`glmnetLRC`: Lasso and elastic-net logistic regression classification with an arbitrary loss function

Landon Sego, Alexander Venzin

March 2016

1 Introduction

The `glmnetLRC` package makes it easy to construct a binary classifier from virtually any number of quantitative predictors that will assign an example, or observation, to one of two classes. It extends the `glmnet` package by making it possible to train lasso or elastic-net logistic regression classifiers (LRC's) using a customized, discrete loss function to measure the classification error. This allows users to assign unique loss values to false positive and false negative errors.

The logistic regression parameter estimates are obtained by maximizing the elastic-net penalized likelihood function that contains several tuning parameters. These tuning parameters are estimated by minimizing the expected loss, which is calculated using cross validation. This approach was originally implemented to automate the process of determining the curation quality of mass spectrometry samples (Amidan et al., 2014).

We present in detail the algorithm used by the `glmnetLRC` package to identify the optimal parameter estimates for a logistic regression classifier (LRC) with variables selection implemented by an elastic-net.

2 The model

We begin by defining a number of variables. Let $i = 1, \ldots, N$ index the observations in a training dataset. Let $y_i = 1$ indicate that observation i belongs to category "1" and $y_i = 0$ indicate that it belongs to category "0". Per the logistic regression model, let

$$P(y_i = 1) \equiv \pi_i = \frac{\exp(\beta_0 + \boldsymbol{\beta}_1^T \mathbf{x}_i)}{1 + \exp(\beta_0 + \boldsymbol{\beta}_1^T \mathbf{x}_i)} \tag{1}$$

where $\mathbf{x}_i = (x_1, \ldots x_K)^T$ is a vector of predictors, or covariates, that influence π_i, β_0 is an intercept, $\boldsymbol{\beta}_1 = (\beta_1, \ldots, \beta_K)^T$ is a vector of logistic regression coefficients, and for notational convenience, $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots \beta_K)^T$.

The estimate of the vector of regression parameters, $\boldsymbol{\beta}$, is influenced by two other tuning parameters, α and λ. For this reason, we will often write $\boldsymbol{\beta}$ as $\boldsymbol{\beta}(\alpha, \lambda)$. The value of $\lambda > 0$ controls the weight of the penalty of the log-likelihood function, while α controls the mixture of the ridge and lasso penalties. The relationship between $\boldsymbol{\beta}$, α, and λ will be clarified below. A final tuning parameter, $\tau \in (0, 1)$, provides a threshold for the LRC such that if $\pi_i > \tau$, observation i is predicted to belong to class "1".

3 Estimating the regression parameters

When we fit the elastic-net logistic regression model to the data, we obtain the estimator $\hat{\boldsymbol{\beta}}(\alpha, \lambda)$. Therefore, let $\hat{\pi}_i \equiv \pi\big(\mathbf{x}_i, \hat{\boldsymbol{\beta}}(\alpha, \lambda)\big)$ denote the predicted probability that $y_i = 1$. Then, if $\hat{\pi}_i > \tau$, the LRC predicts that $y_i = 1$, otherwise it predicts that $y_i = 0$. It will be useful to represent the predicted class of observation i as $\hat{y}_i \equiv f\big(\mathbf{x}_i, \hat{\boldsymbol{\beta}}(\alpha, \lambda), \tau\big) = I_{(\hat{\pi}_i > \tau)}$, where f can be thought of as the LRC. For the elastic-net, the estimate $\hat{\boldsymbol{\beta}}$ is the $\boldsymbol{\beta}$ that maximizes the penalized, binomial log-likelihood function:

$$\hat{\boldsymbol{\beta}}(\alpha, \lambda) = \underset{\boldsymbol{\beta} \in \mathbb{R}^{K+1}}{\arg\max} \left[\ell(\mathbf{x}_i, \boldsymbol{\beta}) - \lambda \left(\frac{1-\alpha}{2} \sum_{k=1}^{K} \beta_k^2 + \alpha \sum_{k=1}^{K} |\beta_k| \right) \right] \tag{2}$$

where the unpenalized log-likelihood is given by

$$\ell(\mathbf{x}_i, \boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^{N} \left[y_i \big(\beta_0 + \boldsymbol{\beta}_1^T \mathbf{x}_i\big) - \log\big(1 + \exp(\beta_0 + \boldsymbol{\beta}_1^T \mathbf{x}_i)\big) \right] \tag{3}$$

