CS 725: Foundations of Machine Learning Homework 1

Soumen Kumar Mondal
23m2157@iitb.ac.in

Naay Balodia
23m2166@iitb.ac.in

August 21, 2023

Abstract

In this assignment, we will implement logistic regression model on a toy datasets for binary classification and linear classification model on a toy datasets for multi class classification. Our key goal in this assignment is to correctly implement these models and analyze the results we obtained.

1 Logistic Regression

1.1 Gradient of Loss Function

The scoring function for logistic regression is typically represented as the linear combination of the input features weighted by the model's learned parameters. Given a set of input features $\bar{x} = (x_1, x_2, \ldots, x_d)$ and the corresponding weights $\bar{w} = (w_1, w_2, \ldots, w_d)$, the scoring function z can be expressed as [2]:

$$z = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_d \cdot x_d \tag{1}$$

Where w_0 is the bias term.

This scoring function computes a linear combination of the input features, incorporating the model's learned weights and a bias term. The output of this scoring function is then used to compute the probability of the positive class in binary logistic regression using the sigmoid function. The sigmoid function is a key component in logistic regression and is used to transform the output of the scoring function into a probability value between 0 and 1. The sigmoid function is defined as:

$$f(\bar{x}; \bar{w}, w_0) = \frac{1}{1 + e^{-(\bar{w} \cdot \bar{x} + w_0)}} \tag{2}$$

In logistic regression, the common loss function used is the log loss or cross-entropy loss. The loss function for a single training example is defined as:

$$L(f(\bar{x}^i; \bar{w}, w_0), y^i) = -y^i \cdot \log(f(\bar{x}^i; \bar{w}, w_0)) - (1 - y^i) \cdot \log(1 - f(\bar{x}^i; \bar{w}, w_0)) \tag{3}$$

Where y^i is the true label of the i^{th} input data point.

Then the average of the total loss over the entire training dataset is given as:

$$J(\bar{w}, w_0) = -\frac{1}{N} \sum_{i=1}^{N} \left[L(f(\bar{x}^i; \bar{w}, w_0), y^i) \right] \tag{4}$$

Where $L(f(\bar{x}^i; \bar{w}, w_0), y^i)$ is substitute from Equation 3. N is the total number of training data points. $J(\bar{w}, w_0)$ is sometimes called the average loss function or cost function.

Three useful results that would be required to derive the gradient of the loss function are given as follows:

$$f'(z) = f(z) \cdot (1 - f(z)) \tag{5}$$

$$\frac{\partial J(\bar{w}, w_0)}{\partial \bar{w}} = \frac{\partial J}{\partial z} \cdot \frac{\partial z}{\partial \bar{w}} \tag{6}$$

$$\frac{\partial J(\bar{w}, w_0)}{\partial w_0} = \frac{\partial J}{\partial z} \cdot \frac{\partial z}{\partial w_0} \tag{7}$$

Where $f(z)$ is the standard sigmoid function defined as per Equation 2 and J is the loss function which is differentiated by following the chain rule.

Using the above three equations, the gradient of the loss function can be derived as follows:

$$J(\bar{w}, w_0) = -\frac{1}{N} \sum_{i=1}^{N} \left[y^i \cdot \log(f(z)) + (1 - y^i) \cdot \log(1 - f(z)) \right] \tag{8}$$

$$\frac{\partial J}{\partial z} = -\frac{1}{N} \sum_{i=1}^{N} \left[y^i \cdot \frac{f'(z)}{f(z)} + (1 - y^i) \cdot \frac{-f'(z)}{1 - f(z)} \right] \tag{9}$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left[y^i \cdot (1 - f(z)) - (1 - y^i) \cdot f(z) \right] \qquad \text{(From Equation 5)} \tag{10}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[f(z) - y^i \right] \tag{11}$$

Applying the chain rule, the gradient [1] can be written as:

$$\frac{\partial J(\bar{w}, w_0)}{\partial \bar{w}} = -\frac{1}{N} \sum_{i=1}^{N} \left[f(\bar{x}^i; \bar{w}, w_0) - y^i \right] \times \left[\bar{x}^i \right] \tag{12}$$

$$\frac{\partial J(\bar{w}, w_0)}{\partial w_0} = -\frac{1}{N} \sum_{i=1}^{N} \left[f(\bar{x}^i; \bar{w}, w_0) - y^i \right] \times [1] \tag{13}$$

Where $z = \bar{w} \cdot \bar{x} + w_0$

1.2 Learning Rate

The learning rate is a crucial hyper-parameter in machine learning algorithms, especially in gradient-based optimization methods like gradient descent. It determines the step size at which the algorithm updates the model's parameters during training. A higher learning rate can lead to faster convergence, but it might also cause overshooting and divergence. Conversely, a lower learning rate can lead to slower convergence, potentially getting stuck in local minima. Finding the right balance is essential for achieving optimal training performance and avoiding convergence issues. Experimenting with different learning rates helps identify the optimal rate that maximizes the model's accuracy and minimizes its loss during training and validation.

The impact of varying learning rates on accuracy and loss was explored for the Logistic Regression (LR) model. The outcomes of these experiments can be summarized as follows:

Training accuracy: Analyzing Figure 1 reveals that the Logistic Regression (LR) model achieves its highest accuracy when the learning rate is set to 10^{-1}. As anticipated, a higher learning rate contributes to swift convergence, requiring fewer than 100 epochs. However, the experiment's scope extends to a higher number of epochs to gauge the impact of a slower learning rate, which demands more epochs for convergence. The figure also underscores that a slower learning rate like 10^{-6} necessitates very large number of epochs for convergence whereas 1000 epochs as shown in the figure are not sufficient. The relationship between learning rate and training performance is not always linear (as evident from the figure), and it's possible to encounter situations where a smaller learning rate (e.g., $10^{-}6$) yields better training accuracy than a larger learning rate (e.g., $10^{-}4$). This phenomenon can be attributed to the concept of the learning rate's impact on optimization.

Training loss: From Figure 2, it is evident that the loss is minimum and achieves convergence in lowest number of epochs when the learning rate is set to 10^{-1}. As expected, a lower learning rate will require more number of epochs to achieve minimum loss.

Validation accuracy: It can be seen from Figure 3 that the validation accuracy is highest when the learning rate is set to 10^{-1}. For lower learning rates, the number of epoch 1000 is insufficient to achieve convergence.

