ECE421S – Introduction to Machine Learning

Assignment 1

Linear and Logistic Regression

Hard Copy Due: February 6, 2019 @ BA3014, 4:00-5:00 PM EST

Code Submission: ece421ta2019@gmail.com

February 6, 2019 @ 5:00 PM EST

General Notes:

* For assignment related questions, please contact Matthew Wong (matthewck.wong@mail.utoronto.ca)
* For general questions regarding Python or Tensorflow, please contact Tianrui Xiao (tianrui.xiao@mail.utoronto.ca) or see him in person in his office hours, Tuesdays, 4:00-6:00 PM in BA-3128 (Robotics Lab)

Please circle section to which you would like the assignment returned

Tutorial Section

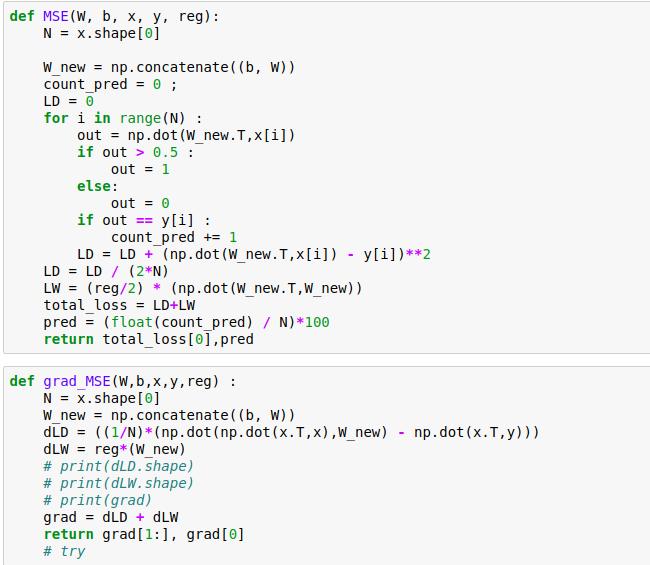
|  |  |  |  |
| --- | --- | --- | --- |
| 001 | 002 | 003 | 004 |
| 005 | 006 | 007 | Graduate |

Group Members

|  |  |
| --- | --- |
| Student Name | Student ID |
| Oishi Bandyopadhyay | 1004530224 |
| Urmi Joshi | 1004822766 |

**Linear Regression**

Starting with first step of algorithm, MSE (Total loss) for every iteration is calculated. Gradient helps to find the trained weights and trained bias of every iteration. Significant changes in bias and weights are observed.

Python Code Snippets:  


Analytical Expression:  
grad = (1/N) (transpose(X)\*X\*w – trans(X)\*y)

Gradient Descent is calculated for linear regression which provides validation and test loss accuracies using Loss functions.

* **Linear Regression without Regularization**

**Effect of Learning Rate:**   
High learning rates train faster and have better accuracies at 5000 iterations. In this case with learning rate 0.05 the train accuracy is 97.89% compared to learning rate 0.01 which is 96.31% and for learning rate 0.0001 which is 94.74 . With higher learning rate the model will train faster but it might miss the convergence point. With lower learning rate model will train slower but will converge very close to the optimal point.

Hence training time:  
 T(alpha = 0.005) < T(alpha=0.001) < T(alpha= 0.0001).  
Final classification accuracy:  
 After 5000 iterations the final classification accuracy of learning rate alpha 0.005, 0.001, 0.0001 are 97.24,95.86,95.17(Test accuracies).

