CS725 - Foundations Of Machine Learning

Homework - 1

Pranab Kumar Paul . Debdoot Roy Chowdhury

Contents

1 Logistic Regression

Logistic regression is a statistical model used for binary classification, to predict a binary outcome (0 or 1, Yes or No, True or False) based on one or more predictor variables. It's a commonly used technique in the field of machine learning and statistics, particularly for problems where you need to determine the probability of an event happening.

The logistic regression model is implemented in this homework uses the sigmoid function to map the linear combination of input features and weights to a probability between 0 and 1. This model also uses the cross-entropy function to calculate the validation set's loss. We have studied the model by tweaking different hyperparameters to optimize the model's accuracy for the validation set.

1.1 Learning Rate

A learning rate is a hyperparameter used in machine learning and optimization algorithms to control the step size at which a model's parameters (e.g., weights) are updated during training.

The primary goal of the majority of gradient descent-based optimisation techniques used in machine learning algorithms is to reduce the loss function by repeatedly altering model parameters or weights. One such hyperparameter, Learning Rate, controls how far we move in the direction of the negative gradient. The following parameters are updated with the aid of the learning rate:

$$w = w - \lambda \Delta L(w) \tag{1}$$

where w is the parameters i.e., weights, λ is the learning rate and $\Delta L(w)$ is the gradient of the loss function.

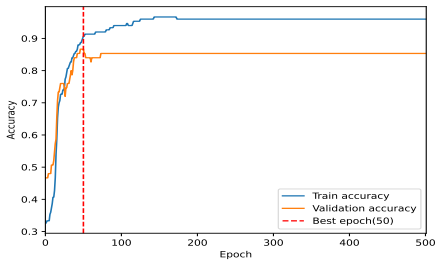
The observed effect of the learning rate on the loss and accuracy in our model is shown below:

1.1.1 Data

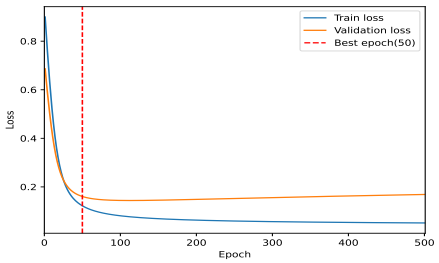
Learning Rate	**Train Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
1e-1	0.05	90.66	0.17	86.67
1e-2	0.09	93.33	0.16	85.33
1e-4	0.23	66.66	0.24	68.00
1e-6	0.41	43.33	0.35	53.33

Table 1: Table showing the observations for the model when epoch = 500 and momentum = 0.

1.1.2 Visualization

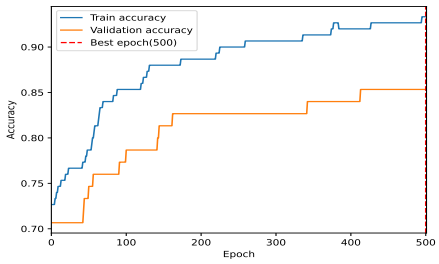


(a) Accuracy Plot

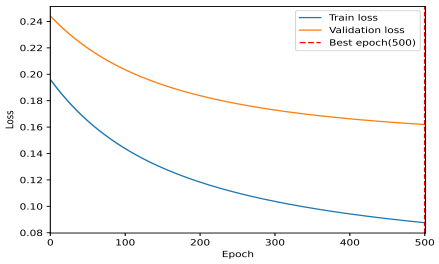


(b) Loss Plot

Figure 1: Learning Rate = 10^{-1}

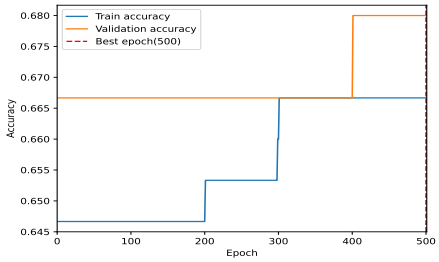


(a) Accuracy Plot

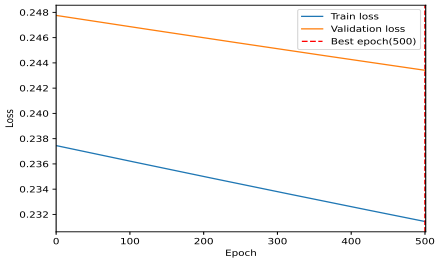


(b) Loss Plot

Figure 2: Learning Rate = 10^{-2}

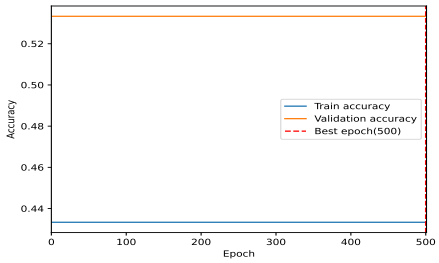


(a) Accuracy Plot

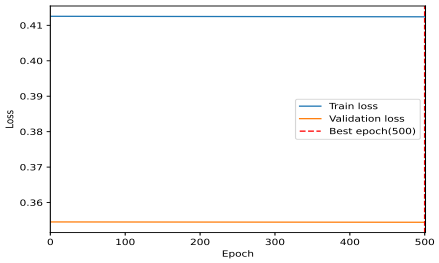


(b) Loss Plot

Figure 3: Learning Rate = 10^{-4}



(a) Accuracy Plot



(b) Loss Plot

Figure 4: Learning Rate = 10^{-6}

1.1.3 Observation

As we can observe from the above figures where we considered the number of epochs to be constant at 500 and the momentum to be constant at 0, the higher the learning rate is, more is the validation accuracy [Fig. 1(a)]. This is because the model will converge fast if the learning rate is high [Fig. 1(b)], while for lower learning rates (like 10^{-4}) the convergence is much slow [Fig. 3(b)], hence providing lower accuracy as it needs more number of epochs to converge and give better accuracy [Fig. 3(a)].

