Sample solutions

Stat 8051 Homework 5

Problem 1: ALR Exercise 11.3

This requires specifying a sequence of models corresponding to the choices of mean function to be prepared. We considered five such mean functions, although many more are possible:

```
> Linf = max(walleye$length)+1
> walleye$var = log(1-walleye$length/Linf)
> lmod.coef = coef(lm(var~age, walleye))
> K = -lmod.coef[2]; t0 = -lmod.coef[1]/lmod.coef[2]
>
> # no period effect
> nlmod1 = nls(length~Linf*(1-exp(-K*(age-t0))),
+              start=list(Linf=Linf, K=K, t0=t0), data=walleye)
>
> # most specific model
> nlmod2 = nls(length ~ (period==1)*Linf1*(1-exp(-K1*(age-t01)))+
+                (period==2)*Linf2*(1-exp(-K2*(age-t02)))+
+                (period==3)*Linf3*(1-exp(-K3*(age-t03))),
+              start=list(Linf1=Linf, Linf2=Linf, Linf3=Linf,
+                         K1=K, K2=K, K3=K,
+                         t01=t0, t02=t0, t03=t0), data=walleye)
>
> # same Linf
> nlmod3 = nls(length~(period==1)*Linf*(1-exp(-K1*(age-t01)))+
+                (period==2)*Linf*(1-exp(-K2*(age-t02)))+
+                (period==3)*Linf*(1-exp(-K3*(age-t03))),
+              start=list(Linf=Linf,
+                         K1=K, K2=K, K3=K,
+                         t01=t0, t02=t0, t03=t0), data=walleye)
>
> # same K
> nlmod4 = nls(length~(period==1)*Linf1*(1-exp(-K*(age-t01)))+
+                (period==2)*Linf2*(1-exp(-K*(age-t02)))+
+                (period==3)*Linf3*(1-exp(-K*(age-t03))),
+              start=list(Linf1=Linf, Linf2=Linf, Linf3=Linf,
```

```
+                         K=K,
+                         t01=t0, t02=t0, t03=t0), data=walleye)
>
> # same t0
> nlmod5 = nls(length~(period==1)*Linf1*(1-exp(-K1*(age-t0)))+
+                (period==2)*Linf2*(1-exp(-K2*(age-t0)))+
+                (period==3)*Linf3*(1-exp(-K3*(age-t0))),
+              start=list(Linf1=Linf, Linf2=Linf, Linf3=Linf,
+                         K1=K, K2=K, K3=K,
+                         t0=t0), data=walleye)
>
> # compare nested models
> anova(nlmod1, nlmod3, nlmod2)
Analysis of Variance Table

Model 1: length ~ Linf * (1 - exp(-K * (age - t0)))
Model 2: length ~ (period == 1) * Linf * (1 - exp(-K1 * (age - t01))) + (period == 2) * Lin
Model 3: length ~ (period == 1) * Linf1 * (1 - exp(-K1 * (age - t01))) + (period == 2) * Li
  Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
1   3195    2211448
2   3191    1994577  4 216871  86.740 < 2.2e-16 ***
3   3189    1963513  2  31064  25.226  1.35e-11 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> anova(nlmod1, nlmod4, nlmod2)
Analysis of Variance Table

Model 1: length ~ Linf * (1 - exp(-K * (age - t0)))
Model 2: length ~ (period == 1) * Linf1 * (1 - exp(-K * (age - t01))) + (period == 2) * Lin
Model 3: length ~ (period == 1) * Linf1 * (1 - exp(-K1 * (age - t01))) + (period == 2) * Li
  Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
1   3195    2211448
2   3191    2014863  4 196585  77.834 < 2.2e-16 ***
3   3189    1963513  2  51350  41.700 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> anova(nlmod1, nlmod5, nlmod2)
Analysis of Variance Table

Model 1: length ~ Linf * (1 - exp(-K * (age - t0)))
Model 2: length ~ (period == 1) * Linf1 * (1 - exp(-K1 * (age - t0))) + (period == 2) * Lin
Model 3: length ~ (period == 1) * Linf1 * (1 - exp(-K1 * (age - t01))) + (period == 2) * Li
  Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
1   3195    2211448
```

```
2   3191    1989989  4 221458  88.779 < 2.2e-16 ***
3   3189    1963513  2  26476  21.500 5.307e-10 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

The model `c1` ignores the period effect. `c5` has separate parameters for each period, and is the most general. Models `c2-c4` are intermediate, setting either the asymptote, rate or start parameters equal. In each case, we use the method suggested in previous problems to get starting values. The five models can be compared using analysis of variance. The most general model seems appropriate, so all three parameters differ in each period. Sample sizes here are very large, so the tests are very powerful and may be detecting relatively unimportant differences.

Problem 2

We first obtain the vanilla linear model:

```
> lmod = lm(medv~., Boston)
```

In the all subsets regression, a model with 11 variables turn out to be the best model, with both AIC and BIC as selection criteria (Figure 1).

