Penalizied Logistic Regression for Classification

Gennady G. Pekhimenko
Department of Computer Science
University of Toronto
Toronto, ON M5S3L1
pgen@cs.toronto.edu

Abstract

Investigation for using different penalty functions (L_1 - absolute value penalty or lasso, L_2 - standard weight decay or ridge regression, weight elimination etc.) on the weights for logistic regression for classification. 5 data sets from UCI Machine Learning Repository were used.

1 Introduction

For supervised learning algorithms over-fitting is usually a potential problem which we need to solve. It is well-known fact that for unregularized discriminative models fit via training error minimization, sample complexity (training set size needed to learn ”well” in some sense) grows linearly with the VCD (Vapnik Chervonenkis Dimension). Moreover, VCD for most models grows linearly in the number of parameters, which usually grows at least linearly in the number of input features. So, if we do not have training set size large relatively to the input dimension, we need some special mechanism - such as regularization, which makes the fitted parameters smaller to prevent over-fitting [4](p.1).

In this paper, we mostly focused on study the typical behavior of two well-know regularization methods: *Ridge Regression* or L_2 penalty function and *Lasso* or L_1 penalty function. L_2 penalty function uses the sum of the squares of the parameters and Ridge Regression encourages this sum to be small. L_1 penalty function uses the sum of the absolute values of the parameters and Lasso encourages this sum to be small.

We are going to investigate these two regularization techniques for classical classification algorithm: Logistic Regression for different data sets.

2 Logistic Regression

Logistic Regression is a popular linear classification method. It's predictor function consists of a transformed linear combination of explanatory variables.

Our model: y - multinomial random variable, **x** - feature vector, θ - weigth vector. y's posterior is the ”softmax” of linear functions of the feature vector **x**.

$$p(y = k|\mathbf{x}, \theta) = \frac{\exp(\theta_k^\top \mathbf{x})}{\sum_j \exp(\theta_j^\top \mathbf{x})} \tag{1}$$

To fit this model we need to optimize the *conditional log-likelihood* $\ell(\theta; D)$:

$$\ell(\theta; D) = \sum_n \log p(y = y^n|\mathbf{x}^n, \theta) = \sum_{nk} y_k^n \log p_k^n \tag{2}$$

where $y_k^n \equiv [y^n == k], p_k^n \equiv p(y = k|\mathbf{x}^n)$.

To maximize the log-likelihood we set it's derivatives w.r.t to θ to zero.

$$\frac{\partial \ell(\theta)}{\partial \theta_i} = \sum_n (y_i^n - p_i^n)\mathbf{x}^n \tag{3}$$

Typically some method like conjugate gradients can be used then to maximize log-likelihood. It needs (2) and (3): value of ℓ and it's derivatives.

3 Regularization

3.1 Penalty Functions

As was mentioned before, using logistic regression without any additional regularization can not guarantee you good results, because of possible over-fitting. This usually happens when the number of observations or training examples (*m*) is not large enough, compared with the number of feature variables (*n*) [3](p.4). So, we need some kind of regularizer in order to get stable classifier. One of the way is using penalization.

The general motivation behind penalizing the likelihood - the approach logistic regression can be regularized in a straight forward manner - is the following: Avoid arbitrary coefficient estimates $\widehat{\theta}$ and a classification that appears to be perfect in the training set, but is poor in the validation set. Penalization aims are improving prediction performance in a new data set by balancing the fit to the data and the stability of the estimates [1](p.4).

Now we are going to penalize logistic regression from (2) in a such way:

$$\ell^*(\theta) = \ell(\theta) - \lambda J(\theta) \tag{4}$$

where

$$J(\theta) = \sum_k \alpha_k \psi(\theta_k), \tag{5}$$

$\alpha_k > 0$.

Usually the penalty function ψ is chosen to be symmetric and increasing on $[0, +\infty]$. Furthermore, ψ can be convex or non-convex, smooth or non-smooth. A good penalty function should result in *unbiasedness*, *sparsity* and *stability* [6](p.20). Most popular penalty functions are in the Table 1. More examples are in [6](p.21).

Now, we are going to consider two special cases of function $J(\theta)$ that are usually used in practice: $J(\theta) = \sum_k |\theta_k|$ (or ℓ_1) and $J(\theta) = \sum_k \theta_k^2$ (or ℓ_2).

