10: Logistic Regression

Sergio Steven Leal Cuellar* and Mateo Rueda Molano†,
Department of Biomedical Engineering, University of the Andes, Bogota, Colombia
Email: *ss.leal10@uniandes.edu.co, †ms.rueda10@uniandes.edu.co,

***Abstract*—The Fer2013 dataset was used to train a model of logistic regression in order to solve a binary classification problem to distinguish happy faces from the other ones. In addition, a multiclass classification problem was done detecting all the classes: angry, disgust, fear, happy, sad, surprise, neutral. For that purpose, sigmoid and softmax activation functions were used for the binary and multiclass classification problem respectively. The results obtained show that the algorithm is learning properly due to the diminishing on the loss for every epoch. Even though the results obtained represent not a high performance, extending the number of epochs will surely cause the algorithm to converge to a lower loss state thus better results will be obtained. In conclusion, the algorithm developed learns the weigths and bias of logistic regression satisfactory and classifies properly in binary and multiclass classification problems.**

I. Introduction

Logistic regression (LR) is defined as a standard probabilistic statistical classification model with high demand in many industries and widely used across different disciplines such computer vision, marketing, social sciences, etc. The difference between logistic regression and linear regression is the outcome. The outcome of logistic regression on a sample is the probability to be positive or negative, where the probability depends on a linear measure of the sample. Therefore, LR is widely used for classification [1]. More formally, for a sample $xi \in Rp$ whose label is denoted as yi , the probability of yi being positive is predicted to be:

$$P(yi = +1) = \frac{1}{1 + \exp(-\beta * T * xi)} \quad (1)$$

Given the LR model parameter β. In order to obtain a parameter that performs well, often a set of labeled samples $[(x1, y1), ..., (xn, yn)]$ are collected to learn the LR parameter which maximizes the induced likelihood function over the training samples.

Several previous works have investigated multiple approaches to robustify the logistic regression (LR). The majority of them are M-estimator based: minimizing a complicated and more robust loss function than the standard loss function (negative log-likelihood) of LR. However, in this article we will use stochastic gradient descents. Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks.

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in R$ by updating the parameters in the opposite direction of the gradient of the objective function $\bigtriangledown * J(\theta)$. The learning rate η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley (local minimum) [2]. Batch gradient descent performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online. It is also shown that when we slowly decrease the learning rate, SGD shows the same convergence behaviour as batch gradient descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively [3].

From these theoretical methods, a classification of emotion images was made with the FER2013 database. In the first part, a method of classifying binary images in happiness vs the rest of emotions was proposed. In the second part, the 7 types of emotions were classified under the described logistic regression algorithms using stochastic gradient descent (SGD). First, the best models were tested by analyzing the stochastic gradient and later, it was evaluated in the test model and in its own natural images [3].

II. materials and methods

A. Dataset description

The Fer2013 dataset consists of 35.887 gray scale images of faces sized as 48x48. Each face has an specific emotions from 7 of the possible ones; angry, disgust, fear, happy, sad, surprise, neutral. The database is distributed in the next amount of images per class; 4593: Angry, 547: Disgust, 5121: Fear, 8989: Happy, 6077: Sad, 4002: Surprise and 6198: Neutral. The annotations are given by the labels 0 - Angry, 1 - Disgust, 2 - Fear, 3 - Happy, 4 - Sad, 5 - Surprise, 6 - Neutral.

Some of the images in the wild that were taken from the Cohn-Kanade (CK and CK+) dataset and some others taken from the internet to sum up a total of 11 images of various faces with emotions.

(a) Cohn-Kanade (CK and CK+) dataset (in the wild sample)

(b) Fer2013 image sample

Figure 1. Comparison of the Fer2013 images and in the wild images.

