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 Keywords

Neural Networks, Logistic Regression, Softmax Regression

2 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:

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

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:

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

$$2)\ \sigma'(x) = \sigma(x)\sigma(-x)$$

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

3 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:

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

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

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

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

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

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

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

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

Hence Proved

4 Logistic Regression

Although we consider logistic regression to be a classification technique, it is called ”regression” because it is used to fit a continuous variable: the probability of the category, given the data. 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.

P $(x \in C_1|x) = \frac{1}{1+exp(-w^T x)}$
P $(x \in C_2|x)$ = 1 P($x \in C_1|x$)

where g w (x) simply notes that the function g is parameterized by w. Note we identify the output y n of the ”network” for a particular example, x n , with g w (x n), i.e., y n = g w (x n). With the hypothesis function defined, we now use the cross entropy loss function (Equation 3) for two categories over our training examples. This equation measures how well our hypothesis function g does over the N data points, E(w) = N X t n ln y n + (1 t n) ln(1 y n). (3) n=1 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. One issue with this cost function is that it depends on the number of training examples. For reporting purposes in this assignment, a more convenient measure is the average error: E(w) = N 1 X n t ln y n + (1 t n) ln(1 y n).

4.1 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.

Now, using the gradient derived for Logistic Regression cross entropy loss, use gradient descent to classify x R 785 (there is one extra dimension for the bias term) for two categories: 2s and 3s. The target is 1 if the input is from the ”2” category and 0 if it is from the other category.

4.2 Method

4.3 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

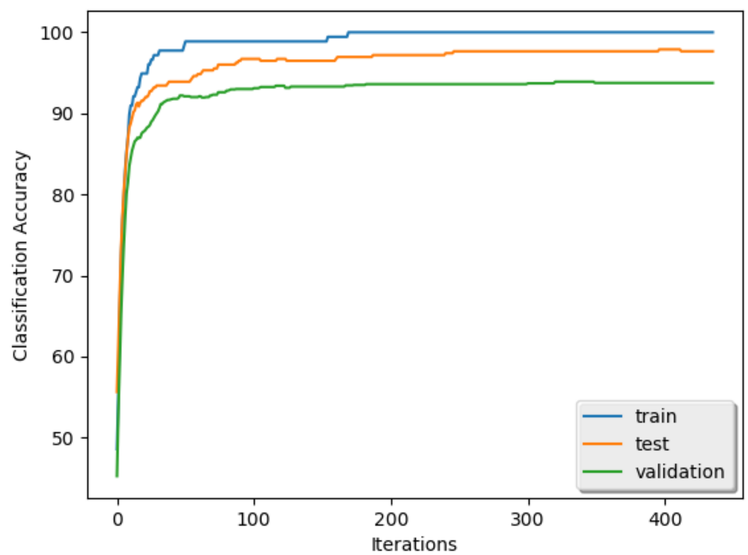

Figure 1: Plot of accuracy in 2 vs 3.

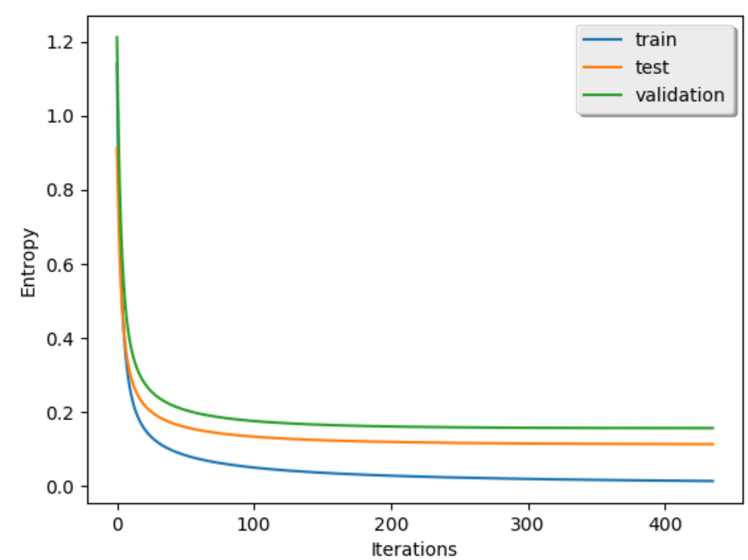

Figure 2: Plot of entropy in 2 vs 3.

