Effect of scaling factor (n) of BigSigma on Root Mean Square Error

n*BigSigma	log(n)	Erms Training	Erms Validation	Erms Testing
0.1	-1	0.64275	0.62826	0.74089
1	0	0.63195	0.61676	0.73049
10	1	0.56557	0.55753	0.63747
100	2	0.55892	0.54646	0.64765
1000	3	0.55063	0.53969	0.62732
10000	4	0.55165	0.54064	0.62567

Table 7: Root Mean Square Error values for different values of n (scaling factor)

Effect of Number of Samples

As it can be seen from the graph below, with increase in number of samples the Erms reduces. For less number of samples, the convergence of the function to the minimum doesn't occur and so the Erms will be high for less number of samples. For 128 samples with learning rate of 0.01 the model has already reached its minimum and so even after increase in the number of the samples there is no improvement in the Erms.

The values of other parameters: λ=10; M = 10; learning rate = 0.01; n = 200

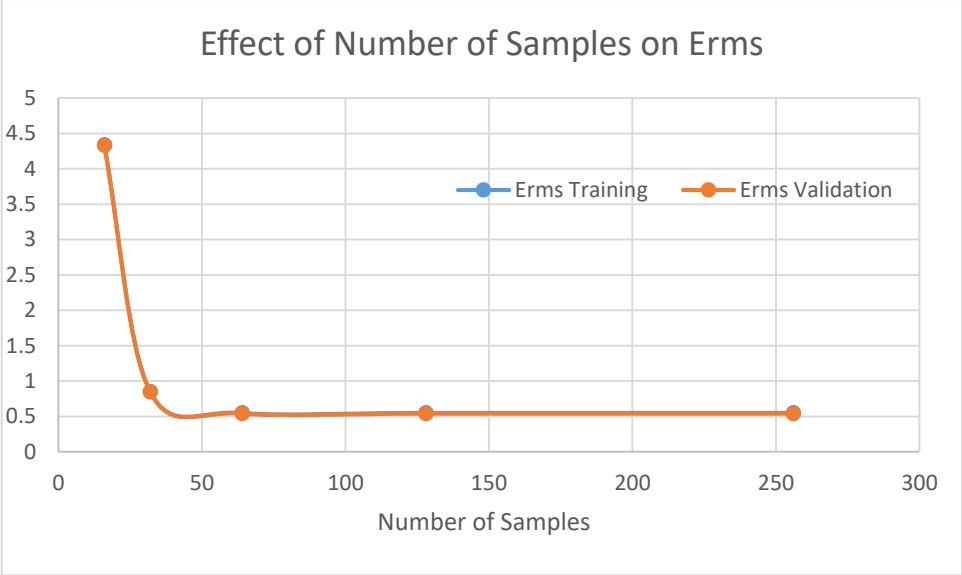

Fig 8: Effect of number of samples on Root Mean Square error

No. of samples	Erms Training	Erms Validation	Erms Testing
16	4.33783	4.32874	4.32536
32	0.85167	0.84606	0.90563
64	0.55165	0.54064	0.62567
128	0.55165	0.54064	0.62567
256	0.55165	0.54064	0.62567

Table 8: Root Mean Square Error values for different values of Number of Samples

Final Parameter Settings

Closed Form parameters:

λ = 0.000005 (as close to 0 as possible to reduce Erms)

M = 10

n = 200 (scaling factor for BigSigma)

Stochastic Gradient Descent(SGD) parameters:

λ=10

Learning Rate = 0.01

M = 10

n = 200

Questions and Answers

Why are we splitting data into Training, Validation and Testing?

We want to check the performance of our model on data which it has not seen (i.e., to predict values of unseen data) but we generally don't have access to future data so we try to simulate this by splitting the data and regarding the testing data as future data.

Why use Moore-Penrose Pseudo Inverse?

In general, for a matrix A to have an inverse, it should be square and it must be non-singular. But the design matrix which we have is not square. So in these type of cases we use Pseudoinverse which is of the form $(A^T A)^{-1} A^T$. The problem of A being a non-square is solved by the term $A^T A$. The problem of singular is also solved, as A is not coming to denominator anymore.