B. Methods description

1) Binary Classification: The method implemented for the binary classification of Happy/Not happy is based on a logistic regression with hypothesis y = Wx + b in which y is the predicted label (0 or 1) for the sample x taking in to account the weights W and the bias b. In order to establish the optimum values of the parameters W and b the loss function of equation 2 was established and the model is optimized by using stochastic gradient descent.

$$J(\theta) = \frac{-1}{M}\sum_{i=1}^{M} Y^i \log(h_\theta(X^i)) + (1-Y^i)\log(1-h_\theta(X^i)) \tag{2}$$

The activation function that was used is the sigmoid function of equation 3 which gives as output binary values (0,1).

$$Sigmoid(z) : S(z) = \frac{1}{1+\exp(-z)} \tag{3}$$

Then, using two different values of learning rate(parameter that establishes the length of every step taken in the gradient descent)and two different batch sizes (number of divisions per epoch)and the loss was analyzed over several epochs. Finally, the resultant predictions were obtained by applying the sigmoid function with a threshold of 0,5 and the results were evaluated using the F1 and average classification accuracy metrics.

2) Multiclass Classification: Considering all the classes/emotions for the multiclass problem (angry, disgust, fear, happy, sad, surprise, neutral) the same logistic model was done. Nevertheless, the shape of the matrices W and b is changed in order to fix the new amount of classes that are going to be considered. For that purpose, the matrix of weights W must match the number of data samples (m = 48x48) for the specified number of classes(n = 7) in order to be able to apply the matrix product between the samples (m = batch size,n = 48x48) and sum up with the the vector of bias b (m = 1, n = 7) which will be added to every row of the resulting vector which will have a size of (m = batch size, n = 7). Then in order to obtain the probability of every sample to belong to an specific class, the softmax function, as described in equation 4, is applied to every row of the resultant matrix and a probability matrix for every sample is obtained in which the sum of rows is 1 and every column is the probability of belonging to every class and every position in the row vector represents the probability of that sample to be classified as specific class (number of column).

$$Softmax(X)_i = \frac{\exp(X_i)}{\sum_j \exp(X_j)} \tag{4}$$

The loss function that was used for this multiclass problem is the cross entropy loss function, from equation 5. This function is defined by the mean value of the sum of losses resulted by the multiplication of the natural logarithm of the probability matrix (softmax(Wx+b)) and the ground truth labels. This can be done by setting a one hot encoding for the annotations in which the label is 1 in the corresponding column. The cross entropy loss function diminishes when prediction is more and more accurate (probability of 1 from softmax, i.e. perfect prediction, means a zero loss due to the natural logarithm). The optimization method that was used to reduce the loss was the same stochastic gradient descent used in the previous binary classification problem.

$$CrossEntropy : H_Y(Y') = \frac{-\sum_i Y_i * \log(Y_i')}{N} \tag{5}$$

Then, the experiment was run using a learning rate of 0.003 and for several epochs until the loss was minimized as much as possible. This high learning rate was used because, due to the shape of the matrix, this operation is far more computationally expensive than the binary problem, so it's necessary that the algorithm converges in less epochs. Finally, the resultant prediction label was established taking the column (position/label) with the highest probability for every sample and the results were evaluated F1 and average classification accuracy metrics.

III. Results

A. FER2013 binary classification

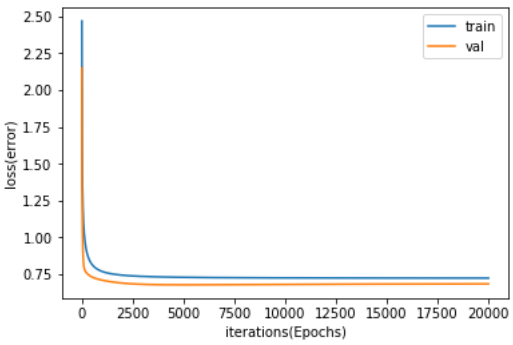

Figure 2. Loss behavior over the epochs for binary classification problem (20000 epochs).

