Anurag Chaudhury

Anurag Chaudhury

997298160

Assignment 3

Logistic Regression

Observations

Number of principal components	Minus LL (average)-Validation set	%age error (validation)
10	0.1908964	7.3 %
20	0.1663364	6.67 %
40	0.1664554	6 %

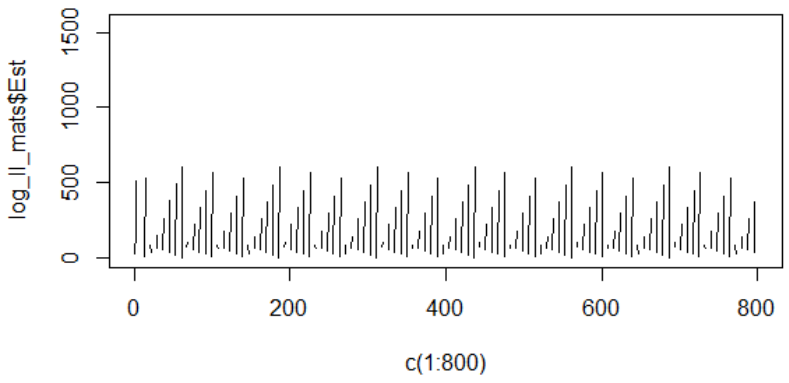
We can see that as increase the number of principal components, the error rate decreases. For K = 50 for instance, the error rate is 4.6 %. However, if we keep increasing the number of principal components, to say 400 we see that the error rate increases to 11.3 % and the minus log likelihood becomes 2.714. This is probably because of “noise” introduced as the additional covariates are not adding any useful information to the model.

Neural Networks

Estimation of learning rate – To estimate the learning rate, I tried fixing the number of principal components and number of iterations as 20 and 800 and then tried different learning rates, to see which caused the minus log likelihood to decrease steadily.

Number of dimensions used in this plot = 20

For a learning rate of 0.5, we see a large number of oscillations in the training estimation as seen below:



Anurag Chaudhury

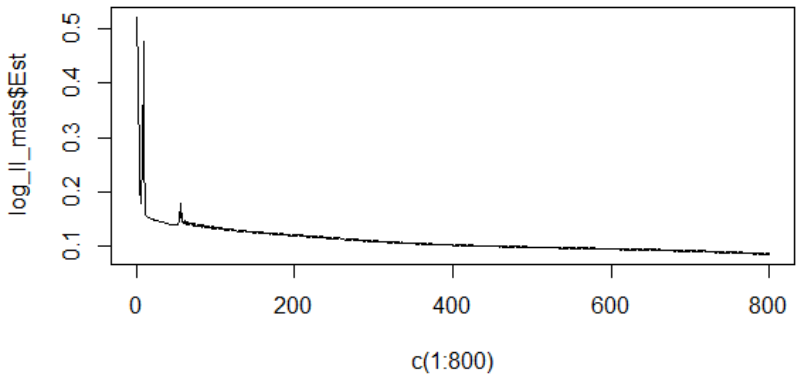


Fig 1 : Minus log likelihood vs number of iterations for training estimation set

Learning rate = 0.01

Num_iterations = 800

Observation : The log likelihood does not decrease steadily for all iterations. It drops and then increases for the estimation set, showing some oscillations later on as well.

For a learning rate of 0.001, we see that the log likelihood decreases steadily throughout all the iterations.

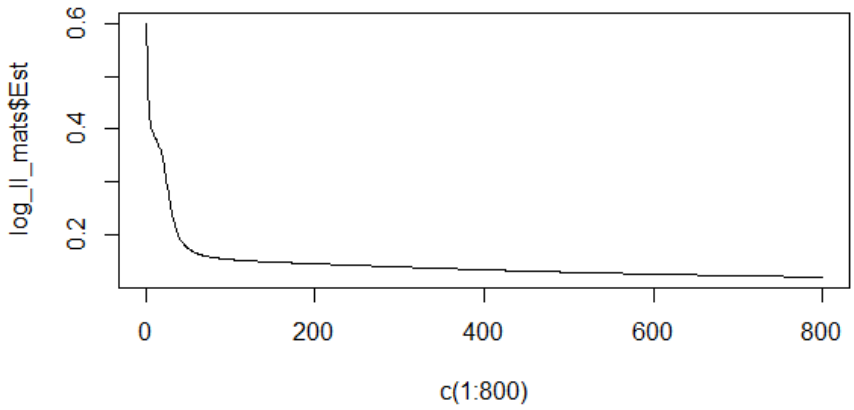


Fig 1 : Minus log likelihood vs number of iterations for training estimation set

```
Minimum minus log likelihood (estimation, average) = 0.11928
Minimum minus log likelihood (validation, average) = 0.1477416
```

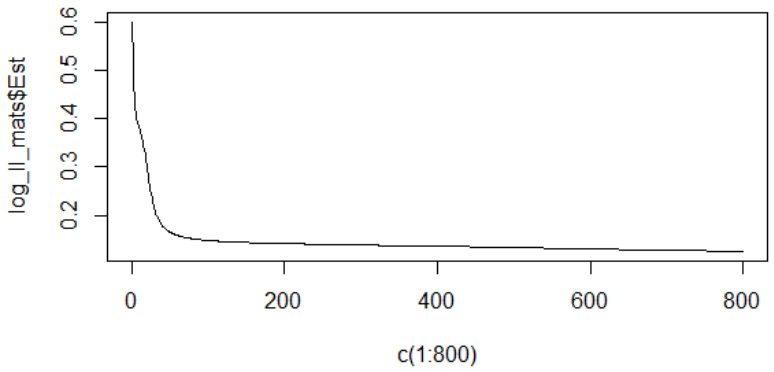


Fig 1 : Learning rate of 0.001 with 800 iterations on 40 component training estimation set