Parenthetically, $\alpha = 1$ is the lasso penalty, $\alpha = 0$ is the ridge regression penalty, and $0 < \alpha < 1$ is a mixture of the two. The penalty in (2) is the one specified in the documentation of the `glmnet` package (Hastie & Qian, 2014).

4 Estimating the tuning parameters

The optimal values of the tuning parameters, α, λ, and τ, are obtained by minimizing the risk, or expected loss, of the LRC, where the risk is calculated via cross validation. Calculating risk requires that we define a discrete loss function, $L(y, \hat{y})$ as follows:

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	0	κ_0
$y = 1$	κ_1	0

with $\kappa_0 > 0$ and $\kappa_1 > 0$ chosen to reflect the severity of false-positive and false-negative errors, respectively. Setting $\kappa_0 = \kappa_1 = 1$ results in the commonly used 0-1 loss function. In the `glmnetLRC` package, L is specified via a call to `lossMatrix()`, and the result is passed to the `lossMat` argument of `glmnetLRC()`. In fact, all four elements of L can be specified as desired, i.e., the diagonal elements are not required to be 0, as suggested above.

Cross validation is accomplished by randomly partitioning the data into M folds (non-overlapping and exhaustive subsets), where each fold is tested using a model trained on the remaining folds. The value of M is controlled by the `cvFolds` argument in `glmnetLRC()`. Following the presentation of Hastie, et al. (2008), let

$$\delta : \{1, \ldots, N\} \to \{1, \ldots, M\} \tag{4}$$

map each observation in the training data to one of the folds. Let $\hat{\boldsymbol{\beta}}^{-m}(\alpha, \lambda)$ represent the estimate of $\boldsymbol{\beta}$ obtained by fitting the elastic-net logistic regression model to all the training data except the m^{th} fold. The cross validation estimate of the risk is given by

$$R(\alpha, \lambda, \tau) = \sum_{i=1}^{N} w_i L\Big(y_i, f\big(\mathbf{x}_i,\ \hat{\boldsymbol{\beta}}^{-\delta(i)}(\alpha, \lambda),\ \tau\big)\Big) \Big/ \sum_{i=1}^{N} w_i \tag{5}$$

where the weights w_i are specified in the `lossWeight` argument to `glmnetLRC()`. The optimal estimates of the tuning parameters are those that minimize the risk:

$$(\hat{\alpha}, \hat{\lambda}, \hat{\tau}) = \underset{\alpha,\lambda,\tau}{\arg\min}\, R(\alpha, \lambda, \tau) \tag{6}$$

In practice, we calculate $(\hat{\alpha},\ \hat{\lambda},\ \hat{\tau})$ by computing (5) over an irregular cube of discrete parameter values, defined by the combination of three vectors: $\boldsymbol{\alpha} \times \boldsymbol{\lambda} \times \boldsymbol{\tau}$. The point in the cube that minimizes the risk becomes the estimate for (α, λ, τ). In the event there are ties for the lowest risk for two or more points in the cube, points with τ nearer to 0.5 are preferred, and if that still doesn't break the tie, points with larger values of λ are preferred because they result in a more parsimonious model with fewer predictors. The values of $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ are specified by the `alphaVec` and `tauVec` arguments of `glmnetLRC()`, respectively. The values of $\boldsymbol{\lambda}$ depend on each α and are chosen algorithmically by `glmnet()`, using the default values for the relevant arguments in `glmnet()` and using the entire training dataset.

