Implementation and Evaluation of a multiclass Logistic Regression model with momentum

Ma Xiang Xiang
Konstantinos Alvertis

Abstract:

The logistic regression is a simple, yet powerful, machine learning model used for binary classification. In this project, we will show an implementation of such a model, featuring a multiclass extension following the one-versus-rest approach, and the introduction of a momentum parameter to accelerate the stochastic gradient descent. After the evaluation of its performance, we will find that momentum has the biggest impact in the initial stages of the learning algorithm.

1. Introduction

Logistic regression is one of the most popular machine learning models, especially in the medical field [1]. This model can be used both singularly as a sole classifier or combined together to create a more complex model like a neural network. Making them an essential concept to understand for any researcher in the area.

The goal of this project is to implement a logistic regression model and investigate the effects on it when a momentum parameter is introduced in the learning algorithm. Traditionally, Logistic Regression (LR) is a binary classifier (0, 1) that computes the probability, given a datapoint, of being class 1. It is, however, possible to extend this model to perform multiclass classification with a one-versus-rest (OvR) approach.

The multiclass extension is accomplished by applying the learning algorithm for each class, and then pick the one with the highest score on the prediction. This work will investigate the OvR approach and evaluate its performance in relation to the scikit-learn's implementation.

This project is organized in 5 sections. In the Background section, the LR model along with the momentum parameter and Softmax are described. The Experiments section presents the experimental results on different datasets comparing our implementation with scikit-learn's. In the Analysis section, all the results from the previous section are discussed. The final section is dedicated to the conclusions.

2. Background

The logistic regression is a linear model, based on statistics, that is used when there are multiple independent variables, or features, that influences the binary output value, also known as the "dichotomous dependent variable" [2].

The logistic regression uses a sigmoid function, returning a value in the range of [0, 1]. We can write the LR model as follows:

$$f(x) = \frac{1}{1 + e^{-(w^T x - t)}}$$

Where w^T represents the transposed weight vector, x the features, and t the bias. The output of this function is treated as a probability or confidence of the model that a certain datapoint is class 1, the prediction simply checks if the confidence is greater than 0.5, if that is the case then the model predicts the class 1, otherwise class 0.

For this project, we have implemented the stochastic gradient descent (SGD), which approximates the original gradient descent that updates the weights iteratively for every datapoint, and *cross-entropy* is used as the loss function. SGD is a myopic algorithm since the gradient update only accounts for the current partial derivative. As a result, it can take a great number of iterations (or epochs) to reach the optimum.

A way to accelerate the gradient descent is to adopt a momentum parameter (Polyak, 1964), which introduces the idea of an accumulating velocity that keeps "moving downhill" the error landscape based on previous updates. It is described as [3]:

$$v_{t+1} = \beta v_t - \alpha \nabla f(\theta_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

where v is the velocity vector which direction depends on past gradients, [0,1] is the momentum parameter, is the learning rate, and ∇f(t) is the gradient. The stochastic gradient descent with momentum is presented below. For the pseudo-code we will use an array notation.

Algorithm 1 Stochastic gradient descent with momentum

```
t=trand(), w=randvector(), m=0.9, a=0.1
for epoch = 1 to maxepochs do
   for each training example (x, y) do
      for each parameter j do
         v[j] = m*v[j - 1] - a(f(x) -y) x[j]
         w[j] = w[j – 1] + v[j]
      end for
      v[t] = m*v[t-1] + a(f(x)-y)
      t = t + v[t]
   end for
end for
```

2.1. Multiclass extension

As mentioned above, this implementation features a multiclass extension. A *one-hot-target* is used in order to convert the multiclass problem into a binary classification. As a result, the new learning algorithm applies the SGD (algorithm 1) for each class. Then the highest scoring class is chosen as the prediction. The learning algorithm is presented below.

Algorithm 2 One-versus-Rest learning algorithm

```
$t$ = trandvector()
$w$ = 2drandvector()
For each class label $i$ do
    one_hot_y = transform_y_to_hot(y, i)
    w[i]=SGD(X, one_hot_y, w , t)
end for
```

3. Experiments

The experiments involve four datasets, of which, two of them, *MNIST* and *breast* will be evaluated thoroughly in this section. The other two are *Glass* (214 examples, 10 features and 7 classes) and *Monks* (556 examples, 6 features, binary). *MNIST* and *Breast* have 60'000 and 569 examples, 784 and 30 features, 10 and 2 classes respectively.