Validation loss: It can be seen from Figure 4 that the validation loss is minimum when the learning rate is set to 10^{-1}. It is also evident that the number of epoch 100 is optimum in terms of loss — increasing from 100 will lead to increase in the loss.

In summary, the learning rate of 10^{-1} is optimal for this LR model as it reaches to convergence in less number of epochs without having significant impact on the accuracy and loss of the LR model. **Therefore, we select learning rate of 10^{-1} as the best learning rate for the Logistic Regression model.**

1.3 Number of Epoch

The number of epochs in machine learning signifies the count of times the entire data set is iterated through during training. The choice of the number of epochs plays a pivotal role in model performance. Too few epochs may lead to under-fitting, where the model hasn't learned enough from the data, while too many epochs might cause over-fitting, where the model captures noise in the training data, failing to generalize well to new data. Striking the right balance by monitoring validation performance helps achieve optimal model training and prevent over-fitting or premature convergence.

The number of epoch are varied while keeping the learning rate same as 10^{-1} which is the best value of learning rate. It can be observed on Figure 5 and Figure 6 that the number of epoch 250 is optimal as it reaches convergence early and it has slightly better accuracy than the number of epoch 100. Nevertheless, the difference in accuracy and loss in both training and validation data set between epoch 100 and epoch 250 is insignificant. However, we choose number of epoch 250 as best because of the uncertainty in unseen data where number of epoch 100 might not be sufficient to reach the convergence. **Therefore, we select number of epoch of 250 as the best number of epoch for the Logistic Regression model.**

1.4 Momentum

Momentum is a critical concept in optimization algorithms that enhances the convergence of machine learning models. By introducing a notion of momentum, the optimization process gains the ability to overcome local minima and accelerate convergence towards optimal solutions. In the context of a Logistic Regression (LR) model, momentum affects the parameter updates during training. The equation $v = \mu v - \lambda \nabla L(w)$ showcases how the gradient descent direction is influenced by the accumulated momentum term (v) and the learning rate (λ). The updated weight parameters ($w = w + v$) then reflect the combined effect of both momentum and learning rate, effectively steering the model towards better convergence paths.

The momentum values are varied while keeping the learning rate same as 10^{-1} which is the best value of learning rate and keeping the number of epoch same as 250 which is the best value of epoch. It can be observed on Figure 7 and Figure 8 that the momentum value of 0.9 helped to achieve the convergence on accuracy earlier than the momentum value of 0. However, the higher momentum value has a slightly lower accuracy than the lower momentum value of 0. Similarly, the loss corresponding to higher momentum value of 0.9 is slightly higher than the loss corresponding to lower momentum value of 0. In summary, it is evident that keeping a high momentum value increases the speed of the training process but there will be a trade off with the accuracy. For this example, since the number of epoch is kept as 250 and there is no problem of slow convergence, we give more weight to the accuracy over speed. **Therefore, we select momentum value of 0 as the best value of momentum for the Logistic Regression model.**

The best hyper-parameters that are selected for the LR model is shown in Table 1.

1.5 Initial Weights

Choosing initial weights properly is crucial as it significantly affects the optimization process and the final performance of a machine learning model. Poor initial weights can lead to slow convergence, getting stuck in local minima, or even divergence. Well-chosen initial weights provide the model with a favorable starting point, enabling faster convergence and better chances of finding globally optimal solutions. Appropriate initialization strategies help enhance training stability, prevent vanishing or exploding gradients, and improve the overall efficiency of the learning process.

Random initialization assigns different starting points to each weight, which breaks symmetry and introduces diversity in the learning process. This diversification aids in capturing distinct features and patterns in the data. Moreover, random initialization prevents the model from converging to the same values, which enhances the model's capacity to learn complex relationships and generalize better to new data. Overall, random initialization promotes effective training by introducing variety and reducing the risk of symmetrical weight updates.

In this example, since there were only 2 input features, we could have initialized the weights manually by plotting the data in 2D plane. One of such manually selected weight could be $[w_0 = 0, w_1 = 1, w_2 = 0]$ since the decision boundary equation would be $x_1 = 0$ for the given training example. However, to keep the model generalized that works on any type of training example (in case we decide to train the model on different set of training data), we select the initial weights by random initialization as described above in this section.

2 Linear Classifier

2.1 Iris Dataset and Logistic Regression on Multi Class Setting

Studying the application of Softmax regression (Multinomial Logistic Regression) on the Iris dataset involves understanding how to extend logistic regression to handle multi-class classification. The Iris dataset contains samples from three species of iris flowers: Setosa, Versicolor, and Virginica. Each sample has four features: sepal length, sepal width, petal length, and petal width. The goal is to classify the iris flowers into their respective species based on these features.

Logistic regression can be extended to handle multi-class classification using techniques like One-vs-Rest (OvR). In multiclass classification problems, the goal is to categorize instances into one of multiple classes. However, many traditional binary classification algorithms are designed to classify instances into two classes. The OvR strategy provides an elegant solution to extend these binary classifiers to handle multiclass scenarios.

The OvR strategy involves breaking down a multiclass classification problem into multiple binary classification sub-problems. For each class in the original problem, a separate binary classifier is trained to distinguish instances of that class from instances of all the other classes combined. This approach effectively transforms the original multiclass problem into a set of binary classification tasks. During training, the instances of the class for which the classifier is being trained are treated as positive examples, while instances from all other classes are considered as negatives. Once all the binary classifiers are trained, classification of a new instance involves running it through each binary classifier. Each classifier produces a score or a probability indicating the likelihood that the instance belongs to its associated class. The class associated with the classifier that produces the highest score is then predicted as the final output class for the instance.

2.2 Learning Rate

The importance of learning rate in the ML model is discussed in Section 1.2. The impact of varying learning rates on accuracy and loss was explored for the Linear Classifier (LC) model. The outcomes of these experiments can be summarized as follows:

Training accuracy: Analyzing Figure 9 reveals that the Linear Classifier (LC) model achieves its highest accuracy when the learning rate is set to 10^{-1}. However, this learning rate of 10^{-1} is highly unstable when at the initial stages of iteration. This can be attributed due to the fact that at initial stages of iteration, the higher learning rate makes a large step size which can lead to oscillating nature of the function evaluation specially if the function is convex function which is the case for LC loss function. As the number of iteration increases, the change in the function becomes smaller and results in a smooth convergence. In order to eliminate the oscillatory behavior of the accuracy corresponding to learning rate of 10^{-1}, at this stage we may think to select a smaller learning rate of 10^{-2} such that there will be no oscillation at any stages of the iteration. Therefore, we may think to select the learning rate of 10^{-2} as the best learning rate. Nevertheless, the accuracy difference between these 2 choices of learning rate is insignificant. Hence, we don't lose any significant amount of accuracy in the training data set by choosing a smaller learning rate. For other smaller learning rates, the accuracy is significantly on the lower side so we ignore them for the design of parameters.

