`glmnetGLR`: Generalized Elastic Net Logistic Regression

Alex Venzin, Landon Sego

March 4, 2014

This document is a short tutorial on how to use the glmnetGLR package. This package was developed as an extension of the capabilities of the `glmnet` package. This document is provided to guide you through the process of installing the R package, reading in the data, training the model, and then applying the model on other data. Note that the beta version of this package will only install correctly on Unix based operating systems (mac OS, Unix, Linux, etc). To install, access the terminal or command line and type "R CMD INSTALL filepath glmnetGLR_" and hit the tab key to complete the filename. Now that the package is installed, we can attach the package to the local environment.

```
> require(glmnetGLR)
```

This package was designed to build generalized classification models using the elastic net algorithm. This tool was originally developed to automate the process of determining the curation quality of proteomics samples for mass spectrometry. We will be using some of the data documented in [?] to demonstrate how to train and use this tool. The elastic net regression model works as both a classification method and a feature selection tool. It incorporates two regularization constraints to balance feature selection and model simplicity using a combination of the ridge penalty (ℓ_2 normalization) and the lasso penalty (ℓ_1 normalization). This generic model can be used whenever the classification problem of interest can be decomposed into a binary decision (i.e., yes/no, 0/1, good/bad, etc). The user needs to supply a handful of arguments to the `trainLLRC` function. These arguments are the true class labels (`truthLabels`), the set of predictor variables as the columns of a matrix or dataframe (`predictors`), and the loss matrix (`lossMat`). Below we demonstrate how to use this function.

```
> # Load the VOrbitrap Shewanella QC data
> data(traindata)
> # Here are the first few observations of the datasets
> traindata[1:5, 1:7]
      Instrument_Category Instrument Dataset_ID Acq_Time_Start Acq_Length
pt701           VOrbitrap VOrbiETD03     251690     12/31/2011         98
pt702           VOrbitrap VOrbiETD03     251706       1/1/2012         98
pt703           VOrbitrap VOrbiETD03     251887       1/4/2012         98
pt704           VOrbitrap VOrbiETD03     252361      1/10/2012         98
pt705           VOrbitrap VOrbiETD04     255284       2/2/2012         99
                                          Dataset Dataset_Type
pt701 QC_Shew_11_06_col2A_30Dec11_Cougar_11-10-11      HMS-MSn
```

```
pt702 QC_Shew_11_06_col2C_30Dec11_Cougar_11-10-11      HMS-MSn

pt703  QC_Shew_11_06_Col2B_4Jan12_Cougar_11-10-11      HMS-MSn

pt704   QC_Shew_11_06_col1_9Jan12_Cougar_11-10-09      HMS-MSn

pt705  QC_Shew_11_06_Col1B_2Feb12_Cougar_11-10-09      HMS-MSn
```

```
> # Here we select the predictor variables
> predictors <- as.matrix(traindata[,9:96])
> # The logistic regression model requires a binary response
> # variable.
> resp <- traindata[,"response"]
> # Set the loss matrix
> lM <- lossMatrix(c("Good","Good","Poor","Poor"),
+                  c("Good","Poor","Good","Poor"),
+                  c(     0,     1,     5,     0))
> # Train the elastic net classifier
> elasticNet <- trainGLR(truthLabels = resp,
+                        predictors = predictors,
+                        lossMat = lM)
```

The call to `trainLLRC` solves for the optimal parameter settings (α, λ, τ) that minimize the expected loss for the elastic net logistic regression model. Keep in mind that the α parameter governs the trade off between the two regularization parameters. When $\alpha = 0$, we are performing ℓ_2 normalization (ridge regression) and when $\alpha = 1$, we are performing ℓ_1 normalization (lasso regression).

```
> elasticNet

Top 10 optimal parameter values for the elastic net logistic regression fit:


   ExpectedLoss alpha  tau    lambda sqErrorTau

1     0.2423077     0 0.80 0.7179983     0.0900

2     0.2430769     0 0.80 0.7880023     0.0900

3     0.2430769     0 0.80 0.6542133     0.0900

4     0.2453846     0 0.80 0.9491515     0.0900

5     0.2453846     0 0.80 0.8648315     0.0900

6     0.2461538     0 0.80 0.5960948     0.0900

7     0.2469231     0 0.80 0.5431394     0.0900

8     0.2484615     0 0.80 1.0416926     0.0900

9     0.2500000     0 0.80 1.1432564     0.0900

10    0.2500000     0 0.85 0.3108043     0.1225
```

Now that the classifier has been properly trained and the optimal parameters have been identified, we are interested in making predictions for new data observations. To make predictions for a new set of observations, the user will need the elastic net regression model (`glmnetGLRobject`) and the set of new observations to be predicted (`newdata`). Additionally, one may wish to carry through a set of the predictor variables (`keepCols`) and the column of ground truth for the classes if available (`truthCol`). Note that the ground truth is not required to make predictions regarding the class of future observations, but is required to compute the metrics for the elastic net logistic regression model.

```
> # load the data for testing
> data(testdata)
> prediction_values <- predict(glmnetGLRobject = elasticNet,
+                              newdata = testdata,
+                              truthCol = 'Curated_Quality',
+                              keepCols = 12:14)
```

The code above will produce a data.frame object containing the value of the predicted class and if specificed, the column of ground truth responses and the variables to carry through during the prediction process in `keepCols`. Note that if the truth vector is not supplied here that it is not possible to compute the metrics for the elastic net model. In particular, there are five metrics that will be calculated with a call to `summary`: sensitivity, specificity, false negative rate, false positive rate, and accuracy. We can call to this function by doing the following.

```
> summary(prediction_values)
```

```
                          Good
sensitivity         0.76315789
specificity         0.90163934
false negative rate 0.23684211
false positive rate 0.09836066
accuracy            0.84848485
```

This extension of `glmnet` with a customizable loss function can be used in any number of scenarios. One concept to keep in mind is that this package produces regularized binomial logistic regression models. These classifiers are a special case of logistic regression models and as such must meet a set of assumptions. First, model regularization is often desired when the underlying problem is 'ill-posed'. The regularization parameter is implemented to perform feature selection and to generate (in theory) a unique, optimal logistic regression model. Secondly, these classifiers are designed to classify variables that fall into exactly two categories. If the response variable of interest consists of more than two classes, then the user is referred to the `glmnet` package that contains the necessary software for building multinomial logistic regression models.

References

[1] LaMarche BL Monroe ME Moore RJ Venzin AM Smith RD Sego LH Payne SH Tardiff MF Amidan BG, Orton DJ. Signatures for mass spectrometry data quality. 2014.