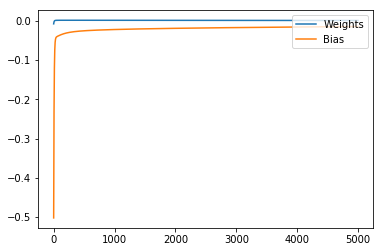
1. When iterations are 5000, alpha is 0.001 and Regularization factor is 0.0

Figure 1(Gradient Graph)

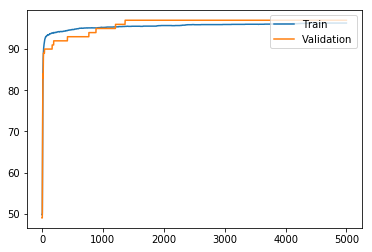


Figure 2(Accuracy Graph)

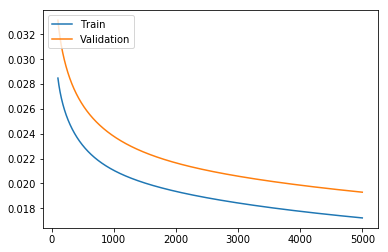


Figure 3(Loss Graph)

Test Accuracy: 95.86

Test Loss: 0.019

In figure 1 Gradient Graph denotes that weights remain constant but bias at the end tends to become very small. While in figure2 Blue line indicates Training set and the green line indicates Validation set. Test Accuracy observed is 95.86. We can see that Validation accuracy is more than Training accuracy. Whereas, the below graph defines the Loss taking place in training and validation. Training loss is less as compared to Validation Loss.

1. When iterations are 5000 and alpha is 0.005

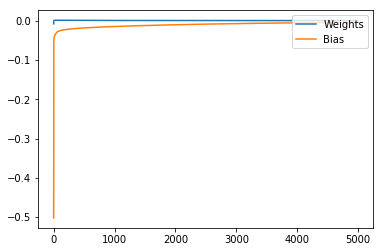


Figure 1 Gradient Graph

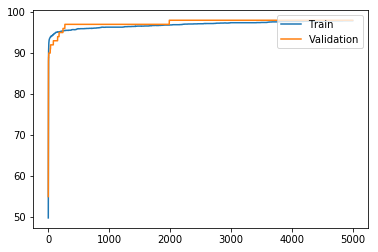


Figure 2 Accuracy Graph

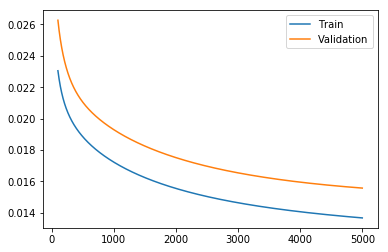


Figure 3 Loss Graph

Test Accuracy: 97.24

Test Loss: 0.017

In figure 1 bias values at the end are quite similar to the weights, almost near to 0.0. In figure 2 it is observed initially, training accuracy was more than validation. Whereas, training gives an incremental graph and validation gives a bumpy graph. At last, both give about same accuracy. In figure 3, training decreases its loss gradually with increasing iterations and at end becomes less than 0.014, while validation shows same pattern as of training but still gives a high loss.

1. When iterations are 5000 and alpha is 0.0

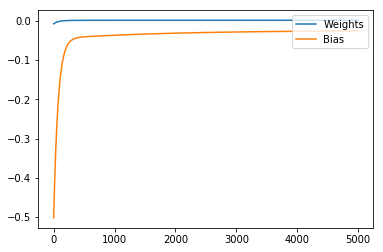


Figure 1 (Gradient Graph)

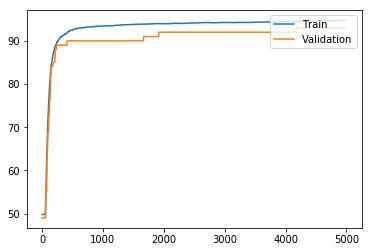


Figure 2 (Accuracy Graph)

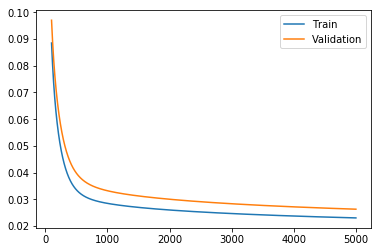


Figure 3 (Loss Graph)

Test Accuracy: 95.17  
 Test Loss : 0.023

In figure 1 bias values don’t tend to zero like weights and previous graph. In figure 2 it is observed training accuracy is more than validation. Whereas, training gives an incremental graph and validation gives a bumpy graph. In figure 3, initially, train and validation had almost same loss, but gradually training decreases its loss with increasing iterations and at end becomes less than 0.002, while validation shows same pattern as of training but still gives a high loss.