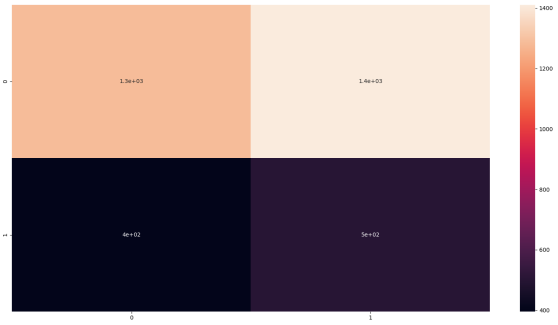

Figure 3. Confusion matrix testing the algorithm on images of fer2013 resulting in ACA of 0.513 and overall F1 of 0.47.

B. Images in the wild binary classification.

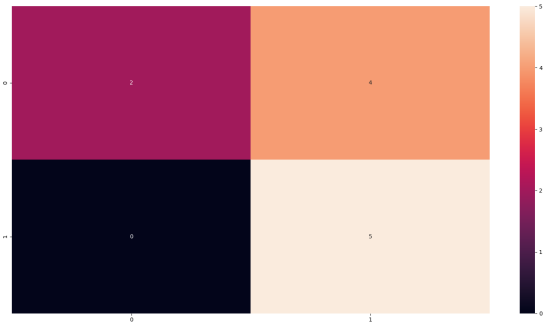

Figure 4. Confusion matrix testing the algorithm on images of fer2013 resulting in ACA of 0.778 and overall F1 of 0.61.

C. FER2013 multiclass classification

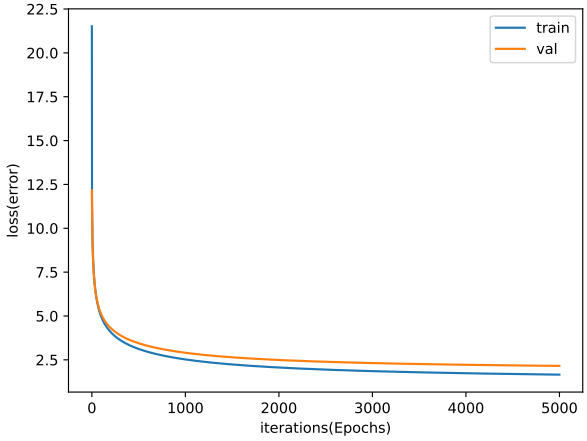

Figure 5. Loss behavior over the epochs for multiclass classification problem for 100 image batch, learning rate of 0.003 and 5000 epochs.

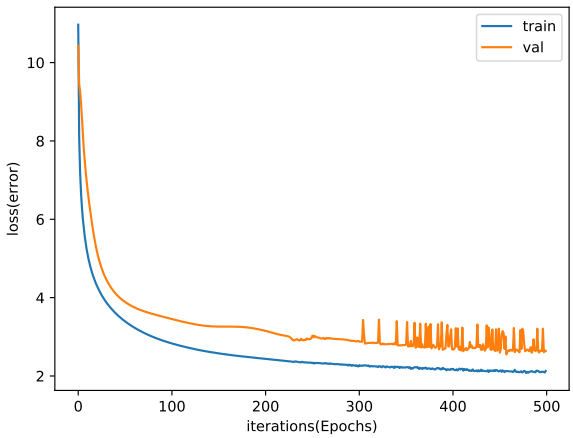

Figure 6. Loss behavior over the epochs for multiclass classification problem for 100 image batch, learning rate of 0.03 and 5000 epochs.

Figure 7. Confusion matrix testing the algorithm on images of fer2013 resulting in ACA of 0.253 and overall F1 of 0.25.

D. Images in the wild multiclass classification

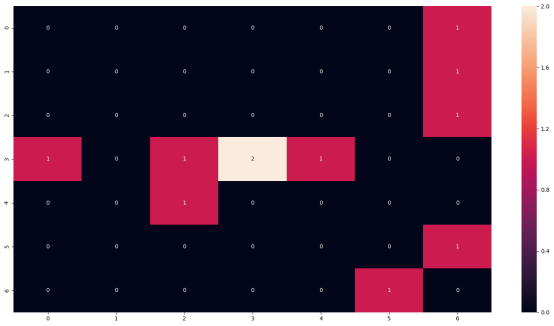

Figure 8. Confusion matrix testing the algorithm on images in the wild resulting in a F1 of 0.57 for happy faces .