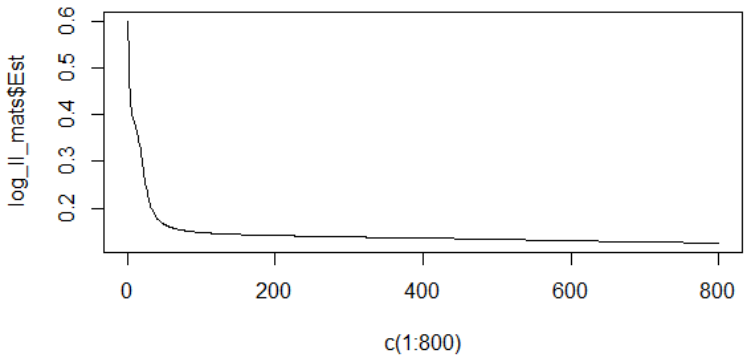


Fig 1 : Learning rate of 0.001 with 800 iterations on 10 component training estimation set

Even if we increase the number of iterations to 5000, the learning rate of 0.001 ensures that no oscillations occur, as seen below:

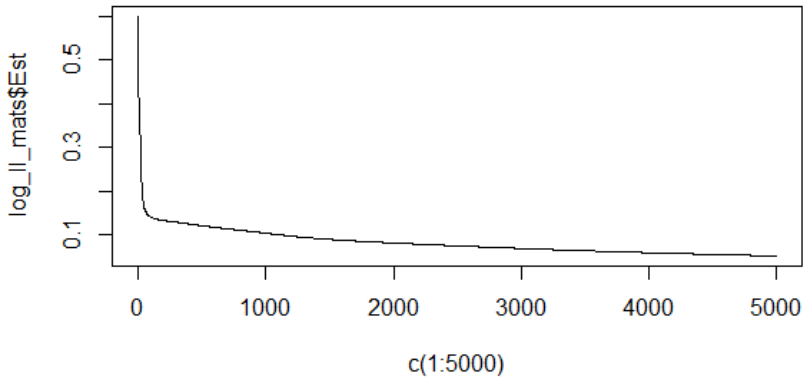


Fig 1 : Learning rate of 0.001 with 5000 iterations on 10 component training estimation set

Anurag Chaudhury

Conclusion: A learning rate of 0.001 allows for a steady decrease in the minus log likelihood.

Analysis of performance on validation set with out penalty

10 principal components

500 iterations & lambda = 0

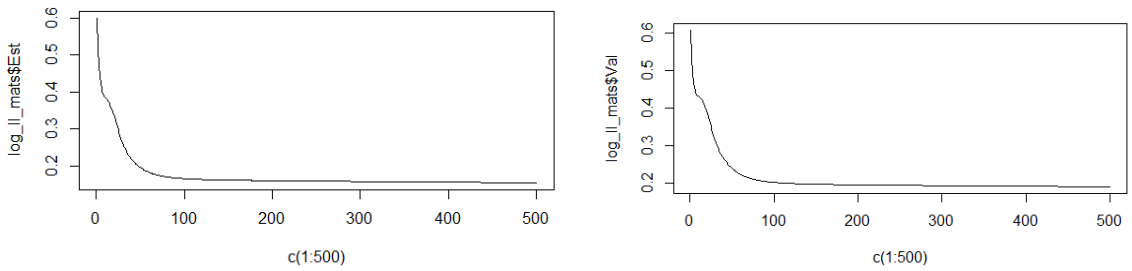


Fig 1 & 2 : Minus log likelihood graphs for the estimation and validation sets respectively (left to right)

For 500 iterations, we see that the validation minus log likelihood is 0.1909 and the error is 7.67%. This might indicate that the network has not stabilized the log likelihood yet and we should run it for more iterations.

5000 iterations & *lambda* = 0

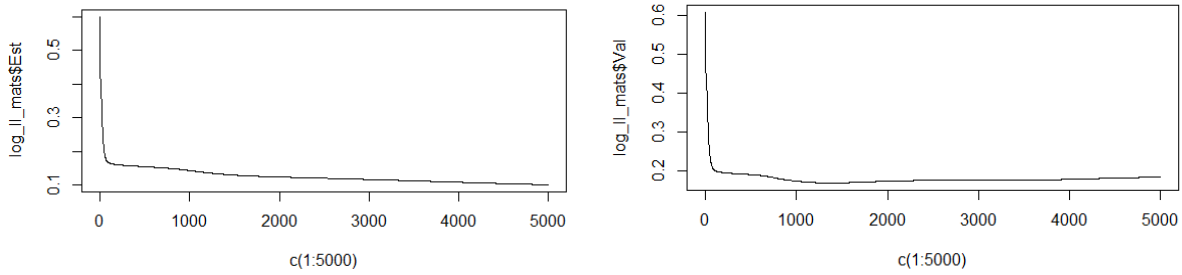


Fig 1 & 2 : Minus log likelihood graphs for the estimation and validation sets respectively (left to right)

In the case of 5000 iterations, the minus log likelihood for the estimation set is still steadily decreasing, however, for the validation set the minus log likelihod has a minimum at 1376th iteration and then starts rising up slowly. This is indicative of overfitting the training set.