Training loss: From Figure 10, it is evident that the loss is minimum and achieves convergence in lowest number of epochs when the learning rate is set to 10^{-1}. As expected, a lower

learning rate will require more number of epochs to achieve minimum loss. The difference between the losses in the training data set is too significant to ignore. For number of epoch of 250, the training loss is nearly double for learning rate of 10^{-2} as compared to learning rate of 10^{-1}. However, since the absolute magnitude of the loss is very small, we give priority to accuracy and non-oscillatory behavior of the accuracy and loss with respect to iterations.

Validation accuracy: It can be seen from Figure 11 that the validation accuracy is highest when the learning rate is set to 10^{-1}. For lower learning rates, the number of epoch 1000 is insufficient to achieve convergence. Nevertheless, the learning rate of 10^{-2} gives the same accuracy at par with the learning rate of 10^{-1}.

Validation loss: It can be seen from Figure 12 that the validation loss is minimum when the learning rate is set to 10^{-1}. It is also evident that the loss is double for the learning rate 10^{-2} like the training loss. Nevertheless, we try to eliminate the oscillatory behavior of the function at all stages of iteration by choosing the lower learning rate.

In summary, the learning rate of 10^{-2} is optimal for this LC model as it reaches to convergence in less number of epochs without having significant impact on the accuracy and loss of the LC model. **Therefore, we select learning rate of 10^{-2} as the best learning rate for the Linear Classifier model.**

2.3 Number of Epoch

The importance of number of epoch in the ML model is discussed in Section 1.3. The impact of varying number of epoch on accuracy and loss was explored for the Linear Classifier (LC) model. The number of epoch are varied while keeping the learning rate same as 10^{-2} which is the best value of learning rate. It can be observed on Figure 13 and Figure 14 that the number of epoch 500 is optimal as it reaches convergence early and it has slightly better accuracy than the number of epoch 500. Nevertheless, the difference in accuracy and loss in both training and validation data set between epoch 250 and epoch 500 is insignificant. However, we choose number of epoch 500 as best because of the uncertainty in unseen data where number of epoch 250 might not be sufficient to reach the convergence. **Therefore, we select number of epoch of 500 as the best number of epoch for the Linear Classifier model.**

2.4 Momentum

The momentum values are varied while keeping the learning rate same as 10^{-2} which is the best value of learning rate and keeping the number of epoch same as 500 which is the best value of epoch. It can be observed on Figure 15 and Figure 16 that the momentum value of 0.9 helped to achieve the convergence on accuracy earlier than the momentum value of 0. It can be seen that, the higher momentum value has a slightly higher accuracy than the lower momentum value of 0. In terms of the loss, there is no difference in the loss between the momentum value of 0.9 and corresponding to lower momentum value of 0. In summary, it is evident that keeping a high momentum value increases the speed of the training process along with the accuracy. **Therefore, we select momentum value of 0.9 as the best value of momentum for the Linear Classifier model.**

The best hyper-parameters that are selected for the LC model is shown in Table 1.

Model Name	Best Learning Rate	Best Number of Epoch	Best Momentum
Logistic Regression	0.1	250	0
Linear Classifier	0.01	500	0.9

Table 1: Best hyper-parameters selected for the LR and LC model

2.5 Initial Weights

The importance of choosing a proper initial weight is discussed in 2.5. In this example, since there are 4 input features, we cannot initialize the weights manually by plotting the data in 2D plane. Since we are unable to visualize the decision boundary, therefore the only possibility to choose the initial weights is by means of random initialization. That's why we select the initial weights by random initialization.

References

[1] Andrew Ng. *Machine Learning Specialization*. 2022. URL: https://youtu.be/vStJoetOxJg.

[2] Aston Zhang et al. *Dive into Deep Learning*. 2021. URL: https://d2l.ai/.

