Part 3

Sara Altman and Hershel Mehta

12/5/2017

```r
library(tidyverse)
library(modelr)
```

```r
data <- read_csv("relapse_subjects.csv", na = c("", "NaN")) %>%
  filter(relIn6Mos %in% c(0, 1))
```

Prediction on the test set

We didn't have enough data for a test set, so we used all our data in Part 2. Therefore, we use leave-one-out cross-validation to get an estimate of the test error.

Classification

```r
test_err <- function(test, mod, threshold) {
  test <- as_tibble(test)

  errors <-
    test %>%
    mutate(
      pred = predict(mod, newdata = test, type = "response"),
      pred_group = ifelse(pred > threshold, 1, 0),
      pred_error = ifelse(relIn6Mos != pred_group, 1, 0)
    )

  return(mean(errors$pred_error))
}

test_err_fn <- function(test, mod, threshold) {
  test <- as_tibble(test)

  errors <-
    test %>%
    mutate(
      pred = predict(mod, newdata = test, type = "response"),
      pred_group = ifelse(pred > threshold, 1, 0),
      false_neg = ifelse(relIn6Mos > pred_group, 1, 0)
    )

  return(mean(errors$false_neg))
}

test_err_fp <- function(test, mod, threshold) {
  test <- as_tibble(test)
  errors <-
    test %>%
    mutate(
      pred = predict(mod, newdata = test, type = "response"),
      pred_group = ifelse(pred > threshold, 1, 0),
      false_pos = ifelse(relIn6Mos < pred_group, 1, 0)
    )

  return(mean(errors$false_pos))
}

set.seed(1)

by_subj_cv <-
  data %>%
  crossv_kfold(37)

#for threshold = .5
by_subj_cv %>%
  mutate(train = map(train, as_tibble),
         model = map(train, ~glm(relIn6Mos ~ nacc_neutral_beta,
                                 family = "binomial",
                                 data = .)),
         error = map2_dbl(test, model, test_err, threshold = .4),
         false_neg_rate = map2_dbl(test, model, test_err_fn, threshold = .4),
         false_pos_rate = map2_dbl(test, model, test_err_fp, threshold = .4)) %>%
  summarise(mean_error = mean(error),
            mean_fn_rate = mean(false_neg_rate),
            mean_fp_rate = mean(false_pos_rate))
```

```
## # A tibble: 1 x 3
##   mean_error mean_fn_rate mean_fp_rate
##        <dbl>        <dbl>        <dbl>
## 1  0.2432432   0.08108108    0.1621622
```

Our estimated test error is .24.

Regression

```r
data %>%
  filter(relIn6Mos == 1) %>%
  crossv_kfold(k = 15) %>%
  mutate(
    mod = map(train, ~lm(obstime ~ naccR_drugs_beta, data = .)),
    rmse = map2_dbl(mod, test, rmse)
  ) %>%
  summarise(
    avg_rmse = mean(rmse)
  )
```

```
## # A tibble: 1 x 1
##   avg_rmse
##      <dbl>
## 1 34.82884
```

Our estimated test RMSE is 34.8.

Inference

(a)

Logistic regression model fitted on all data

```
fit_all_data <- glm(relIn6Mos ~ nacc_neutral_beta,
                    family = "binomial",
                    data = data)

summary(fit_all_data)
```

```
##
## Call:
## glm(formula = relIn6Mos ~ nacc_neutral_beta, family = "binomial",
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0199  -0.6582  -0.2791   0.6293   2.1539
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.1309     0.4246  -0.308  0.75791
## nacc_neutral_beta  25.5718     8.3959   3.046  0.00232 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 49.961  on 36  degrees of freedom
## Residual deviance: 34.802  on 35  degrees of freedom
## AIC: 38.802
##
## Number of Fisher Scoring iterations: 5
```

- The coefficient on nacc_neutral_beta is significant at the $\alpha = .001$ level. The p-value is .004. This means that, assuming the null hypothesis is true, the probability of observing a Wald statistic as extreme as 2.869 is .004. Here, the null hypothesis is that the coefficient on `nacc_neutral_beta` is 0.
- We only have one covariate. If the coefficient on this covariate really is 0, then the chance it will be significant at the $\alpha = .05$ level is 5% (and is 1% at the $\alpha = .01$ level). Therefore, the chance is low that this one covariate would be significant if the null were true. In contrast, if we had 100 covariates, we would expect 5 of them to be significant at the $\alpha = .05$ level and so we might not believe our results if it turned out that 5 of our coefficient were significant.

