Group 1.6
Final Report
Battle of the Momentums

Akhil Ravichandran, Sriram Vellangallor Subramanian

Abstract—The objective of this project is to design experiments for synthetic data sets, and build two two-class machines for logistic regression one using hinge loss and the other using negative log-likelihood loss. This would enable us to the study effect of momentum on gradient descent to obtain the required minima in a logistic regression environment.

Index Terms—Max-margin machine, logistic regression , classification problem, gradient-descent algorithm, Nesterovs Accelerated Gradient, Polyaks Classical Momentum

1 Introduction

"According to Wikipedia, logistic regression, or logit regression, or logit model is a regression model where the dependent variable (DV) is categorical. This article covers the case of binary dependent variablesthat is, where it can take only two values, such as pass/fail, win/lose, alive/dead or healthy/sick. Cases with more than two categories are referred to as multinomial logistic regression, or, if the multiple categories are ordered, as ordinal logistic regression. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Thus, it treats the same set of problems as probit regression using similar techniques, with the latter using a cumulative normal distribution curve instead. Equivalently, in the latent variable interpretations of these two methods, the first assumes a standard logistic distribution of errors and the second a standard normal distribution of errors."[1]

2 Algorithm

According to Wikipedia, Logistic regression is computed based on the following sigmoid formula:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

t can be expressed as:

$$t = \beta_0 + \beta_1 x$$

t can be also our n-deminsional linear function with respect to x

$$t = \beta_0 + \beta_1 x + \beta_2 x^2 + +\beta_3 x^3 .. + \beta_n x^n$$

the logistic function and required function for predicting y or dependent variable bases on n Independent variables is:

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

The inverse of this function, i.e. the logit, is defined as:

$$g(F(x)) = \ln\left(\frac{F(x)}{1 - F(x)}\right) = \beta_0 + \beta_1 x,$$

By exponentiating both the sides,

$$\frac{F(x)}{1 - F(x)} = e^{\beta_0 + \beta_1 x}.$$

This is the equivalent.

3 Logistic Regression

3.1 Pseudocode

1) Setup
   - Initialize, the unknown function to a weight vector, assume 0
   - Set the stopping criteria to earlier of the events viz. 100 iterations and reaching a threshold value (assume 10^{-3})
2) Iteratively compute the weight vector till the convergence condition is satisfied with respect to cost function designed for loss.

4 Hinge loss

Wikipedia states "In machine learning, the hinge loss is a loss function used for training classifiers. The hinge loss is used for "maximum-margin"[1] classification, most notably for support vector machines (SVMs). For an intended output t = 1 and a classifier score y, the hinge loss of the prediction y is defined as

$$\ell(y) = \max(0, 1 - t \cdot y)$$

Note that y should be the ”raw” output of the classifier’s decision function, not the predicted class label. For instance, in linear SVMs, y=w.f x +b, where (w,b) are the parameters of the hyperplane and x is the point to classify.

It can be seen that when t and y have the same sign (meaning y predicts the right class) and —y—1, the hinge loss l(y)=0, but when they have opposite sign, l(y) increases linearly with y (one-sided error).”

4.1 Pseudocode

```
1) Setup
    • Initialize the loss and gradient to 0
2) For each point in the dataset compute the gradient
   descent and update the weights.
3) Return the computed loss value to the invoking
   logistic regression function.
```

5 Gradient Descent algorithm

According to Dr. Bryan Travis Smith, “Gradient descent algorithm assumes you have some function, and we will assume that it is some data (x) and weight (w) variables where the data if fixed but the weights are to be determined. The function $f(x, w)$ gives us some value for a given data, weight combination and we want to find the weights that give us the lowest value. Assuming the function is appropriately well-behaved, we can choose some weights then continue to update the weights using the following update rule:

$$w^i = w^i - \alpha \frac{\partial}{\partial w^i} f(x, w)$$

The index i is the index of the i^{th} feature in x . The negative sign is because we are trying to descend down the cost function. If the slope/derivative is positive, we want to move in the negative direction. If the slope/derivative is negative, we want to move in the positive direction.
The cost function if log-likelihood is used is as follows:

$$-ln\mathcal{L}(w) = -\sum_k [y_k ln(p(y_k = 1|x_k, w) \\ +(1 - y_k) ln(1 - p(y_k = 1|x_k, w))]$$

$$\frac{\partial}{\partial w^i} ln\mathcal{L}(w) - \sum_k (y_k - p(y_k = 1|x_k, w)) x_i^k$$

