Stat 495 - Midterm Practice Questions

Add 1

A random variable X is a member of an exponential family if its density $f(x|\theta)$ can be written as:

$$f(x|\theta) = a(\theta)b(x)\exp\left[c(\theta)d(x)\right],$$

where $a()$ and $c()$ are functions only of the parameter θ, and $b()$ and $d()$ are functions of x.

Verify that the Binomial distribution is an exponential family. Note, X $Bin(n, p)$ has pmf given by:

$$P(X = x|p) = (n \text{ choose } x)p^x(1-p)^{n-x}.$$

Add 2

Describe how to obtain a bootstrap distribution for a chosen statistic and obtain a (say) 95 percent confidence interval for its related parameter of interest.

Add 3

Explain the plug-in principle in relation to obtaining the standard error of a sample proportion.

Add 4

Explain what a Bayesian conjugate family is and what its benefits are.

Add 5

Using the Chapter 8 Lab data set for context, explain how to perform a permutation test to see whether the mean age differs for individuals with monthly income > 7500 versus individuals with less than that monthly income.

```r
credit <- read.csv("https://awagaman.people.amherst.edu/stat495/creditsample.csv", header = T)
```

Add 6

The diabetes data set (loaded below) is used to demonstrate ridge regression in Chapter 7.

```
diabetes <- read.csv("http://web.stanford.edu/~hastie/CASI_files/DATA/diabetes.csv", header = TRUE)
diabetes <- select(diabetes, -X)
```

The data was pre-processed. The text states that the predictors were standardized to mean 0 and sum of squares 1, and the response had it's mean subtracted off (i.e. it was centered), but not scaled.

```
n <- nrow(diabetes)
diabetes2 <- mutate(diabetes, prog = scale(prog, scale = FALSE),
                    age = scale(age)/sqrt(n-1),
                    sex = scale(sex)/sqrt(n-1),
                    bmi = scale(bmi)/sqrt(n-1),
                    map = scale(map)/sqrt(n-1),
                    tc = scale(tc)/sqrt(n-1),
                    ldl = scale(ldl)/sqrt(n-1),
                    hdl = scale(hdl)/sqrt(n-1),
                    tch = scale(tch)/sqrt(n-1),
                    ltg = scale(ltg)/sqrt(n-1),
                    glu = scale(glu)/sqrt(n-1))
```

```
#so you can see the matrix algebra to verify sum of squares 1
x <- model.matrix(prog ~ ., diabetes2)[, ]
y <- diabetes2 %>% select(prog) %>%  unlist() %>%  as.numeric()
S <- t(x) %*% x
diag(S)
```

```
## (Intercept)         age         sex         bmi         map          tc
##         442           1           1           1           1           1
##         ldl         hdl         tch         ltg         glu
##           1           1           1           1           1
```

Note that while you'd usually want a training/test set here, the book used the entire data set, so you should do that to try to verify their work.

(a) Use OLS to predict *prog* using *diabetes2* and verify you obtain the results in Table 7.3.

```
mod1 <- lm(prog ~ ., data = diabetes2)
msummary(mod1)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.40e-14   2.58e+00    0.00   1.0000
## age         -1.00e+01   5.97e+01   -0.17   0.8670
## sex         -2.40e+02   6.12e+01   -3.92   0.0001 ***
## bmi          5.20e+02   6.65e+01    7.81  4.3e-14 ***
## map          3.24e+02   6.54e+01    4.96  1.0e-06 ***
## tc          -7.92e+02   4.17e+02   -1.90   0.0579 .
## ldl          4.77e+02   3.39e+02    1.41   0.1604
## hdl          1.01e+02   2.13e+02    0.48   0.6347
```

```
## tch          1.77e+02   1.61e+02    1.10   0.2735
## ltg          7.51e+02   1.72e+02    4.37  1.6e-05 ***
## glu          6.76e+01   6.60e+01    1.02   0.3060
##
## Residual standard error: 54.2 on 431 degrees of freedom
## Multiple R-squared:  0.518,  Adjusted R-squared:  0.507
## F-statistic: 46.3 on 10 and 431 DF,  p-value: <2e-16
```

```
round(coef(mod1), 2)
```

```
## (Intercept)         age         sex         bmi         map          tc
##        0.00      -10.01     -239.82      519.84      324.39     -792.18
##         ldl         hdl         tch         ltg         glu
##      476.75      101.04      177.06      751.28       67.63
```