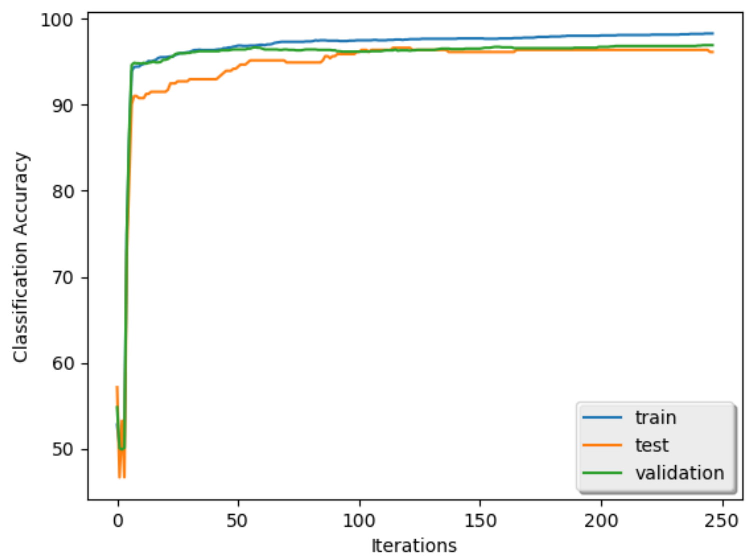

Figure 3: Plot of accuracy in 2 vs 8.

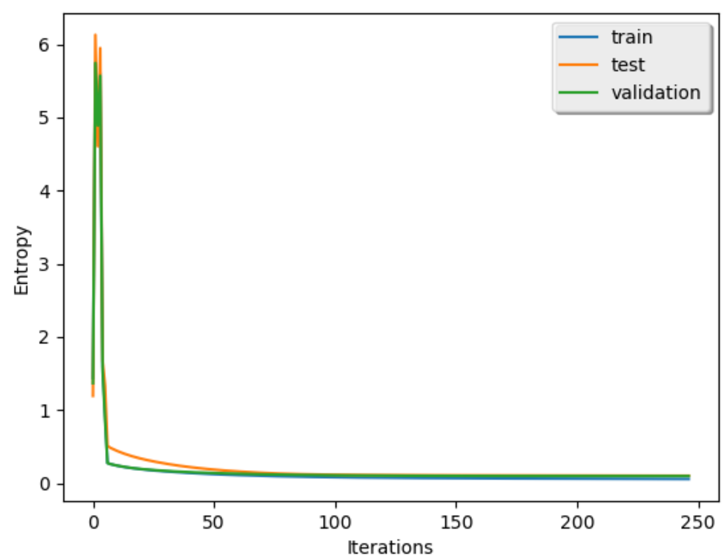

Figure 4: Plot of losses in 2 vs 8.

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.

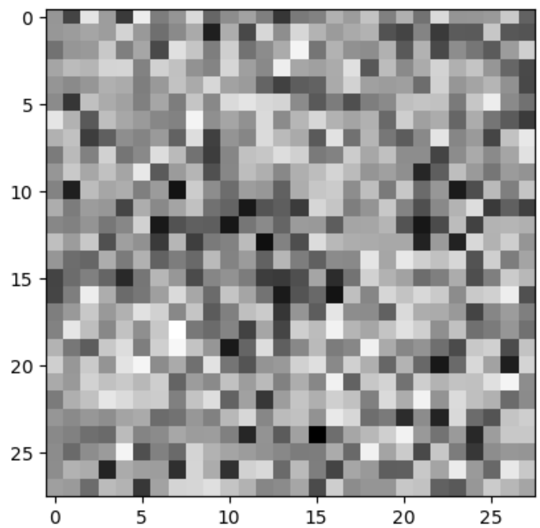

Figure 5: Sample figure caption.

