Logistic Regression

Signal Data Science

Logistic regression is the most commonly encountered linear classification method. We'll learn about logistic regression by returning to the Columbia speed dating dataset, a rich dataset with many different avenues to explore.

Unregularized logistic regression

First, we'll see how to use unregularized logistic regression for two-class classification.

Loading the data

The dataset you'll be using in this assignment is an aggregated form of the full speed dating dataset; you've worked with a simplified form of this dataset before (with fewer variables). Refer to the documentation in `speeddating-documentation.txt` for a description of the new variables.

- Use `read.csv()` to load `speeddating-aggregated.csv` in the `speed-dating` dataset.
- Use `complete.cases()` to determine the number and proportion of rows in the data with `NA`s. Clean the data by using `na.omit()` to remove all rows with `NA`s.

Technical details

You can run logistic regression with `glm()`. It can be used in the same fashion as `lm()`, except for logistic regression you must pass in the additional parameter `family="binomial"`. Additionally, the column representing the binary class which you want to predict must either be (1) a numeric column taking on values 0 and 1 or (2) a factor.

The `pROC` package provides a function, `roc()`, which plots the receiver operating characteristic (ROC) curve given the results of a logistic regression fit. The output of `roc()` can be passed into `plot()` or directly printed to display the area under the ROC. Note that `roc()` accepts *probabilities* as inputs, but the predictions made with a logistic regression model will be in the form of *log-odds*, which must be converted into probabilities with

$$P = \frac{\exp L}{1 + \exp L}$$

where L is a log-odds and P is the corresponding probability.

Analyzing the speed dating dataset

When working through the following questions, examine and interpret the coefficients of each logistic regression model. In addition, examine the area under the ROC curve as well as the shape of the ROC curve itself.

- Predict gender in terms of the 17 self-rated activity participation variables.
- Restrict to the subset of participants who indicated career code 2 (academia) or 7 (business / finance). Predict membership in either class in terms of the 17 activities.
- Restrict to the subset of participants who indicated being Caucasian (`race == 2`) or Asian (`race == 4`). Predict membership in either class in terms of the 17 activities.

Regularized logistic regression

We can also use L^p regularization with the logistic regression cost function. In this part of the assignment you'll be predicting decisions (`dec`) of speed dating participants in terms of interactions between partners' attributes.

Starter code is located at `speedDatingDecisionStarter.R` and the associated dataset is `speeddating-full.csv` in the `speed-dating` folder. We'll be using the *unaggregated* version of the dataset, with data about *both* the person being rated *and* each individual person doing the rating.

The first portion of the code creates a data frame with race and career code for both partners on each date, as well as the frequency with which the person making a decision expressed interest in seeing a partner again (`decAvg`), the frequency with which others expressed interest in the partner (`decPartnerAvg`), and the average attractiveness rating of the partner (`attrPartnerAvg`).

We'll be doing our cross validation at the level of speed dating events, so that there no participants appear in each of the train set and test set for any cross validation fold. The function `crossValidate()` in the starter code performs a grid search for `glmnet()` over values of α and λ, returning the area under the ROC for each pair (α, λ), and can correspondingly be used to find the best choice of coefficients.

- Using `dummy.data.frame()` from the `dummies` package, create a data frame `dums1` with dummy variables corresponding to the participant making the decision and another data frame `dums2` with dummy variables corresponding to the partner being decided on. You only need to expand race and career code out into dummy variables.
- Create a data frame `dums` by calling `cbind()` on `dums1` and `dums2`. Make sure not to repeat any overlapping columns, nor to include the original race and career columns. To this data frame, add interaction terms for
  - (race of decider) x (attractiveness of partner),
  - (career of decider) x (attractiveness of partner),
  - (race of decider) x (race of partner), and
  - (career code of decider) x (career code of partner),

  with column names formed by calling `paste(name1, name2, sep = ":")`.