From the loss plots given above, we can analyse that when the learning rate is 10^{-1} (Fig. 1(b))the convergence is way faster than the rest loss plots [Fig. 2(b) - Fig. 4(b)] because of the higher learning rate. But, if we increase the number of epochs, then we can get a better accuracy even for lower learning rates because the model will have more number of ephocs to converge.

In conclusion, when the learning rate is low, more epochs are required for the model to converge for giving better accuracy.

1.2 Number of Epochs

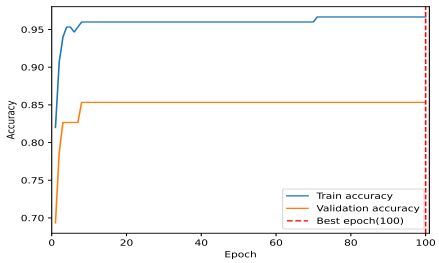
In machine learning lingo Epochs represents the no. of times the entire training dataset is processed by a learning algorithm. It's an essential hyperparameter in training of machine learning models. During each epoch, the model updates its parameters based on the training data to minimize a specific loss or error metric. This repetitive process helps the model learn patterns and relationships in the data.

1.2.1 Data

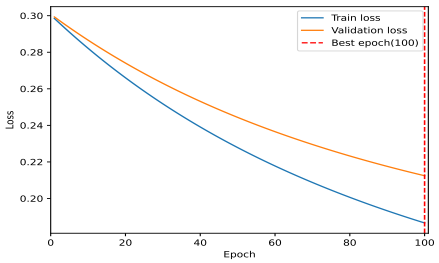
No. of Epochs	**Train Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
100	0.19	96.66	0.21	85.33
250	0.13	96.66	0.17	86.67
500	0.10	95.33	0.15	88.00
1000	0.08	92.66	0.15	89.33

Table 2: Table showing the observations for the model when Learning Rate = 10^{-2} and momentum = 0

1.2.2 Visualization

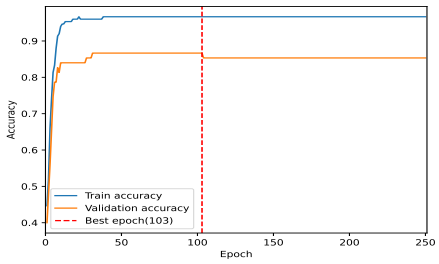


(a) Accuracy Plot

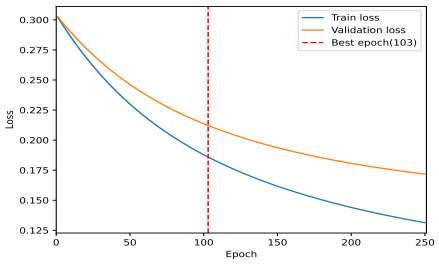


(b) Loss Plot

Figure 5: Number of Epochs = 100

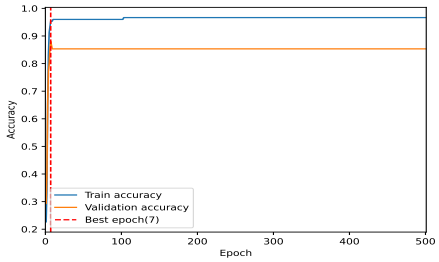


(a) Accuracy Plot

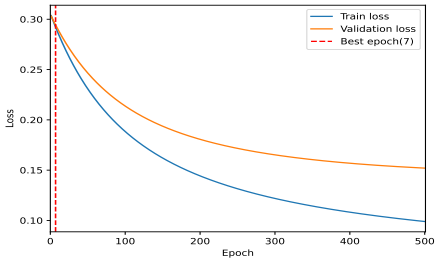


(b) Loss Plot

Figure 6: Number of Epochs = 250

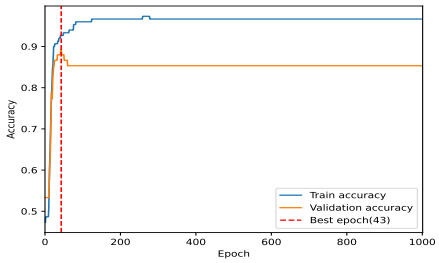


(a) Accuracy Plot

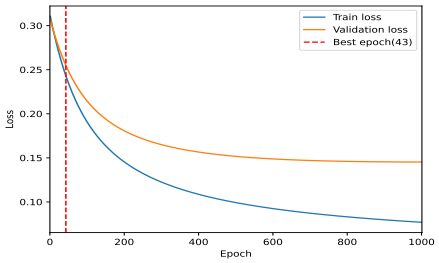


(b) Loss Plot

Figure 7: Number of Epochs = 500



(a) Accuracy Plot



(b) Loss Plot

Figure 8: Number of Epochs = 1000

1.2.3 Observation

We can observe from the above figures where we considered the learning rate to be constant at 10^-2 and momentum to be constant at 0 in our experiment, the more the number of epochs, the higher the validation accuracy and the lower the validation loss. This is because, as we are increasing the number of epochs we letting the model more time to converge. That's why as the number of epochs are increasing the model gets ample amount of time to train itself and gives a better loss and accuracy for the validation data.