Therefore, we can say that as alpha comes near to 0.0, Training data gives more accuracy than validation and with that shows less test loss. When alpha tends to 0.0, Loss Graph shows very small amount of train loss.

* **Linear Regression with Regularization**

**Effect of Regularization Parameter:**  
With the introduction of the regularization parameter the model starts to under-fit. With increase in the regularization parameter the model under-fits even more. This can be clear from the loss plots of the three-regularization parameter shown below. The gap between the train and the validation increases because of under-fitting.  
Therefore:   
 Final Accuracy(Train) - Final Accuracy(Validation/Test) with reg 0.5 >   
 Final Accuracy(Train) - Final Accuracy(Validation/Test) with reg 0.1 >  
 Final Accuracy(Train) - Final Accuracy(Validation/Test) with reg 0.001   
  
Regularization parameter is tuned after looking at the validation set for the same reason. When the plots of the training loss and the validation loss do not come close then it means that the graph has under-fit. And if in the graph the validation loss comes very close to the train loss and then starts to increase, this means that the model has over-fit for the train dataset.

1. Regularization = 0.001

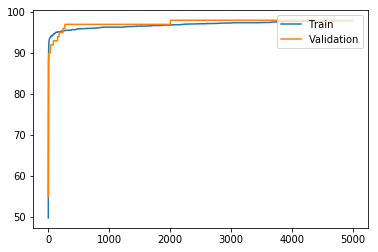


Figure 1 (Accuracy Graph)

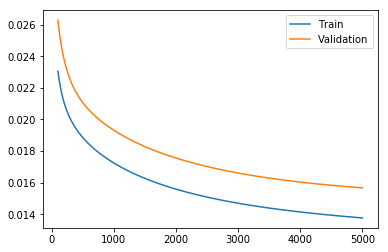


Figure 2(Loss Graph)

Test Loss:0.017  
 Test accuracy: 97.24

In figure 1, validation is more accurate than train. But validation line is found to be a bit bumpy, whereas train line goes smooth enough. In figure 2, we can see a high difference between train loss and validation loss.

1. Regularization = 0.1

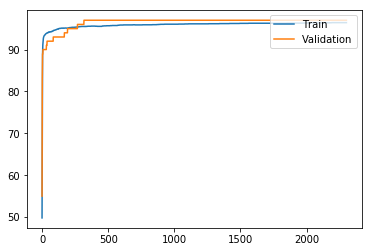


Figure 1 (Accuracy Graph)

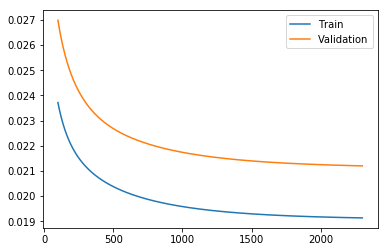


Figure 2 (Loss Graph)

Test Loss:0.021  
 Test accuracy: 95.86

Figure 1, is same as that of pervious one with regularization 0.001. But in figure 2, Loss has increased as compared to about loss graph. It shows with an increase of 4% comparative to above loss graph.

1. Regularization = 0 .5

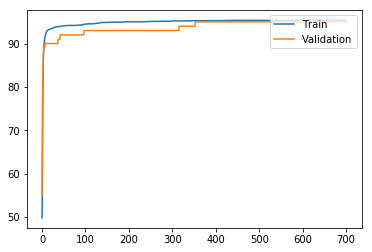


Figure 1 (Accuracy Graph)

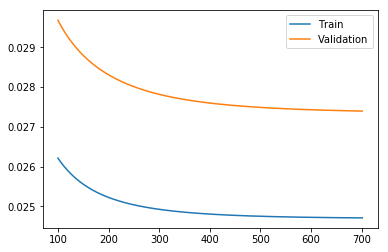


Figure 2(Loss Graph)

Test Loss:0.025  
 Test accuracy: 95.172

Figure 1, shows initially train accuracy is high comparative to validation accuracy for iterations from 0 to 400. But later on, both accuracies become close to each other. Loss graph shows a huge difference between train loss and validation loss.