5 The final LRC model

So far in this discussion, we have made reference to a single random partition of the data, δ, into M folds. Naturally, the estimates of the tuning parameters depend on δ. A different partition will yield different estimates of the tuning parameters. To ensure the final LRC is robust to the random partitioning process, we repeat the training process for multiple partitions, $\delta_1, \ldots, \delta_J$, producing $(\hat{\alpha}_j, \hat{\lambda}_j, \hat{\tau}_j)$ for $j = 1, \ldots, J$. The value of J is controlled by the `cvReps` argument in `glmnetLRC()`. We subsequently refer to the repetition of the cross validation process as cross validation replication.

Calling the `plot()` method on the object returned by `glmnetLRC()` shows a pairs plot and univariate histogram of the various $(\hat{\alpha}_j, \hat{\lambda}_j, \hat{\tau}_j)$. This plot illustrates the consistency (or lack thereof) of the tuning parameter estimates across cross validation replicates.

Once $\hat{\alpha}_j$, $\hat{\lambda}_j$, and $\hat{\tau}_j$ are identified for all the cross validation replicates, the final estimate of the tuning parameters is obtained by calculating the median of each one separately:

$$(\hat{\alpha}^\star, \hat{\lambda}^\star, \hat{\tau}^\star) = \big(\underset{j}{\text{median}}(\hat{\alpha}_j),\ \ \underset{j}{\text{median}}(\hat{\lambda}_j),\ \ \underset{j}{\text{median}}(\hat{\tau}_j)\big) \tag{7}$$

The final estimator $\hat{\boldsymbol{\beta}}(\hat{\alpha}^\star, \hat{\lambda}^\star)$ is obtained by fitting all the training data (via (2)) using the final estimates of the tuning parameters (7), which gives rise to the final LRC:

$$f^\star \equiv f\big(\mathbf{x},\ \hat{\boldsymbol{\beta}}(\hat{\alpha}^\star, \hat{\lambda}^\star),\ \hat{\tau}^\star\big) \tag{8}$$

Calling the `predict()` method on the object returned by `glmnetLRC()` uses $f^\star$ to classify new observations. Likewise, calling the `coef()` method on the object returned by `glmnetLRC()` returns $\hat{\boldsymbol{\beta}}(\hat{\alpha}^\star, \hat{\lambda}^\star)$.

6 The cross validation estimate of the risk

A measure of overall performance for the LRC is provided by the cross validation estimate of the risk. Using the final tuning parameter estimates $(\hat{\alpha}^\star, \hat{\lambda}^\star)$ defined by (7), a corresponding set of regression parameter estimates, $\hat{\boldsymbol{\beta}}_j^{-m}(\hat{\alpha}^\star, \hat{\lambda}^\star)$, are obtained using (2) for each of the M folds in replicate j. The estimate of the risk for replicate j is given by applying (5) as follows:

$$R_j = \sum_{i=1}^{N} w_i L\Big(y_i,\ f\big(\mathbf{x}_i,\ \hat{\boldsymbol{\beta}}_j^{-\delta_j(i)}(\hat{\alpha}^\star, \hat{\lambda}^\star),\ \hat{\tau}^\star\big)\Big) \Big/ \sum_{i=1}^{N} w_i \tag{9}$$

The value of R_j is calculated for $j = 1, \ldots, J$ and summarized using the mean and standard deviation in the usual way:

$$\bar{R} = \frac{1}{J}\sum_{j=1}^{J} R_j, \qquad \sigma_R = \sqrt{\frac{\sum_{j=1}^{J}(R_j - \bar{R})^2}{J-1}} \tag{10}$$

The values of $\bar{R}$ and σ_R are obtained by calling `glmnetLRC()` with the argument `estimateLoss = TRUE` and then printing the resulting object.

References

Amidan, B. G., Orton, D. J., LarMarche, B. L., Monroe, M. E., Moore, R. J., Venzin, A. M., … Payne, S. H. (2014). Signatures for mass spectrometry data quality. *Journal of Proteome Research*, *13*(4), 2215-2222.

Hastie, T., & Qian, J. (2014). *Glmnet Vignette.* Retrieved 20 January 2016, from `http://cran.fhcrc.org/web/packages/glmnet/vignettes/glmnet_beta.html`

Hastie, T., Tibshirani, R., & Friedman, J. H. (2008). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer-Verlag.