Handwritten Digits Classification using Logistic and Softmax Regression

Sriram Ravindran
A53208651
sriram@ucsd.edu

Ojas Gupta
A53201624
ogupta@ucsd.edu

Abstract

Here in this project we have implemented the famous Logistic Regression which is a two class classifier and its extension Softmax Regression which is a multiclass classifier. Both of these methods are trained and tested on famous handwritten MNIST dataset via gradient descent. We used the first 20000 data points as our training set out of which we have kept the 10 percent i.e. 2000 as our hold out set. On training our system we have tested on the first 2000 test data points so as to evaluate our work. On using the Logistic Regression, we have attained an accuracy of " " while classifying 2 vs 3 and an accuracy " " while classifying 2 vs 8. On applying 10 way classification, we get an accuracy of " ". To enhance classification and generalization we have used regularization as well.

1 Derivation of Gradient for Logistic Regression

We will derive the gradient for logistic regression by using the following predefined variables given in the document.

Given:

$$y^n = \frac{1}{1 + exp(-w^T x^n)} \text{(Sigmoid function)}$$
$$E(w) = -\sum_N t^n ln y^n + (1 - t^n) ln(1 - y^n)$$

To Prove:

$$\frac{\partial E^n(w)}{\partial w_j} = -(t^n - y^n) x_j^n$$

Let's recall the properties of Sigmoid Function, if $\sigma(x)$ is a sigmoid function then following two properties hold:

$$\sigma(x) = 1 - \sigma(-x)$$

$$\sigma'(x) = \sigma(x)\sigma(-x)$$

Derivation:

$$E^n(w) = -(t^n ln y^n + (1-t^n) ln(1-y^n))$$

$$\frac{\partial E^n(w)}{\partial w_j} = -(\frac{t^n}{y^n} - \frac{1-t^n}{1-y^n})\frac{\partial y^n}{\partial w_j}$$

$$\frac{\partial E^n(w)}{\partial w_j} = -(\frac{t^n - y^n}{y^n(1-y^n)})\frac{\partial y^n}{\partial w_j}$$

$$\frac{\partial E^n(w)}{\partial w_j} = -(\frac{t^n - y^n}{y^n(1-y^n)})y^n(1-y^n)\frac{\partial w^T x}{\partial w_j} \text{(Properties of Sigmoid)}$$

$$\frac{\partial E^n(w)}{\partial w_j} = -(t^n - y^n)\frac{\partial w^T x^n}{\partial w_j}$$

$$\frac{\partial E^n(w)}{\partial w_j} = -(t^n - y^n)x_j^n$$

Hence Proved

2 Derivation of Gradient for Softmax Regression

We will derive the gradient for softmax regression by using the following predefined variables given in the document.

Given:

$$y_k^n = \frac{exp(a_k^n)}{\sum_k' exp(a_k'^n)}$$
$$a_k^n = w_k^T x^n$$

$$E = \sum_n \sum_{k=1}^c t_k^n ln(y_k^n)$$

To Prove:

$$-\frac{\partial E^n(w)}{\partial w_j^k} = (t_k^n - y_k^n)x_j^n$$

Derivation:

$$E^n(w) = -\sum_{k'=1}^c t_{k'}^n ln(y_{k'}^n)$$

$$\frac{\partial E^n(w)}{\partial w_{jk}} = -\sum_{k'=1}^c \frac{\partial(t_{k'}^n ln(exp(w_{k'}^T x^n)) - t_{k'}^n ln(\sum_{k''} exp(w_{k''}^T x^n)))}{\partial w_{jk}}$$

$$\frac{\partial E^n(w)}{\partial w_{jk}} = -\sum_{k'=1}^c \frac{\partial(t_{k'}^n w_{k'}^T x^n - t_{k'}^n ln(\sum_{k''} exp(w_{k''}^T x^n)))}{\partial w_{jk}}$$

$$\frac{\partial E^n(w)}{\partial w_{jk}} = -t_k^n x_j^n - \sum_{k'=1}^c \frac{\partial(t_{k'}^n ln(\sum_{k''} exp(w_{k''}^T x^n)))}{\partial w_{jk}}$$

Only one $t_{k'}$ is 1, all others are 0s.

