CS 725: Foundations of Machine Learning (Autumn 2023)— Homework 1 Exploring Logistic Regression and Linear Classification

Anuj Asati 23M0763
Frederic J Maliakkal 23M0745

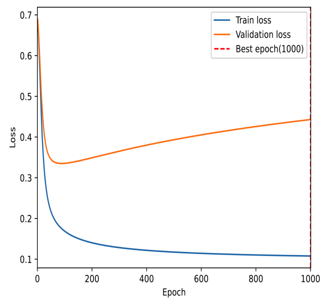
August 2023

Chapter 1

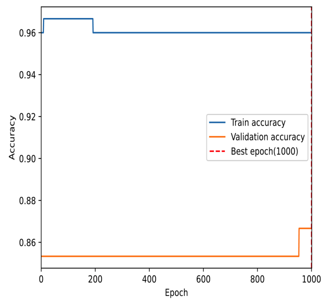
Logistic Regression

1.1 Learning Rate

When the learning rate is small (not too small), like 10^{-1} and 10^{-2}, the accuracy is high and our model performs decently well, but when the learning rate is too small, i.e., 10^{-4} and 10^{-6}, then the accuracy is low and the model performs bad.



(a) Loss



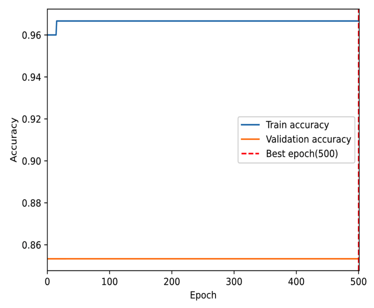
(b) Accuracy

Figure 1.1: Learning Rate

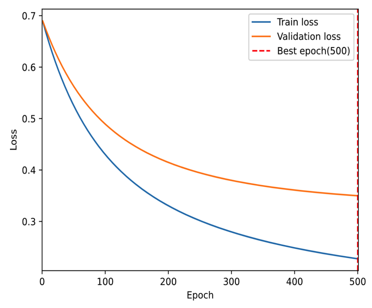
As we can see in the figure that, for the loss, with the changes in epochs, the validation loss initially goes down and then subsequently went up and then converges at 0.42 at the end of 1000 epochs. The training loss on the other hand, just went down and it converges at 0.10 giving us a very less training loss as compared to the validation loss for the 1000 epochs.

1.2 Number of Epochs

When the number of epochs is very low like '100' epochs then the accuracy is low. By low we mean that the model does not get enough iterations to converge



(a) Accuracy at 500 epochs

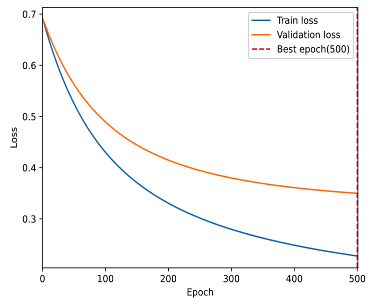


(b) Loss at 500 epochs

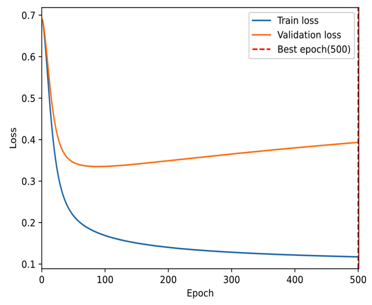
Figure 1.2: Number of Epochs

to a minima. As the number of epochs increases the model starts to perform better. At '250' epochs the model performs much better compared to '100' epochs. At '500' epochs the model gives the best accuracy and performs best as compared to the lower iteration values and then after that the epochs count does not have much significant changes, like in '1000' epochs, it performs similar to the model at '500' epochs.