IV. Discussion

A. Binary classification

The first stage of the results of the binary classification was to analyze the behavior of the decline of the loss function, which had to be minimized in the validation set. This can be

seen in the section of appendices where the multiple graphs are shown evaluating the decrease of the loss function with respect to the number of epochs and to 2 different batch sizes and learning rates. It was found according to the theory that a low learning rate allows to reach a minimum and prevent shocks between the minimums. It is for this reason that the function does not present variances like those in figure 8, since both learning rate with which the model was tested were quite low. However, the learning rate was much better under 0.003 with the same times and this is due to the fact that the gradient decreases faster and, as it does not present large variations, it drops faster than a learning rate model of 0.00001.

Therefore, the model used was the one shown in figure 2 with a batch size of 50 and a learning rate of 0.003 for the test set. The results were not what was expected since the ACA and the F1 were quite low being a problem of binary classification (figure 3, so we would have to evaluate itself if the variance is very high between the test and validation set or if simply because of randomness, the parameters found were not suitable for the test set. However, in the demo set the ACA and F1 improved considerably (figure 4), although no more general conclusions can be drawn because there were only 11 images. However, it is worth to do more experiments increasing the number of epochs. While it is true that the number of losses at more than 20,000 epochs tended to stabilize, this number never reached a global minimum where descent was static.

B. Multiclass classification

As can be seen in figure 5, for every iteration (epoch) of the algorithm the model is getting smaller losses every time which is an important indication that the model is learning. Nevertheless, the F1 and ACA obtained is not high enough, this can be explained by the number of epochs set for the model to adjust it's parameters which was too small and parameters couldn't get optimized far more. For that reason, establishing more epochs for the model to learn will represent a direct improvement in the resultant metrics but due to the computational expensiveness of the operations on these bigger matrix, the computation time will also be significantly increased. In addition, as can be seen in figure 23, as expected the algorithm faces problems when big learning rates are set, even though it decreases the loss faster, it is unable to diminish far more than $\sim$2 of loss because such a learning rate causes the algorithm to "bounce" around the real minimum of the function and that's why the loss goes up and down every epoch. To avoid that behavior, smaller learning rates must be established and bigger epochs must be set.

Considering the confusion matrix obtained for the fer2013, as can be seen in figure 7, it can be said that those classes that result being the most challenging are the disgust faces. On the other hand, those classes which tend to be the easiest were the happy faces. This behavior can be explained due to how disgust faces tend to look similar faces of other classes like angry, fear or sad faces. Also, as can be seen in figure 8, even though few epochs were used an the loss barely descend to $\sim$1.6the algorithm performs well in classifying happy faces but not that well on the other classes. This might be explained by the similarities around the other classes and how different happy faces are from the other ones.

V. Conclusions

In conclusion, the algorithm developed is suitable of learning the proper weights and bias for the logistic regression model due to the constant decreasing of the loss over every single iteration which indicates that the algorithm is learning after every re calculation of the parameters and the gradient descent is working properly. Nevertheless, the epochs that were used for the training of the method were not enough and the method didn't converge to it's optimum minimum. Because of that, is suitable to let the algorithm train for more epochs but this may cause to raise the computational cost, in order to avoid that a dynamic learning can be set, first a big one and then a small learning rate. This may cause the algorithm to rapidly decrease it's loss but then the decrease on the learning rate may help to avoid "loss bounces" as the ones presented for the learning rate of 0,03. In addition, considering bigger batches may increase the overall performance of the method because the current batch size might be not representative at all for the data and more data points should provide more useful information but as a consequence the computational cost will increase due to the fact that more random access memory would be required in order to process bigger amount of data at the same time.