From the accuracy and loss plots, [Fig. 8(a) & 8(b)] where the number of epochs is 1000, the model is converged better than the rest three of epochs we have considered above. When number of epochs is 100 [Fig. 5(a) & 5(b)], the model can be seen as not at all converged, thus giving a greater loss and less accuracy. Whereas, we increase the epoch to 250 [Fig. 6(a) & 6(b)] model is slightly better converged than the previous one, but still needs more epochs to converge, and thus it is also not enjoying a better validation loss and accuracy. As we increase the epoch to 500 [Fig. 7(a) & Fig. 7(b)] and 1000 [Fig. 8(a) & 8(b)] the model almost gets converged and thus provides a very less validation loss and enjoys a good accuracy than the previous ones.

In conclusion, more number of epochs gives the model time to converge to an optimal set of model parameters that minimize the chosen loss function and thus provide better accuracy. But, a very high number of epochs can also make the model overfit and reduce the accuracy of the validation set.

1.3 Momentum

For updating the weight parameters, we are performing the gradient descent using the equation $w = w - \lambda \Delta L(w)$ where λ is the chosen learning rate and w is the weight parameters. A modification to this update rule called "momentum update" enjoys better convergence rate. This requires maintaining an extra set of parameters called v where v is initialized to zero at the same time w is initialized. Then the actual update is written as follows:

$$v = \mu v - \lambda \Delta L(w) \tag{2}$$

$$w = w + v \tag{3}$$

Where μ is the additional parameter called momentum.

1.3.1 Data

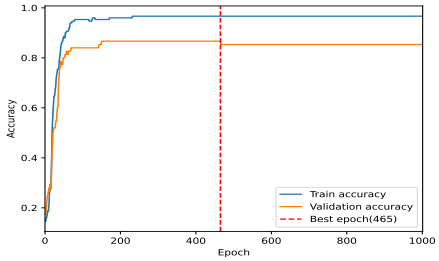
Momentum	**Tain Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
0	0.08	96.66	0.15	85.33
0.9	0.05	97.33	0.19	88.00

Table 3: Table showing the observations for the model when Learning Rate = 10^{-2} and No. of Epochs = 1000

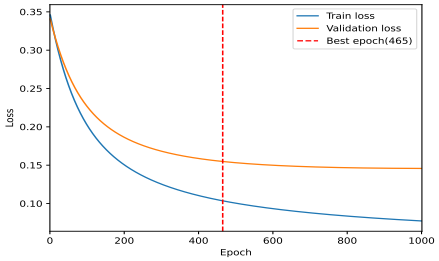
Momentum	**Tain Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
0	0.26	95.33	0.27	84.00
0.9	0.19	96.66	0.22	88.00

Table 4: Table showing the observations for the model when Learning Rate = 10^{-4} and No. of Epochs = 1000