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

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, section 3.2 contains method used to solve the problem. Results ans Discussion is done in section 3.3 and 3.4 respectively.

5.1 Introduction

In this part of the problem, we have created a multi-class classifier which classifies a data point into 10 different classes.

5.2 Method

5.3 Results and Discussion

5.4 Tables

All tables must be centered, neat, clean and legible. Do not use hand-drawn tables. The table number and title always appear before the table. See Table **??**.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

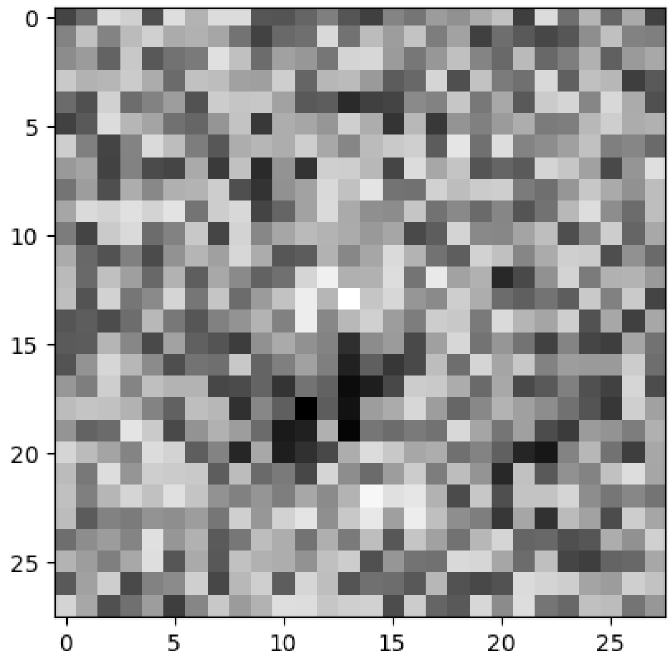

Figure 6: Sample figure caption.

Table 1: Sample table title

PART	DESCRIPTION
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

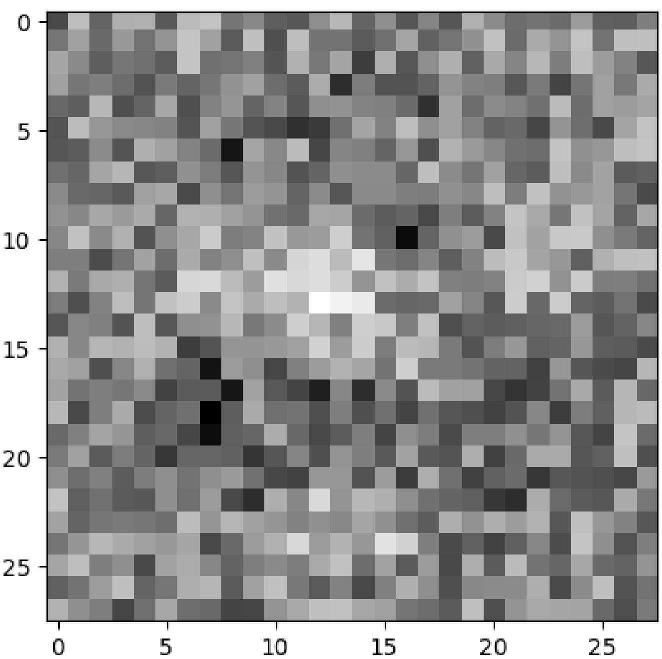

Figure 7: Sample figure caption.