References

[1] S. Ruder, "An overview of gradient descent optimization algorithms," *Insight Centre for Data Analytics, NUI Galway*, p. arXiv:137.154, 2017.
[2] J. C. C. Darken and J. Moody, "Learning rate schedules for faster stochastic gradient search," *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, p. arXiv:36.44, 1992.
[3] I. Sergey and S. Christian, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Microsoft Research*, p. arXiv:66, 2015.

Appendix A
Extra figures

A. Batch size: 50 learning rate: 0.0003

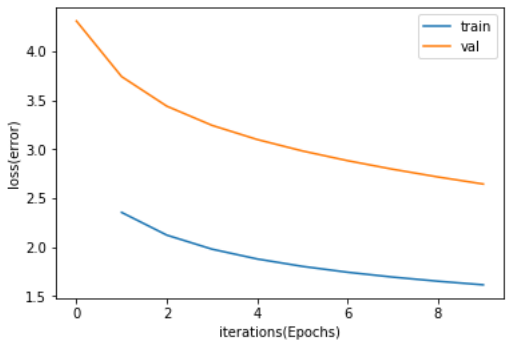

Figure 9. Loss behavior over the epochs for binary classification problem for 10 epochs.

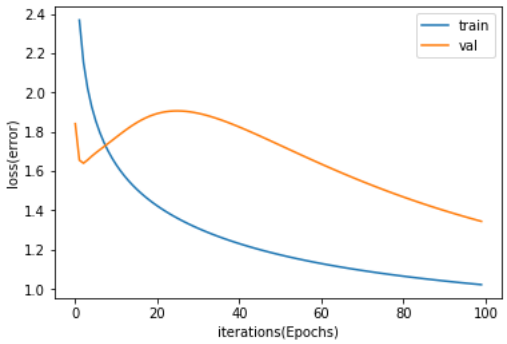

Figure 10. Loss behavior over the epochs for binary classification problem for 100 epochs.

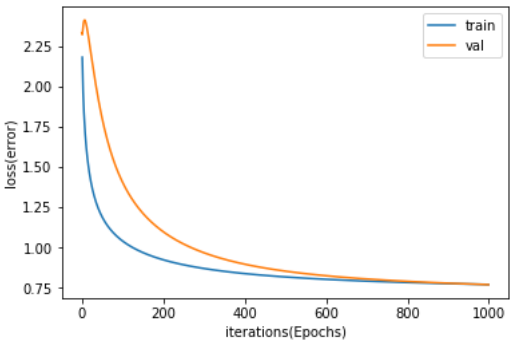

Figure 13. Loss behavior over the epochs for binary classification problem for 1000 epochs.

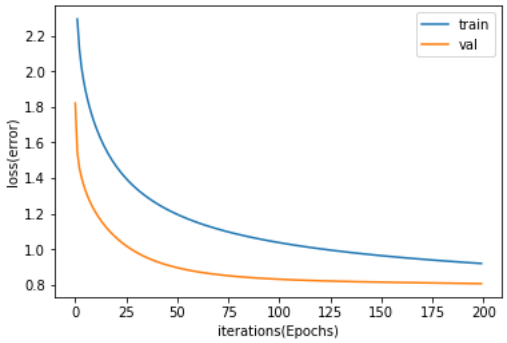

Figure 11. Loss behavior over the epochs for binary classification problem for 200 epochs.

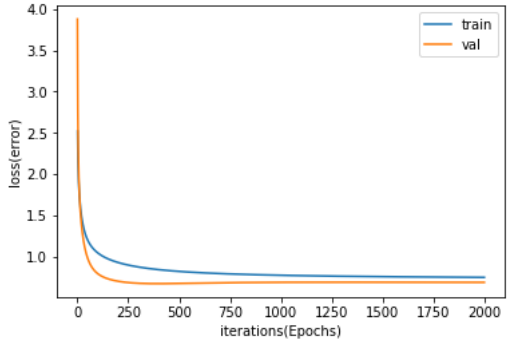

Figure 14. Loss behavior over the epochs for binary classification problem for 2000 epochs.

