Logistic Regression with L_2 Regularization

Adrian Guthals, David Larson, Jason Yao
CSE 250B: Project #1
University of California, San Diego

January 24, 2013

Abstract

Logistic Regression with L_2 Regularization is analyzed as a classification learning method for data sets with real-valued features and binary labels. Stochastic Gradient Descent (SGD) and Limited-memory BFGS (L-BFGS) are used to find the regression parameters such that the Log Conditional Likelihood is maximized. The methods are applied to two common data sets: USPS-N (images of hand-written digits) and Web (statistics on HTML web pages). For the USPS-N data set, SGD had a (lower/equal/higher) test error than L-BFGS (xxx vs xxx) while for the Web data set, blah. These results indicate that blah blah blah about the two gradient methods as well as blah about logistic regression.

1 Introduction

Machine Learning (ML) is a sub-discipline of Artificial Intelligence (AI) that focuses on the development and analysis of methods that produce models from data. An often referenced example application of ML is email spam filtering, but there is a wide variety of real-world problems where ML has been successful used, from object detection using a Microsoft Kinect sensor [1] to forecasting of power output from a photovoltaic solar farm [2].

ML algorithms are plentiful in number, but some are more suitable than others depending on the type of problem. Logistic regression is a ML algorithm well-suited to problems involving a set of real-valued inputs and a binary (Bernoulli) output. A real-world example of such a problem is whether a baseball player will hit a home run based on a collection of statistics (e.g. batting average, height, weight). The statistics are the real-valued inputs, whether he hits a home run is the binary output (1=yes, 0=no), and the probability of whether a player hits a home run is the determined using logistic regression.

In this paper we analyze logistic regression's capability for classification learning using a previously study as a baseline [3]. Aside from recreating the previous study's results, we also extend their work to comparing two gradient-based optimization methods for use in conjuction with the logistic regression: Stochastic Gradient Descent (SGD) and Limited-memory Broyden-Fletcher-Goldfarb-Shannon (L-BFGS).

2 Design and Analysis of Algorithms

2.1 Logistic Regression

Given a set of real-valued features x with binary labels y (a training set), we can define logistic regression model as:

$$p_i = p(y_i|x_i; \beta) = \frac{1}{1 + \exp - [\sum_{j=0}^{d} \beta_j x_i j]} \tag{1}$$

where d is the dimensionality of a training example, and β_j are parameters of the regression. To learn the logistic regression, we need deteremine $\vec{\beta}$ such that the Log Conditional Likelihood (LCL) is maximized. In more formal notation:

$$\widehat{B} = \text{argmax}_\beta \text{LCL} \tag{2}$$

where LCL is defined as:

$$\text{LCL} = \sum_{i:y_i=1} log(p_i) + \sum_{i:y_i=0} log(1 - p_i) \tag{3}$$

and is maximized when its derivative with respect to each parameter β_j is zero. For a single example the partial derivative of the LCL is:

$$\frac{\partial}{\partial \beta_j} log L(\beta; x, y) = (y - p)x_j \tag{4}$$

2.2 L_2 Regularization

In order to prevent overfitting of machine learning, a penalty was imposed to regulate the values of the parameters. A regularization constant μ was introduced into the LCL objective function:

$$\hat{B} = \text{argmax}_\beta LCL - \mu ||\beta||_2^2 \tag{5}$$

where $||\beta||_2^2$ is the L_2 norm of the parameter vector. With this revision the derivative of the LCL becomes:

$$\frac{\partial}{\partial \beta_j} [log p(y|x; \beta) - \mu \sum_{j=0}^{d} \beta_j^2] = (y - p)x_j - 2\mu\beta_j \tag{6}$$

2.3   Gradient-Based Optimization

To efficiently maximize LCL, two gradient-based optimization methods are used: SGD and L-BFGS. The SGD was coded from scratched in Matlab while we used the popular minFunc toolbox [4] for the L-BFGS. For each of these algorithms, the input training data is formatted as a set of n examples $(x_1, \ldots, x_n)$ where each x_i is a real-valued vector of d features. Each x_i is correlated to a binary (Bernoulli) outcome y_i by a global parameter vector β of length $d + 1$.