3.2 ℓ_2-regularization

Over-fitting often tends to occur when the fitted model has many feature variables with relatively large weights in magnitude. To prevent this situation we can use weight decay method (or ridge regression). Let $J(\theta) = \sum_k \theta_k^2$ (or ℓ_2). The result of using such a function $J(\theta)$ is classifier with smaller values of weights and often better generalization ability. We can also prune this classifier: weights with magnitudes smaller than some certain threshold can be considered redundant and removed. This regularization technique is very easy to implement and often works pretty well. It is still very often used for different applications: in [8] for detecting genes interaction, in [6] for classification of microarray data, in [1] for gene expression analysis.

3.3 ℓ_1-regularization

However, ℓ_2 usually leaves most of the weights θ_i non-zero that can be a problem when we have a lot features and want to get sparse θ-vector. It can be also a native characteristic of our data, when we have a large amount of features, but only relatively small number of them is sufficient to learn the target concept. So, Let $J(\theta) = \sum_k |\theta_k|$ (or ℓ_1). In this case a lot of weights can become zero, so, that gives a model with relatively sparse vector θ. We can think about this quality as a possibility to select important features [3](p.6). Logistic model with sparse vector of weights is simpler and

Table 1: Examples of penalty functions

Penalty function	Smoothness at 0	Convexity	Authors
$\psi(\theta) = \|\theta\|$	yes	$\theta'(0^+) = 1$	Rudin(1992)
$\psi(\theta) = \|\theta\|^\alpha, \alpha \in (0, 1)$	yes	$\theta'(0^+) = \infty$	Saquib(1998)
$\psi(\theta) = \alpha\|\theta\|/(1 + \alpha\|\theta\|)$	yes	$\theta'(0^+) = \alpha$	Geman(92, 95)
$\psi(\theta) = \|\theta\|^\alpha, \alpha > 1$	yes	yes	Bouman(1993)
$\psi(\theta) = \alpha\theta^2/(1 + \alpha\theta^2)$	no	yes	McClure(1987)

more parsimonious and it's not surprising that it can outperform ℓ_2, especially when the number of observation is smaller than the number of features [4](p.5).

In this paper we are going to compare both these regularization methods on different common classification data sets (UCI data sets).

4 Experiments

4.1 Data sets

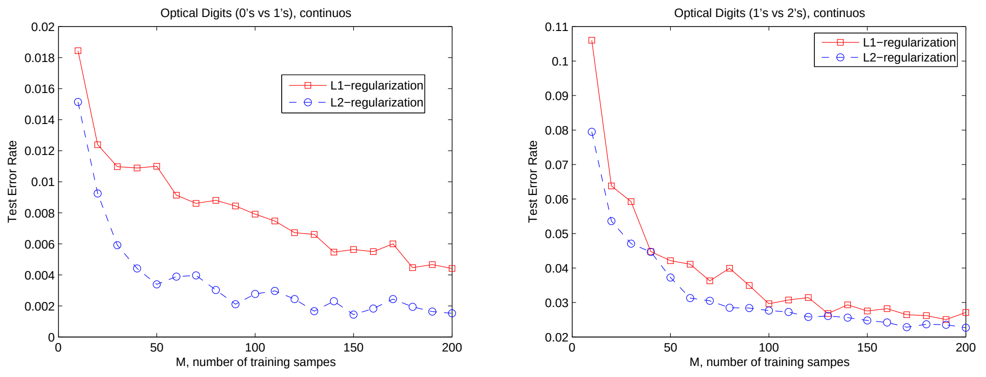

Figure 1: (A), (B)

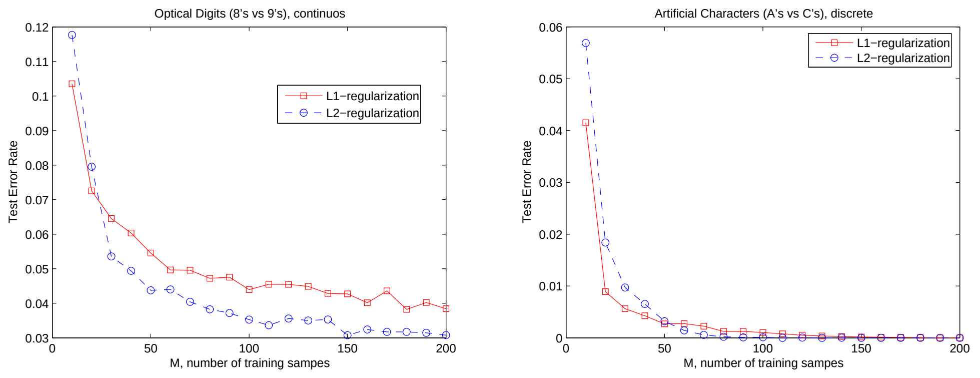

Figure 2: (C), (D)