Min. estimation minus log likelihood (average) = 0.100

Anurag Chaudhury

10000 iterations & lambda = 0

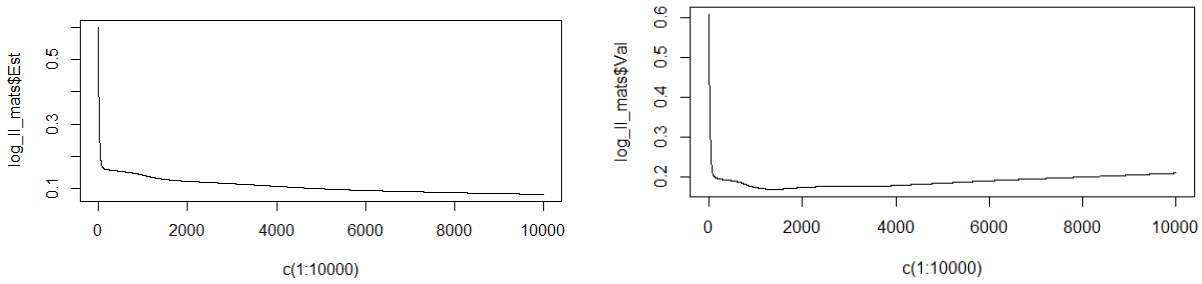


Fig 1 & 2 : Minus log likelihood graphs for the estimation and validation sets respectively (left to right)

Min. estimation minus log likelihood (average) = 0.083

Since, the log likelihood for the estimation set only changed very marginally 0.1 to 0.083 from 5000 to 10000 iterations, there does not seem to be much to be gained by increasing the number of iterations further. The net decrease is 0.0034 for 1000 iterations, which indicates that it has relatively stabilized. Furthermore, the validation minus log likelihood (average) does not change by increasing the number of iterations each time. This makes sense as the network starts overfitting the training estimation set after the same number of iterations.

Taking into consideration that 500 iterations does not show good results for 10 principal components (the minus log likelihood has not stabililzed. This is shown in the summary table at the end of this section), 20 components and 40 components are run with 5000 and 10000 iterations only.

20 principal components

5000 iterations & lambda = 0

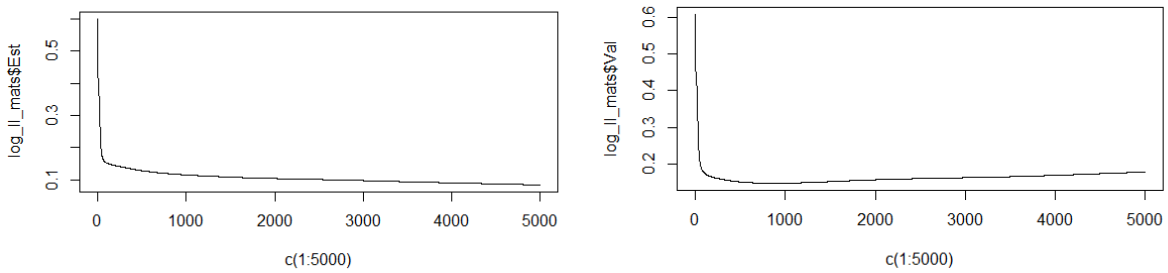


Fig 1 & 2 : Minus log likelihood graphs for the estimation and validation sets respectively (left to right)

10000 iterations & lambda = 0

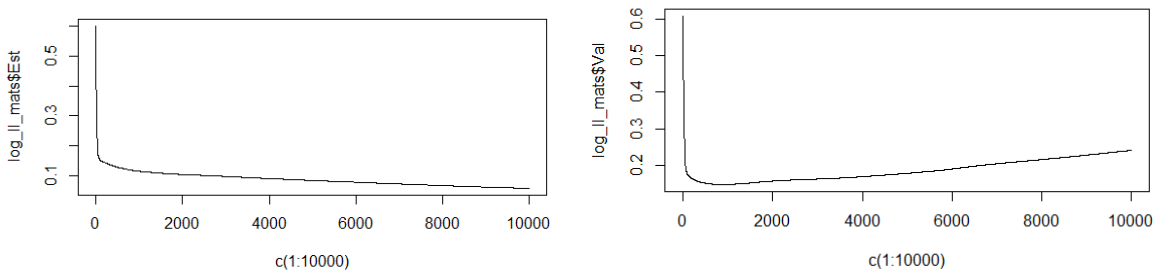


Fig 1 & 2 : Minus log likelihood graphs for the estimation and validation sets respectively (left to right)

40 principal components

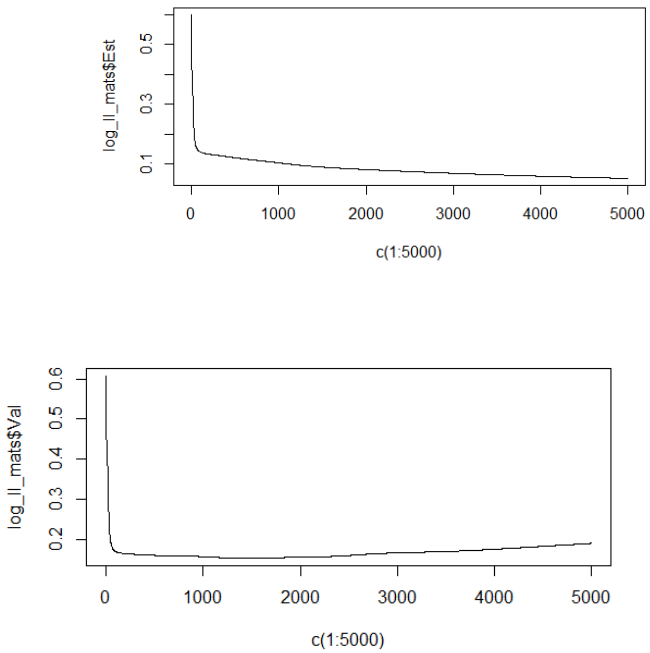


Fig 1 & 2 : Minus log likelihood graphs for the estimation and validation sets respectively (top to bottom)

