**Project 4 Machine learning**

Binomial logistic regression with multi-dimensional input

*Submitted by* Siddhartha Gupta, Sittal Aryal and Trey Smith

**Abstract:**

In this report we provide a summary of our implementation of binomial logistic regression with multi-dimensional inputs using MATLAB R2017b. We will first discuss the critical aspects of our implementation including our choice of data set and implementation strategy. This is followed by the evaluation of our implementation using synthetic data and then for the case of real world data. Finally, we provide a general summary of our results and discuss the training and testing times.

**Introduction:**

Binomial logistic regression can be used as a binomial classifier to discriminate between data. It works by fitting a sigmoidal curve to the input data which can be multi-dimensional. The standard logistic curve is expressed as:

The output in the case of logistic regression is always between (0,1) because of the monotonic and asymptotic nature of the sigmoidal curve. For a given target vector *t,* and feature vector x logistic functions can give the probability . Thus, logistic regression can be used to model the probability of the target being 0 or 1 given experimental data. This makes it useful for classification purposes.

The generalized linear logistic function for our case can be written as:

Here, *w* indicates the weight vectors and x indicates the input data or the feature vector. Y is the target variable whose probability needs to be estimated given X.

**Implementation of logistic regression model:**

There are various approaches by which a logistic regression model’s parameter can be estimated given a training set of input and output data such as Newton’s method, gradient descent, stochastic gradient descent, minibatch etc. In this study we have chosen the gradient descent and the stochastic gradient descent method to perform logistic regression. Details of these algorithms can be found in the literature. Gradient descent and stochastic descent were chosen for their simplicity of implementation and for comparing two different approaches. Other approaches such as gradient descent with regularization require extra hyperparameters which may need to be optimized and as a result we decided not to go for them.

We first apply both the gradient descent and the stochastic descent algorithm to synthetic data and compare the results. Finally, we apply the above two algorithms to real data with a feature vector length of 5 (multi-dimensional). We validate our results using hold-out method in the case of real data and show the time taken by various algorithms.

Additional, we also applied *fminunc* which is a custom MATLAB function to benchmark our implementation of the algorithms for the case of real-data. We now begin by discussing the case of synthetic data.

**Hardware and software:**

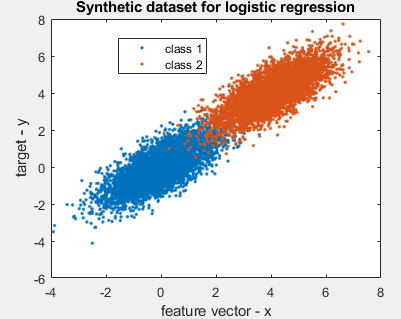
For the purpose of this exercise we used MATLAB R2017B on an Alienware laptop with Windows 10 and 32 GB RAM and core i7 processor with 4GB graphic card. This enabled us to easily run iterations of the order of ~ 3\*107 in small amounts of time (few minutes).

**Part A. Validation of logistic regression using synthetic data:**

The synthetic data was generated using multivariate normal random numbers with means of [0 0] and [4 4] and standard deviation of [1 0.75]. The data obtained is shown in **figure 1 a** and clearly indicates overlap between the two otherwise distinct regions.

Both Stochastic descent implementation and Gradient descent implementation were tested on the synthetic data. In this preliminary case, we have calculated accuracy based on prior knowledge of the data. Therefore, we have defined accuracy for this preliminary case as:

**Classification accuracy % = mismatch with original classification / (total data-points) \*100**

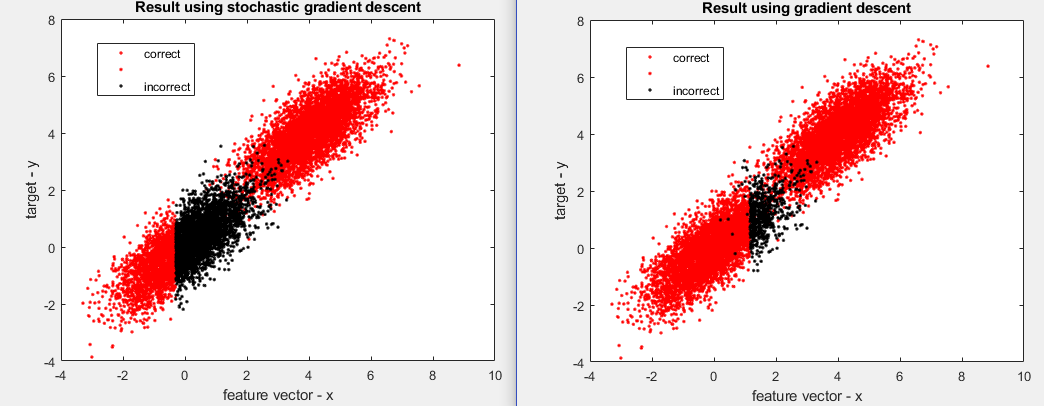
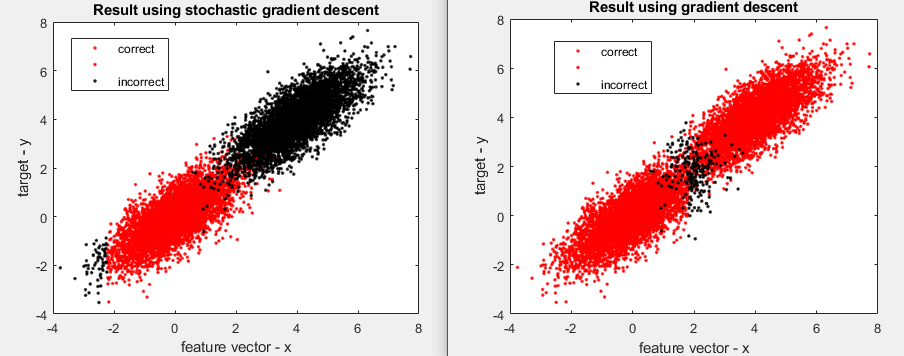