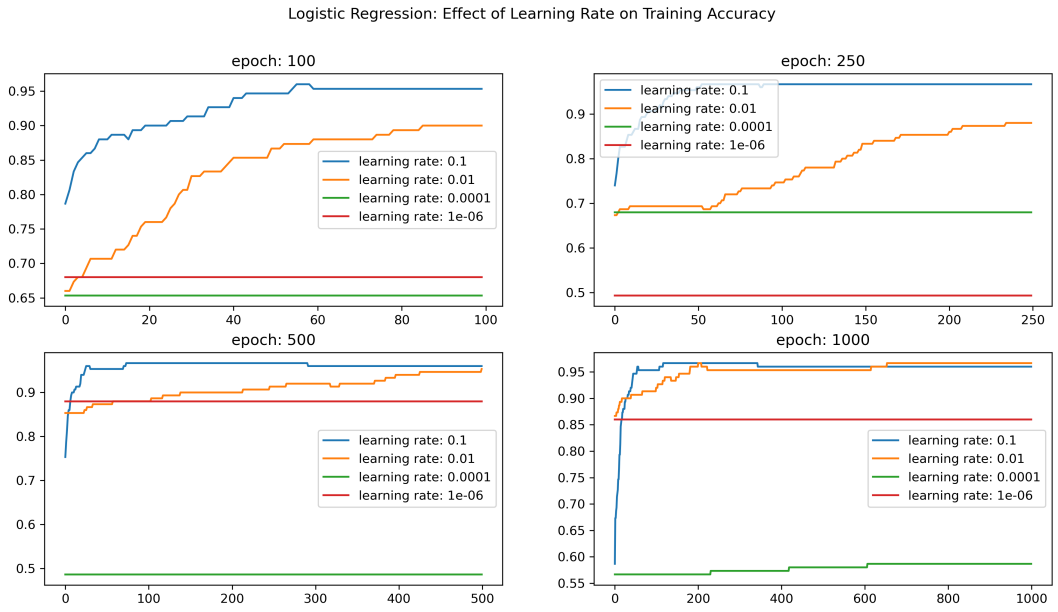

Figure 1: Effect of learning rate on training accuracy of LR model

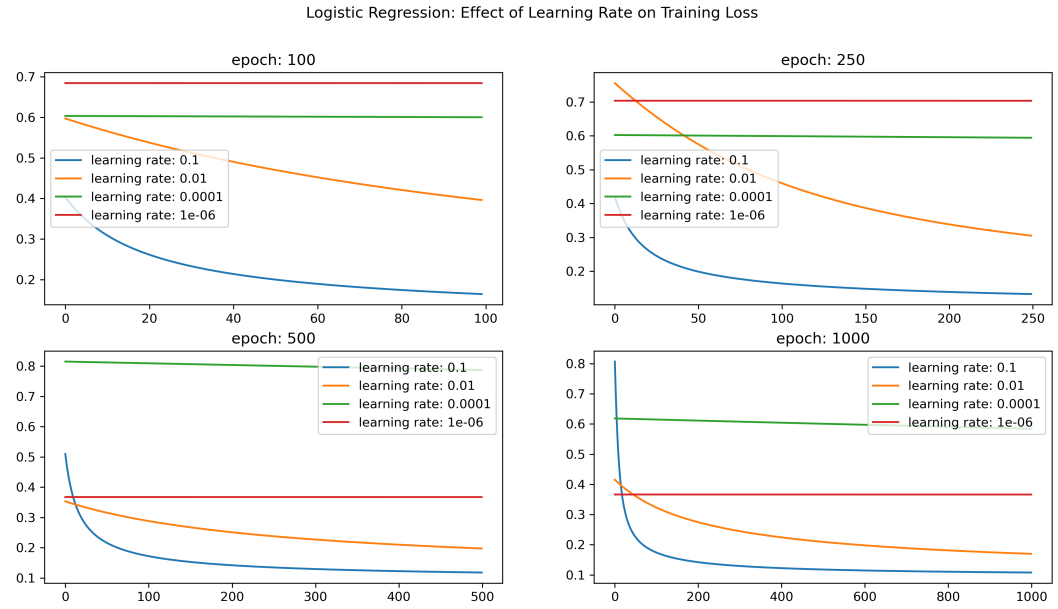

Figure 2: Effect of learning rate on training loss of LR model

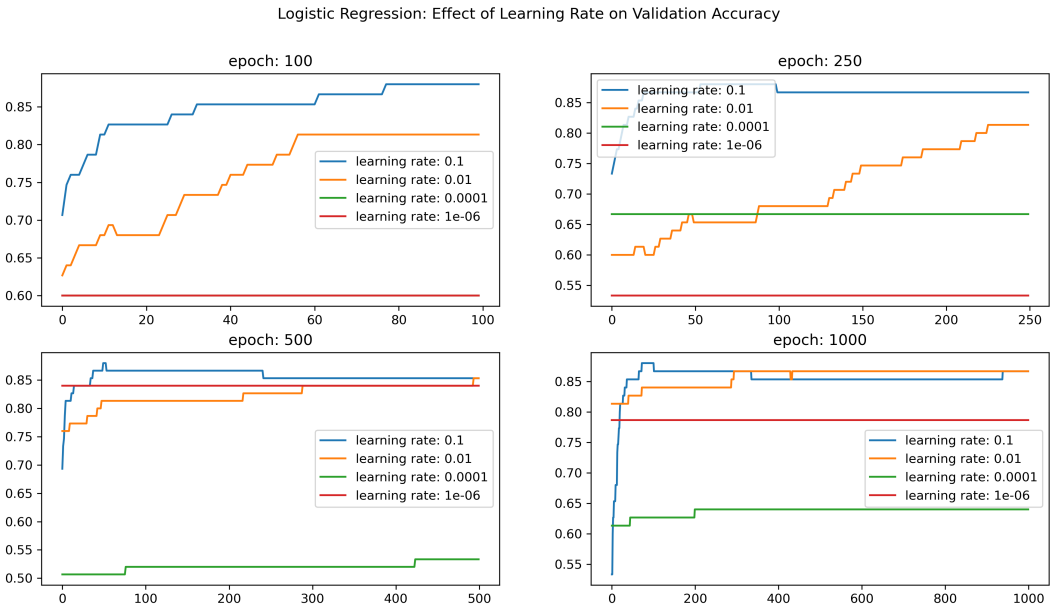

Figure 3: Effect of learning rate on validation accuracy of LR model