1.3.2 Visualization

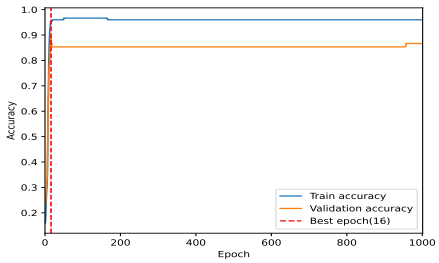


(a) Accuracy Plot

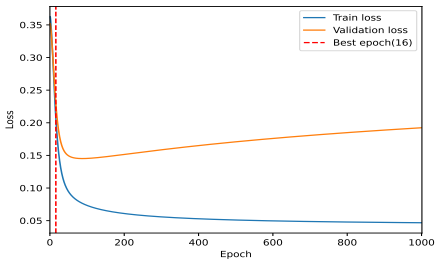


(b) Loss Plot

Figure 9: Momentum = 0 & Learning Rate = 10^{-2}

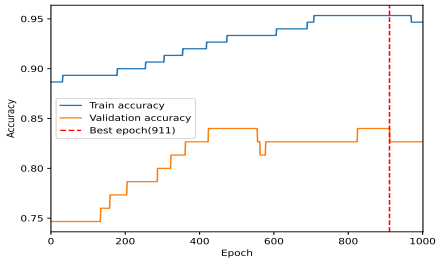


(a) Accuracy Plot

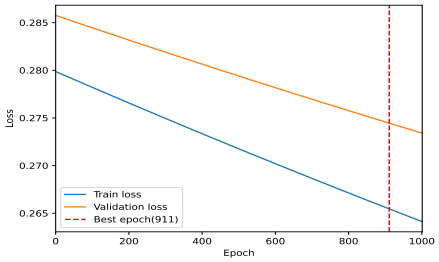


(b) Loss Plot

Figure 10: Momentum = 0.9 & Learning Rate = 10^{-2}

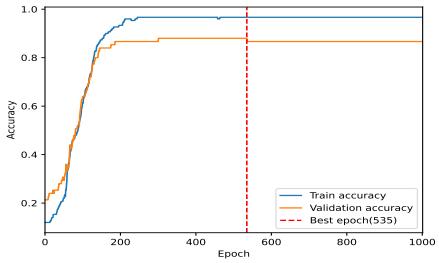


(a) Accuracy Plot

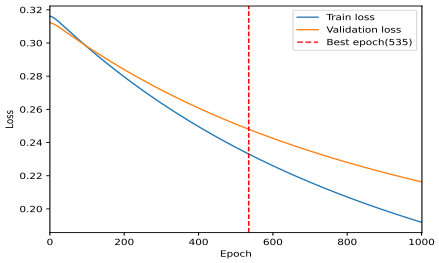


(b) Loss Plot

Figure 11: Momentum = 0 & Learning Rate = 10^{-4}



(a) Accuracy Plot



(b) Loss Plot

Figure 12: Momentum = 0.9 & Learning Rate = 10^{-4}

1.3.3 Observation

We can observe from the above figures where we considered the number of epochs as constant at 1000, we took two cases where two different learning rates are used (10^{-2}&10^{-4}) for observing the change in loss and accuracy due to momentum. In Fig. 9 and Fig. 10 where we set the learning rate to be 10^{-2}, it can be seen that when momentum (μ) is 0, the model achieved its expected accuracy and loss gradually at a slower pace in comparison to the case when the μ is set to 0.9. This is due to the case as momentum helps accelerate the convergence of the model. In Fig. 11 and 12, where we decreased the learning rate to 10^{-4}, it can be noticed that the model converges much slower than the previous case. In conclusion, increasing momentum leads to accelerating of the convergence of the model. Thus, a decrease in momentum produces lower accuracy and high loss value.

1.4 Mathematical Derivation of Gradient Descent

Gradient Descent is an optimization technique which is used to minimize the loss function. This optimization technique is used on the implementation of this logistic regression model. To perform gradient descent we need to calculate the gradient of the loss function w.r.t the model's parameter W. The gradient is a vector that contains the partial derivatives of the loss w.r.t each parameter. The loss function measures the dissimilarity between the predicted probabilities and the actual binary levels. It is defined as follows:

$$L(W) = -\frac{1}{m}\left(Y log\hat{Y} + (1-Y)\, log\left(1-\hat{Y}\right)\right) \tag{4}$$

where m is the no. of data points, Y is the actual output level and $\hat{Y}$ is the predicted output level by the model. Here the $\hat{Y}$ is defined as follows:

$$\hat{Y} = \sigma(Z) = \frac{1}{1+exp(-Z)} \quad \& \quad \frac{\partial}{\partial Z}(\sigma(Z)) = \sigma(Z)(1-\sigma(Z)) \tag{5}$$

where σ is the sigmoid function that is used to squeeze the predicted output value between 0 and 1. Here Z is defined as $(W^T X)$ where X is the input dataset and W is the trainable model parameters that is updated according to the gradient $\frac{\partial L}{\partial W}$. Hence the derivation of the gradient i.e., $G = \Delta L(W) = \frac{\Delta L}{\Delta W}$ is shown below:

$$\begin{aligned}
G = \frac{\Delta L}{\Delta W} = \frac{\partial L}{\partial W} &= \frac{\partial}{\partial W}\left[-\frac{1}{m}\left(Y\log\hat{Y} + (1-Y)\log\left(1-\hat{Y}\right)\right)\right] \\
&= -\frac{1}{m}\left[\frac{\partial}{\partial W}\left(Y\log\hat{Y}\right) + \frac{\partial}{\partial W}(1-Y)\log\left(1-\hat{Y}\right)\right] \\
&= -\frac{1}{m}\left[\frac{\partial}{\partial W}\left(Y\log\left(\sigma\left(WX\right)\right)\right) + \frac{\partial}{\partial W}\left((1-Y)\log\left(1-\sigma\left(WX\right)\right)\right)\right] \\
&= -\frac{1}{m}\left[\frac{Y}{\hat{Y}}\frac{\partial}{\partial W}\left(\sigma\left(WX\right)\right) + \frac{(1-Y)}{(1-\hat{Y})}\frac{\partial}{\partial W}\left(1-\sigma(WX)\right)\right] \\
&= -\frac{1}{m}\left[\frac{Y}{\hat{Y}}\sigma(WX)(1-\sigma(WX))\frac{\partial}{\partial W}(WX) + \frac{(1-Y)}{(1-\hat{Y})}\frac{\partial}{\partial W}\sigma(WX)\right] \\
&= -\frac{1}{m}\left[\frac{Y}{\hat{Y}}\hat{Y}(1-\hat{Y})X - \frac{(1-Y)}{(1-\hat{Y})}\frac{\partial}{\partial W}\sigma(WX)\right]
\end{aligned}$$

$$= -\frac{1}{m}\left[Y(1-\hat{Y})X - \frac{(1-Y)}{(1-\hat{Y})}\sigma(WX)(1-\sigma(WX))\frac{\partial}{\partial W}(WX)\right]$$
$$= -\frac{1}{m}\left[Y(1-\hat{Y})X - \frac{(1-Y)}{(1-\hat{Y})}\hat{Y}(1-\hat{Y})X\right]$$
$$= -\frac{1}{m}\left[Y(1-\hat{Y}) - (1-Y)\hat{Y}\right]X$$
$$= -\frac{1}{m}\left[Y - Y\hat{Y} - \hat{Y} + Y\hat{Y}\right]X$$
$$G = -\frac{1}{m}\left[Y - \hat{Y}\right]X$$

Where G is the gradient of the loss function, $\hat{Y}$ is the probability of the class labels and Y is the actual predicted class labels.

2 Linear Classification

Linear classification in machine learning is a fundamental technique for solving binary and multi-class classification problems. It aims to find a linear decision boundary that separates data points of different classes. The key idea is to find a hyperplane (a subspace of one less dimension than the input space) that best separates the data points of different classes in the feature space.

Here, we implemented a multi-class linear classification method using Softmax regression technique to predict the outcome of the IRIS dataset.