Our experiments used data sets from the UCI Machine Learning Repository. The first data set was the Optical Digits data set that was the extraction of handwritten digits from a preprinted form. It consisted of 3823 training samples and 1797 test ones; had 64 attributes and 10 classes (all digits). The second data set was the Artificial Characters database generated by first order theory that described 10 letters of the English alphabet (A,C,D,E,F,G,H,L,P,R). There were 1000 instances (100

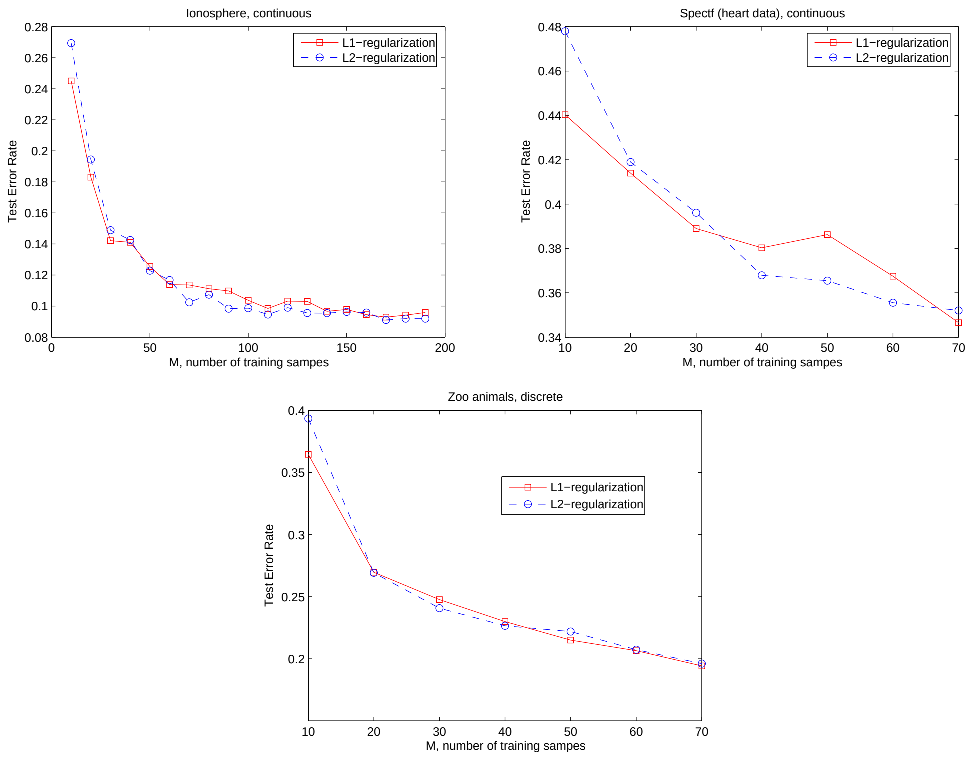

Figure 3: (E), (F), (G)

per class) as learning set and 5000 instances (500 per class) as test one. Every instance had 96 binary attributes (12*8 matrix).
Then, we used Johns Hopkins University Ionosphere database of radar returns from the ionosphere. There were two classes for this database: ”good” radar return (showing evidence of some type of structure in the ionosphere) and ”bad” return (signals passed through the ionosphere). Number of Instances: 351. Number of Attributes: 34 (all of them are continuous). We used 200 of them for training (carefully split almost 50% positive and 50% negative) and the rest for test.
Fourth data set (SPECTF) described diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. It had 267 instances that were described by 45 attributes.
And the last one was Zoo database (classification for animals in the zoo). It had 101 instances (we divided them 80 to 21: learning and test respectively) and 17 attributes (15 boolean attributes and 2 numeric).

4.2 Results

We used simple self-written Logistic Regression algorithm (with two variants of regularizing). For determining optimal λ parameter we used cross-validation. Results of 7 experiments are in Figures 1, 2, 3.
These experiments show the difference in error rate for ℓ_1 and ℓ_2 regularization methods. Because we expected to see the difference when the number of samples was comparatively small to the number of features, executions for different amount of training instances were made (from 10 to 200 (or less) with step 10). To avoid problems with ”bad” choice, for every number of training samples we made 100 random splits of training data and then reported the average error for all cases. So, our test error is formally the generalization error (for 100 random splits' experiments). The difference (ℓ_1 outperforming) was not so dramatic as in [4], but in [4](p.5) they considered cases for 1 and 3 relevant features when the total amount of features is from 100 to 1000, that was not true for all of our experiments.