$$\frac{\partial E^n(w)}{\partial w_{jk}} = -t_k^n x_j^n - \frac{\partial(ln(\sum_{k''} exp(w_{k''}^T x^n)))}{\partial w_{jk}}$$

$$\frac{\partial E^n(w)}{\partial w_{jk}} = -t_k^n x_j^n - \frac{exp(w_k^T x^n)}{\sum_{k''} exp(w_{k''}^T x^n))} x_j^n$$

$$\frac{\partial E^n(w)}{\partial w_{jk}} = -t_k^n x_j^n - y_k^n x_j^n$$

$$\frac{\partial E^n(w)}{\partial w_{jk}} = -(t_k^n - y_k^n) x_j^n$$

Hence Proved

3 Logistic Regression

Logistic regression can be modeled as using a single neuron reading in an input vector (1,x) $\in \mathbb{R}^{d+1}$ and parameterized by weight vector $w \in \mathbb{R}^{d+1}$. d is the dimensionality of the input, and we tack on a "1" at the beginning for a bias parameter, w_0 . The neuron outputs the probability that x is a member of class C_1.

$$\text{P}\ (x \in C_1|x) = \frac{1}{1 + exp(-w^T x)}$$

$$\text{P}\ (x \in C_2|x) = 1\ \ \text{P}(x \in C_1|x)$$

Cross entropy loss function for two categories over the training examples is given as following.

$$E^n(w) = -(t^n ln y^n + (1 - t^n) ln(1 - y^n))$$

Here, t^n is the target or teaching signal for example n. Our goal is to optimize this cost function via gradient descent. This cost function is minimized at 0 when $t^n = y^n$ for all n.

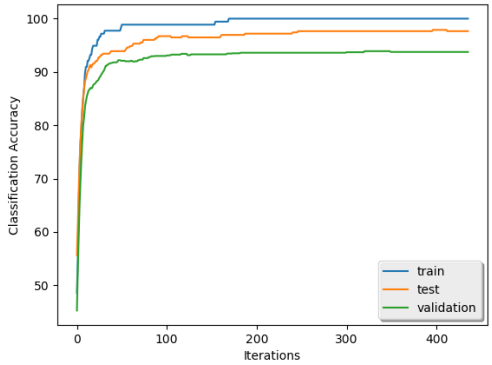
3.1 Introduction

Using the gradient derived for Logistic Regression cross entropy loss, we will first use gradient descent to classify for categories: 2's and 3's, 2's and 8's.

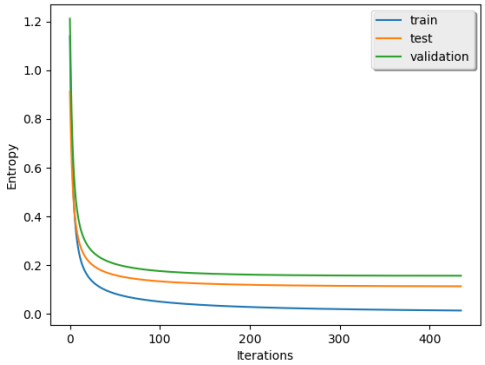
We have taken the data from the famous MNIST handwritten data, of which we used the first 20000 as our training data set, and 2000 from the testing data set. Out of the training data set we have excluded 2000 data points for the hold out set.

3.2 Results and Discussion

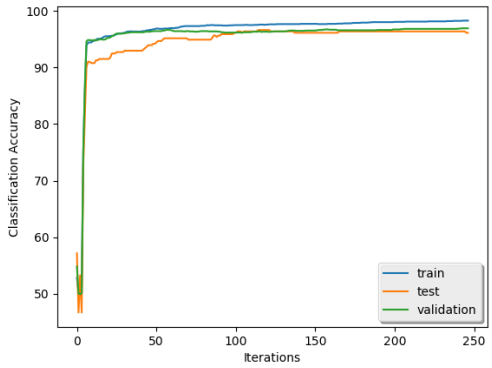
We have calculated the entropy and classification accuracy of 2 vs 3 and 2 vs 8 data set and the results are found to be quite impressive. Following are the graphs plotted for them. Figure 1 and 2 shows the entropy and classification accuracy of 2 vs 3



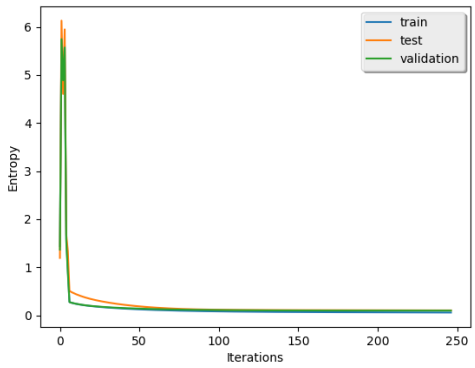
(a) Plot of accuracy in 2 vs 3.