2.1 Learning Rate

A learning rate is a hyperparameter used in machine learning and optimization algorithms to control the step size at which a model's parameters (e.g., weights) are updated during training.

By iteratively altering model parameters, such as weights, the main goal of the majority of gradient descent-based optimisation techniques used in machine learning algorithms is to minimise the loss function. The magnitude of the step we take in the direction of the negative gradient is determined by one of the hyperparameters known as Learning Rate. The following parameters are updated with the aid of the learning rate:

$$w = w - \lambda \Delta L(w) \tag{6}$$

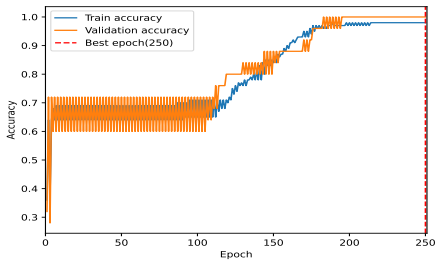
where w is the parameters i.e., weights, λ is the learning rate and $\Delta L(w)$ is the gradient of the loss function. The observed effect of the learning rate on the loss and accuracy in our model is shown below:

2.1.1 Data

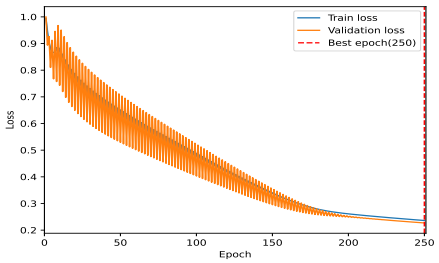
Learning Rate	**Train Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
1e-1	0.24	98.00	0.23	100.00
1e-2	0.54	93.00	0.53	92.00
1e-4	1.08	64.00	1.08	72.00
1e-6	1.10	31.33	1.09	40.00

Table 5: Table showing the observations for the model when epoch = 250 and momentum = 0.

2.1.2 Visualization

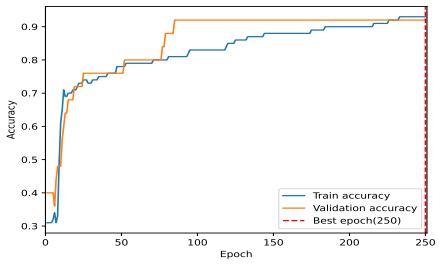


(a) Accuracy Plot

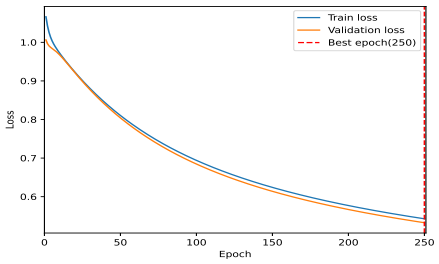


(b) Loss Plot

Figure 13: Learning Rate = 10^{-1}

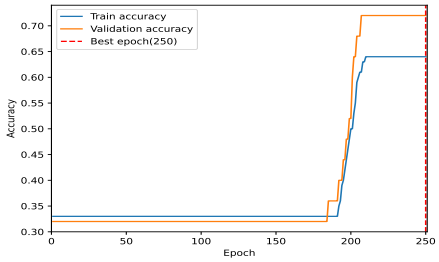


(a) Accuracy Plot

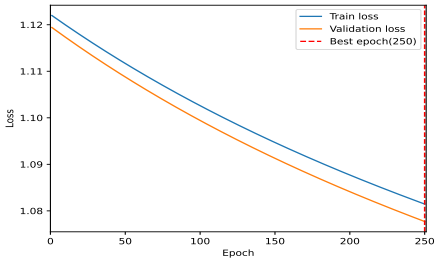


(b) Loss Plot

Figure 14: Learning Rate = 10^{-2}

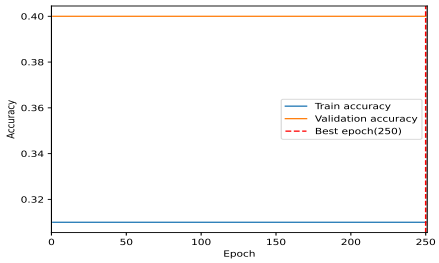


(a) Accuracy Plot

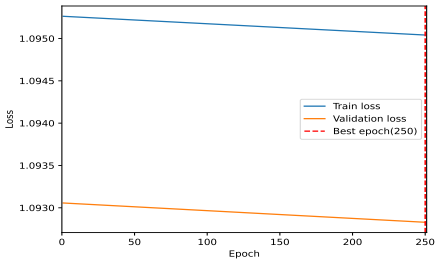


(b) Loss Plot

Figure 15: Learning Rate = 10^{-4}



(a) Accuracy Plot



(b) Loss Plot

Figure 16: Learning Rate = 10^{-6}

2.1.3 Observation

The figures above, where we assumed the number of epochs to be constant at 250 and the momentum to be constant at 0, show that the validation accuracy increases with increasing learning rate [Fig. 13(a)]. This is due to the fact that the model will quickly converge if the learning rate is large [Fig. 13(b)], however for lower learning rates (say 10^{-4}), the convergence is considerably slower [Fig. 15(b)], resulting in poorer accuracy as it requires a greater number of epochs to converge and provide improved accuracy [Fig. 15(a)].

We can deduce from the loss plots presented above that, due to the greater learning rate, the convergence occurs much more quickly when the learning rate is 10^{-1} [Fig. 13(b)], than it does for the other loss plots [Fig. 14(b) to Fig. 16(b)]. The model will have more scope to converge if the number of epochs are increased, though this will result in lower learning rates and greater accuracy. This is similar to the result we have seen earlier in the logistic regression.