The cost function if hinge loss is used is as follows:

$$\ell(y) = \max(0, 1 - t \cdot y)$$

$$\frac{\partial \ell}{\partial w_i} = \begin{cases} -y_i \cdot x & \text{if } y_i w \cdot x < 1 \\ 0 & \text{otherwise} \end{cases}$$

The variables x and y are the features and labels for the training data, and k is the $k^t h$ example in the training data.

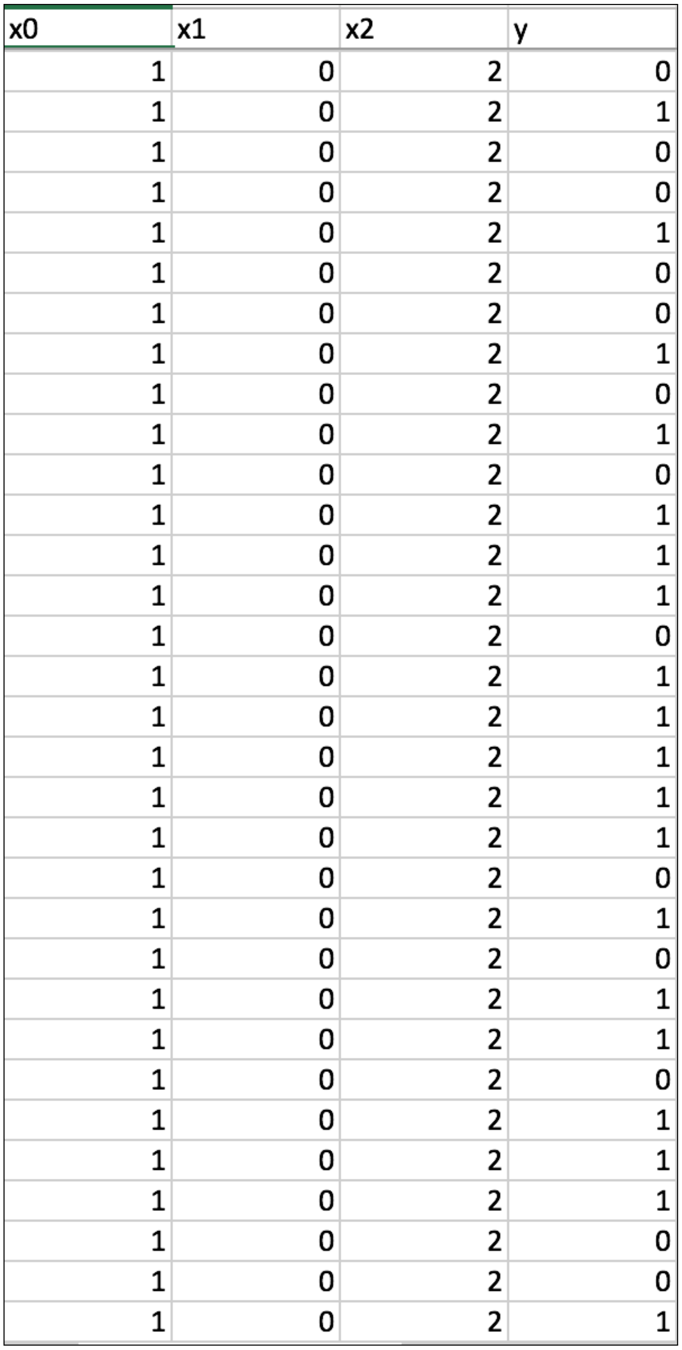
The update rule will be”:

$$w^i = w^i + \alpha \sum_k (y_k - p(y_k = 1|x_k, w)) x_i^k$$

5.1 Pseudocode

```
1) Setup
    • Initialize the weights to zeros.
    • Initialize loss to infinite.
2) Compute hinge loss using algorithm ¡algonum¿
3) Check difference of hinge loss and old loss
4) If loss value is greater than defined threshold
    • Update old loss to current value of hing loss
    • Repeat steps 2-4
```