10000 iterations & lambda = 0

Anurag Chaudhury

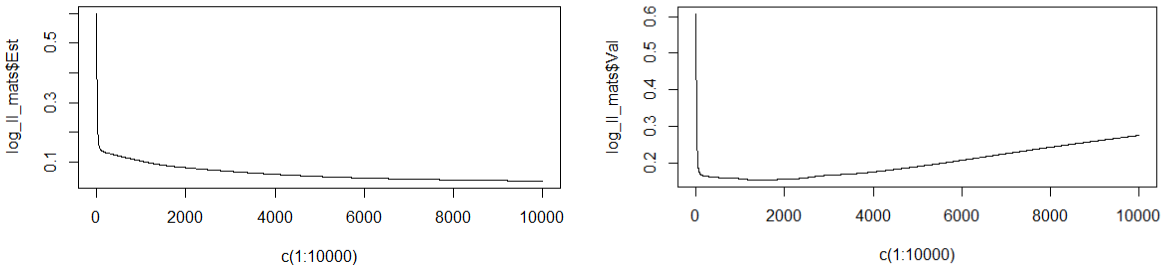


Fig 1 & 2 : Minus log likelihood graphs for the estimation and validation sets respectively (left to right)

Summary of performance on validation set without penalty

Number of PC	Iterations	Validation (-log ll) average	Validation classification error % age	Best index for validation
10	500	0.191	7.67	500
10	5000	0.1694751	6	1376
10	10000	0.1694751	6	1376
20	5000	0.1476413	4.3	874
20	10000	0.1476413	4.3	874
40	5000	0.153169	4	1422
40	10000	0.153169	4	1422

This phenomenon in which the best index for validation occurs before training is over is due to the fact that the network is overfitting the training set after that point. For instance, for 40 components, after 1422 iterations, the net is overfitting the training data and the validation minus log likelihood increases. We could technically stop training as soon as we see that the validation minus log likelihood is increasing – early stopping.

Furthermore, we see that the performance on 40 principal components in terms of minus log likelihood alone, is a bit worse than for 20 principal components. This is perhaps because the additional 20 components do not add any useful information to the model and essentially mislead the computations slightly.

Analysis of performance on validation set with penalty

The observations below compare the performance of different principal components against different penalty magnitudes.

Lambda = {0.001, 0.01, 0.05, 0.1 , 1} were the penalty magnitudes used.

From the above discussion, it is reasonable to choose 10000 iterations.

Also tried lambda = 1000 for 20 components to illustrate underfitting for high penalty scenario.

10 COMPONENTS

Lambda = 0.001

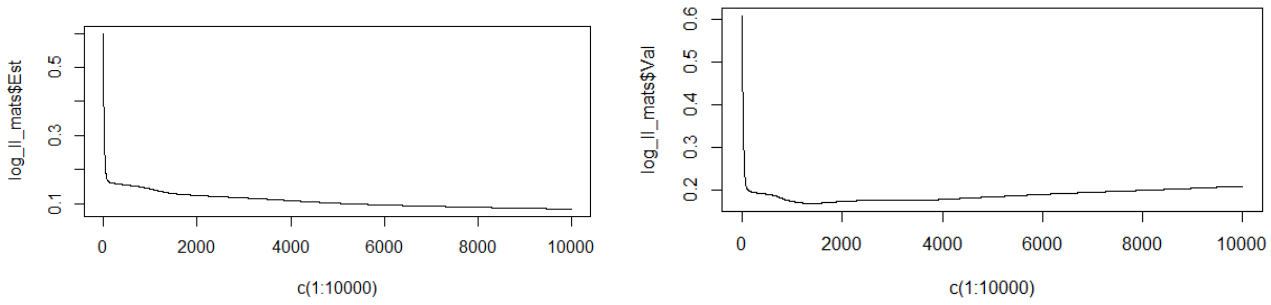


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.001)(left to right)

Lambda = 0.01

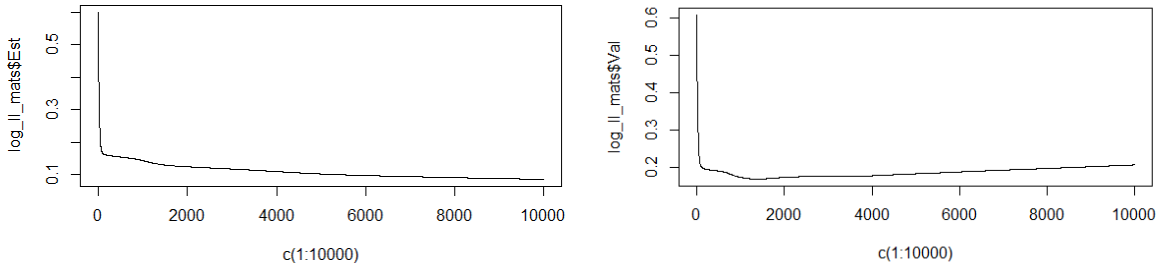


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.01)(left to right)

Lambda = 0.05

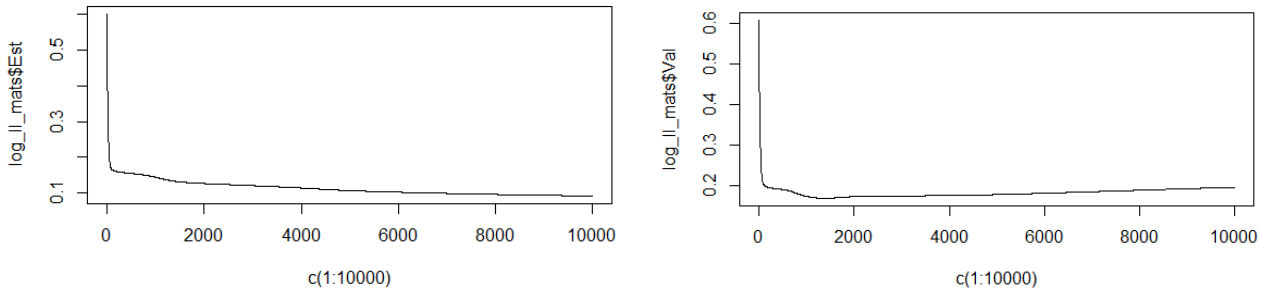


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.05)