If not said otherwise, the default parameter for LR are: 0.9 for momentum, 0.1 alpha, normalization based on mean and standard deviation, *5features* epochs. The data is split with K=*5* fold cross-validation. From now on, we will refer to scikit-learn implementation as SKLR.

3.1. Evaluating the momentum parameter

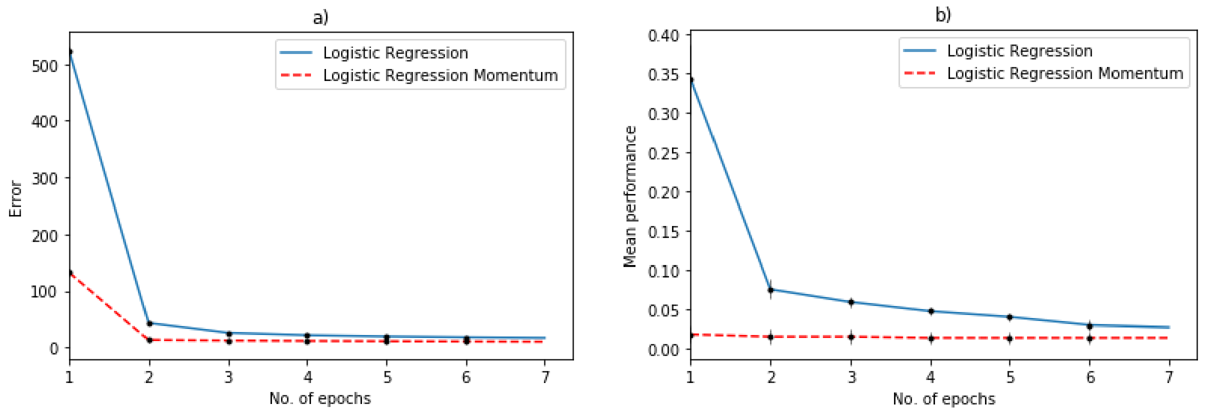

Figure 1. a) Cross Entropy error, b) Mean performance

The Figure 2 an illustrates the cross-entropy error per epoch. The blue line refers to the Errors of Logistic Regression and the red line refers to the momentum logistic regression with m=0.9. The error values collected in every epoch. As can be seen clearly from the diagram above the Error sum plummeted sharply in the first loop and then continued to decrease. Also, confidence intervals are present in the above diagram, but they have small values and are not visible. The Figure 2 b above depicts the mean performance for these two models for each epoch. To collect our measurements, we have been fitting the model repeatedly and in every epoch of training procedure of the algorithm the value "1-accuracy" was calculated which stored in array. Subsequently the mean values and the confidence intervals were calculated and plotted.

To evaluate the Logistic Regression algorithm, a comparison with the SKlearn implementation was done. Five folds were used for linear_model.Logistic Regression model to calculate the accuracy-1.

3.2. Evaluation against scikit-learn

This experiment compares our implementation of the logistic regression to the scikit-learn library. K fold cross-validation, with *K=3*, is performed to get the mean ROC curve. By splitting the dataset differently multiple times, it is possible to observe how the classifier is affected by these changes. The area under the curve (AUC), [0,1], is "the probability of that the classifier, given a randomly chosen positive example, will predict a higher score than a randomly chosen negative example" [5].