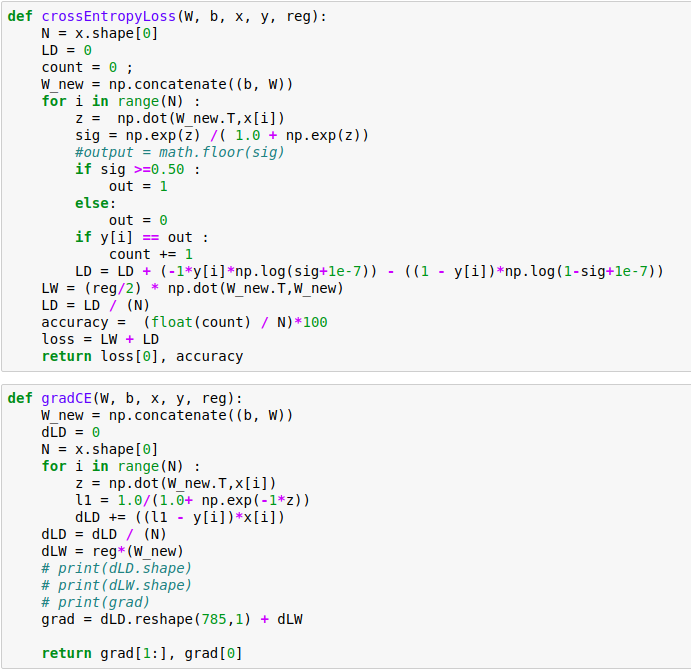
Therefore, as regularization increases, it can be stated that loss increases between train and validation, whereas difference in accuracy between those two decreases. It can also be seen, as Regularization increases, the train loss and validation loss also increases.

**Comparing Batch GD with Normal Equation:**  
The normal Equation gives better performance, for train error and accuracy compared to Batch GD. But it has higher losses and lower accuracies for test and validation this is because the model overfits for the train data. Whereas, model performs better for Batch GD on test and validation data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Train Loss | Validation Loss | Test Loss | Train Accuracy | Validation Accuracy | Test Accuracy |
| Normal Equation | 0.0093 | 0.023 | 0.028 | 99.37 | 96.00 | 94.48 |
| Batch GD(alpha = 0.005) | 0.01 | 0.02 | 0.017 | 97.89 | 97.85 | 97.24 |

**Logistic Regression**

In logistic regression, gradient descent is implemented which provides cross entropy loss and its gradient (weights and bias) for every iteration.

**Python Code snippet:**  


**Analytical Expression:**  
Gradient = (1/N) \* Σ (h(z) - y)\*x  
where h(x) = sig(trans(W)\*x + b) and sig(z) = 1/(1 + e^(-z))

1. Alpha = 0.005 and regularization = 0.1

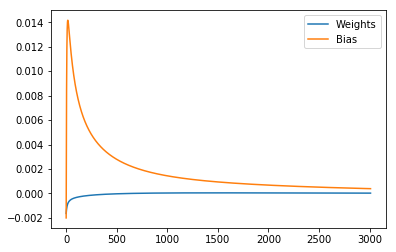


Figure 1(Gradient Graph)



Figure 2(Accuracy Graph)

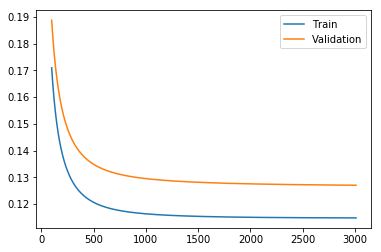


Figure 3(Loss Graph)

Test Loss:0.12  
 Test Accuracy:97.93

Figure 1, weights are increasing gradually form negative to 0.0 and bias show a tremendous increase from negative to 0.014 initially. Later on, bias gradually decrease to approximate 0.0. In Figure 2, at initial stage validation shows increase in accuracy than train but later on, both come at same pace. Figure 3, shows that train loss is less but comparing train and validation loss initially it was less but increasing number of iteration it increased.

