Regularized Logistic Regression for Mind Reading with Parallel Validation

Heikki Huttunen, Jukka-Pekka Kauppi, Jussi Tohka
Tampere University of Technology
Department of Signal Processing
Tampere, Finland
{jukka-pekka.kauppi,heikki.huttunen,jussi.tohka}@tut.fi

Abstract

In our submission to the mind reading competition of ICANN 2011 conference, we concentrate on feature selection and result validation. In our solution, the feature selection is embedded into the regression by adding a ℓ_1 penalty to the classifier cost function. This can be efficiently done for regression problems using the LASSO, which generalizes also to classification problems in the logistic regression classification framework. A special attention is paid to the evaluation of the performance of the classification by cross-validation in a parallel computing environment.

1 Introduction

Together with the ICANN 2011 conference, a competition for classification of brain MEG data was organized. The challenge was to train a classifier for predicting the movie being shown to the test subject. There were five classes and the data consisted of 204 channels. Each measurement was one second in length and the sampling rate was 200 Hz. From each one-second measurement, we had to derive discriminative features for classification. Since there were only a few hundred measurements, the number of features will easily exceed the number of measurements, and thus the key problem is to select the most suitable features efficiently. We tested various iterative feature selection methods including the *Floating Stepwise Selection* [4] and *Simulated Annealing Feature Selection* [1], but obtained the best results using Logistic Regression with ℓ_1 penalty also known as the *LASSO* [5].

2 Logistic Regression with the LASSO

Our classifier is based on the logistic regression model for class probability densities. The logistic regression models the PDF for the class $k = 1, 2, \ldots, K$ as

$$p_k(\mathbf{x}) = \frac{\exp(\boldsymbol{\beta}_k^T \mathbf{x})}{1 + \sum_{j=1}^{K} \exp(\boldsymbol{\beta}_j^T \mathbf{x})}, \text{ for } k \neq K, \text{ and} \tag{1}$$

$$p_K(\mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{K} \exp(\boldsymbol{\beta}_j^T \mathbf{x})}, \tag{2}$$

where $\mathbf{x} = (1, x_1, x_2, \ldots, x_p)^T$ denotes the data and $\boldsymbol{\beta}_k = (\beta_{k0}, \beta_{k1}, \beta_{k2}, \ldots, \beta_{kp})^T$ are the coefficients of the model.

The training consists of estimating the unknown parameters $\boldsymbol{\beta}_k$ of the regression model, which can then be used to predict the class probabilities of independent test data. The simplest approach for estimation is to use an iterative procedure such as iteratively reweighted least squares (IRLS).

In the mind reading competition the number of variables is large compared to the number of measurements. If additional features are derived from the measurement data, the number of parameters p to be estimated easily exceeds the number of measurements N. Therefore, we have to select a subset of features that is the most useful for the model. Our first attempt was to find the optimal subset iteratively using simulated annealing, but soon we decided to use a method, with feature selection embedded into the cost function used in the classifier design. LASSO *(Least Absolute Shrinkage and Selection Operator)* regression method enforces sparsity via ℓ_1 -penalty, and in a least squares model the constrained LS criterion is given by [5, 3]:

$$\min_{\beta} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_1, \tag{3}$$

where $\lambda \geq 0$ is a tuning parameter. When λ is small, this is identical to the OLS solution. With large values of λ, the solution becomes shrunken and sparse version of the OLS solution where only a few of the coefficients β_j are non-zero.

The LASSO has been recently extended for logistic regression [2], and a Matlab implementation is available at `http://www-stat.stanford.edu/~tibs /glmnet-matlab/`.

We experimented with numerous features fed to the classifier, and attempted to design discriminative features using various techniques. Our understanding is that those ended up being too specific for the the first day data and eventually a simplistic solution turned out to be the best, resulting in the following features:

- The detrended mean, i.e., the parameter $\hat{b}$ of the linear model $y = ax + b$ fitted to the time series.

- The standard deviation of the residual of the fit, i.e., stddev$(\hat{y} - y)$.

Both channels were calculated from the raw data; we were unable to gain any improvement from the filtered channels. Since there were initially 204 measurements, this makes a total of 408 features from which to select.

3 Results and Performance Assessment

An important aspect for the classifier design is the error assessment. This was challenging in the mind reading competition, because only a small amount (50 samples) of the test dataset was released with the ground truth. Additionally, we obviously wanted to exploit it also for training the model. In order to simulate the true competition, we randomly divided the 50 test day samples into two parts of 25 samples. The first set of 25 samples was used for training, and the other for performance assessment. Since the division can be done in $\binom{50}{25} > 10^{14}$ ways, we have more than enough test cases for estimating the error distribution.

The remaining problem in estimating the error distribution is the computational load. One run of training the classifier with cross-validation of the parameters takes typically 10-30 minutes. If, for example we want to test with 100 test set splits, we would be finished after a day or two. For method development and for testing different features this is certainly too slow.

