Multinomial Logistic Regression

You'll be formally learning about multinomial logistic regression today.

Previously, you used binomial logistic regression to do *two-class classification*, where you modeled the log-odds ratio 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 ratio associated with falling into each category as being linearly related to our predictor variables.

Multinomial logistic regression is sometimes called *softmax regression*.[1]

Using multinomial logistic regression

You can use 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, you can just set `lambda=0`, because we aren't using very many predictors relative to the number of rows (so overfitting isn't a big problem).

The coefficients for the model can be accessed with `coef(fit, s=lambda)` as usual.

Converting to 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 ratio 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 ratios $\exp(L_i)$. 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, $\sum_i P_i = 1$.

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

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 $\exp(L_i)$.

Speed dating dataset

Return to the aggregated OkCupid dataset from Week 2.

- 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.
- Use your model to make predictions on the entire dataset and look at the principal components of the resulting log-odds ratios. Interpret the results.
- 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 ratios) and a row number `rownum`, returning row `rownum` converted into *probabilities*.

OkCupid dataset

Think of one or two aspects of the OkCupid data to explore using multinomial logistic regression. Do so, incorporating your findings into your writeup of your results.

Here are some examples of things to model (but feel free to explore as you desire):

- Race in terms of extracted factors
- Income bracket in terms of extracted factors