From [Fig. 13], we can see that the loss and accuracy is oscillating in the graph. This indicates that the learning rate is too high and the model is not converging efficiently. Actually a higher learning rate causes the optimization process to overshoot and the model keeps moving back and forth.

In conclusion, more epochs are needed for the model to converge for improved accuracy when the learning rate is low.

2.2 Number of Epochs

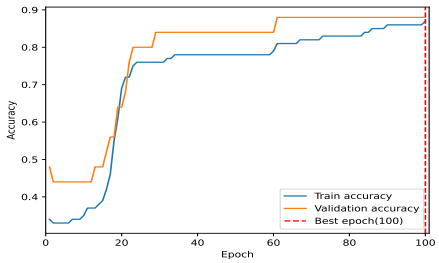
In the language of machine learning the number of epochs indicates how frequently a learning algorithm has gone over the full training dataset. This hyperparameter is crucial for training machine learning models. A specified loss or error measure is minimised throughout each epoch by the model, which modifies its parameters depending on training data. The model can discover patterns and connections in the input data.

2.2.1 Data

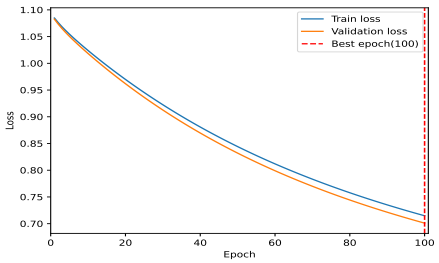
No. of Epochs	**Train Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
100	0.71	87.00	0.70	88.00
250	0.55	97.00	0.54	96.00
500	0.45	97.00	0.44	96.00
1000	0.36	98.00	0.35	100.00

Table 6: Table showing the observations for the model when Learning Rate = 10^{-2} and Momentum = 0

2.2.2 Visualization

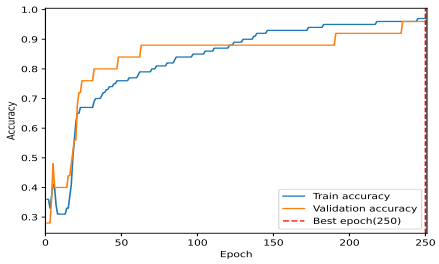


(a) Accuracy Plot

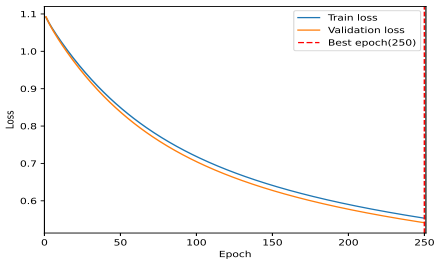


(b) Loss Plot

Figure 17: Number of Epochs = 100

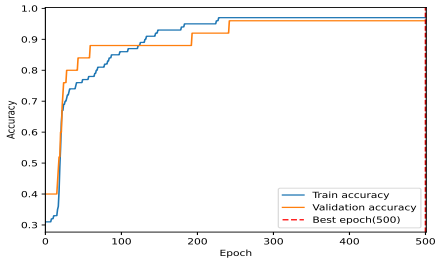


(a) Accuracy Plot

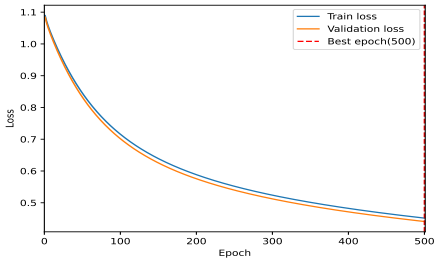


(b) Loss Plot

Figure 18: Number of Epochs = 250

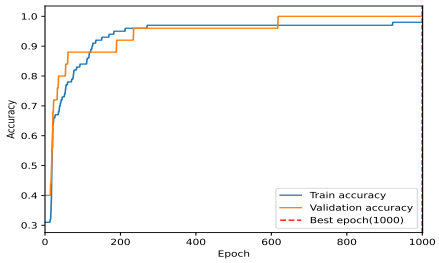


(a) Accuracy Plot

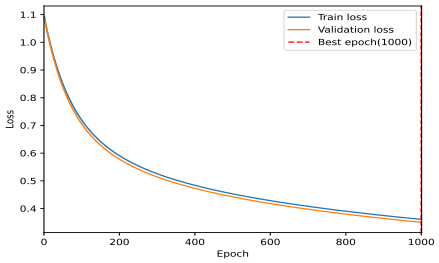


(b) Loss Plot

Figure 19: Number of Epochs = 500



(a) Accuracy Plot



(b) Loss Plot

Figure 20: Number of Epochs = 1000

2.2.3 Observation

We can observe from the above figures where we considered the learning rate to be constant at 10^{-2} and momentum to be constant at 0 in our experiment, the more the number of epochs, the higher the validation accuracy and the lower the validation loss. This is because, as we are increasing the number of epochs we letting the model more time to converge. That's why as the number of epochs are increasing the model gets ample amount of time to train itself and gives a better loss and accuracy for the validation data.