Lambda = 0.1

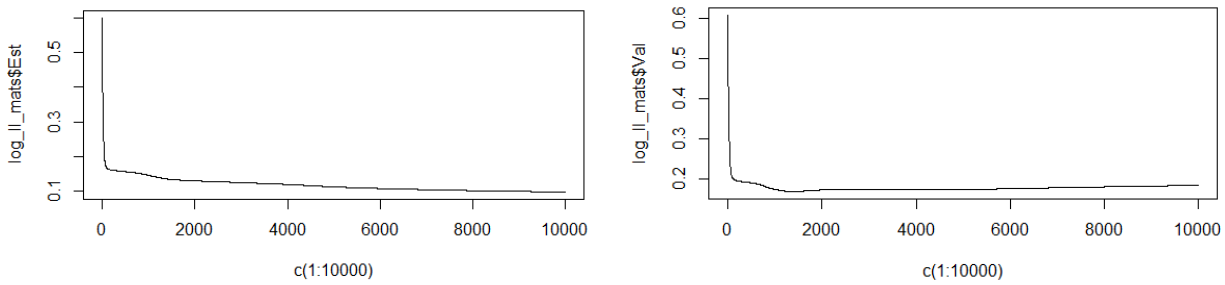


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.1)

Lambda = 1

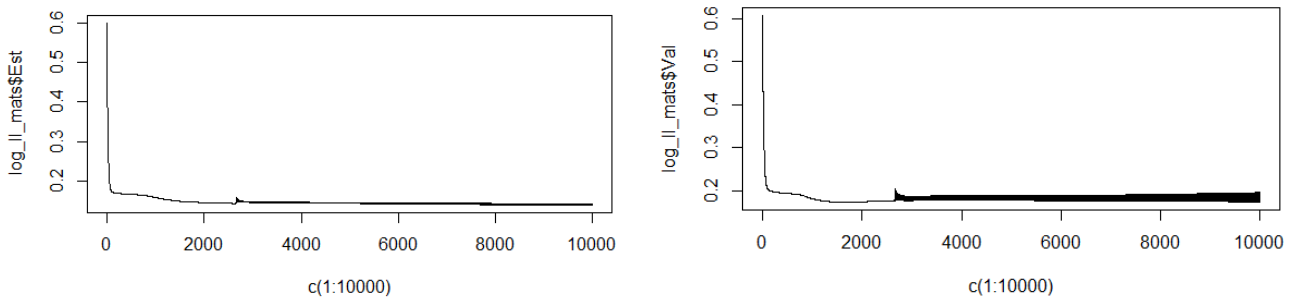


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 1)

Summary - 10 principal components & lambda value comparison

Lambda	Minimum minus log likelihood	Min. validation error %	Best index (min.validation minus log likelihood)
0.001	0.1694745	6	1376
0.01	0.1694877	6	1377
0.05	0.1695651	6	1380

0.1	0.1696933	6	1386
1	0.1732961	6	1560

For any value of lambda, with 10 components we see that the log likelihood only gets worse i.e the minus log likelihood increases. For 1 especially, we see that the value goes up quite a bit and in the graphs corresspondingly we see that there are oscillations in the values as we go to 10000 iterations for both validation and estimation sets.

From this , we can say that the penalty is not helping the model in the case of 10 principal components. Despite the fact,that the best index is shifted rightward.

20 COMPONENTS

Trying different values of lambda for 20 components.

Lambda = 0.001

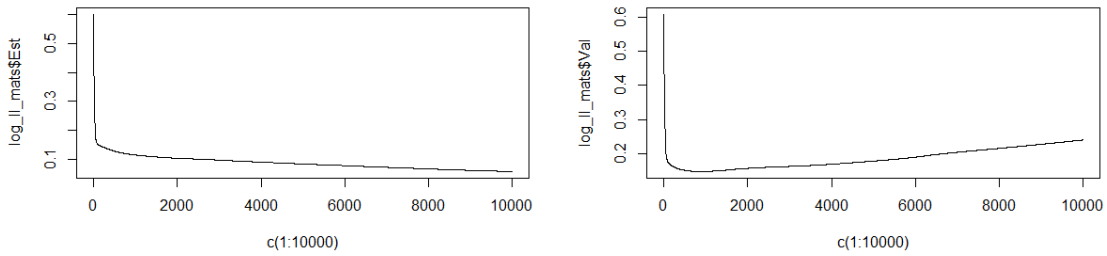


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.001)

Lambda = 0.01

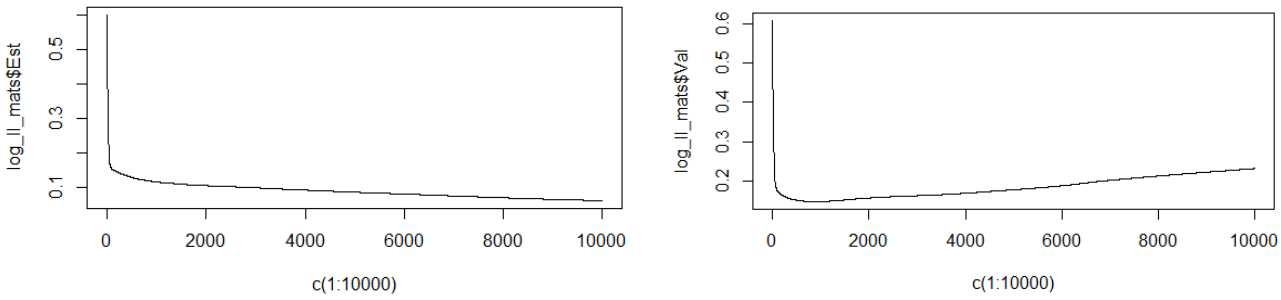


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.01)