6 About the Dataset



x0	x1	x2	y
1	0	2	0
1	0	2	1
1	0	2	0
1	0	2	0
1	0	2	1
1	0	2	0
1	0	2	0
1	0	2	1
1	0	2	0
1	0	2	1
1	0	2	0
1	0	2	1
1	0	2	1
1	0	2	1
1	0	2	0
1	0	2	1
1	0	2	1
1	0	2	1
1	0	2	1
1	0	2	1
1	0	2	0
1	0	2	1
1	0	2	0
1	0	2	1
1	0	2	1
1	0	2	0
1	0	2	1
1	0	2	1
1	0	2	1
1	0	2	0
1	0	2	0
1	0	2	1

Fig. 1. Data Used

The data set as in our Figure 1 is a set of random values generated using C using the rand() function using time, process ID as the seed within the [specified] range. It consists of 10 random data features, and the final column specifies the class label which is either 0 or 1.
We have considered a 50/50 set of both final label data(0,1),That is we have generated random data equally for our required dependent variables as this will give us the required amount of training data to train our logit function for all cases of the requested labels.

The data set consists of randomized numbers generated as follows depicted in

- A vector for the bias value
- 6 vectors of random integers generated between specific range
- An outcome vector

Also to avoid heavy co-linearity we have used random data sets generated in non-overlapping ranges for each dimension.2500 entries are generated, each for an outcome of 0 and 1. The rows are shuffled and stored as a CSV.Such a synthetic dataset provides a coverage of random values representing high dimensional data with equally partitioned outcomes
We have also used a k-fold cross-validation method to train our logit function and consider test data to predict the required dependent variables using the generated weight vectors.This enables us to use a varied data-set containing data from all parts of our sample to train our logit function. Hence our trained weight vectors must satisfy various partitions of data which could contain various anomalies and unique cases.[11]

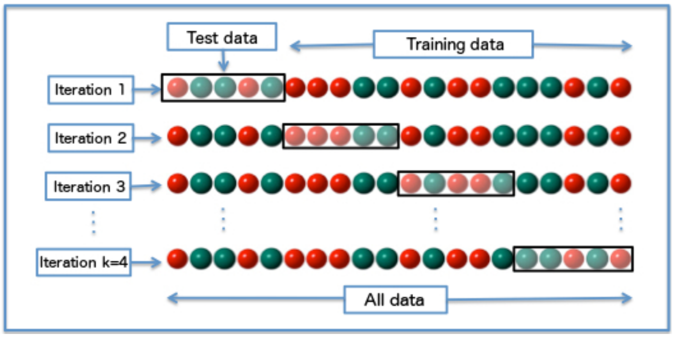


Fig. 2. K-fold Cross Validation[11]

7 Momentum

In our Study of Momentums we have used the suggested and required Polyaks Momentum and Nesterovs Accelerated Gradient Method which is used along with gradient decent calculation on every iteration to train our weight vectors.

7.1 Polyaks momentum

Polyaks momentum helps to converge the gradient descent function (loss) faster than the regular update technique for gradient descent