  To save on computational time, remove those columns with 20 or fewer entries, with `dums = dums[,colSums(dums) > 20]`.
- Form a `features` data frame by extracting `decAvg`, `decPartnerAvg`, `attrPartnerAvg`, and all of the dummy columns from the data frame `dums`.

Keep the following in mind as you work: (1) If your `glmnet()` call returns an error indicating `NA` or `NaN` values in the predictors, use `is.nan()` to check for and filter out columns with `NaN`s. This occurs when `scale()` is used on constant columns (with standard deviation 0). (2) If your `glmnet()` model with a single λ value fails to converge, run `glmnet()` without specifying the `lambda` parameter, and subsequently pass in the desired λ value to `predict()` and `coef()` directly via the `s` parameter.

- For each of males and females, use `crossValidate()` to find the optimal values of α and λ for predicting `dec` in terms of the features. Then inspect the coefficients of the model corresponding to the best values of α and λ and discuss interpretation of the results with your partner.

Multinomial logistic regression

Finally, we'll learn how to use multinomial logistic regression today (sometimes called *softmax regression*[1]).

[1]This comes from the usage of the *softmax function*, which is a continuous approximation of the indicator function.

Previously, you used binomial logistic regression to do *two-class classification*, where you modeled the log-odds associated with a binary outcome as being linearly related to a number of predictor variables. The technique of *multinomial logistic regression* is a straightforward extension of this: our outcome variable has more than two categories, and we model the log-odds associated with falling into each category as being linearly related to our predictor variables.

Technical details

You can run multinomial logistic regression with `glmnet(x, y, family="multinomial")`, where `x` is a scaled matrix of predictors and `y` is a numeric vector representing a categorical variable. In the following, we can simply use *unregularized* logistic regression because the number of predictors is relatively low. To do so, **don't** pass in `lambda=0` to `glmnet()`; instead, call `glmnet()` without specifying the `lambda` parameter, and when calling `predict()` or `coef()` on the output of `glmnet()`, simply set `s=0`.

One can think of multinomial logistic regression as being equivalent to fitting an individual logistic regression model testing for membership in each class. The resulting log-odds are the same for each individual class, but they need to be combined together properly to obtain probabilities.

Suppose that we've fit a multinomial logistic regression model to some data and made predictions on the dataset. Now, for each particular row, we have a log-odds L_i associated to each outcome i. We sometimes want to convert to *probabilities* P_i, which ought to be proportional to the exponentiated log-odds $\exp(L_i)$. We can exponentiate and obtain just $\exp(L_i)$, but those values might not necessarily sum to 1: $\sum_i \exp(L_i) \neq 1$. This is a problem, because probabilities have to sum to 1, that is, $\sum_i P_i = 1$.

To resolve this, we divide each $\exp(L_i)$ by the proper *normalization factor.* That is, we can compute

$$P_i = \exp(L_i) / \sum_i \exp(L_i)$$

which makes all the values of P_i sum to 1 as desired while still being proportional to $\exp(L_i)$.

Analyzing the speed dating dataset

Return to the aggregated speed dating dataset. Again, in the following, there's no need to use regularization because we'll be working with a relatively small number of features.

- Use `table()` on the career code column to find the four most common listed careers in the dataset.
- Restricting to those four careers, predict career in terms of self-rated activity participation and average ratings by other participants. Interpret the coefficients of the resulting linear model. Visualize them with `corrplot()`.
  - You can combine the output of `coef()` with `cbind()`, `do.call()`, and `as.matrix()` as input into `corrplot()`. Be sure to plot just the coefficients, *not* the intercepts of the linear models.
- Write a function `probabilities(preds, rownum)` that takes in a matrix `preds` of predictions generated from multinomial logistic regression (*i.e.*, a matrix of log-odds) and a row number `rownum`, returning row `rownum` converted into *probabilities*. Verify that the output always sums to 1 as expected.