2.3.1   SGD

Our SGD implementation first randomized the order of input examples to avoid repeated computation of random numbers and partitioned the input data into $x_1 \ldots x_k$ *training* examples and $x_{k+1} \ldots x_n$ *validation* examples. Then sequential mini-batches of size $\kappa < k$ taken from the training set were used to update the parameter vector β (initialized to all zero values) by the following equation. The constant μ quantifies the trade-off between maximizing likelihood and minimizing parameter values for L_2 Regularization. The update rule for β is:

$$\beta := \beta + \frac{\lambda}{\kappa} [-2\mu\beta + \sum_{i=1}^{\kappa} (y_i - p_i)x_i] \tag{7}$$

After each update of β, absolute change in the objective $\widehat{\beta}$ was computed over all validation examples with the following function.

$$\widehat{\beta} = \mu\|\beta\|^2 + \sum_{i=k+1}^{n} -\log(p_i^{y_i}(1-p_i)^{1-y_i}) \tag{8}$$

Convergence was reached when change in the objective reached a value less than ω. Convergence was also reached if the total number of epochs was greater than ε. With this configuration, the time to run the SGD is O(nd).

To improve the performance of the SGD, cross-validation was added. The pseudo-code for SGD with cross-validation is as follows:

1. Initialize all parameter values β_j

2. Initialize all conditional probabilities p_i

3. Initialize the objective function and its difference value

4. Randomly divide the example set into two groups: testing set of m rows and validation set of (n-m) rows

5. while(objective function difference $\geq$ theshold AND number of epochs < maximum epochs allowed

   (a) Randomly pick a sample of s rows out of the testing set
   (b) Update β_j for all features
   (c) Update the objective function

The algorithm was run ten times to generate 10 vectors of parameters. The ten vectors of parameters were averaged to calculate one vector (10-fold cross-validation). The result cross-validated vector was used to calculate the test error and its variance over all the instances. The test error was defined as the average value of the loss function:

$$\text{test error} = \sum_{i=1}^{n} -log(y_i|x_i; \beta) \tag{9}$$

2.3.2 L-BFGS

Limited-memory Broyden-Fletcher-Goldfarb-Shannon (L-BFGS) is a quasi-Newton optimization method used to find local extrema. L-BFGS provides performance gains over Newton's method by approximating (instead of solving directly) the Hessian matrix of the objective function.

3 Design of Experiments

We apply logistic regression to two training data sets:

1. USPS-N: images of hand-written digits as 256-dimensional vectors [5]

2. Web: statistics on HTML web pages [6]

3.1 Pre-Processing Training Data

3.1.1 USPS-N

The USPS-N data set consists of 10 tables, where each table has features and labels for one digit. We are only interested in digit 9 for this study, so we concat all 10 tables together into one training set, set the labels for all examples for digit 9 to 1 and set the labels for all other examples to 0 (i.e. only examples from digit 9 have positive labels). Also, features of this data set are RGB color values and so we normalize all features relative to the max RGB value (255).

Table 1: Comparison of hyperparameters of SGD and their test error. λ was evaluated using a 10% subset of each training set with a fixed value of μ and the optimal λ was chosen such that the test error was minimized. Once λ was chosen, μ was varied to determine the μ that minimizes the test error.

Training Set	λ	μ
USPS-N	1.0	1.0
USPS-N	1.0	1.0
USPS-N	1.0	1.0
Web	1.0	1.0
Web	1.0	1.0
Web	1.0	1.0

3.1.2   Web

The Web data set has features with binary values, which are compatible with logistic regression. The only pre-processing required for the Web data set was removal of non-essential characters (e.g. colons).

3.2   Hyperparameters

3.2.1   Learning Rate

SGD uses a fixed learning rate λ to control the change the change in LCL. We ran the SGD method using a range of λ values over a 10% subset of each training data set, using the test error of the SGD as a metric for determining which λ is optimal. The optimal λ for the USPS-N and Web training data sets was YYY and YYY.