(b) Plot of entropy in 2 vs 3.

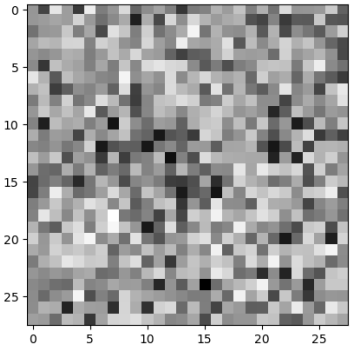


(a) Plot of accuracy in 2 vs 8.

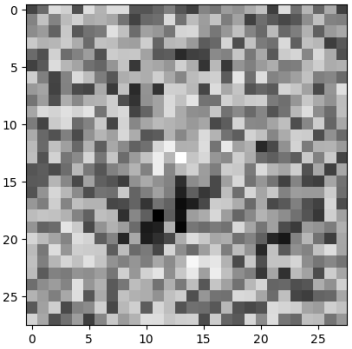


(b) Plot of losses in 2 vs 8.

As the ”hold-out” set is previously unseen, it acts as a good replacement for the testing data, as can be seen from the results.



(a) Subtraction of 2 vs 3 Classifier and 2 vs 8 Classifier.



(b) 2vs3.

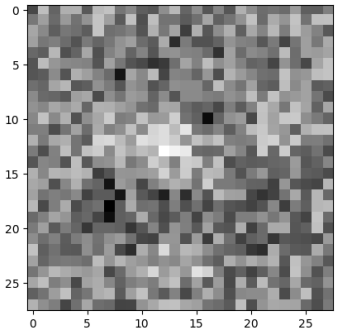


Figure 4: 2 vs 8 Classifier weights as an image.

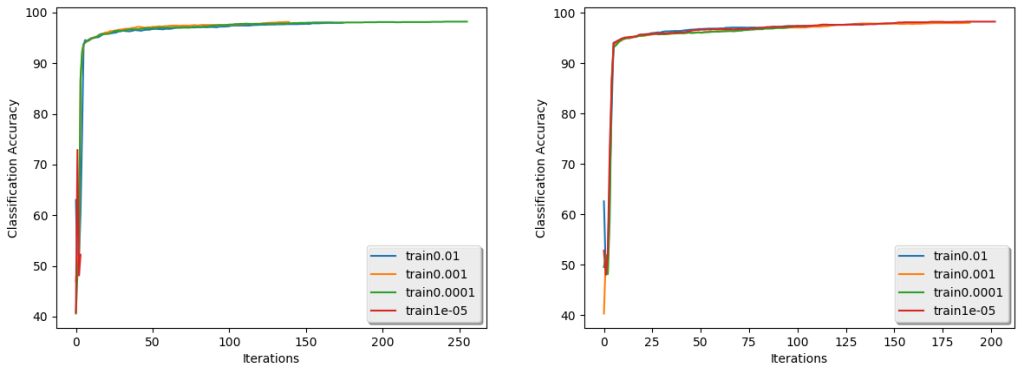
d) While a pattern is not clear from the image, the darker a cell in the image, the more those cells contribution towards two-ness in the image. The difference image must contain features (dark cells) which are neither in three nor in eight.

4 Regularization

For L2: $-\frac{\partial E(w)}{\partial w_j} = \sum_n (t^n - y^n) x_j^n + \sum_{ij} 2\lambda w_{ij}$

For L1: $-\frac{\partial E^n(w)}{\partial w_j} = \sum_n (t^n - y^n) x_j^n + \sum_{ij} \lambda |w_{ij}| / w_{ij}$

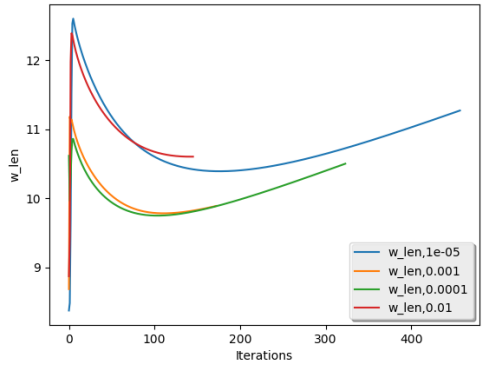
b) From our training experience, they are pretty much similar in performance. The accuracy difference is within 1%. Accuracies were approx 97% for both L1 and L2 regularization.