1. Alpha = 0.005 and regularization = 0.0

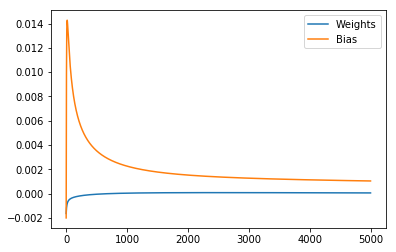


Figure 1(Gradient Graph)



Figure 2(Accuracy Graph)

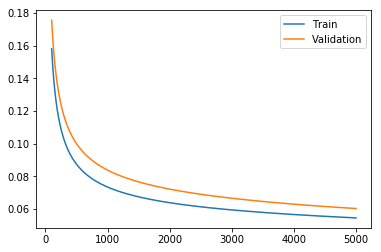


Figure 3(Loss Graph)

Test Loss:0.08  
 Test Accuracy:97.93

Figure 1 and Figure 2, behaves similar to that of previous gradient graph and accuracy graph. Therefore, we can say that change in regularization would not affect gradient graph and accuracy graph. But in figure 3, it shows as regularization increases difference between train and validation loss decreases. Validation Loss is also seemed to be decreased in model.

**Comparisons of Linear and Logistic Regression:**

Therefore, comparing with linear regression, it is seen that logistic regression gives a good accuracy. The model convergences better for logistic is better than linear regression. This is because cross entropy loss updates the gradient value based on the probability that a given point is classified to a region. If the model assigns high probability to a misclassified point the model is penalized more with the gradient.

**Batch Gradient vs SGD and Adam**

* **Stochastic Gradient Descent (SGD) and Adam : Gradient Descent Optimization Algorithm**

SGD is updated Gradient Descent, with iterative method (that is of Batch size 1 or mini-batch) for optimizing a differential function. It is used to reduce the loss function. SGD is used as a part of computation in Adam. Adam is an Adaptive Learning rate optimization algorithm which uses SGD for its Momentum because in SGD it iterates using moving average of gradient.

Analytical Expression of Adam :

Mt  = B1 \* Mt-1 + (1 - B1 ) \* gt

Vt = B2 \* Vt - 1 + (1 – B2 ) \* (gt ^2)

M and V are moving averages, g is the gradient and B are the hyper- parameters.

In this assignment, Adam is implemented using tensorflow library. Here, SGD uses mini-batch size.

**SGD with Batch size 500, 700 iterations, reg = 0.0 and alpha is 0.001**

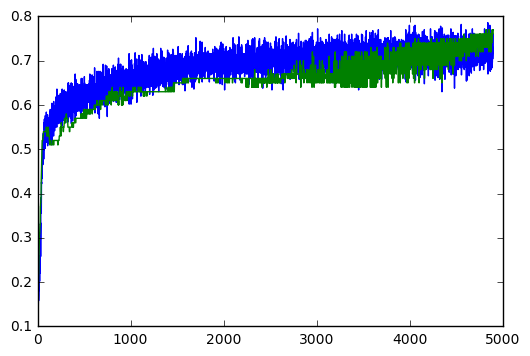
**** Test Accuracy: 0.7034483

Figure 1 (Accuracy Graph)

Figure 1 shows, Training accuracy increases gradually but after a point it becomes constant. Whereas, Validation accuracy from mid-point of total iterations it keeps on increasing linearly. In middle, few batch of data does not show good accuracy. But, it makes process faster.