(b) Use ridge regression with lambda = 0.1 and verify you obtain the results in Table 7.3.
(c) Explain what cross-validation is (generally), and how it can be used to help choose a lambda in ridge regression.
(d) Use CV with ridge regression to select your lambda running the default grid through glmnet (i.e. don't specify a grid). What lambda is selected?

```
x2 <- model.matrix(prog ~ ., diabetes)[, -1]
y2 <- diabetes %>%
  select(prog) %>%
  unlist() %>%
  as.numeric()

set.seed(1)
cv.out <- cv.glmnet(x2, y2, alpha = 0)
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 4.516
```

```
ridge_mod <- glmnet(x2,y2, alpha = 0)
round(coefficients(ridge_mod, s = bestlam), 3)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                   s1
## (Intercept) -256.424
## age           -0.007
## sex          -20.847
## bmi            5.437
## map            1.066
## tc            -0.165
## ldl           -0.078
## hdl           -0.664
## tch            4.190
## ltg           99.256
## glu            0.334
```

(e) Implement LASSO with CV to select your lambda running the default grid through glmnet. What lambda is selected?

```r
set.seed(1)
cv.out1 <- cv.glmnet(x2, y2, alpha = 1)
bestlam1 <- cv.out1$lambda.min
bestlam1
```

```
## [1] 0.0972943
```

```r
lasso_mod <- glmnet(x2,y2, alpha = 1)
round(coefficients(lasso_mod, s = bestlam1), 3)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                   s1
## (Intercept) -322.932
## age           -0.021
## sex          -22.361
## bmi            5.635
## map            1.103
## tc            -0.744
## ldl            0.433
## hdl           -0.026
## tch            5.403
## ltg          138.094
## glu            0.275
```

(f) Implement a regression tree to predict *prog*. What cp corresponds to the default tree?

```r
set.seed(1)
prog.rpart <- rpart(prog ~ ., data = diabetes2, method = "anova")
printcp(prog.rpart)
```

```
##
## Regression tree:
## rpart(formula = prog ~ ., data = diabetes2, method = "anova")
##
## Variables actually used in tree construction:
## [1] bmi glu hdl ltg map
##
## Root node error: 2621009/442 = 5930
##
## n= 442
##
##         CP nsplit rel error xerror    xstd
## 1  0.29154      0    1.0000 1.0029 0.05027
## 2  0.08523      1    0.7085 0.7617 0.04708
## 3  0.05660      2    0.6232 0.6789 0.04386
## 4  0.03066      3    0.5666 0.6211 0.03927
## 5  0.02031      4    0.5360 0.6164 0.04099
## 6  0.01569      5    0.5157 0.6360 0.04121
## 7  0.01345      6    0.5000 0.6521 0.04242
## 8  0.01101      8    0.4731 0.6605 0.04308
## 9  0.01055      9    0.4621 0.6589 0.04326
## 10 0.01000     10    0.4515 0.6625 0.04393
```

(g) Explain why we don't need a *glm* here to predict *prog*. `prog` is a continuous, numeric variable so OLS is fine. GLMs are used when there is logistic, poisson for counts, etc.

(h) Which model of these do you prefer? How do the fits differ? (Hint: Besides comparing coefficients/involved variables, you can compare MSEs.)

```
# for OLS
54.16^2
```

```
## [1] 2933.31
```

```
# for ridge
ridge_pred <- predict(ridge_mod, s = bestlam, newx = x2)
mean((ridge_pred - y2)^2)
```

```
## [1] 2881.33
```

```
# for lasso
lasso_pred <- predict(lasso_mod, s = bestlam1, newx = x2)
mean((lasso_pred - y2)^2)
```

```
## [1] 2862.19
```

```
# for tree
fittedtree <- predict(prog.rpart)
mean((fittedtree - diabetes2$prog)^2)
```

```
## [1] 2677.44
```