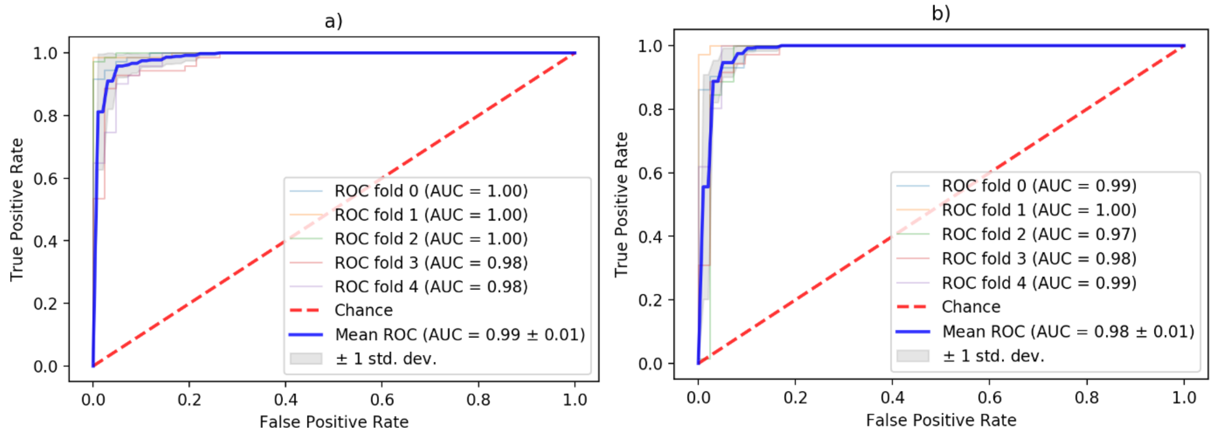

Figure 2. ROC analysis of the breast dataset. a) ROC curves with AUC areas for the scikit-learn implementation. b) ROC curves with AUC areas for LR.

For the multiclass case, the dataset is MNIST. The ROC curve is plotted for each class by considering a series of binary problems, similar to the mechanism of the OvR approach. The macro-average is defined as the "unweighted average of effectiveness across all categories" [6]. It is being used over the micro-average since the former treats every class equally [6] and we are considering a well-balanced MNIST subset.

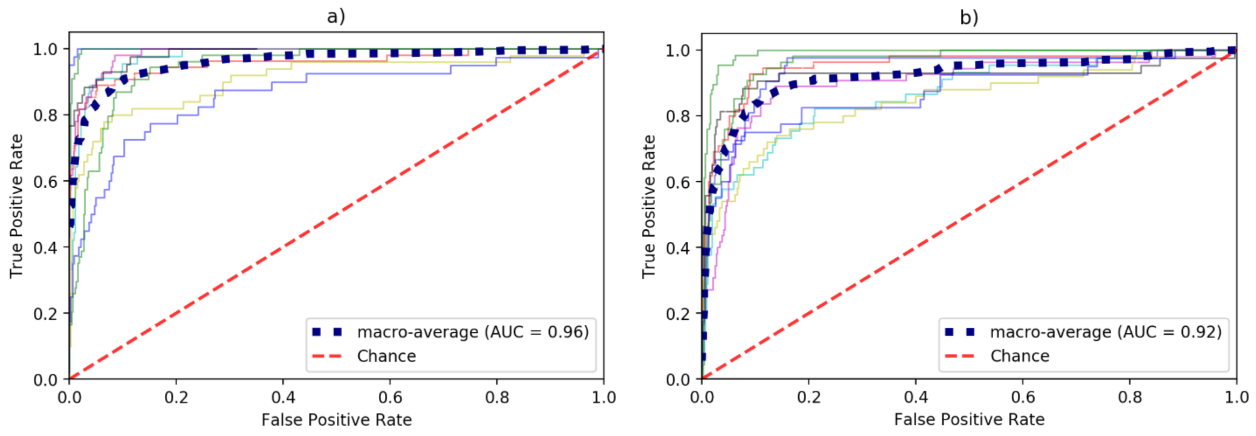

Figure 3. ROC analysis for multiclass classification. ROC curve is plotted for each class and then macro-average curve is calculated. a) Scikit-learn LR results. b) LR results.

Lastly, a summary of the accuracy is presented.

Task	SKLR	LR	LR with 0.9M
Breast (binary)	0.949	0.975	0.965
Monks (binary)	0.393	0.571	0.554
Glass (multi)	0.589	0.570	0.598
MNIST (multi)	0.824	0.752	0.774

Table 1. Average accuracy for each dataset with the corresponding model.