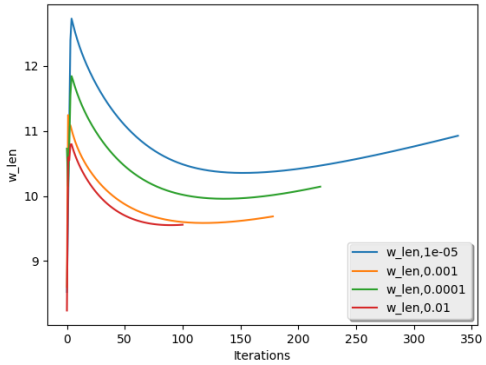
(a) Training Accuracy with L1 regularization with different lambda.

(b) Training Accuracy on L2 regularization with different lambda.

c) Ideally, as training progresses, the norm of the weights must be higher if lamda is lower. We see this is the case with L2, however, in our case the L1 showed an anomaly, with the red line coming above the others. This will not be the case if the initialization is kept same.

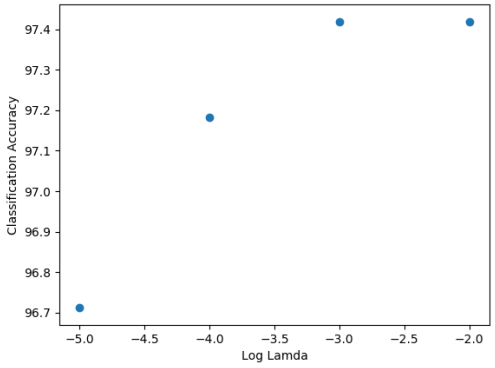


(a) Length of weight vector on L1.

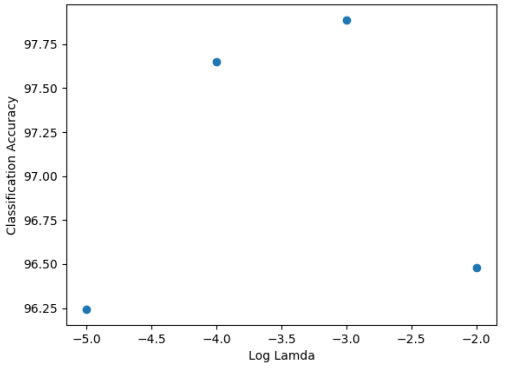


(b) Length of weight vector on L2.

d) For L1 regularization, lamda = 0.001 gives best test accuracy. For L2 regularization, again, lamda = 0.001 gives best test accuracy. Both cases approx 98

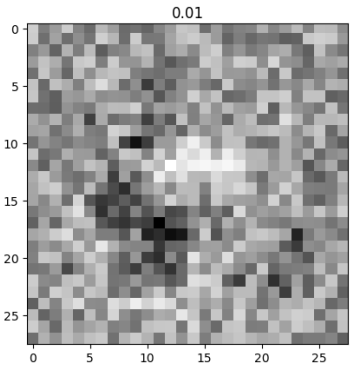


(a) Testing error vs logλ on L1.

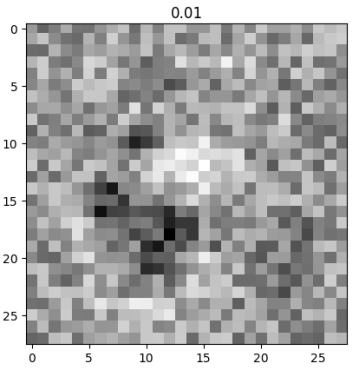


(b) Testing error vs logλ on L2.

e) For L2 regularization case, we see a bunch of dark cells together in lower left corner, getting most darkest in the best case. For L1, we see dark cells interspersed as expected. L1 makes the model sparse, thus many cells get lighter. Also consistent with L2, as we expect contribution from a lot more sources than L1. $lambda$=0.01

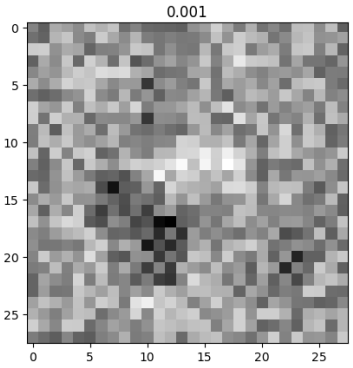


(a) Testing error vs logλ on L1.