Add 7

The Spam email data set (loaded below) is used to demonstrate logistic regression and classification trees in Chapter 8. Note that while you'd usually want a training/test set here, the book used the entire data set, so you should do that to try to verify their work.

```
spam <- read.csv("http://web.stanford.edu/~hastie/CASI_files/DATA/SPAM.csv", header = TRUE)
```

(a) Why does it not make sense to use OLS to predict *spam*? It doesn't make sense to use OLS to predict spam because `spam` is a 0/1 variable and OLS doesn't have a way to constrain the response variable to be 0/1.
(b) Use logistic regression to predict *spam*. Verify you obtain the results in Table 8.3.

```
spamdata <- select(spam, -spam, -testid)
spamdata <- data.frame(scale(spamdata))
spamdata <- mutate(spamdata, spam = spam$spam)
```

```
loglm <- glm(spam ~ ., data = spamdata, family = "binomial")
```

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

```
summary(loglm)
```

```
##
## Call:
## glm(formula = spam ~ ., family = "binomial", data = spamdata)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.127  -0.203   0.000   0.114   5.364
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.2653     1.9908   -6.16  7.2e-10 ***
## make         -0.1189     0.0707   -1.68  0.09239 .
## address      -0.1881     0.0894   -2.10  0.03536 *
## all           0.0575     0.0556    1.03  0.30076
## X3d           3.1412     2.1025    1.49  0.13517
## our           0.3782     0.0685    5.52  3.3e-08 ***
## over          0.2418     0.0684    3.53  0.00041 ***
## remove        0.8919     0.1303    6.85  7.6e-12 ***
## internet      0.2285     0.0675    3.39  0.00071 ***
## order         0.2046     0.0794    2.58  0.00996 **
## mail          0.0822     0.0468    1.76  0.07923 .
## receive      -0.0515     0.0600   -0.86  0.39066
## will         -0.1192     0.0638   -1.87  0.06177 .
## people       -0.0240     0.0693   -0.35  0.72956
## report        0.0485     0.0457    1.06  0.28885
## addresses     0.3200     0.1878    1.70  0.08837 .
## free          0.8577     0.1203    7.13  1.0e-12 ***
## business      0.4262     0.1000    4.26  2.0e-05 ***
```

```
## email          0.0639     0.0622    1.03  0.30453
## you            0.1444     0.0622    2.32  0.02033 *
## credit         0.5339     0.2744    1.95  0.05167 .
## your           0.2905     0.0630    4.61  3.9e-06 ***
## font           0.2065     0.1669    1.24  0.21584
## X000           0.7865     0.1651    4.76  1.9e-06 ***
## money          0.1887     0.0718    2.63  0.00853 **
## hp            -3.2097     0.5228   -6.14  8.3e-10 ***
## hpl           -0.9226     0.3899   -2.37  0.01797 *
## george       -39.6236     7.1155   -5.57  2.6e-08 ***
## X650           0.2399     0.1072    2.24  0.02526 *
## lab           -1.4752     0.8909   -1.66  0.09774 .
## labs          -0.1506     0.1432   -1.05  0.29297
## telnet        -0.0687     0.1943   -0.35  0.72374
## X857           0.8374     1.0787    0.78  0.43757
## data          -0.4105     0.1733   -2.37  0.01784 *
## X415           0.2200     0.5273    0.42  0.67649
## X85           -1.0940     0.4196   -2.61  0.00912 **
## technology     0.3719     0.1244    2.99  0.00280 **
## X1999          0.0197     0.0743    0.27  0.79082
## parts         -0.1317     0.0934   -1.41  0.15847
## pm            -0.3760     0.1664   -2.26  0.02384 *
## direct        -0.1066     0.1272   -0.84  0.40221
## cs           -16.2716     9.6074   -1.69  0.09033 .
## meeting       -2.0617     0.6429   -3.21  0.00134 **
## original      -0.2791     0.1805   -1.55  0.12198
## project       -0.9785     0.3292   -2.97  0.00295 **
## re            -0.8016     0.1575   -5.09  3.6e-07 ***
## edu           -1.3295     0.2447   -5.43  5.5e-08 ***
## table         -0.1774     0.1266   -1.40  0.16096
## conference    -1.1474     0.4603   -2.49  0.01267 *
## ch.           -0.3143     0.1077   -2.92  0.00350 **
## ch..1         -0.0509     0.0674   -0.75  0.45066
## ch..2         -0.0719     0.0917   -0.78  0.43291
## ch..3          0.2832     0.0728    3.89  0.00010 ***
## ch..4          1.3120     0.1737    7.55  4.2e-14 ***
## ch..5          1.0318     0.4780    2.16  0.03088 *
## crl.ave        0.3803     0.5976    0.64  0.52451
## crl.long       1.7771     0.4912    3.62  0.00030 ***
## crl.tot        0.5116     0.1365    3.75  0.00018 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6170.2  on 4600  degrees of freedom
## Residual deviance: 1815.8  on 4543  degrees of freedom
## AIC: 1932
##
## Number of Fisher Scoring iterations: 13
```

```
tally(~ spam, data = spamdata)
```

```
## spam
```

```
##  TRUE FALSE
##  1813  2788
```

```
1813/4601
```

```
## [1] 0.394045
```

```
logaug <- loglm %>%
  broom::augment(type.predict = "response")
logaug <- mutate(logaug, binprediction = round(.fitted,0))
tally(~binprediction, data = logaug)
```

```
## binprediction
##    0    1
## 2860 1741
```

```
with(logaug, table(spam, binprediction))
```

```
##        binprediction
## spam       0    1
##   FALSE 2666  122
##   TRUE   194 1619
```