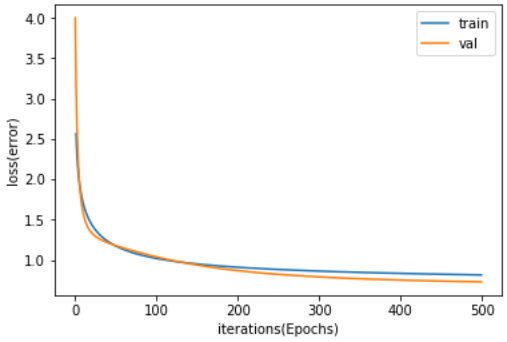

Figure 12. Loss behavior over the epochs for binary classification problem for 500 epochs.

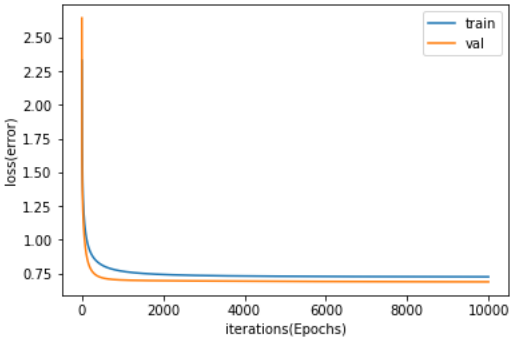

Figure 15. Loss behavior over the epochs for binary classification problem for 10000 epochs.

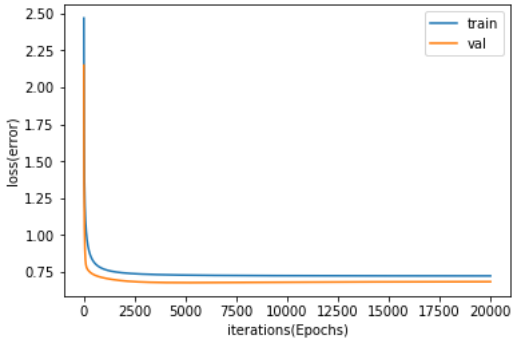

Figure 16. Loss behavior over the epochs for binary classification problem for 20000 epochs.

B. batch size: 100 learning rate: 0.00001

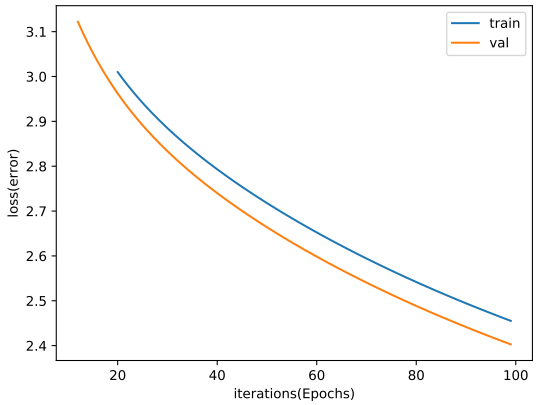

Figure 17. Loss behavior over the epochs for binary classification problem for 100 epochs.

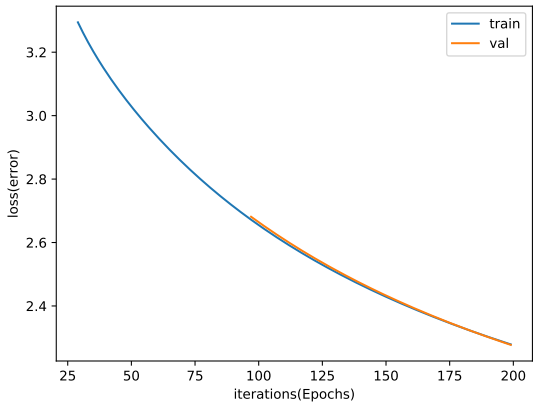

Figure 18. Loss behavior over the epochs for binary classification problem for 200 epochs.