$$v_{t+1} = \mu v_t - \varepsilon \nabla f(\theta_t)$$

$$\theta_{t+1} = \theta_t + v_{t+1}$$

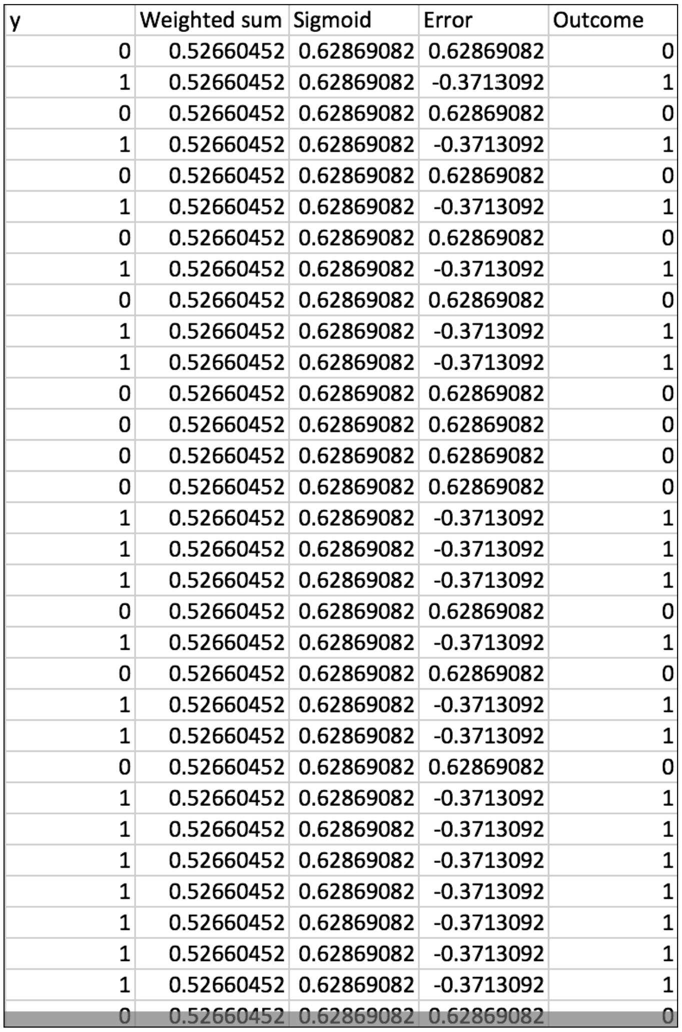
7.2 Nesterovs Accelerated Gradient Method

Nesterovs accelerated momentum helps to converge the gradient descent function faster than Polyaks momentum. Our understanding of this method is the feedback of momentum that is passed to the successive iterations of gradient descent

$$v_{t+1} = \mu v_t - \varepsilon \nabla f(\theta_t + \mu v_t)$$

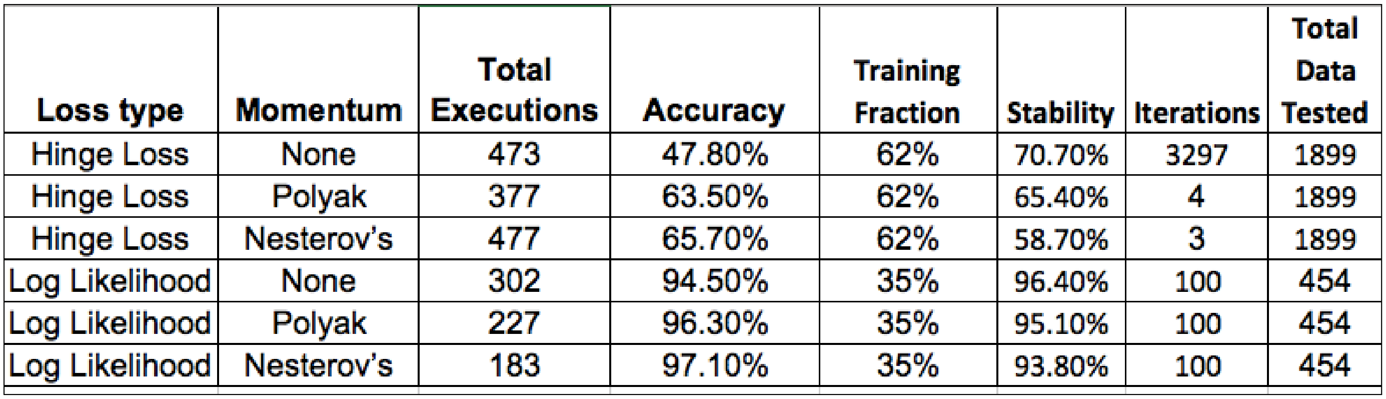
$$\theta_{t+1} = \theta_t + v_{t+1}$$

8 Results And Conclusion



y	Weighted sum	Sigmoid	Error	Outcome
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
0	0.52660452	0.62869082	0.62869082	0
0	0.52660452	0.62869082	0.62869082	0
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
1	0.52660452	0.62869082	-0.3713092	1
0	0.52660452	0.62869082	0.62869082	0