4. Analysis

4.1. Evaluation of the momentum parameter

Making several experiments with different values of momentum we concluded that as we decreased the value of momentum the Error and Mean performance is getting closer to the Error and Mean performance of Logistic Regression they almost match with momentum 0.1. On the other hand when we increase the momentum both Error and Mean performance differentiate from the Logistic Regression's graphs respectively.

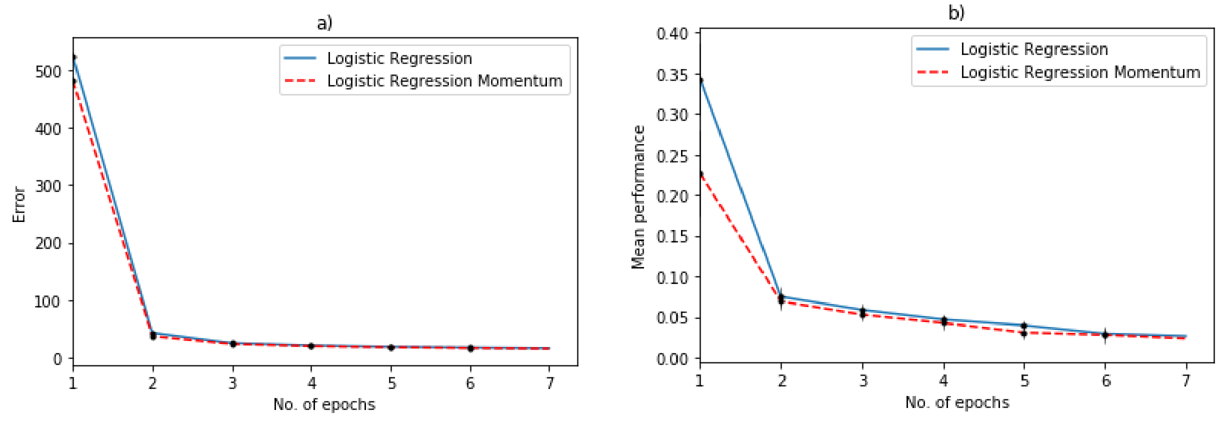

Figure 4 a) Cross entropy error and b) mean performance, momentum 0.1

4.2. Evaluation against sklearn

This experiment shows that our implementation of the logistic regression is on par with sklearn's. In the binary classification problem, the performance of LR versus SKLR is within the margin of error, as it is shown in the figure 3.2.1, they share the same mean AUC. Even though, it can be argued that LR is slightly more stable since it does not have as much of variations as SKLR between the cross-validation folds.

Meanwhile, in the multiclass case, SKLR performs better by a fair margin, presumably because of the more sophisticated normalization techniques (L1 and L2) and more optimized learning algorithms [7]. Which are coded in C, making SKLR orders of magnitude faster than our implementation in python.

5. Conclusions

From the above analysis, we can conclude that we have correctly implemented both the standard logistic regression model as well as the multiclass variant using OvR. The momentum parameter revealed to be a minor improvement of the SGD, since it only affects the error in the first few steps, without any major benefit in the convergence to the minima.

The multiclass extension's performance is still not as good as the results that was achieved in the binary model, mainly because of the initial assumption that transforms the problem back to a simpler one. Future work could explore other approaches to the multiclass case. For example, it is possible to change the sigmoid function with a Softmax function, enabling the model to be a true multiclass classifier since its output is the probability for each class, with the sum equal to 1.

References

Journal of Biomedical Informatics Volume 35, Issues 5–6, October 2002, Pages 352-359.

Expert Systems with Applications Volume 34, Issue 1, January 2008, Pages 366-374.

Geoffrey Hinton. On the importance of initialization and momentum in deep learning, 2003.

Book, Introduction to Machine Learning Ethem Alpaydin.

Fawcett, Tom (2005); An introduction to ROC analysis, Pattern Recognition
Ferri C., Hernández-Orallo J., Salido M.A. (2003) Volume under the ROC Surface for Multi-class Problems. In: Lavrač N., Gamberger D., Blockeel H., Todorovski L. (eds) Machine Learning: ECML 2003.

Scitkit-learn documentation. LogisticRegression.