1.3 Momentum



(a) Momentum: 0



(b) Momentum: 0.9

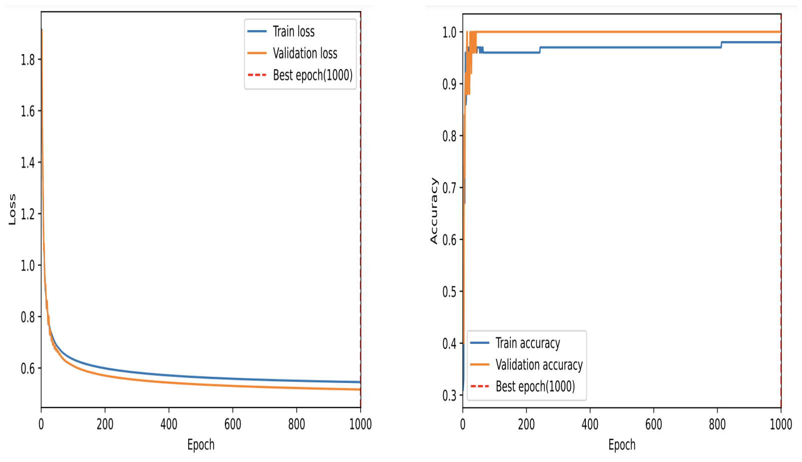
Figure 1.3: Momentum

We have worked with two values of momentum here, that is, 0 and 0.9. With the momentum set to 0, we reach the high accuracy with more number of epochs as compared when momentum is set to 0.9. This help in saving the computational work of machine and we save much power for doing the same work.

Chapter 2

Linear Classifier

2.1 Learning Rate



(a) Loss (b) Accuracy

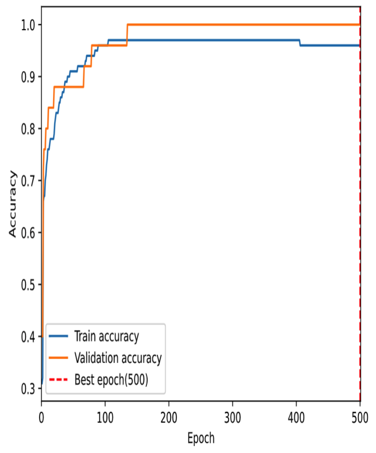
Figure 2.1: Learning Rate

When the learning rate is small (not too small) say 10^{-1} , the accuracy is high and our model performs decently well, but when the learning rate is too small, i.e., 10^{-4} and 10^{-6}, then the accuracy is low and the model performs bad.

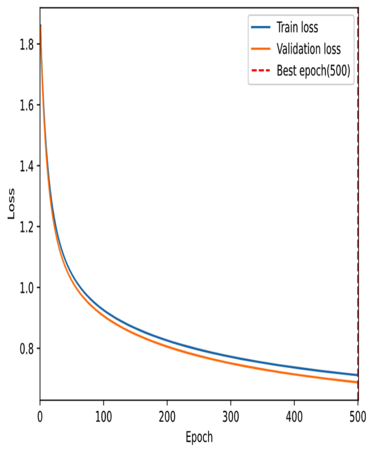
The number of epochs needed to reach the best accuracy also increases as we decrease the learning rate from 10^{-1} to lower values like 10^{-4} or 10^{-6}. Even though we are able to receive the perfect accuracy of 100 with lower (too small) learning rates, the loss was minimum for the small but not too small learning rate which we found to be 10^{-1}.

As we see in the figure 2.1 the best loss and accuracy is received when we use the learning rate 10^{-1}. We are able to achieve a prefect accuracy on the validation data using the learning rates 10^{-1} and 10^{-2}.

2.2 Number of Epochs



(a) Accuracy at 500 epochs



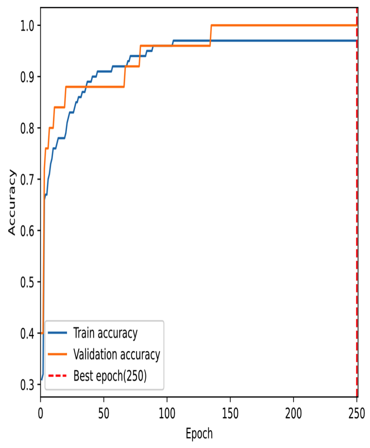
(b) Loss at 500 epochs

Figure 2.2: Number of Epochs

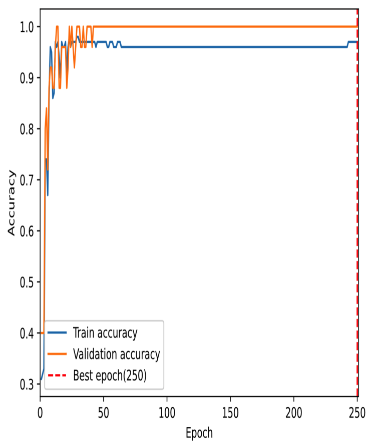
When the learning rate is optimal we are able to achieve the prefect 100 accuracy even with very less number of epochs say 30 as it is evident from the figure 2.1(b). Even though the accuracy is high we are able to reduce our loss when we add more number of epochs as evident from the figure 2.1(a). As the number of epochs increases the loss is gradually reducing which leads to better accuracy.