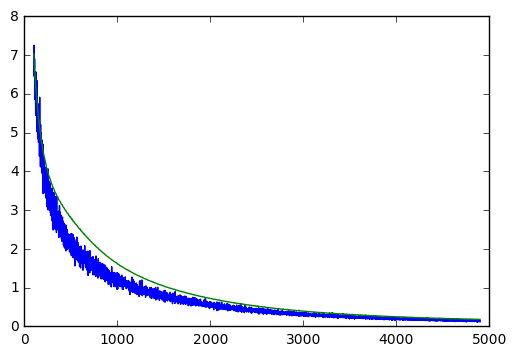
 Test loss: 0.13968188

Figure 2(Loss Graph)

In Figure 2, we can see initially, train and validation both had a high amount of loss of approx 7. With increase in iterations loss for both decreased and eventually showed same amount of loss.

* **SGD with different Batch Size**
* Accuracy Graph

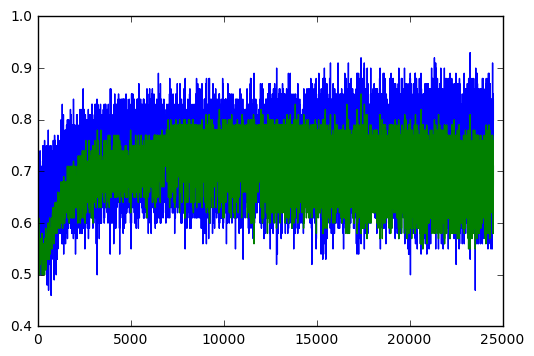
 Test Accuracy: 0.7241379

Figure 1(Accuracy Graph, Batch size 100)

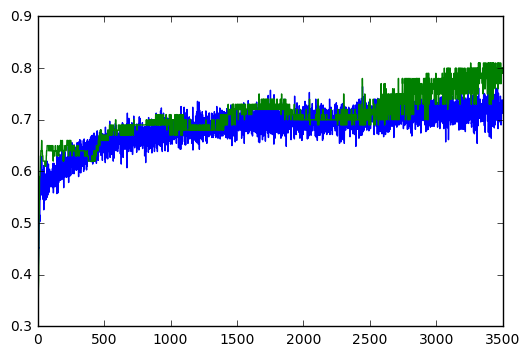
 Test Accuracy: 0.71034485

Figure 2 (Accuracy Graph, Batch size 700)

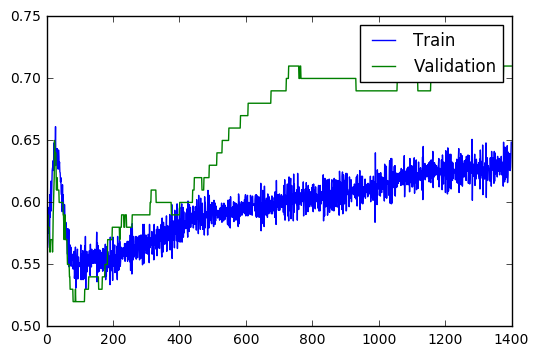
Test Accuracy: 0.6689655

Figure 3(Accuracy Graph, Batch size 1750)

In this accuracy graph Figure1, throughout the iterations all batch show similar accuracy, while train batches at points give high accuracy. When batch size is 700 (figure 2), validation accuracy seems to increase after certain iterations, while train accuracy shows minimal difference in accuracy. Figure 3, huge change is seen in validation accuracy. Graph shows that both train and validation drop at one point, but later on validation gives very good accuracy.