Fig. 4. Calculated Data



Loss type	Momentum	Total Executions	Accuracy	Training Fraction	Stability	Iterations	Total Data Tested
Hinge Loss	None	473	47.80%	62%	70.70%	3297	1899
Hinge Loss	Polyak	377	63.50%	62%	65.40%	4	1899
Hinge Loss	Nesterov's	477	65.70%	62%	58.70%	3	1899
Log Likelihood	None	302	94.50%	35%	96.40%	100	454
Log Likelihood	Polyak	227	96.30%	35%	95.10%	100	454
Log Likelihood	Nesterov's	183	97.10%	35%	93.80%	100	454

Fig. 3. Battle Of Momentum Results

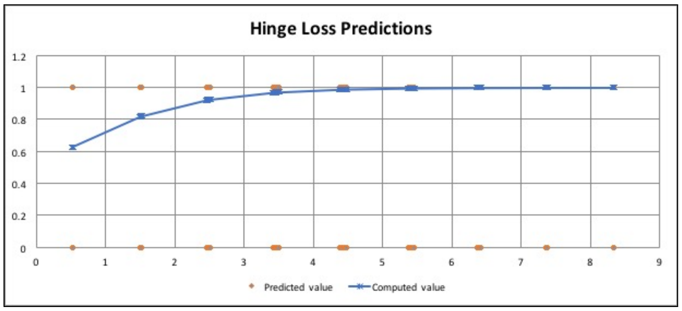


Fig. 5. Predicted Hinge Loss For Synthetic Data Used

The Figure 4 shows our weight vector and given x vector calculations along with the same.We additonally calculated the predicted sigmoid values,erroa and depicted the same sigmoid values generated in in 5.

The Figure 3 depicts our Momentum results where we have listed the required comparison.We have noted that by using Momentum the number of iterations for calculating gradient descent has reduced significantly and has helped us converge to our required results quickly.But we have also observed that stability of the obtained weight vectors and the prediction has varied differently for hinge loss and log-likelihood loss.

Acknowledgment

The authors wish to thank Dr. Baoxin Li , Teaching Assistant Ragav Venkatesan and Teaching Assistant Parag Chandakkar for their lectures and continuous support to learn Fundamentals of Statistics Learning.

References

[1] ”Logistic regression”, En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Logistic_regression. [Accessed: 19- Nov- 2016].

[2] ”Implementing Logistic Regression From Scratch Part 2: Python Code”, Bryan Travis Smith, Ph.D, 2016. [Online]. Available: https://bryantravissmith.com/2015/12/29/implementing-logistic-regression-from-scratch-part-2-python-code/. [Accessed: 19- Nov- 2016].

[3] ”Hinge loss”, En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Hinge_loss. [Accessed: 19- Nov- 2016].

[4] ”Logistic Regression Basic Concepts — Real Statistics Using Excel”, Real-statistics.com, 2016. [Online]. Available: http://www.real-statistics.com/logistic-regression/basic-concepts-logistic-regression/. [Accessed: 19- Nov- 2016].

[5] W. Dwinnell, W. Dwinnell and V. profile, ”Logistic Regression”, Matlabdatamining.blogspot.com, 2016. [Online]. Available: http://matlabdatamining.blogspot.com/2009/03/logistic-regression.html. [Accessed: 19- Nov- 2016].

[6] ”Multinomial logistic regression - MATLAB mnrfit”, Mathworks.com, 2016. [Online]. Available: https://www.mathworks.com/help/stats/mnrfit.html. [Accessed: 19- Nov- 2016].

[7] 2016. [Online]. Available: https://onlinecourses.science.psu.edu/stat504/node/149. [Accessed: 19- Nov- 2016].

[8] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. ICML (3), 28:11391147, 2013.

[9] ”Logarithmic Loss — Kaggle”, Kaggle.com, 2016. [Online]. Available: https://www.kaggle.com/wiki/LogarithmicLoss. [Accessed: 19- Nov- 2016].

[10] ”Margin (machine learning)”, En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Margin_(machine_learning). [Accessed: 19- Nov- 2016].

[11] ”Cross-validation (statistics)”, En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Cross-validation_(statistics). [Accessed: 06- Dec- 2016].

[12] Notes for EE392o - Subgradient Methods - Stephen Boyd, Lin Xiao, and Almir Mutapcic - Stanford University, Autumn, 2003