In Figure 1 there is a result for classification 0 vs 1 and 1 vs 2 digits. In this execution ℓ_2 was a bit better than ℓ_1 (0.2-0.4%). The reasons for this are, first, digits are not similar (in case of 8 vs 9 situation is different) and, second, we used conjugate gradient method for both methods. In [2](p.5-6) & [4](p.3-4) specific algorithms were used for ℓ_1 case. [3](p.6-8) gives a short overview of methods that can be used to increase performance for ℓ_1-regularization.
In Figure 2 we see the result for 8 vs 9 digits (C) and ”a” vs ”c” letters (D). When the number of samples is small (10-30) relatively to the number of features (64,96), ℓ_1 outperformed ℓ_2; and then error rate is almost equal.
In Figure 3 are the results for experiments with 3 last data sets: Ionosphere, Spectf and Zoo (E,F,G respectively). For (E) & (F) results are similar to (C) & (D) - for the case when number of samples is small (10-30) ℓ_1 outperformed ℓ_2. For (G) results for both methods are almost the same.

5 Short Overview of Related Works

In 3.2 we've mentioned 3 papers that used ℓ_2 regularization. In [4](p.7-8) Ng proved an interesting fact that for logistic regression with ℓ_1 regularization sample complexity grows only *logarithmically* in the number of irrelevant features, while ℓ_2 grows at least linearly (in the worst case). [3] proposed specialized method for solving ℓ_1-regularized logistic regression in every efficient way for all size problems. In [8](p.24) there is a comparison of logistic regression with naive bayes using few data sets that we used in this paper' experiments. [1](p.6-7) gives efficiency' comparison of using different methods (cross-validation, *Akaike Information Criterion etc*.) for choosing λ parameter. [2](p.6-7) is a very good natural example of the case when ℓ_1 is much quicker than ℓ_2, because of the sparsity (Text Categorization problem).

6 Concluding Remarks

In this paper we compared two well-known and widely used methods of regularization. The idea was to understand which method is preferable for different types of data, different amount of training samples and which additional algorithms can improve these methods. According to this research (and using results from the papers in the References), we can conclude that ℓ_1 is a good choice when you do not have enough training samples and your input vector has relatively many features or when you need sparsity (for weight vector) to decrease the time that your algorithm needs to get the result. On the other hands, ℓ_2 is still a good choice for the tasks where most of the weights during learning process are relevant. It can be interesting also to investigate the same things for other classifiers and ML algorithms like *Neural Networks* (NN), *Support Vector Machines* etc. Part of these experiments was also done for NN and the results are very similar (digits and characters data sets).

References

[1] M.G. Schimek, (2003) Penalized Logistic Regression in Gene Expression Analysis, *Proc. The Art of Semi-parametrics Conference, Oct. 2003.*

[2] Genkin A., Lewis D.D., & Madigan D. (2005) Sparse Logistic Regression for Text Categorization *DIMACS Working Group on Monitoring Message Streams. Project Report, April 2005.*

[3] Koh K., Kim S.-J.& Boyd, S. (2006) An Interior-Point Method for Large-Scale ℓ_1-Regularized Logistic Regression *Submitted to Journal of Machine Learning Research, July 2006.*, pp. 1-38.

[4] Ng. A. J., (2004) Feature selection, L1 vs. L2 regularization, and rotational invariance *In Proceedings of the Twenty-first International Conference on Machine Learning, 2004.*

[5] Lee S.-I., Lee H., Abbeel P. & Ng. A. J., (2006) Efficient L1 Regularized Logistic Regression. *In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06), 2006.*

[6] Antoniadis A. (2003) Penalized Logistic Regression and Classification of Microarray Data *Computational and Statistical Aspects of Microarray Analysis, Workshop, Milan, May 2003 p.1-32*

[7] Mitchell T.M. Machine Learning Course 10-701, Logistic Regression *Center for Automated Learning and Discovery, Carnegie Mellon University September 29, 2005*

[8] Park M.Y. & Hastie T. (2006) Penalized Logistic Regression for Detecting Gene Interactions *Technical Report, Dept. of Statistics, Stanford University, September 2006.*