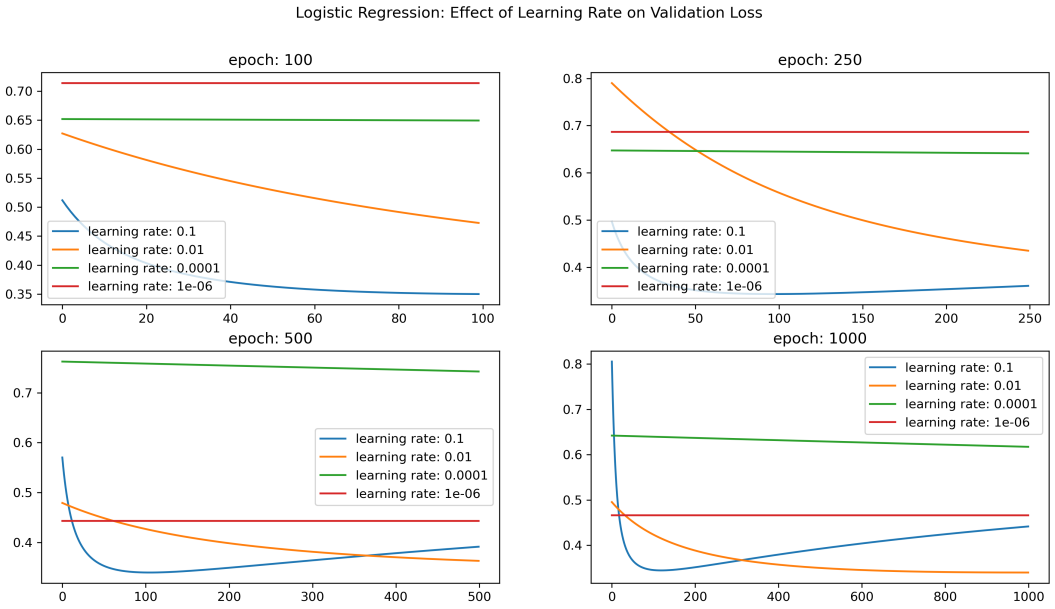

Figure 4: Effect of learning rate on validation loss of LR model

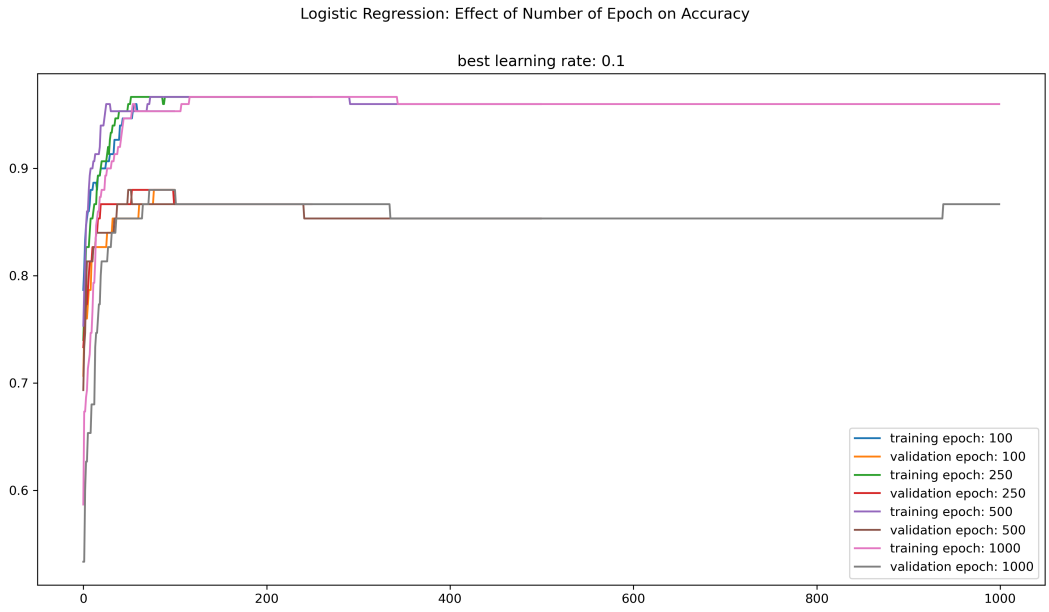

Figure 5: Effect of number of epoch on accuracy of LR model

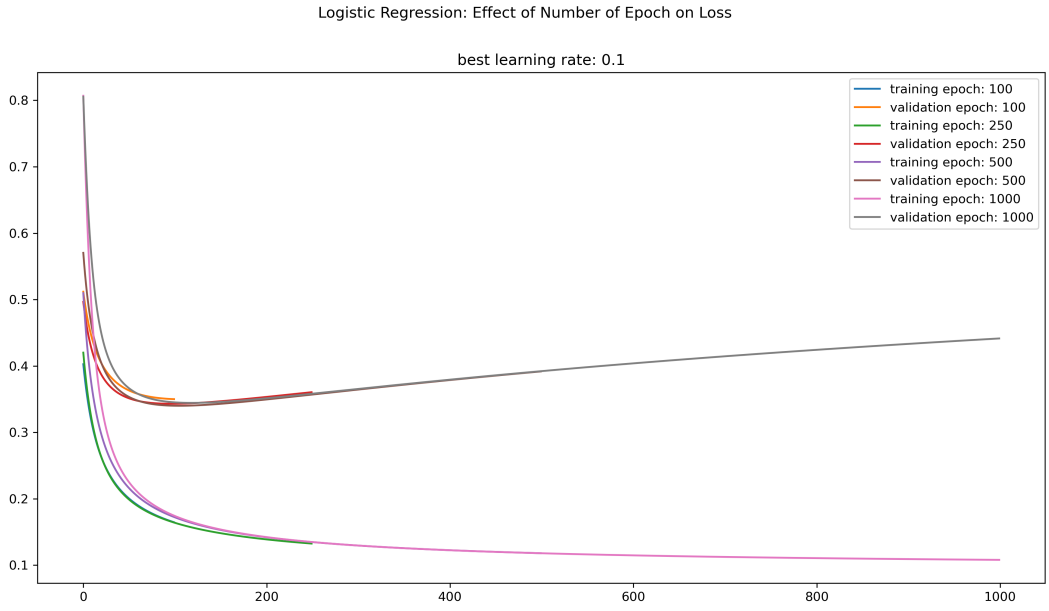

Figure 6: Effect of number of epoch on loss of LR model

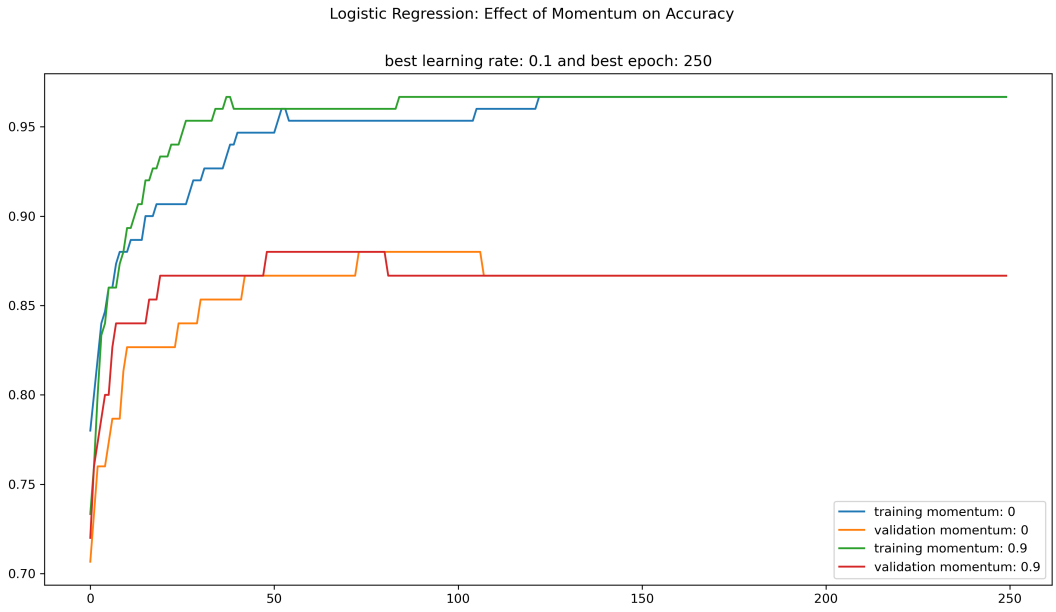