3.2.2   Regularization

Once we determined an optimal λ, we tried a range of μ values with a randomly selected 30% subset of each training data set. We selected μ such that the test error was minimized, which was YYY and YYY for the USPS-N and Web training data sets.

3.2.3   Convergence

Convergence was reached when change in the objective function was less than ω or the total number of epochs was greater than ε (which ever came first)

4   Results of Experiments

Results of experiments. Comparison of methods and data sets.

4.1   USPS-N

How did SGD and L-BFGS perform on the USPS-N data sets.

4.2   WEB
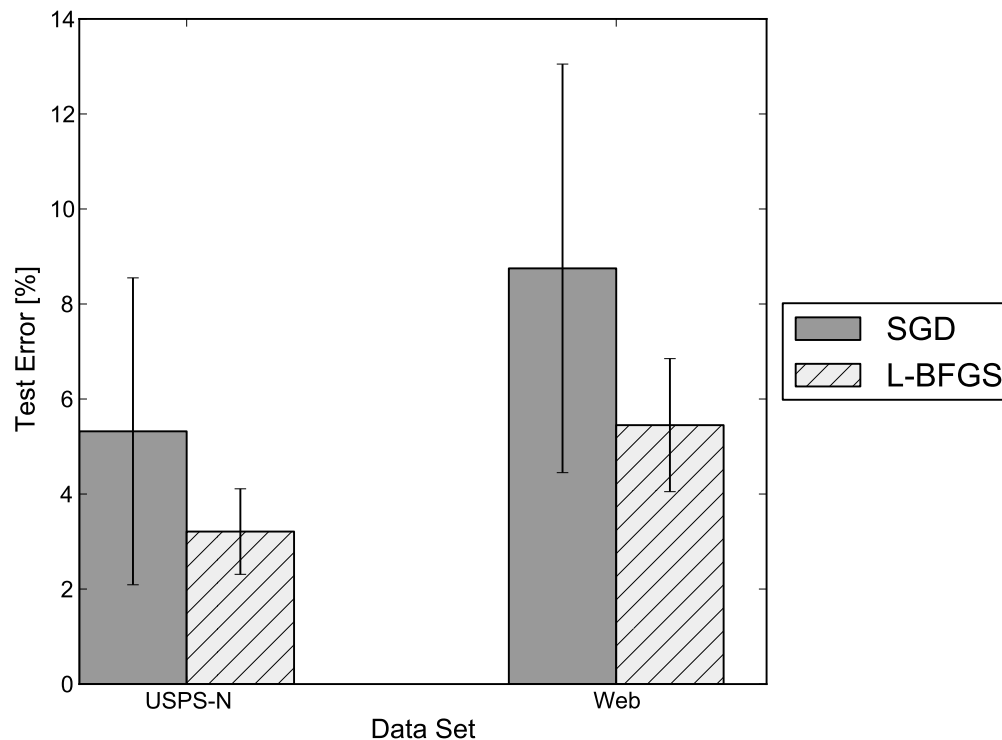
How did SGD and L-BFGS perform on the Web data sets.

Figure 1: Comparison of SGD and L-BFGS for USPS-N (left) and Web (right) data sets.

5   Findings and Lessons Learned

Findings and lessons learned.

References

[1] A. Janoch, S. Karayev, Y. Jia, J. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3-d object dataset: Putting the kinect to work," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, nov. 2011, pp. 1168 –1174.

[2] H. T. Pedro and C. F. Coimbra, "Assessment of forecasting techniques for solar power production with no exogenous inputs," *Solar Energy*, vol. 86, no. 7, pp. 2017 – 2028, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X12001429

[3] N. Ding and S. Vishwanathan, "t-logistic regression," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., 2010, pp. 514–522.

[4] M. Schmidt. [Online]. Available: http://www.di.ens.fr/~mschmidt/

[5] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, May 1994.

[6] M. Pazzani, M. J., and D. Billsus, "Syskill and webert: Identifying interesting web sites," in *Proceedings of the National Conference on Artificial Intelligence*, 1996.