From our homework assignment we found out that even if the learning rate is small the weights are approaching the minima but with less speed which in turn wants us to increase the number of epochs which leads to wasting lots of machine power.

2.3 Momentum



(a) Momentum: 0



(b) Momentum: 0.9

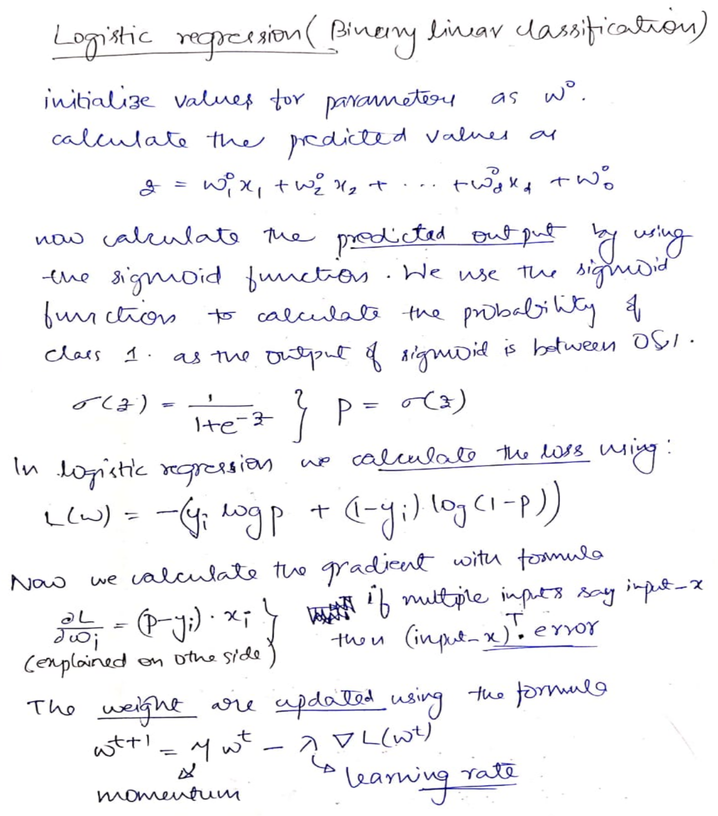
Figure 2.3: Momentum

If we use momentum as 0 it takes more number of epochs to reach better accuracy. With the introduction of 0.9 momentum the system reaches higher accuracy with less number of epochs.

The momentum factor allows us to reach the minima much faster which results in less wastage of machine power.

Chapter 3

Gradient derivation



Logistic regression(Binary linear classification)

initialize values for parameters as w^0.
calculate the predicted values as

$$z = w_1^0 x_1 + w_2^0 x_2 + \cdots + w_d^0 x_d + w_0^0$$

now calculate the predicted output by using the sigmoid function. We use the sigmoid function to calculate the probability of class 1. as the output of sigmoid is between 0&1.

$$\left.\sigma(z) = \frac{1}{1+e^{-z}}\right\} \; p = \sigma(z)$$

In logistic regression we calculate the loss using:

$$L(w) = -\left(y_i \log p + (1-y_i) \log(1-p)\right)$$

Now we calculate the gradient with formula

$$\frac{\partial L}{\partial w_i} = (p-y_i) \cdot x_i$$

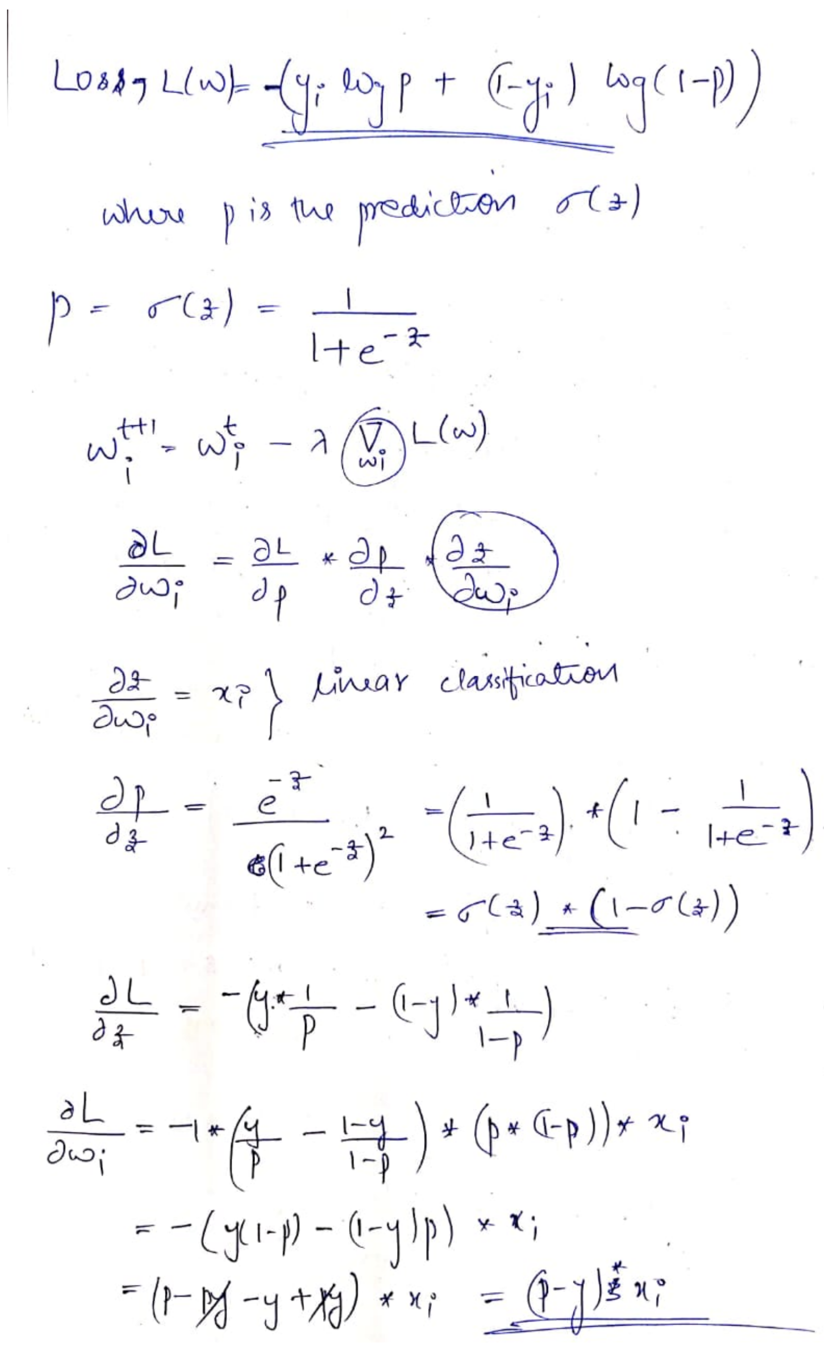
(explained on other side)

~~[illegible]~~ if multiple inputs say input_x then $(\text{input\_x})^T \cdot \text{error}$

The weight are updated using the formula

$$w^{t+1} = \underset{\text{momentum}}{\gamma}\, w^t - \underset{\text{learning rate}}{\lambda}\, \nabla L(w^t)$$

Figure 3.1: Loss, Gradient, Update weight, Sigmoid



Loss, $L(w) = \underline{-(y_i \log p + (1-y_i)} \log(1-p))$

where p is the prediction $\sigma(z)$

$$p = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$w_i^{t+1} = w_i^t - \lambda \boxed{\nabla_{w_i}} L(w)$$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial p} * \frac{\partial p}{\partial z} * \boxed{\frac{\partial z}{\partial w_i}}$$

$$\left.\frac{\partial z}{\partial w_i} = x_i \right\} \text{linear classification}$$

$$\frac{\partial p}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right) * \left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= \sigma(z) \underline{* (1} - \sigma(z))$$

$$\frac{\partial L}{\partial z} = -\left(y * \frac{1}{p} - (1-y) * \frac{1}{1-p}\right)$$

$$\frac{\partial L}{\partial w_i} = -1 * \left(\frac{y}{p} - \frac{1-y}{1-p}\right) * (p * (1-p)) * x_i$$

$$= -(y(1-p) - (1-y)p) * x_i$$

$$= (p - \cancel{py} - y + \cancel{py}) * x_i = \underline{(p-y) * x_i}$$

Figure 3.2: Gradient derivation

Chapter 4

Study on Iris

We used 1-vs-all strategy for linear multi class classification. In this strategy we consider individual classes as separate logistic regression problems and find the weights for separate classes.

In the given problem there were 3 classes. The given true label ranges from the values [0,2]. From this true label we construct 3 vectors one corresponding to every class and we mark the corresponding label 1 if it matches with the true label value or else we mark it as 0. Using this newly constructed vectors we reduce the linear classification for multi class into a binary linear classification problem and carry onto update weights using logistic regression technique.

We consider one class as winner of for a particular input if its predicted probability is higher than other classes and above a certain threshold.