Figure 7: Effect of momentum on accuracy of LR model

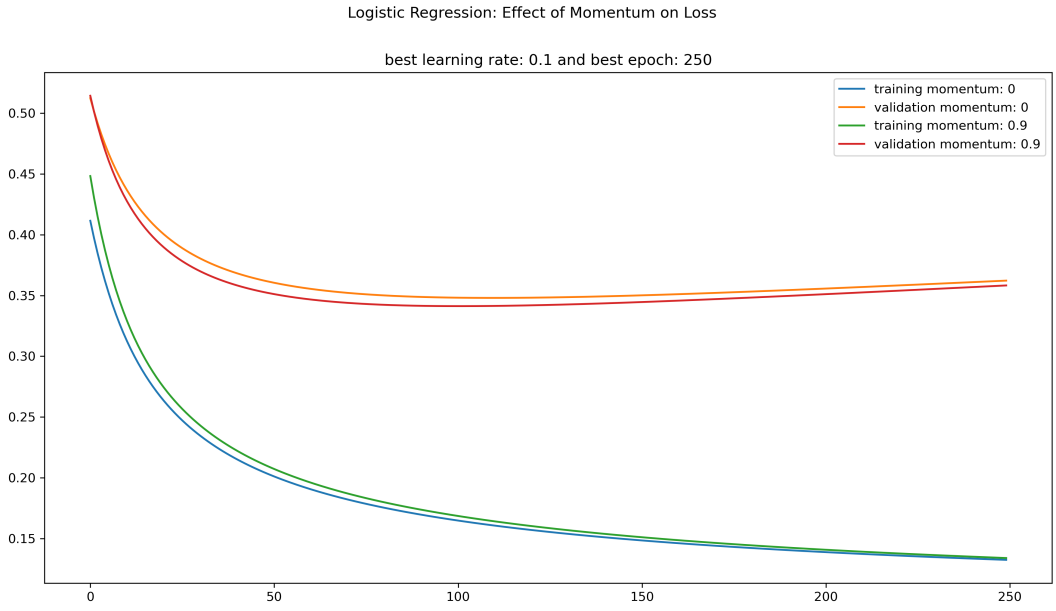

Figure 8: Effect of momentum on loss of LR model

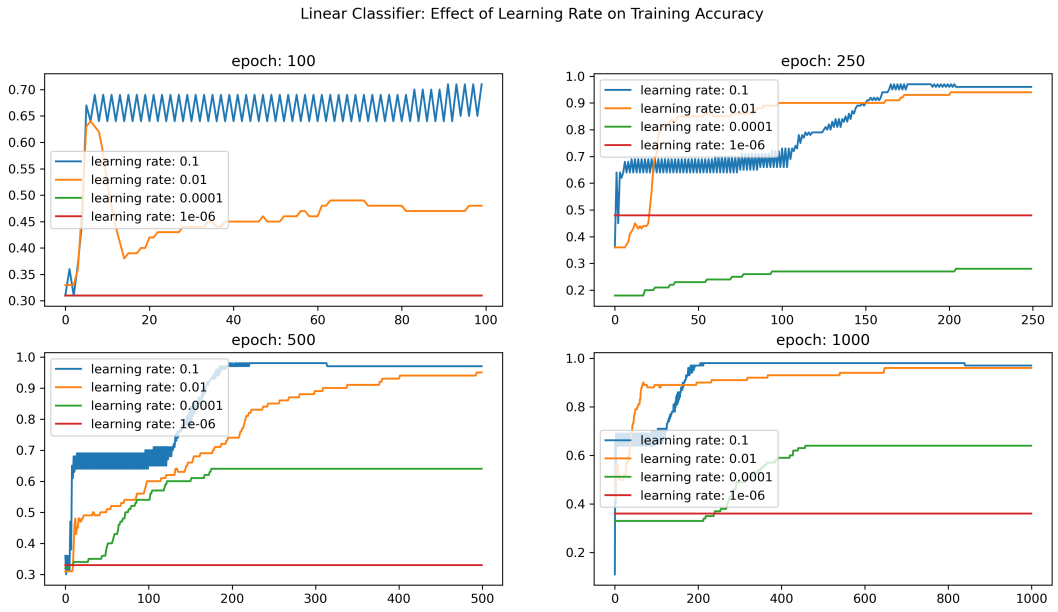

Figure 9: Effect of learning rate on training accuracy of LC model

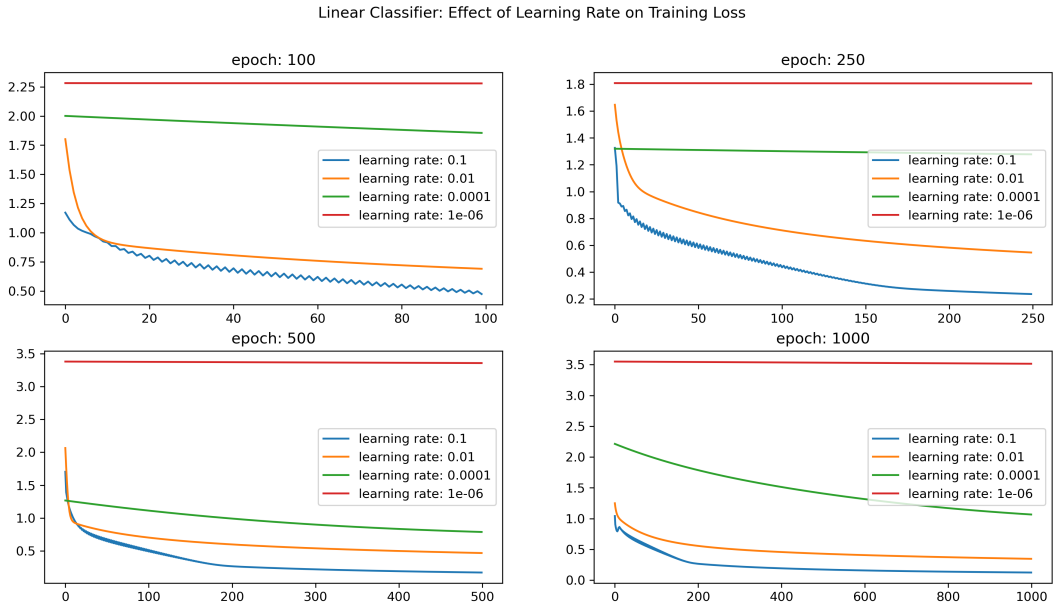

