STATS216v Introduction to Statistical Learning
Stanford University, Summer 2017

Problem Set 2

Due: Friday, July 21

Remember the university honor code. All work and answers must be your own.

---

1. *Grading notes: 12 points total. 6 points for each part.*

   (a) Let's say that if a patient has heart attack then the response variable takes value $Y = 1$. So we have

   $$\mathbb{P}(Y = 1) = \frac{\exp(\beta_0 + \beta_1 \times 143 + \beta_2 \times 17)}{1 + \exp(\beta_0 + \beta_1 \times 143 + \beta_2 \times 17)} = 0.092$$

   (b)

   $$\begin{aligned} \frac{\exp(-3.8 + .007w)}{1 + \exp(-3.8 + .007w)} &= .1 \\ \exp(-3.8 + .007w) &= .1(1 + \exp(-3.8 + .007w)) \\ .9\exp(-3.8 + .007w) &= .1 \\ \exp(-3.8 + .007w) &= .1/.9 \\ -3.8 + .007w &= -2.20 \\ w &= 228.6 \end{aligned}$$

2. *Grading notes: 21 points total. 3 points for each part (a), (b) and (c) and 12 points for part (d). 3 points for each part of part (d).*

   (a) The model pays a penalty of $\lambda|\beta_j|$ for including term β_j in the model. This discourages the sum of the magnitude of the coefficients from being large. In other words it favors models with small sum of the coefficients.

   (b) The OLS estimate $\hat{\beta}_{OlS}$

   (c) All the coefficients become zero in this case.

   (d) In each of the following parts briefly explain that what will happen as we increase the λ from 0 to ∞.

      i. The training RSS will increase steadily since we are going from a very flexible model to a model that can just capture the mean.

      ii. Initially, with $\lambda = 0$, the model starts with the OLS estimates, which minimize the sum of the squared residuals for the training data, and so generally does not perform so well on the test data, which results in relatively high RSS. As λ is increased, the values of the coefficients are reduced and so is the overfitting, which lessens the RSS. Eventually, as the coefficients approach zero, the model loses too much of its flexibility and the test RSS increases.

iii. When $\lambda = 0$ we have the OLS estimates. As λ increases, the model becomes simpler which results in less variance for the coefficients.

iv. We start with the OLS estimates, and as λ grows the model loses flexibility by shrinking the coefficients to zero. As a result, the model fits less accurately to the training data, and so the bias increases.

3. *Grading notes: 16 points total, each part 4 points.*

(a) Due to the randomness in the data, the coefficient of a feature (for example, β_{income}) is a random quantity. The standard deviation of this random quantity is known as the standard error.

(b) We have the following output:

```
Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4725  -0.1444  -0.0574  -0.0211   3.7245

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1
    1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8
```

The estimated standard error for `income` is 4.99e-06 and the estimated standard error for `balance` is 2.27e-04.

(c) Sample function is shown in the appended R code.

(d) We have the following output:

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = n_reps)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01  1.024721e-02 4.824698e-01
t2*  2.080898e-05 -1.103654e-06 4.948009e-06
t3*  5.647103e-03  2.076323e-05 2.643337e-04
```

The estimated standard error for `income` is 4.95e-06 and the estimated standard error for `balance` is 2.64e-04.

(e) The estimated standard errors for R income are remarkably close. The estimated standard errors for the intercept and `balance` are reasonably close, coming within about 20%.

4. *Grading notes: 26 points total. 4 part (a), 1 part (b), 4 points for each of the parts(c) to (e), 3 points for (f) to (h)*

   (a) Lag 2 and the intercept appear to be statistically significant.

   (b) When the model predicts 'down' then the prediction was correct 54/102 times. When the model predicts 'up' the prediction was correct 557/ 987 times.

   (c) When the model predicts 'down' then the prediction was correct 10/19 times. When the model predicts 'up' the prediction was correct 75/137 times. This is a mis-classification rate of .455.

   (d) The confusion matrix is identical to the one from logistic regression. LDA and logistic regression have similar structure, so with this much data it is plausible that they ended up with the same predictions.

   (e) When the model predicts 'down' then the prediction was correct 28/65 times. When the model predicts 'up' the prediction was correct 47/91 times. This is a mis-classification rate of .519.

   (f) The LDA and logistic regression model have identical mis-classification rate, and it is lower than that of the knn model. The first two models appear to be doing a little bit better.

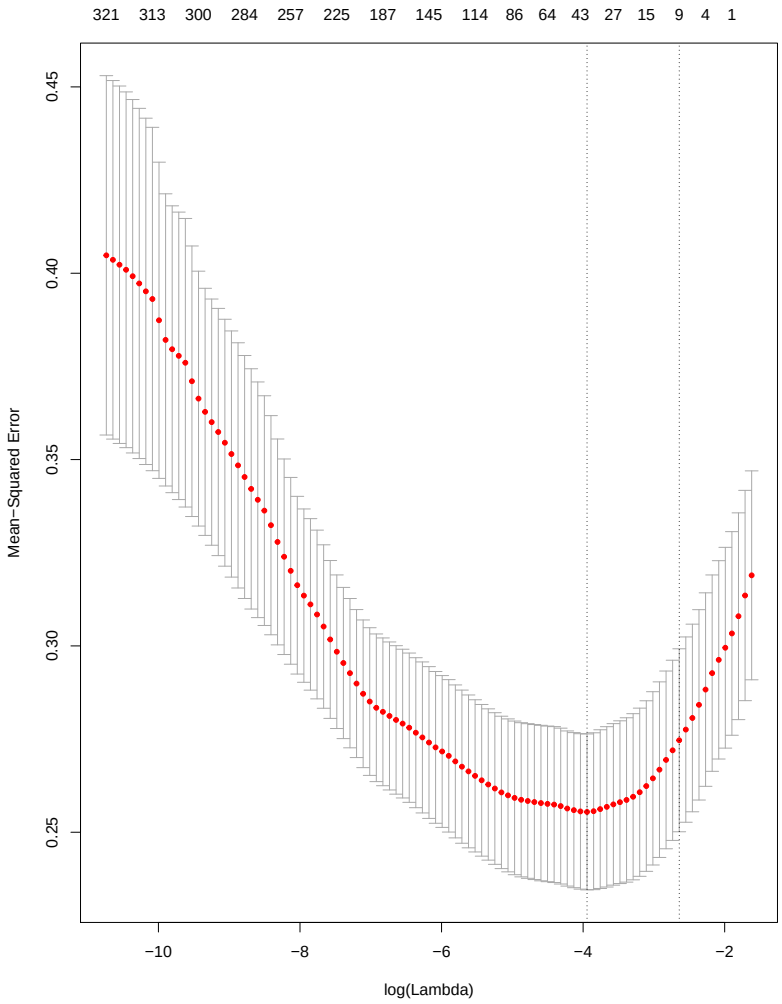
   Note that simply predicting 'up' every time would have better accuracy than the KNN model. This suggests that the KNN model is quite bad in this case.

   (g) LDA is assuming that the features X are generated from different normal distributions depending on the class of Y. When this is correct, it will do better. For example, imagine classifying two species of birds based on their wingspan. In this case, if we expect that the wingspans follow different normal distributions for each species, then LDA is probably better suited. Logistic regression can also suffer from numerical instabilities if the classes are perfectly separable, which doesn't happen with LDA.

   (h) KNN is a more flexible model, and may be able to fit nonlinear shapes that logistic regression cannot. Suppose that we are trying to predict whether a person is a college student based on their age. Given sufficient data, KNN would be able to predict 'yes' for people between $18 - 22$ and 'no' for others. Logistic regression, which is a linear classifier, would always classify people below a certain age as 'yes' and everyone else as 'no', which would perform poorly in this setting.

5. *Grading notes: 25 points total. 4 points part (a), 2 points part (b), 4 points (c), 2 points (d), 3 points (e), 10 points part (f) which is divided into 3, 1 and 3 for each of its parts and 3 points on comments on the models.*

(b) The plot should be as the one below. 1 point for a visibly wrong plot if the implementation is correct (i.e. wrong seed).



(c) The value is 0.0712.

(d) There are 9 nonzero predictors.