(c) Obtain a confusion matrix for the logistic regression. How well is the model doing?
(d) Use a classification tree to predict *spam*. Attempt to obtain the tree in Figure 8.7. Note that you may need to prune or adjust control parameters to obtain this tree. If you cannot obtain the tree, obtain one you want to work with for part e below.

```
set.seed(495)
spam.rpart <- rpart(factor(spam) ~ ., method = "class", data = spamdata)
printcp(spam.rpart)
```

```
##
## Classification tree:
## rpart(formula = factor(spam) ~ ., data = spamdata, method = "class")
##
## Variables actually used in tree construction:
## [1] ch..3   ch..4   crl.tot free    hp      remove
##
## Root node error: 1813/4601 = 0.394
##
## n= 4601
##
##        CP nsplit rel error xerror    xstd
## 1 0.47656      0    1.0000 1.0000 0.01828
## 2 0.14892      1    0.5234 0.5433 0.01535
## 3 0.04302      2    0.3745 0.4468 0.01425
## 4 0.03089      4    0.2885 0.3271 0.01254
## 5 0.01048      5    0.2576 0.2796 0.01172
## 6 0.01000      6    0.2471 0.2653 0.01145
```

```r
#rplot.plot(spam.rpart)
```

(e) Obtain a confusion matrix for your classification tree. How do your error rates compare to those reported in Figure 8.7?

```r
spampredict <- predict(spam.rpart, type = "class")
table(spampredict, spamdata$spam)
```

```
##
## spampredict FALSE TRUE
##       FALSE  2654  314
##       TRUE    134 1499
```

(f) Which method do you prefer for predicting *spam*? The tree has a little worse performance, but is much easier to understand so we would probably use thr tree.

CASI 8.6

```
galaxy <- read.table("http://web.stanford.edu/~hastie/CASI_files/DATA/galaxy.txt", header = TRUE)
```

Data set description: Table of counts of galaxies binned into categories defined by redshift and magnitude. The column labels are log-redshift values, and the row labels magnitude.

Fit the Poisson regression model (8.39) to the galaxy data.

(Note that some data wrangling is required based on how the data is provided.)

```
galaxy2 <- mutate(galaxy, m = 18:1)

galaxydata <- galaxy2 %>%
  tidyr::pivot_longer(-m, names_to = "log_r", values_to = "count")

galaxydata <- mutate(galaxydata, r = c(rep(1:15, 18)))
```

Add 8

```
data(Credit)
Credit <- select(Credit, -ID)
```

You may want to look over the help file for this data set. Note that the number of observations stated there is inaccurate. Our goal is to predict Balance. Do not worry about transforming Balance or any of the other predictors here.