Lambda = 0.05

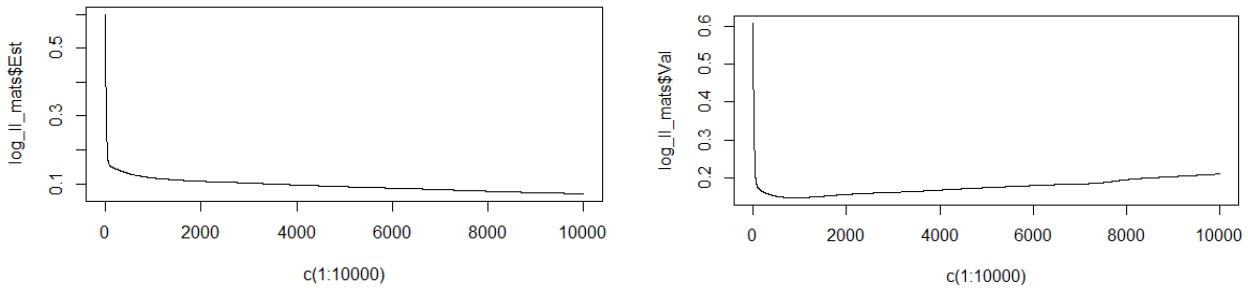


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.05)

Lambda = 0.1

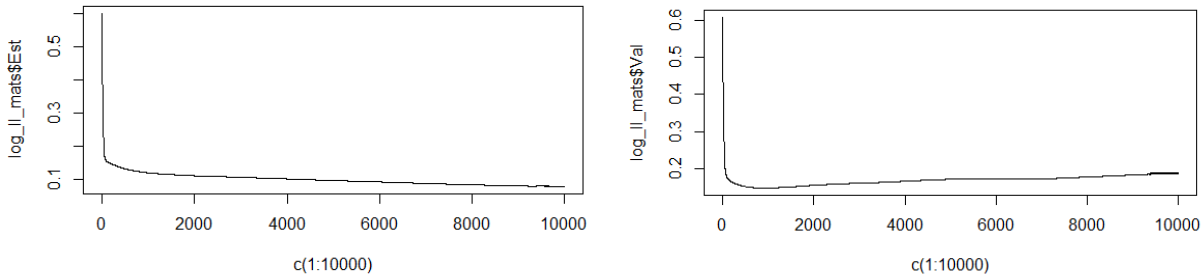


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.1)

Lambda = 1

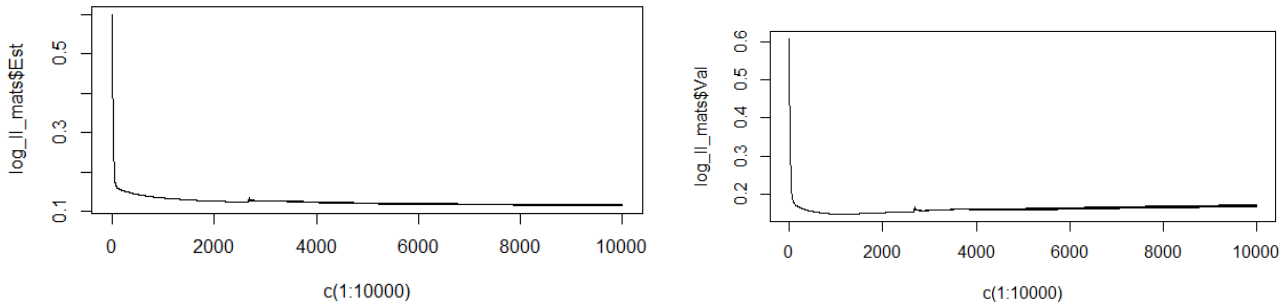


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 1)

Lambda = 1000

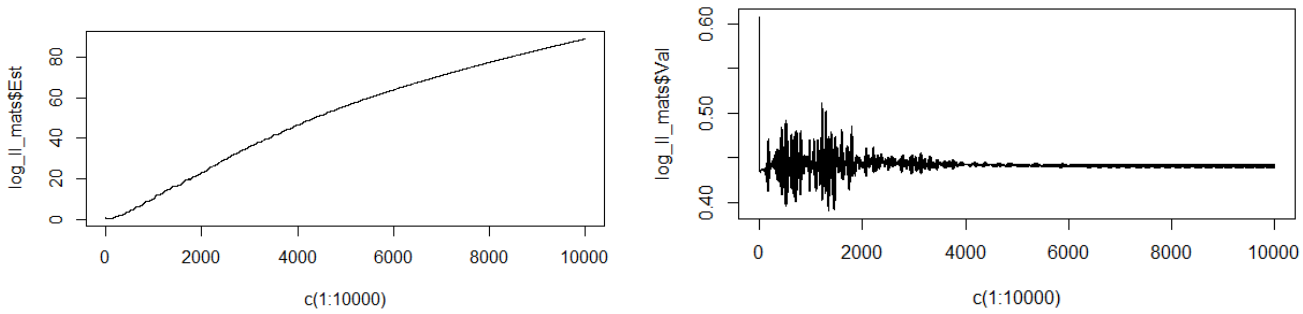


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 1000)

Summary – 20 principal components and lambda value comparison

Lambda	Min. minus log likelihood-validation	Min. validation error %	Best Index(for validation)
0.001	0.1476689	4.3	874
0.01	0.1476055	4.3	877
0.05	0.1474695	4.3	890
0.1	0.1473162	4.3	905
1	0.1473273	4	1067
1000	0.391	15.67	1342

We see that with the presence of a penalty, the best index for validation slowly shifts. This makes sense, as the penalty is dampening the weights and effectively reducing the overfitting, so the optimal index for validation shifts rightward..