(b)

```
set.seed(1)

by_subj_cv <-
  data %>%
  crossv_kfold(10)

#for threshold = .5
by_subj_cv %>%
  mutate(train = map(train, as_tibble),
         model = map(train, ~glm(relIn6Mos ~ nacc_neutral_beta,
                                 family = "binomial",
                                 data = .)),
         summary = map(model, summary),
         p_value = map_dbl(summary, ~.$coefficients[8]),
         sig_05 = ifelse(p_value < .05, 1, 0),
         sig_01 = ifelse(p_value < .01, 1, 0),
         sig_001 = ifelse(p_value < .001, 1, 0))  %>%
  summarise(prop_sig_05 = mean(sig_05),
            prop_sig_01 = mean(sig_01),
            prop_sig_001 = mean(sig_001)) %>%
  knitr::kable()
```

prop_sig_05	prop_sig_01	prop_sig_001
1	1	0

- We didn't have enough data for a test set and used cross-validation in the prediction part of the project. Therefore, we looked at significance of our coefficient in 10 folds.

- The coefficient on `nacc_neutral_beta` is significant in 100% of the folds at the $\alpha = .05$ level and at and is significant 90% of the time at the $\alpha = .01$ level.
- Note that here, we run 31 hypothesis tests. Therefore, if the null is true, we should expect 5% of our tests (1.55) to be significant at the $\alpha = .05$ level.

(c)

Our original 95% confidence interval for our `nacc_neutral_beta` coefficient is [11.29, 45.05] as we can see from the table below.

```
confint(fit_all_data) %>%
  knitr::kable()
```

```
## Waiting for profiling to be done...
```

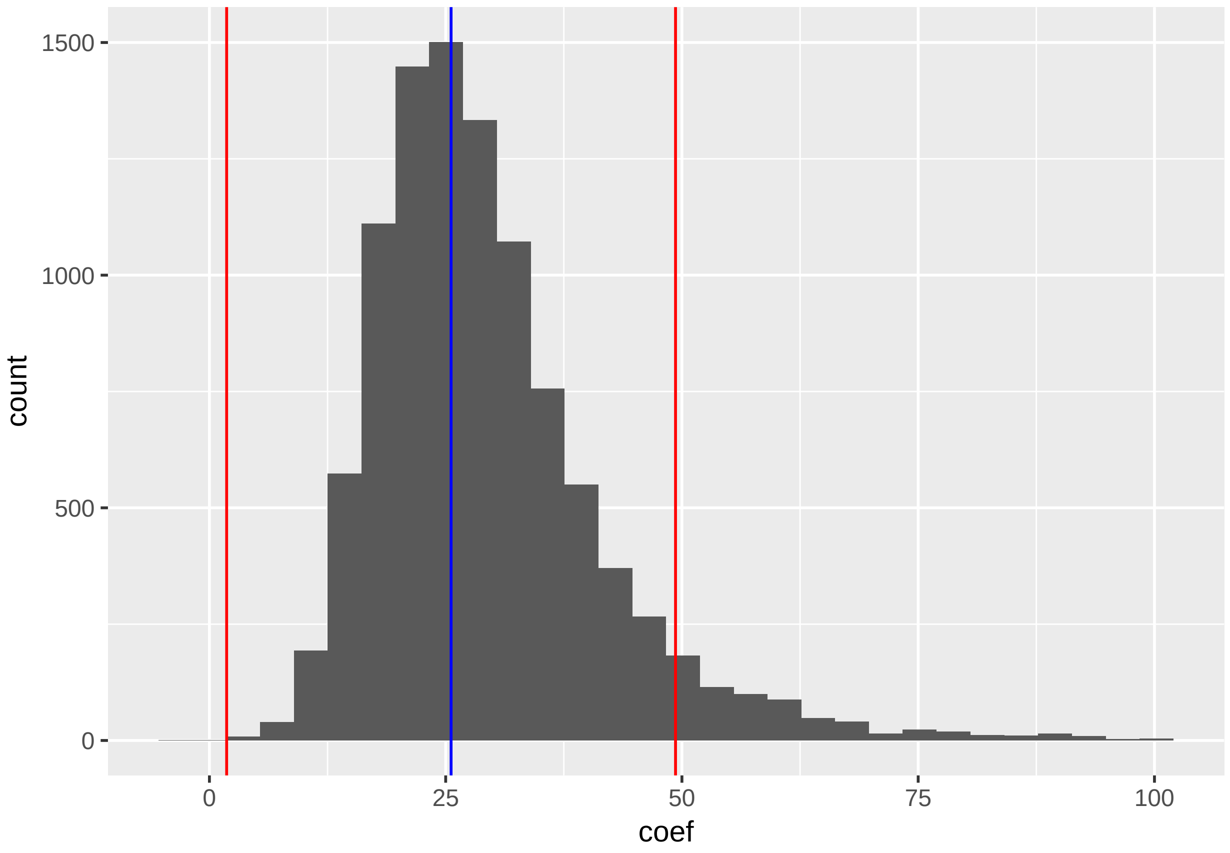
	2.5 %	97.5 %
(Intercept)	-0.974094	0.7311982
nacc_neutral_beta	11.292804	45.0456063

```
mod_glm_bootstrap <-
  data %>%
  modelr::bootstrap(10000) %>%
  mutate(
    mod_glm = map(strap, ~glm(relIn6Mos ~ nacc_neutral_beta,
                              family = "binomial",
                              data = .,
                              control = list(maxit = 50)
                              )),
    coef = map_dbl(mod_glm, ~coef(.)[["nacc_neutral_beta"]])
  ) %>%
  filter(coef <= 100)
```

```
bootstrap_glm_sd <- sd(mod_glm_bootstrap$coef)
```

```
mod_glm_bootstrap %>%
  ggplot(aes(x = coef)) +
  geom_histogram() +
  geom_vline(xintercept = 25.5718, color = "blue") +
  geom_vline(xintercept = 25.5718 - 1.96 * bootstrap_glm_sd, color = "red") +
  geom_vline(xintercept = 25.5718 + 1.96 * bootstrap_glm_sd, color = "red")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



When finding our bootstrapped 95% confidence interval, we run into an error in many of our bootstrapped samples where "fitted probabilities numerically 0 or 1 occurred". We believe this occurs because we have perfect linear separation in some of our bootstrapped samples, which may occur since our n is quite small. This is interesting because it means that in some of our samples, our coefficient perfectly separates all cases of relapsers from non-relapsers, but this may simply occur as a product of bootstrapping on a small sample of data. Also as a result of that, our model converges to very extreme coefficient values, making it difficult to interpret confidence intervals.

To deal with that problem, I attempted to filter out some of the most extreme values (coefficient values >= 100), and made the plot of the 95% confidence interval (i.e., [`{r} 25.5718 - 1.96 * bootstrap_glm_sd`, `{r} 25.5718 + 1.96 * bootstrap_glm_sd`]) above. The plot shows a larger 95% confidence interval than our standard glm output, which you'd expect because of the extreme values created by the perfect linear separation issue mentioned above.

(d)