**Figure 1 Synthetic data with 10,000 points showing the two classes with overlap.**

**Table 1: A study of performance of stochastic gradient descent vs. gradient descent on synthetic data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Number of Iterations | Accuracy | Step Size | Time |
| Stochastic Gradient Descent | 300000 | 69.29 | 3\*(10^-5) | 38.82 |
| Gradient Descent | 300000 | 93.76 | 3\*(10^-5) | 42.33 |
| Stochastic Gradient Descent | 3000000 | 49.38 | 3\*(10^-5) | 423.5363 |
| Gradient Descent | 3000000 | 97.62 | 3\*(10^-5) | 458.5091 |

(a)



(b)

**Figure 2 (a) Stochastic decent (left) and gradient descent right after 300000 iterations showing match (red) and mismatch (black) (b) The same classification with 3000000 iterations showing improvement in gradient descent**

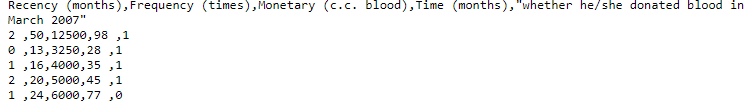
The choice of step size and number of iterations were made after various trials and keeping in mind the time-complexity of the problem and the computational feasibility.

**Part B. Performance of logistic regression on real data:**

To carry out logistic regression on real data, we obtained data about blood transfusion service center such that the total number of instances is 748 and the number of attributes was 5. The data was obtained from:

https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

A snapshot of the data columns with their headers is shown below.



To classify this data using our model we decided to use 3\*107 iterations each with a step size of 3\*10-5. The step size and the number of iterations were chosen heuristically based on our experience with the synthetic data. However, it does not reflect a choice which was made through a parametric study and thus there is a lot of scope for improvement.

We applied the gradient descent method in order to find the regression parameters. We also applied the *fminunc* function of MATLAB in order to independently check our implementation for this case.

We used all 4 attributes of the donor as our input parameters and the target variable as the probability of blood donation (0 or 1). In order to validate the model obtained after logistic regression we decided to use the **hold-out validation.**  Since our data set is relatively small ( ~ 748 points) and easy to split, hence hold-out is a fast and convenient choice for us.

Validation using hold-out method: We implemented hold-out method by splitting the data into 80% as training set and 20% as test set in an arbitrary manner. The training set was used to learn the model paramaeters and then the model was applied to the test case. The accuracy for both the training and test cases were calculated separately as the number of mismatches as a percentage of the data set.

The validation using the test and training set was done for results obtained using the MATLAB built in function as well as for Gradient descent.

The accuracy for the test case using *fminunc* or gradient descent was found to be ~ 91%, however the accuracy for the training case was lower at 75% . This accuracy is merely a measure of the mismatch or the accuracy of correctly classifying an observation. Lastly, while we agree that k-fold validation would be a more rigorous method of validation, we adopted holdout method for its simplicity. In order to check if the hold-out method was consistent we carried out the 80-20 split multiple times in a random way and measured the accuracy of applying logistic regression. In general, since we were able to acquire 80 – 90% accuracy, we believe that hold-out method is sufficient to assess the performance of our implementation.

Cross entropy loss function: We also implemented cross-entropy loss function calculations to quantify the loss estimate. Our results showed gradient descent performing better than stochastic descent in terms of number of iterations. It may be possible that computational overhead might change the balance of which algorithm is overall more efficient.

**Part C: Training and testing times**

For the case of real data, the training and testing times are a sensitive function of convergence / maximum number of iterations, as well as computational algorithm implemented. In our simplistic implementation of the gradient decent we observed time requirements of ~ 90 seconds for 3\*107 iterations in the training step. For the test case, the time required to calculate the test error was negligible < 0.001 s. Hence, the training step is the most time consuming by at least 3 orders of magnitude in our case.

Similarly, when we used the MATLAB function finance in place of our own implementation, the time required for the training step was ~ 0.36 s and the time required for test case was < 0.001 s. These results reflect the robustness of optimized algorithms in dealing with machine learning problems where a speedup of almost 30 times is achieved by using custom functions.

**Summary:**

In our study we looked at developing binomial logistic regression models using synthetic as well as real data. We also tried multiple algorithms and highlighted their advantages and disadvantages. We used hold-out validation in case of our real data as it affords simplicity and ease of use. In the case of learning algorithm, we implemented both the stochastic gradient and the gradient descent method for the synthetic data. However, since stochastic descent is sensitive to the choice of number of iterations and step size, we decided to go with gradient descent for the case of real data. We observe that as we increased number of iterations, our accuracy increased. However, we do realize that a lot of optimization needs to be done in our case to arrive at an efficient choice of step size and number of iterations. In sum, our regression models can achieve ~ 75% – 97 % accuracy for the choice of parameters.

**References:**

Wikipedia contributors. "Machine learning." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 1 Apr. 2018. Web. 3 Apr. 2018.

“Deep Learning.” *Coursera*, Deeplearning.ai, [www.coursera.org/specializations/deep-learning](http://www.coursera.org/specializations/deep-learning)?

Bishop, Christopher M., and Tom M. Mitchell. "Pattern Recognition and Machine Learning." (2014).