Furthermore, the minus log likelihood also slowly improves as we increase the penalty, however the minimum validation error does not change. For lambda = 1 we see that underfit starts to occur as the value for the minimum minus log likelihood increases slightly.

So lambda as 0.1 is good for 20 principal components.

For lambda = 1000, the model underfits the training set completely and predicts all 0s for the validation set which makes sense as the weights should go down considerably.

40 PRINCIPAL COMPONENTS

Lambda = 0.001

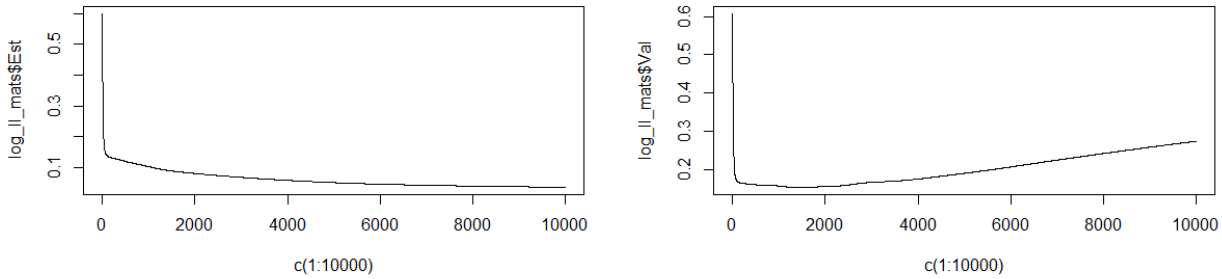


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.1)

Lambda = 0.01

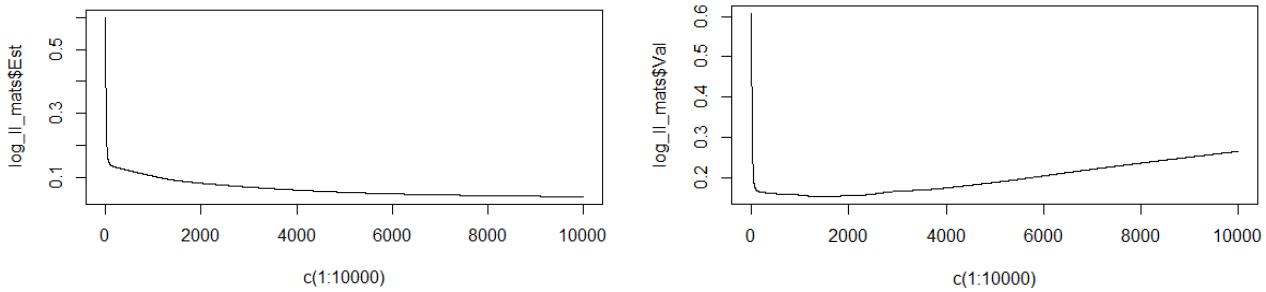


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.01)

Lambda = 0.05

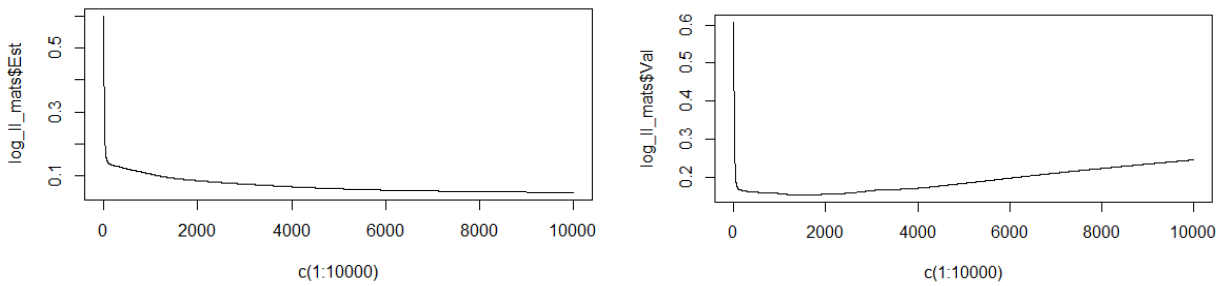


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.05)

Lambda = 0.1

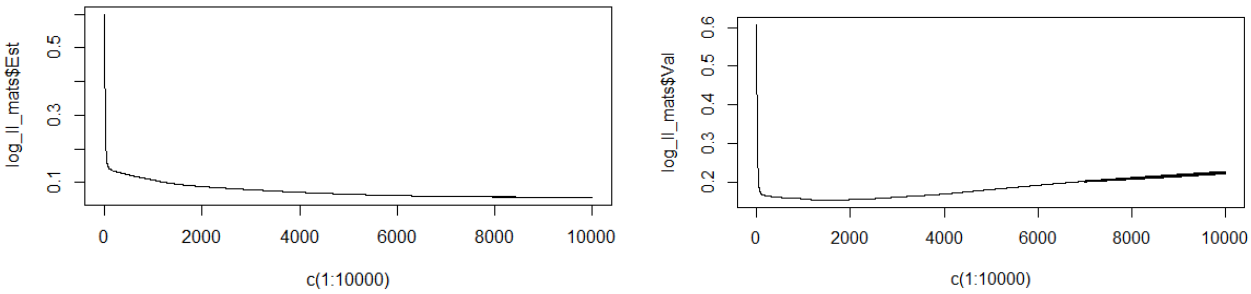


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 0.1)