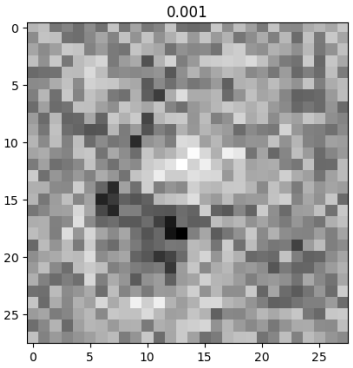


(b) Testing error vs logλ on L2.

$lambda$=0.001

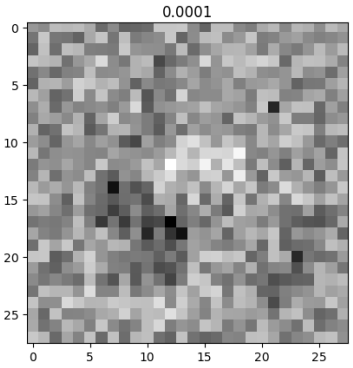


(a) Testing error vs logλ on L1.

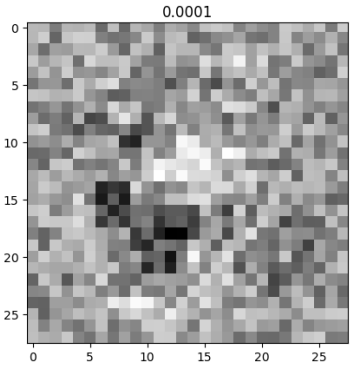


(b) Testing error vs logλ on L2.

$lambda$=0.0001

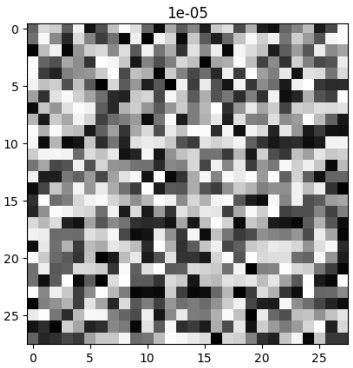


(a) Testing error vs logλ on L1.

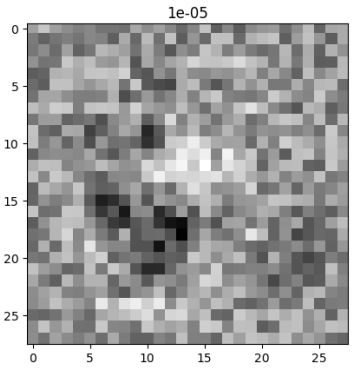


(b) Testing error vs logλ on L2.

$lambda$=0.00001



(a) Testing error vs logλ on L1.



(b) Testing error vs logλ on L2.

5 Softmax Regression

Softmax regression is the generalization of logistic regression for multiple (c) classes. Now given an input x^n , softmax regression will output a vector y^n , where each element, y_k^n represents the probability that x^n is in class k.

$$y_k^n = \frac{exp(a_k^n)}{\sum_k^{'} exp(a_k^{'n})}$$
$$a_k^n = w_k^T x^n$$

Here, a_k^n is called the net input to output unit y_k. Note each output has its own weight vector w_k . With our model defined, we now define the cross-entropy cost function for multiple categories:

$$E = \sum_n \sum_{k=1}^{c} t_k^n ln(y_k^n)$$

Again, taking the average of this over the number of training examples normalizes this error over different training set sizes.

Further information is distributed as section 3.1 contains the introduction to the problem. Results and Discussion is done in section 3.2.

5.1 Introduction

In this part of the problem, we have created a multiclass classifier which classifies a data point into 10 different classes. We have taken the data from the famous MNIST handwritten data, of which we used the first 20000 as our training data set, and 2000 from the testing data set. Out of the training data set we have excluded 2000 data points for the hold out set. We haven't done mini-batch gradient descent because we got convergence over about 400 iterations.

5.2 Results and Discussion

Here, as instructed in the part(a) we have used a hold out set and Regularization in the softmax regression problem so as to classify handwritten data set in 10 classes.