Tampere University of Technology uses a grid computing environment developed by Techila Oy (`http://www.techila.fi/`). The system allows

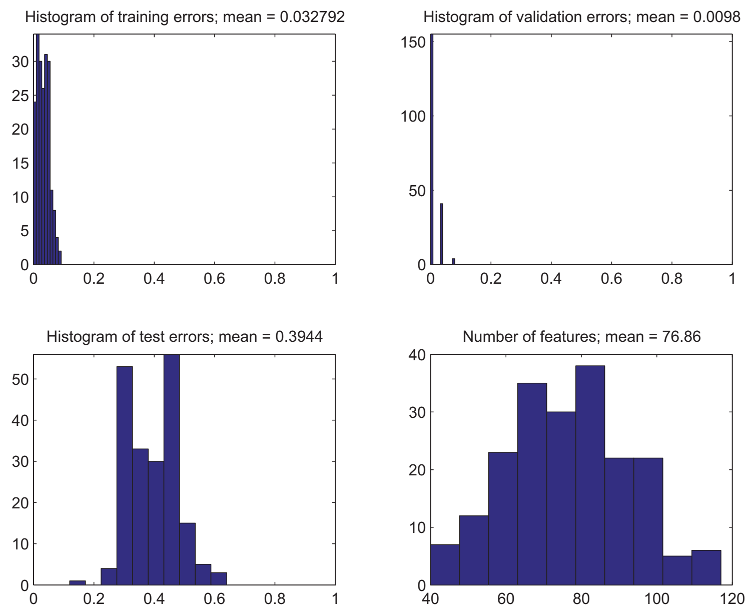

Figure 1. An example result of the error estimation. Figures (a-c) show the error distribution for the first day training data, for the 25 samples of second day data used for training and for the rest of the second day data, respectively. Figure (d) shows the number of features used on the average for the model. In this experiment we run the training for 200 test cases.

distributing the computation tasks to all computers around campus. Since our problem is parallel in nature, incorporating the grid into the performance assessment was easy: A few hundred splits of the test set were done, and one processor in the grid was allocated for each case.

Figure 1 illustrate an example test run. The performance can be assessed from the error distribution for the test data shown on Figure on the bottom left, which in this case is 0.394, or 60.6 % correct classification. We believe that the final result will be slightly better, because the final classifier is trained using all 50 samples of the second day data.

The error for the validation data (top right figure) is very small. This is because the samples were weighted such that second day data has a higher weight in the cost function.

After preparing the final submission, we studied the distribution of the predicted classes for the test data. The test samples should be roughly class-balanced, so a very uneven distribution could indicate a problem (such as a bug) in the classification. We also considered the possibility of fine tuning the regularization parameter based on the balancedness of the corresponding classification result. As the balancedness index we used the ratio of the cardinalities of the largest and smallest classes in

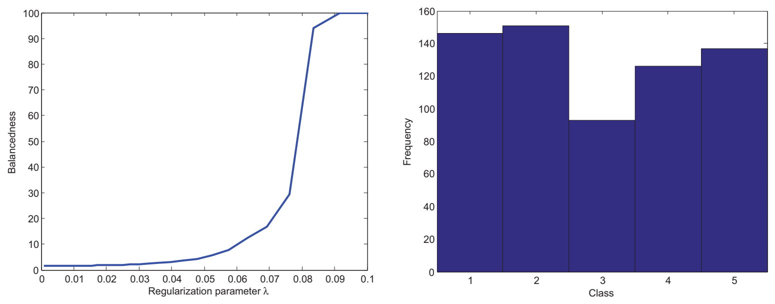

Figure 2. Left: The balancedness index for different values of regularization parameter. Right: The predicted class histogram for the test data.

the predicted result. The balancedness indicator is plotted as a function of the regularization parameter λ in Figure 2 (left).

The result clearly emphasizes the old rule: regularization increases the bias but decreases the variance. This can be seen from the curve in that less regularization (small λ) improves the class-balance (indicator close to unity). However, since it seems that the regularization parameter $\lambda = 0.0056$ selected using cross validation is at the edge of the well balanced region, we decided not to adjust the CV selection. The final predicted class distribution is shown in Figure 2 (right).

Bibliography

[1] Debuse, J.C., Rayward-Smith, V.J.: Feature subset selection within a simulated annealing data mining algorithm. Journal of Intelligent Information Systems 9, 57–81 (1997), 10.1023/A:1008641220268

[2] Friedman, J.H., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(1), 1–22 (2010)

[3] Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: Data mining, inference, and prediction. Springer Series in Statistics, Springer (2009)

[4] Pudil, P., Novovičová, J., Kittler, J.: Floating search methods in feature selection. Pattern Recogn. Lett. 15(11), 1119–1125 (Nov 1994)

[5] Tibshirani, R.: Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society, Series B 58, 267–288 (1994)