Lambda = 1

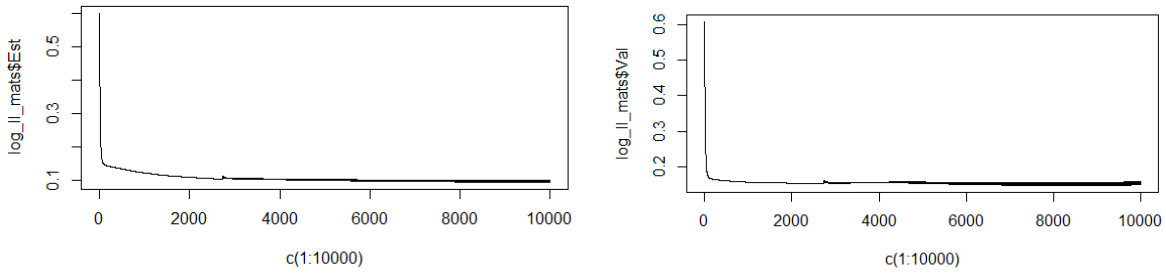


Fig.1&2 : Estimation & validation data minus log likelihood(lambda = 1)

Summary - 40 Principal components lambda value comparison

Lambda	Min. minus log likelihood-validation	Min. validation error %	Best Index(for validation)
0.001	0.1531647	4	1423
0.01	0.1531289	4	1426
0.05	0.1530435	4	1438
0.1	0.1530342	4	1449
1	0.1510234	4.3	8649

Similar to the situation explained for 20 principal components, the best index for validation shifts to the right (index with minimum minus log likelihood) .Corresspondingly, the minimum minus log likelihood also improves, marginally as we change lambda. However, when we transition from lambda = 0.1 to 1, there is a decrease in the minus log likelihood but the minimum validation error increases by 0.3 %. Taking this into consideration, lambda = 0.1 appears to be the best choice for 40 principal components.

Anurag Chaudhury

Summary discussion: For very small lambda, the effect is very similar to no penalty,and the minimum log likelihood and best index are very similar to the without penalty case. However, as we change the penalty going up slowly, we reach a point where the increase in lambda causes underfitting and many weights start dropping to quite small values.

Test Data

The best models for MLP from above are :

Learning rate = 0.001

Number of iterations = 10000

Number of PC.	Lambda	Minimum validation minus LL (average)	Minimum validation error %
10	0	0.1694751	6
20	0.1	0.1473162	4.3
40	0.1	0.1530342	4

According to the log likelihood only, we would go with 20 components but taking into consideration validation error 40 would be ideal. For 20 the results are :

Proposed model

Number of PC	Lambda	Test -LL (average)	Test error %
20	0.1	0.1503509	4.48 %

Now comparing it with the other best models :

Number of PC	Lambda	Test - LL (average)	Test error %
10	0	0.1620431	4.98 %
40	0.1	0.175652	4.127 %

In this case we see that the 40 component model with the slightly worse minus log likelihood but better validation error did better (4.127 vs 4.48 % error)

Now, to compare it with other models from the without penalty section:

Number of PC	Lambda	Test –LL(average)	Test error %
20	0	0.1504763	4.53
40	0	0.1769780	4.022

Again, despite 40 components with 0 penalty producing a slightly higher minus log likelihood on validation than with penalty, the error rate is lower , even though the minus log likelihood on the test set is high.

Comparison with all other models

Number of PC	Lambda	Test-LL(average)	Test error %
20	0.001	0.1507529	4.53 %
20	0.01	0.1507149	4.48 %
20	1	0.1500833	4.58%
40	0.001	0.1769678	4.022 %
40	0.01	0.1769223	4.073 %
40	1	0.1620022	4.73 %
10	0.001	0.1602677	4.98 %
10	0.01	0.1602585	4.98 %
10	0.1	0.1601296	5.04%
10	1	0.1594567	5.14 %

From logistic regression, the best results from validation were produced for 20 and 40 principal components. Therefore applying them to the test set we get :

Number of components	Minus log likelihood-Test data(average)	Test error %
10	0.1884397	6.41 %
20	0.1762244	6.06 %
40	0.179645	5.34 %

Further discussion

From the above output it is clear that minus log likelihood alone does not prove a model's quality. This is perhaps because we are modelling a binary response and even when the minus log likelihood is high the error rate in classification might be low.

Tuning parameters was not very easy. The learning rate was easy to find, as it was based on just seeing if the minus log likelihood for the estimation set decreased steadily. Number of iterations too was not tedious, as it involved finding around what point did the minus log likelihood for the training estimation set stabilize. Penalty was more involving, to compare with the without penalty case, a seed was fixed to compare the without penalty and with penalty situations. Using this, it could be shown how the penalty was changing the minus log likelihood for different principal component scenarios. Different seeds would yield slightly different results for both log likelihood and validation error.

Training was reasonably fast depending on the number of iterations. For 10000 iterations, it would take a few minutes for the computations to be performed. I increased performance marginally by computing the validation

log likelihoods in tandem with the training ones (using the weights from backprop. And running mlp_forward with them and the validation set).

The penalty, depending on its magnitude and the number of principal components helped or didn't. In the case of 10 components, it did not add anything useful to the average minus log likelihood on the validation set. However, for 20 and 40 components, a value of 0.1 proved to improve the minus log likelihood slightly over many iterations. From the observations in the test set, the best performance however was achieved by 40 principal components without any penalty associated – 4.022 % error.

Performance on the validation set was a reasonably good indicator of performance on test set. For 10 components, the performance of the validation set indicated that it would not do better than 20 or 40 on the test set and that was proven to be the case. However, for 20 and 40 components, the minus log likelihood for validation implied 20 with 0.1 penalty would do best. But, going by validation error instead 40 with 0.1 penalty would do better. From the observations across all models we see that the validation error was a fairly good indicator of what models would perform best. However, it turned out that it was not an exact predictor as clearly the 40 component model with no penalty was the best model overall.

Comparison with logistic regression:

Since, logistic regression also performed reasonably well we can probably say that the decision boundary between the classes is very close to linear , and as such the MLP does not decrease the error that much further.