```
data_betas <-
  data %>%
  select(
    relIn6Mos,
    contains("beta"),
    -starts_with("naccR"),
    -starts_with("naccL")
  )
```

```
mod_glm_betas <- glm(relIn6Mos ~ ., family = "binomial", data = data_betas)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(mod_glm_betas)
```

```
##
## Call:
## glm(formula = relIn6Mos ~ ., family = "binomial", data = data_betas)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.318e-05  -2.110e-08  -2.110e-08   2.110e-08   2.212e-05
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -49.68   85519.44  -0.001    1.000
## nacc_drugs_beta       1209.89 1225903.58   0.001    0.999
## nacc_food_beta       -1950.77 2071865.39  -0.001    0.999
## nacc_neutral_beta     1976.18 2023951.49   0.001    0.999
## mpfc_drugs_beta       -247.70 1101855.62   0.000    1.000
## mpfc_food_beta         736.56 1146329.69   0.001    0.999
## mpfc_neutral_beta     -167.65  650452.85   0.000    1.000
## vta_drugs_beta         333.21 1639399.72   0.000    1.000
## vta_food_beta         -443.89 1069001.80   0.000    1.000
## vta_neutral_beta       439.08 1668977.94   0.000    1.000
## acc_drugs_beta        1241.10 1331858.90   0.001    0.999
## acc_food_beta          100.09 1833584.41   0.000    1.000
## acc_neutral_beta     -1201.30 2237031.11  -0.001    1.000
## ains_drugs_beta       -735.35 2187084.12   0.000    1.000
## ains_food_beta        -275.57 3448176.99   0.000    1.000
## ains_neutral_beta     1123.92 1668063.10   0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 4.9961e+01  on 36  degrees of freedom
## Residual deviance: 3.3496e-09  on 21  degrees of freedom
## AIC: 32
##
## Number of Fisher Scoring iterations: 25
```

We again see the perfect linear separation issue mentioend above when working with all the different beta values. This demonstrates why we took care to build parsimonious models in part 2 and 3, and that when working with small data, you must take a lot of care when adding features.

However to illustrate whether significant coefficients changed on a larger model, one interesting model to compare is one that includes all the different "neutral betas" in different brain regions.