From the accuracy and loss plots, [Fig. 20(a) & 20(b)] where the number of epochs is 1000, the model converged better than the rest of the cases of epochs we have considered above. When number of epochs is 100 [Fig. 17(a) & 17(b)], the model can be seen as not at all converged, thus giving a greater loss and less accuracy. Whereas, we increase the epoch to 250 [Fig. 18(a) & 18(b)] model is slightly better converged than the previous one, but still needs more epochs to converge, and thus it is also not enjoying a better validation loss and accuracy. As we increase the epoch to 500 [Fig. 19(a) & Fig. 19(b)] and 1000 [Fig. 20(a) & 20(b)] the model almost gets converged and thus provides a very less validation loss and enjoys a good accuracy than the previous ones.

In conclusion, the model may not have enough iterations if the number of epochs is too low for it to learn any significant patterns from the data. Underfitting is a condition where the model performs poorly on both the training and validation datasets. If there are too many epochs, the model may begin to memorise the training data rather than drawing generalisations from it. Overfitting is a situation in which a model performs well on training data but badly on unobserved validation or test data.

2.3 Momentum

For updating the weight parameters, we are performing the gradient descent using the equation $w = w - \lambda \Delta L(w)$ where λ is the chosen learning rate and w is the weight parameters. A modification to this update rule called "momentum update" enjoys better convergence rate. This requires maintaining an extra set of parameters called v where v is initialized to zero at the same time w is initialized. Then the actual update is written as follows:

$$v = \mu v - \lambda \Delta L(w) \tag{7}$$

$$w = w + v \tag{8}$$

Where μ is the additional parameter called momentum.

2.3.1 Data

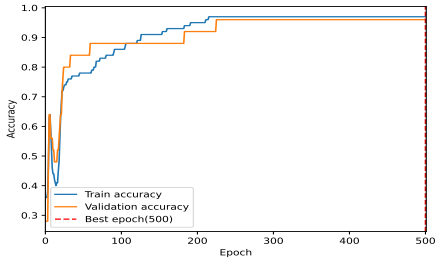
Momentum	**Tain Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
0	0.45	97.00	0.44	96.00
0.9	0.17	97.00	0.16	100.00

Table 7: Table showing the observations for the model when Learning Rate = 10^{-2} and No. of Epochs = 500

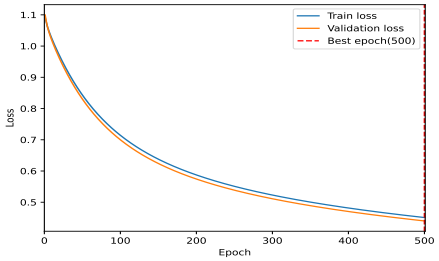
Momentum	**Tain Loss**	**Train Accuracy(%)**	**Validation Loss**	**Validation Accuracy(%)**
0	1.05	53.00	1.05	64.00
0.9	0.85	77.00	0.83	84.00

Table 8: Table showing the observations for the model when Learning Rate = 10^{-4} and No. of Epochs = 1000