Here the type of Regularization from which we received the maximum accuracy of hold out set was L2 and the value of λ = 0.00001. Here We checked the method for other values of λ as well mainly [0.1, 0.01, 0.001, 0.0001, 0.0000] with both L1 and L2 Regularization and found L2, 0.00001 to the most appropriate one.

As asked in the part 6(b), following is the graph of loss function with λ=0.00001.

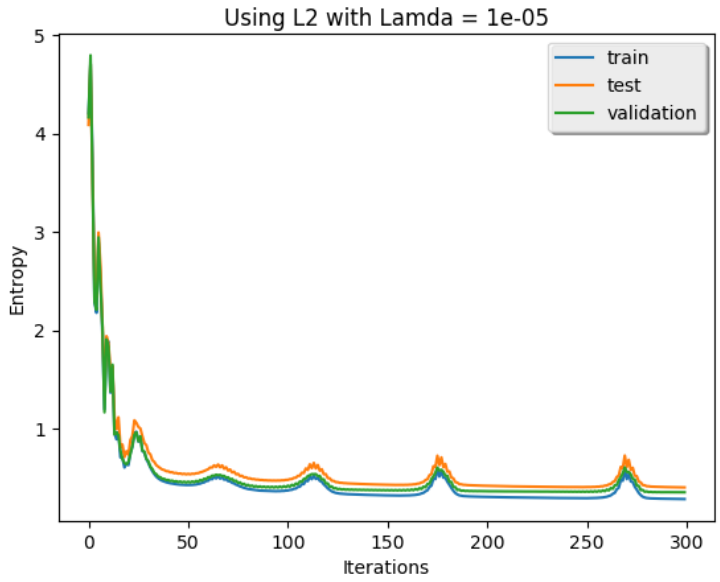


Figure 12: Part 6(c):Entropy vs Iterations of Softmax Regression with λ=0.00001.

As asked in the part 6(c), following is the graph of classification accuracy with λ=0.00001. Accuracy was about 92.16%.

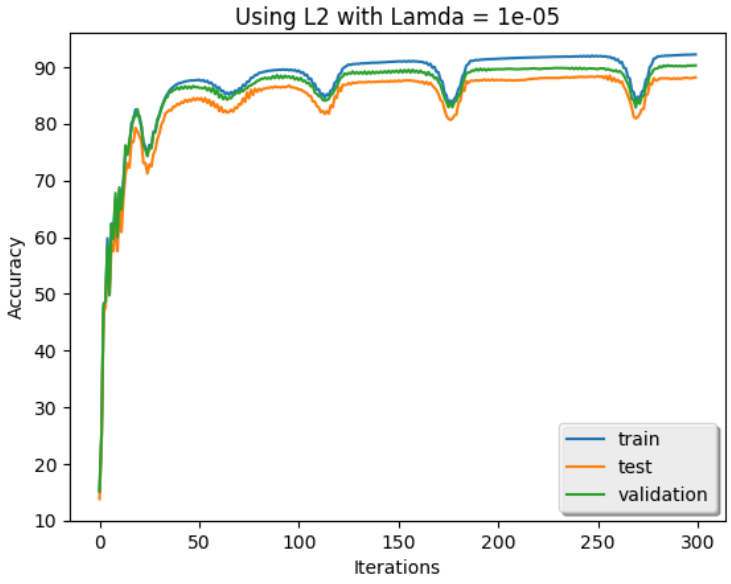


Figure 13: Part 6(b):Classification Accuracy of Softmax Regression with λ=0.00001.

6 Final observations

Logistic regression and Softmax regression were implemented on Python and tested with MNIST. The characteristics of the final weights, effect of regularization, effect due to weight of penalty, λ

were studied. We conclude that Softmax regression is a good method for multiclass classification and logistic regression is a good method for binary classification for simple problems like MNIST.

7 Contribution

7.1 Ojas Gupta

I have written the code for Softmax Regression and executed it with the MNIST dataset. I trained the softmax regression to classify a given data point into one of the 10 classes. Logistic Regression part is done by my partner Sriram. All the remaining parts were distributed equally among us such as Derivations, regularization and report making as mentioned in the assignment.

7.2 Sriram Ravindran

I wrote the code for Logistic Regression and executed it with the data from the MNIST data set. Along with training the 2vs3 and 2vs8 dataset, I plotted the accuracy and loss graphs. Apart from Logistic Regression and Softmax Regression all the other parts such as Derivation, Regularization, creating latex report in NIPS format has been done by both me and my partner together so as to divide the load equally among ourselves.