(a) Create an appropriate training/test split using a 75/25 percent split.
(b) Perform best subsets and choose a model based on an appropriate descriptive statistic. Compute the test MSE.

(c) Perform forward selection and choose a model based on an appropriate descriptive statistic. Compute the test MSE.

(d) Perform the lasso and choose a model based on an appropriate descriptive statistic. Compute the test MSE.

(e) Fit an elastic-net penalty with alpha = 0.5, and choose a model based on an appropriate descriptive statistic. Compute the test MSE.

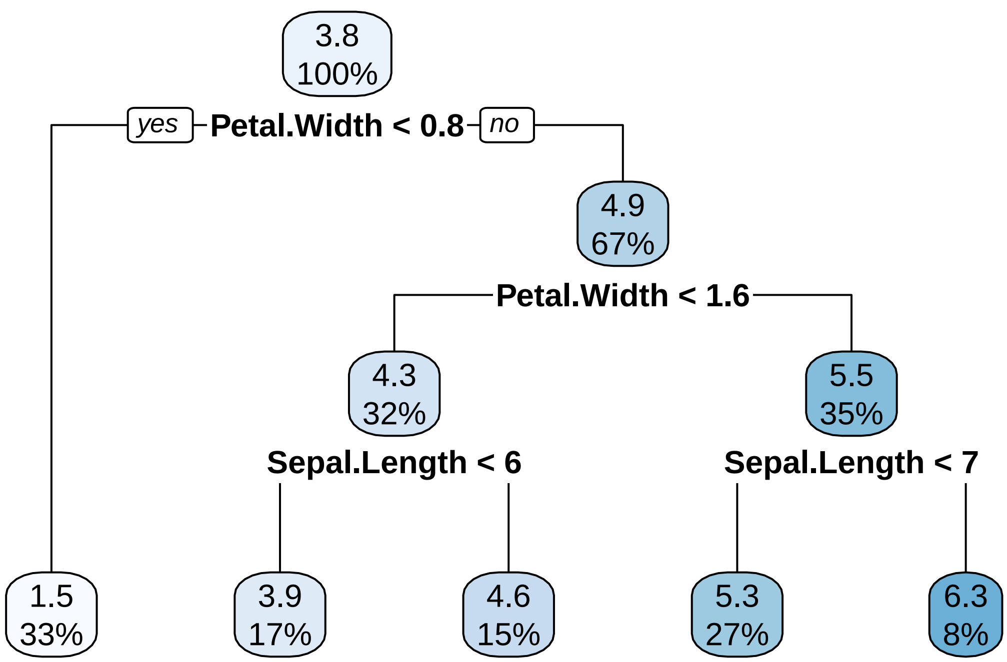
(f) How do your models compare? Which model do you prefer? Why? (Should be able to compare coefficients.)

Add 9

```
data(iris)
glimpse(iris)
```

```
## Rows: 150
## Columns: 5
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
## $ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s~
```

```
# for parts a and b
iris.control <- rpart.control(minbucket = 10, minsplit = 30)
iris.rpart <- rpart(Petal.Length ~ . - Species, data = iris, method = "anova",
                    control = iris.control)
rpart.plot(iris.rpart)
```




```
# for part c
iris[121,]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 121          6.9         3.2          5.7         2.3 virginica
```

```r
# for part d sketch
favstats(~ Petal.Width, data = iris)
```

```
##  min  Q1 median  Q3 max    mean       sd   n missing
##  0.1 0.3    1.3 1.8 2.5 1.19933 0.762238 150       0
```

```r
favstats(~ Sepal.Length, data = iris)
```

```
##  min  Q1 median  Q3 max    mean       sd   n missing
##  4.3 5.1    5.8 6.4 7.9 5.84333 0.828066 150       0
```

(a) Is this a classification or a regression tree? How do you know?
(b) Explain what the minbucket and minsplit control options do.
(c) What is the residual for observation 121? (Residual = observed - predicted response value)
(d) Only 2 variables are used in the tree. Sketch the hypercubes formed in 2-D space, labeling them with their predicted response values.