```
data_neutral_betas <-
  data %>%
  select(
    relIn6Mos,
    contains("neutral_beta"),
    -starts_with("naccR"),
    -starts_with("naccL")
  )
```

```
mod_glm_neutral_betas <- glm(relIn6Mos ~ ., family = "binomial", data = data_neutral_betas)
```

```
summary(mod_glm_neutral_betas)
```

```
##
## Call:
## glm(formula = relIn6Mos ~ ., family = "binomial", data = data_neutral_betas)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.363  -0.610  -0.249   0.691   1.876
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.8693     0.8018  -1.084  0.27825
## nacc_neutral_beta  34.2792    12.7162   2.696  0.00702 **
## mpfc_neutral_beta  -2.0960     2.7916  -0.751  0.45277
## vta_neutral_beta    4.9033     3.8896   1.261  0.20745
## acc_neutral_beta   -3.3247     5.3258  -0.624  0.53246
## ains_neutral_beta   1.2983     6.6506   0.195  0.84523
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 49.961  on 36  degrees of freedom
## Residual deviance: 30.951  on 31  degrees of freedom
## AIC: 42.951
##
## Number of Fisher Scoring iterations: 6
```

We can see here that `nacc_neutral_beta` is still significant but its p-value is slightly less than the original single variable glm above. This makes sense when considering the variables we added are activations during the same cues in different brain regions (e.g., `ains_neutral_beta` are the betas extracted from the anterior insula, a brain region thought to be involved in emotion). Based on prior knowledge of how these regions are connected, you would expect, for instance, the NAcc and Anterior Insula to be connected, so they would likely be correlated. Indeed, we can check the correlation in our data

```
cor(data_neutral_betas$nacc_neutral_beta, data_neutral_betas$ains_neutral_beta)
```

```
## [1] 0.3919509
```

So, compared to the single variable regression we tried above, the regression that includes different brain regions like the Anterior Insula, which may be correlated with the NAcc, can knock down the coefficient on the NAcc.

(e)

Post selection inference may be a problem here. We fit lots of different models during part 2, and are only reporting our p-values for one such model. Therefore, we may be favorably biasing our p-value. One way to correct for this would be to validate our findings on new data (i.e., the test set). However, as explained previously, we don't have a test set and this option is not available.

(f)

In our case, there is no causal relationship between NAcc activity when looking at neutral cues and relapse in 6 months, since it is impossible (and unethical) to experimentally manipulate relapse in 6 months. Since we believe addiction is the underlying mechanism increasing chance of relapse, there are many complex factors related to addiction that may be confoudning our ability to infer causality. For instance, various emotional, personality, and socio-economic factors play a role in addiction, so the variations in these factors may be creating confounds for relapse. In our study, we simply hope to find a reliable neural signal of risk for relapse within 6 months.

Discussion

These models could be used for both predication and inference. The models could be used by clinicians to predict if a stimulant-dependent veteran will relapse after rehabilitation. The models could also be used to make inferences about the neuroscience of addiction.

Our models should hold up well over time. There's no obvious reason to suspect that time will affect the relationship between nucleus accumbens activity and relapsing. However, updating our data and then refitting the regression model after more time has passed would be useful. This is because our current analysis removes those who hadn't relapsed at the time the data set was completed. However, this doesn't mean that they never relapsed. It would be useful to continually follow-up with them (if possible), update our data, and then refit our regression model. Note that this isn't relevant to our current classification model, since that predicts relapse in 6 months.

Users of our models (e.g., clinicians helping stimulant addicts) should be aware that are data set was small and therefore our models are vulnerable to overfitting. Although we used cross-validation to get an estimate of our test error, this is likely still an underestimate of the true test error. Therefore, users of the model should be cautious about making important decisions (e.g., whether or not to give a patient extra treatment or to release them) using our model, and should definitely not rely solely on our models' predictions.

If we could recollect our data, we would try to increase the number of patients in the study. We would also increase the number of brain regions from which activity was recorded. Our current data includes covariates collected during follow-ups (e.g., obstime, relapse). We would add a survey to assess general well-being to the covariates collected during the follow-up.

If we were to analyse the same data set again, we would try a) using all by-trial data to create a hierarchical model (our current models just use our by-subject data set) and b) use survival analysis to model time to relapse.