Figure 10: Effect of learning rate on training loss of LC model

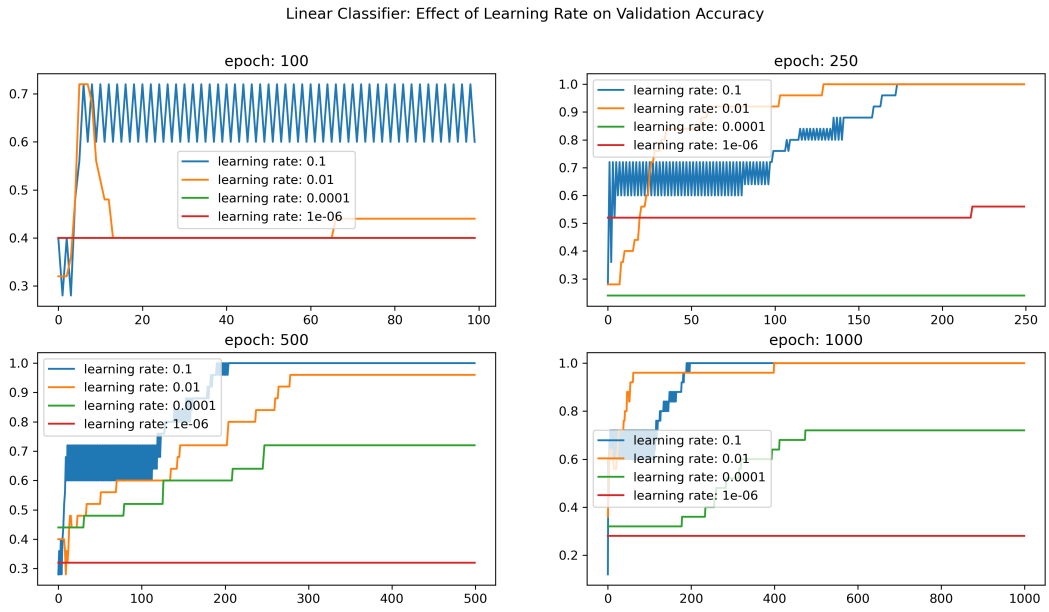

Figure 11: Effect of learning rate on validation accuracy of LC model

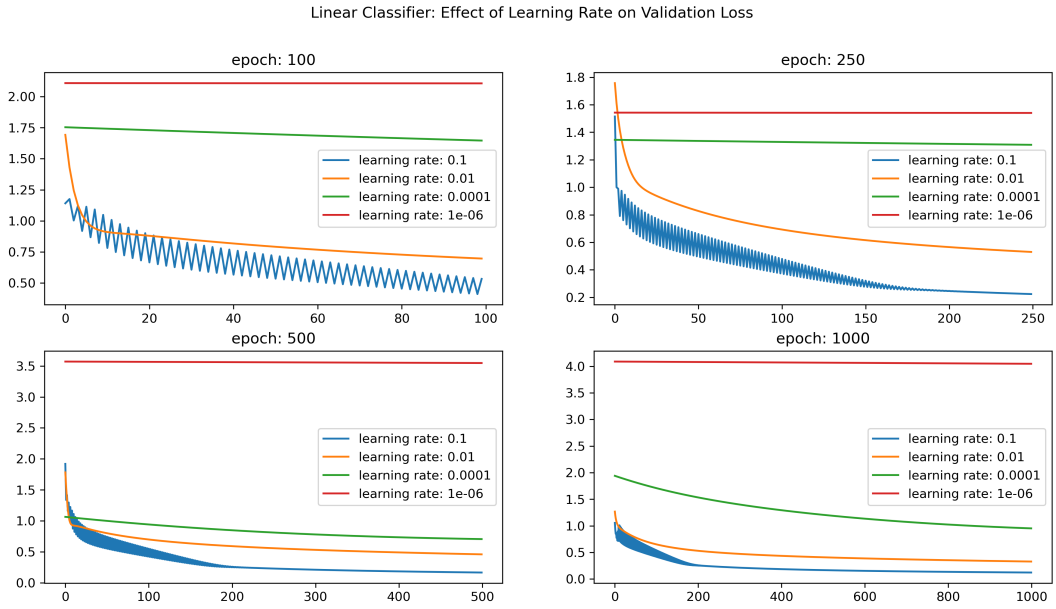

Figure 12: Effect of learning rate on validation loss of LC model

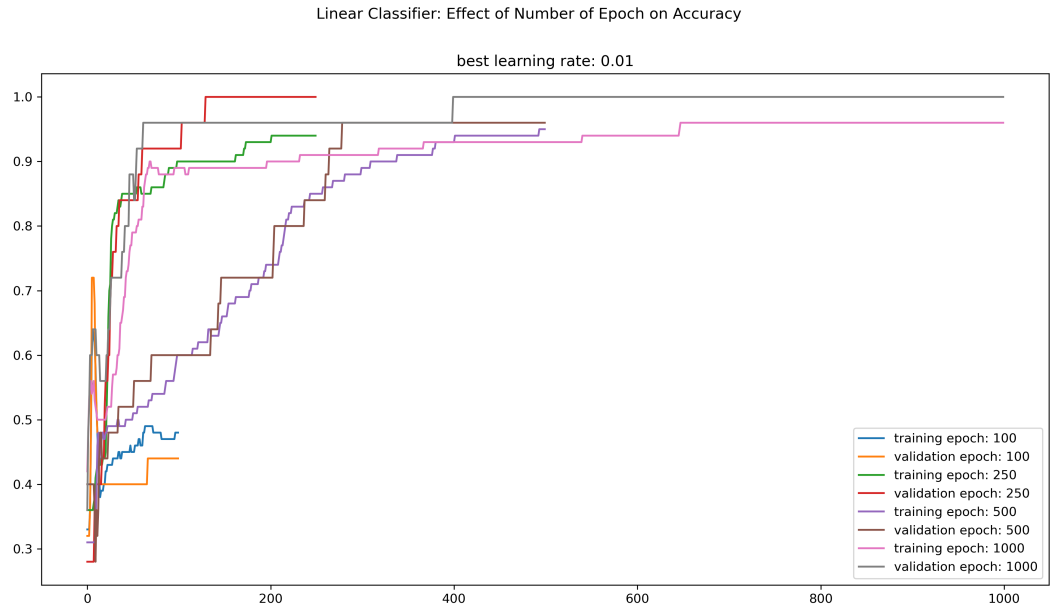