```
[1] "Onset.Delta"            "sd.alsfrs.score"        "alsfrs.score.slope"     "
    last.speech"
[5] "fvc.liters.slope"       "min.slope.alsfrs.score" "sum.slope.alsfrs.score" "
    min.slope.speech"
[9] "sum.slope.speech"
```

(e) The RMSE is 0.527.

(f) All 323 predictors are nonzero. This is not surprising, as ridge regression does not induce sparse models. The RMSE is 0.541, which is very similar to the lasso RMSE.

Appendix: Relevant R code (for Questions 4-5)

Question 3

```
library(ISLR)
library(boot)
data("Default")

# part a
logistic_model <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(logistic_model)

#part b
boot.fn <- function(data_set, indices) {
  coef(glm(default ~ income + balance, data = data_set, family = "binomial", subset =
      indices))
}

#part c
set.seed(2017)
n_reps <- 100
boot_se <- boot(Default, boot.fn, n_reps)
boot_se
```

Question 4

```
library("ISLR")
Weekly <- Weekly

#part a
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
               data = Weekly, family = binomial)
summary(glm.fit)

# part b
glm.probs <- predict(glm.fit, type = "response")
glm.pred <- rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Weekly$Direction)

#part c
Weekly_train <- Weekly[Weekly$Year < 2008, ]
Weekly_test <- Weekly[Weekly$Year >= 2008, ]
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3, data = Weekly_train, family = binomial)
summary(glm.fit)
glm.probs <- predict(glm.fit, newdata = Weekly_test, type = "response")
glm.pred <- rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Weekly_test$Direction)

#part d
library("MASS")
lda.fit <- lda(Direction ~ Lag1 + Lag2 + Lag3, data = Weekly_train, family = binomial)
lda.fit
lda.pred <- predict(lda.fit, newdata = Weekly_test, type = "response")$class
table(lda.pred, Weekly_test$Direction)

#part e
library("class")
train_x <- Weekly_train[, 2:4]
train_y <- Weekly_train$Direction
test_x <- Weekly_test[, 2:4]
knn_fit <- knn(train = train_x, test = test_x, cl = train_y, k=1)
table(knn_fit, Weekly_test$Direction)
```

Question 5

```
rm(list=ls())
setwd('~/Box Sync/statshomework/stats216v/hw2/source/')

#(a)
library(glmnet)
load("als.RData")
set.seed(2017)
lasso.cv = cv.glmnet(x = as.matrix(train.X), y = train.y)

#(b)
plot(lasso.cv)

#(c)
my.lambda = lasso.cv$lambda.1se

#(d)
nonzero = predict(lasso.cv, s = 'lambda.1se', type = 'nonzero')
colnames(train.X)[nonzero[,]]

#(e)
lasso.test.pred = predict(lasso.cv, as.matrix(test.X), s = 'lambda.1se')
sqrt(mean((lasso.test.pred - test.y)^2))

#(f)
set.seed(2017)
ridge.cv = cv.glmnet(x = as.matrix(train.X), y = train.y, alpha=0)
nonzero.ridge = predict(ridge.cv, s = 'lambda.1se', type = 'nonzero')
colnames(train.X)[nonzero.ridge[,]]
ridge.test.pred = predict(ridge.cv, as.matrix(test.X), s = 'lambda.1se')
sqrt(mean((ridge.test.pred - test.y)^2))
```