2.3.2 Visualization

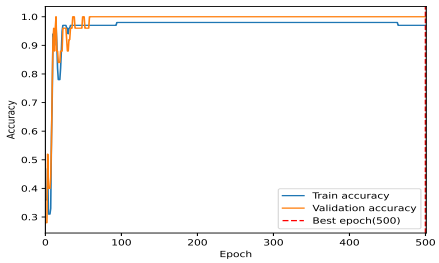


(a) Accuracy Plot

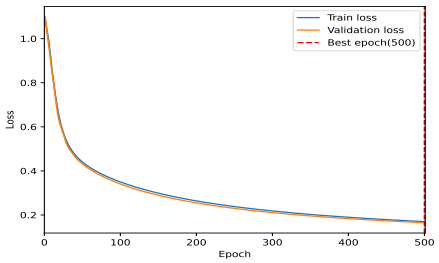


(b) Loss Plot

Figure 21: Momentum = 0 & Learning Rate = 10^{-2}

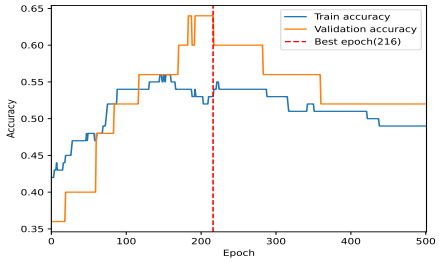


(a) Accuracy Plot

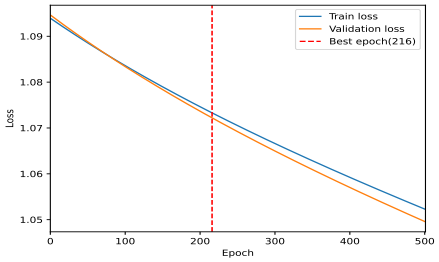


(b) Loss Plot

Figure 22: Momentum = 0.9 & Learning Rate = 10^{-2}

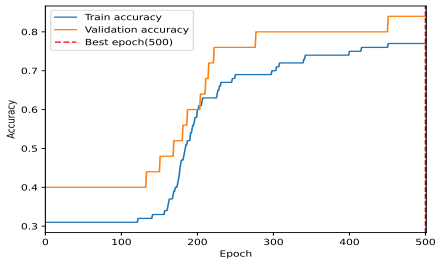


(a) Accuracy Plot

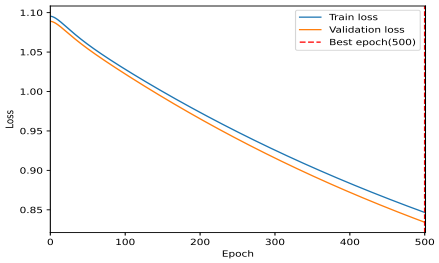


(b) Loss Plot

Figure 23: Momentum = 0 & Learning Rate = 10^{-4}



(a) Accuracy Plot



(b) Loss Plot

Figure 24: Momentum = 0.9 & Learning Rate = 10^{-4}

2.3.3 Observation

We can observe from the above figures where we considered the number of epochs as constant at 500, we took two cases where two different learning rates are used (10^{-2}&10^{-4}) for observing the change in loss and accuracy due to momentum. In Fig. 21 and Fig. 22 where we set the learning rate to be 10^{-2}, it can be seen that when momentum (μ) is 0, the model achieved its expected accuracy and loss gradually at a slower pace in comparison to the case when the μ is set to 0.9. This is due to the case as momentum helps accelerate the convergence of the model. In Fig. 11 and 12, where we decreased the learning rate to 10^{-4}, it can be noticed that the model converges much slower than in the previous case. In conclusion, increasing momentum leads to accelerating the convergence of the model. Thus, a decrease in momentum produces lower accuracy and a high loss value.

2.4 Study on Iris Dataset

The iris dataset is contained with 100 data points in the training set and 25 data points in the validation set. The features (attributes) included in the Iris dataset are:

- Sepal Length: Length of the sepal in centimetres.
- Sepal Width: Width of the sepal in centimetres.
- Petal Length: Length of the petal in centimetres.
- Petal Width: Width of the petal in centimetres.

The dataset involves multi-class classification, making it suitable for testing and understanding algorithms that work with multiple classes. The Iris dataset is visualised as below:

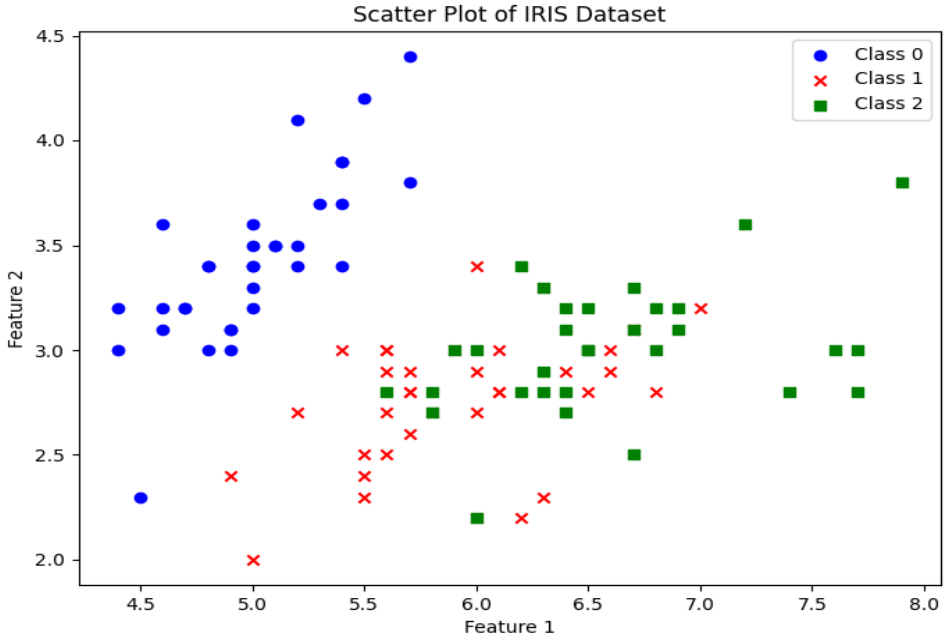


Figure 25: Iris dataset

2.5 Logistic Regression for Multi-Class

Logistic regression can be adapted for multiclass classification using various methods such as:

- **One-Vs-Rest(OvR):** In this approach for each class we will train a separate binary logistic regression classifier which will output 1 for one respective class and 0 for the rest of the classes. So for k output labels we will need to train k binary logistic model. The class for which the model gives the highest probability is the final predicted class.

- **Softmax Regression:** Softmax Regression is also known as the multinomial logistic regression. It allows us to handle multiple output labels ($Y^i \in \{1, 2, \dots K\}$, where K is the number of classes) as opposed to the binary logistic classification where K=2.

  Given an input X, softmax regression will estimate the probability of that input for a particular class label K using the softmax function σ which will return a probabilistic value between 0 and 1 for each of the class labels. Class label achieving the highest probability using the function σ for X is taken as the predicted label $\hat{Y}$.

  The Softmax Function is defined as below:

$$\sigma(Z)_i = \frac{e^{Z_i}}{\sum_{j=1}^{k} e^{Z_j}} \tag{9}$$

  where k is the no. of class labels, $Z_i = W_i^T X$ where W_i is the weight parameter of the i^{th} class label and X is the features of one input data point. For this multi-class logistic regression the loss function is defined as below:

$$L(W) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} Y_k^{(i)} \log(\hat{Y}_k^{(i)}) \tag{10}$$

  where m is the no. of data points K is the no. of class, $Y_k^{(i)}$ is the actual K^{th} class level value i.e., 0 or 1 for the i^{th} input feature and $\hat{Y}_k^{(i)}$ is the probability for the J^{th} class level value for the i^{th} input. Here we need to minimize the loss function $L(W)$ using Gradient Descent which we implemented here as follows:

$$G = \frac{\partial L}{\partial W} = -\frac{1}{m} X^T (Y - \hat{Y}) \tag{11}$$

  Where G is the gradient of the loss function, Y is the One-Hot-Encoding (OHE) matrix for the class labels and $\hat{Y}$ is the probability of the class labels.