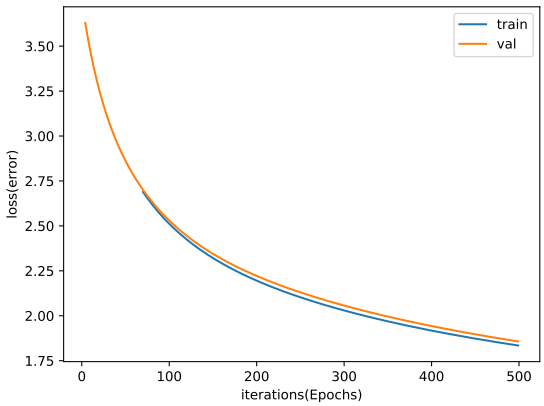

Figure 19. Loss behavior over the epochs for binary classification problem for 500 epochs.

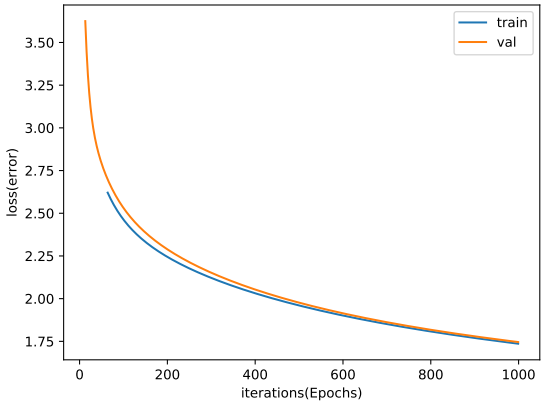

Figure 20. Loss behavior over the epochs for binary classification problem for 1000 epochs.

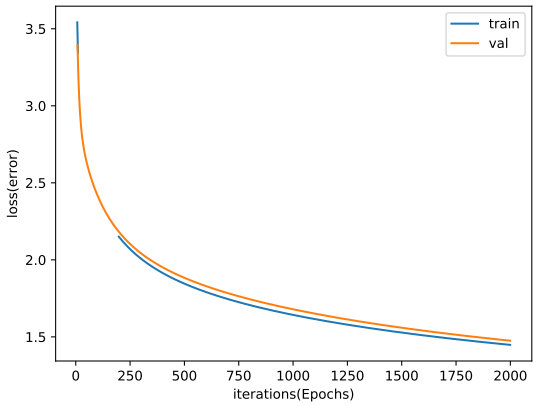

Figure 21. Loss behavior over the epochs for binary classification problem for 2000 epochs.

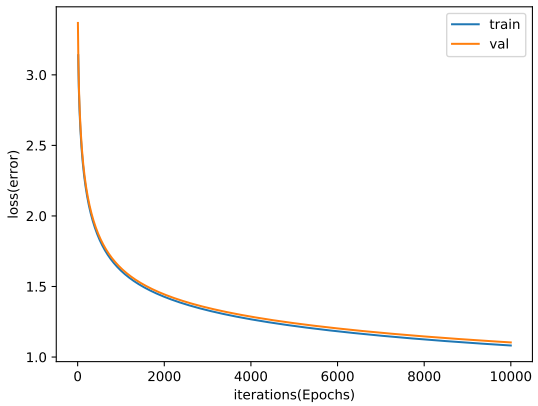

Figure 22. Loss behavior over the epochs for binary classification problem for 10000 epochs.

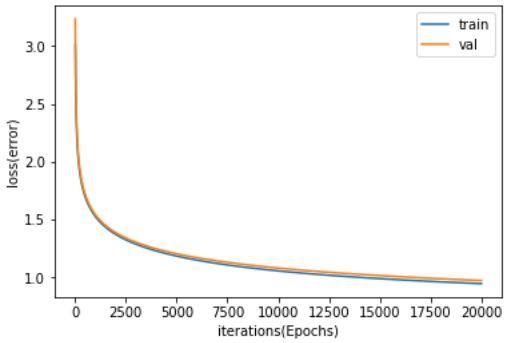

Figure 23. Loss behavior over the epochs for binary classification problem for 20000 epochs.