* Loss Graph

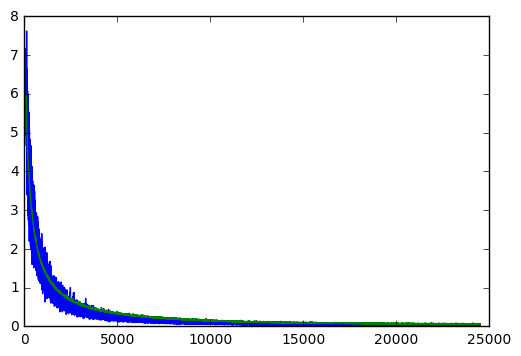
 Test loss: 0.07663254

Figure 1(Loss Graph, Batch size 100)

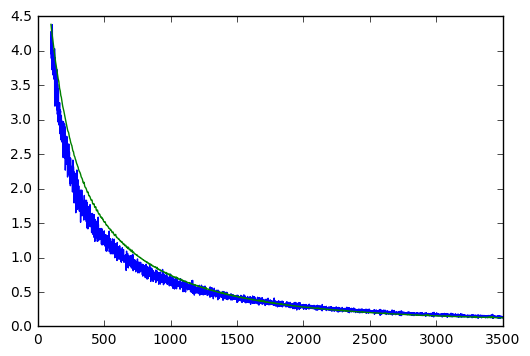
Test loss: 0.20089601

Figure 2(Loss Graph, Batch size 700)

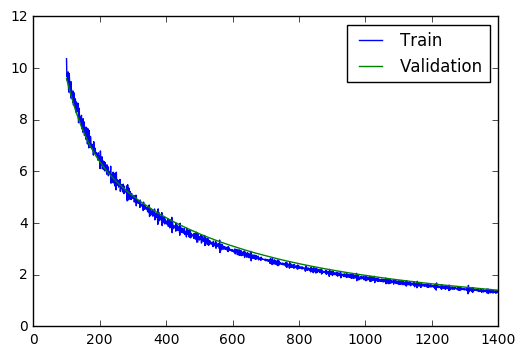
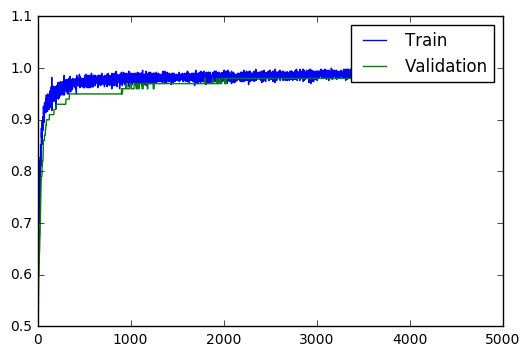
Test loss: 1.1022667

Figure 3 (Loss Graph, Batch size 1750)

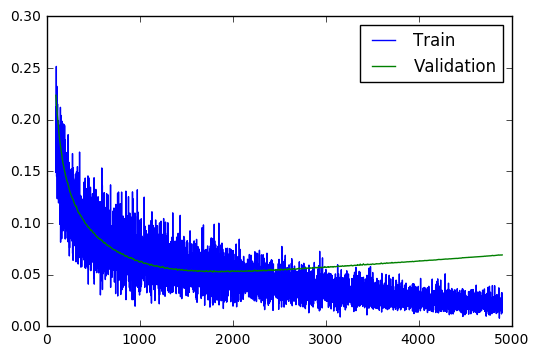
In Figure 1 Train and Validation loss produce a linear decrease till 10000 iterations and later on remains constant to 0. It takes about 25000 iterations to get state of 0.0 loss. Whereas, as batch size (Figure 2) increases we can observe that at one moment validation loss seems to be greater comparative to train loss. Therefore, at end we can see that as batch size increases, validation and train loss become same.

1. Batch size 1750

* **Cross Entropy**

Test Accuracy: 0.9724138

In cross entropy, initially bumpy accuracy is found, but later on matches with the train accuracy.

Test loss: 0.12820636

Loss Graph, shows that initially validation loss is minimum, but later it gradually increases, whereas training loss keeps decreasing.

Going through both the models, Linear Regression gives good accuracy, but Logistic Regression gives better accuracy.

* **SGD vs Batch GD**

Here, Batch Gradient uses the whole dataset, whereas SGD divides data into mini batches and then uses those samples.