Figure 13: Effect of number of epoch on accuracy of LC model

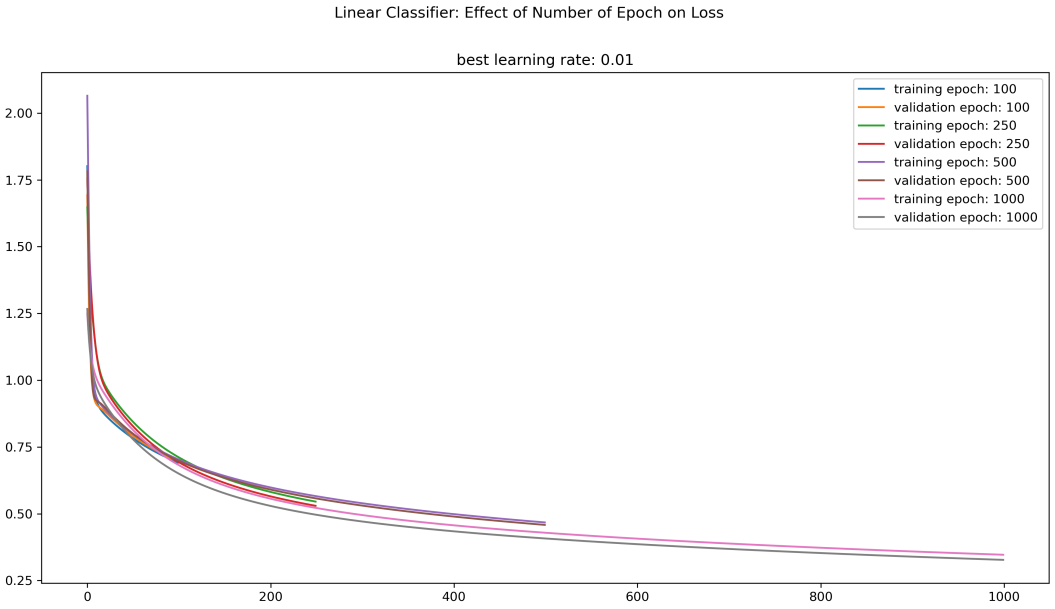

Figure 14: Effect of number of epoch on loss of LC model

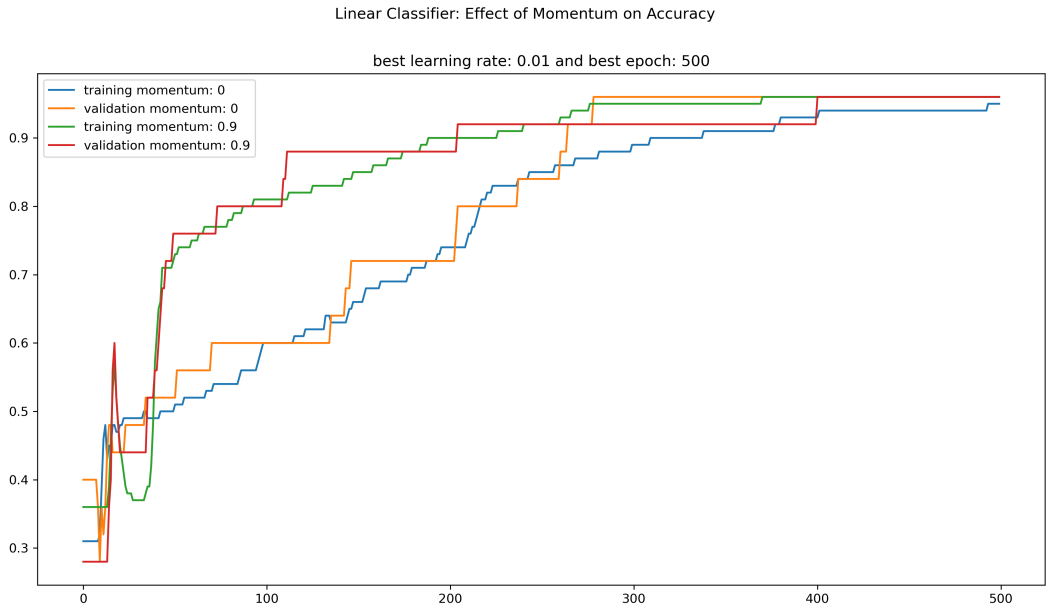

Figure 15: Effect of momentum on accuracy of LC model

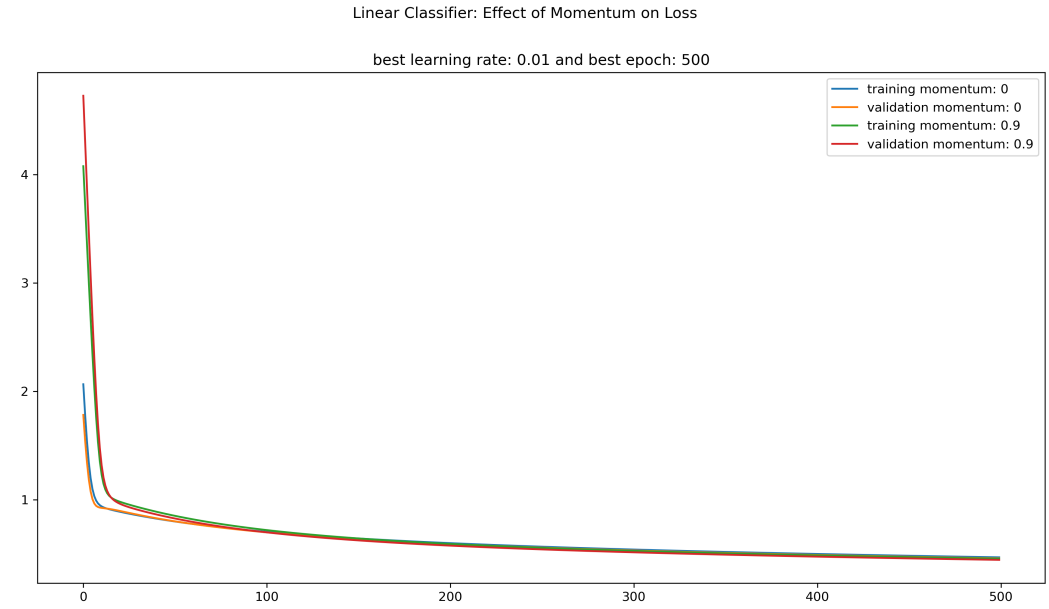

Figure 16: Effect of momentum on loss of LC model