Logistic Regression with L_2 Regularization

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

1 INTRODUCTION

This paper will attempt to recreate the results by [1].

Highlight what Logistic Regression, why is it used and where is it used (real-world applications). Also mention potential pitfalls of the method (e.g. overfitting).

2 DESIGN AND ANALYSIS OF ALGORITHMS

Stochastic gradient descent (SGD) and Limited-memory Broyden-Fletcher-Goldfarb-Shannon (L-BFGS) were each implemented in Matlab to maximize the total conditional log likelihood of each training data set. In brief, SGD incorporated a fixed learning rate λ to control the change in log likelihood, which was averaged over random mini-batches of size κ. To control over-fitting, logistic regression was used and change in the objective function was evaluated on a separate validation dataset containing 30% of all training examples selected at random. 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). (Add brief overview of L-BFGS and cite MinFunc's implementation)

minFunc[2]

For each of these algorithms, the input training data is formatted as a set of n examples $x_i \ldots x_n$ where each x_j 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 $d+1$. We assume this correlation follows the model below where $x_{i0} = 1$ for all i.

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

2.1 L2 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:

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

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

$$\frac{\partial}{\partial \beta_j}[logp(y|x;\beta) - \mu \sum_{j=0}^{d} \beta_j^2] = (y - p)x_j - 2\mu\beta_j \tag{3}$$

2.2 STOCHASTIC GRADIENT DESCENT

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.

$$\beta := \beta + \frac{\lambda}{\kappa}\,[-2\mu\beta + \sum_{i=1}^{\kappa}(y_i - p_i)\,x_i] \tag{4}$$

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

$$\widehat{\beta} = \mu\|\beta\|_2 + \sum_{i=k+1}^{n} - \log(p_i^{y_i}(1 - p_i)^{1-y_i}) \tag{5}$$

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).

2.2.1 Cross-Validation

SGD with cross-validation is represented by the following pseudo-code:

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 with 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:

$$\text{test error} = \sum_{i=1}^{n} -log(y_i|x_i;\beta) \tag{6}$$

2.3 LIMITED-MEMORY BFGS

Limited-memory Broyden-Fletcher-Goldfarb-Shannon (L-BFGS) is a quasi-Newton optimization method used to find local extrema.

3 DESIGN OF EXPERIMENTS

3.1 PRE-PROCESSING TRAINING DATA

normalization, concatation?

3.1.1 USPS-N

Digit 9 vs other digits

3.1.2 Web

anything special?

Table 1: Hyperparameters.

	λ	μ
SGD	1.0	1.0
SGD	0.1	1.0
SGD	0.01	1.0

3.2 HYPERPARAMETERS

3.2.1 Learning Rate

How did we determine λ (the learning rate).

3.2.2 Regularization

How did we determine μ (the regularization constant).

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.

4.2.1 Figures

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] 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.

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