```
> n = nrow(Boston); p = ncol(Boston)-1
> require(leaps)
> subsetObj = summary(regsubsets(medv~., Boston, nvmax=p))
> k = 1:p
> bicvals = subsetObj$bic
> aicvals = bicvals - k*log(n) + 2*k
> which.min(bicvals)
[1] 11
> which.min(aicvals) # both are same model
[1] 11
>
> # list of variable indices in best model
> best.ind = which(subsetObj$which[which.min(bicvals),-1])
> BostonBest = Boston[,c(best.ind,14)]
>
> lmod.as = update(lmod, data=BostonBest)
```

Comparing the smaller model with the full linear model, we see that the two extra variables in the full model do not have any significant effect.

```
> anova(lmod.as, lmod) # compare with larger model
Analysis of Variance Table

Model 1: medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +
    black + lstat
```

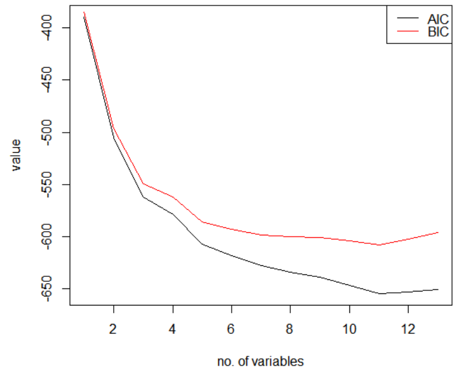

Figure 1: Plot of AIC and BIC values for all subsets linear regression on Boston housing data

```
Model 2: medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
    tax + ptratio + black + lstat
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1    494 11081
2    492 11079  2    2.5794 0.0573 0.9443
```

Next we build the LASSO and ridge regression models. All but two variables, `indus` and `age`, have non-zero coefficients in the LASSO model. Note that these two variables also got left out in the best all subsets regression model.

```
> X = as.matrix(Boston[,-14])
> y = as.matrix(Boston[,14])
> cv.lasso = cv.glmnet(X, y, alpha=1)
> lmod.lasso = glmnet(X, y, alpha=1)
> coef(lmod.lasso, s=cv.lasso$lambda.min)
14 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  34.248405456
crim         -0.097424931
zn            0.041034124
indus         .
chas          2.681496880
nox         -16.205420456
rm            3.872832939
age           .
dis          -1.386908239
rad           0.248393350
tax          -0.009648008
ptratio      -0.928430214
black         0.009000473
lstat        -0.522491506
```

```
>
> ## Ridge
> cv.ridge = cv.glmnet(X, y, alpha=0)
> lmod.ridge = glmnet(X, y, alpha=0)
> coef(lmod.ridge, s=cv.ridge$lambda.min)
14 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  28.001475824
crim         -0.087572712
zn            0.032681030
indus        -0.038003639
chas          2.899781645
nox         -11.913360479
rm            4.011308385
age          -0.003731470
dis          -1.118874607
rad           0.153730052
tax          -0.005751054
ptratio      -0.854984614
black         0.009073740
lstat        -0.472423800
```

To compare performance of the methods, we write two different functions to take care of least square type models (`geterr`) and `glmnet`-type models(`geterr.glmnet`):

```
geterr = function(data, model, nrep){
  n = nrow(data); p = ncol(data)
  mspe.vec = rep(0, nrep)

  for(i in 1:nrep){
    train = sample(1:n, floor(n/2), replace=F)
    halfmodel = update(model, data=data[train,])
    preds = predict(halfmodel, newdata=data[-train,])
    err = data[-train,p] - preds
    mspe.vec[i] = mean(err^2)
  }
  mean(mspe.vec)
}

geterr.glmnet = function(data, alpha, nrep){
  n = nrow(data); p = ncol(data)
  mspe.vec = rep(0, nrep)

  for(i in 1:nrep){
    train = sample(1:n, floor(n/2), replace=F)
```

```
    halfmodel = cv.glmnet(x=data[train,-p], y=data[train,p], alpha=alpha)
    preds = predict(halfmodel, newx=data[-train,-p], s="lambda.min")
    err = data[-train,p] - preds
    mspe.vec[i] = mean(err^2)
  }
  mean(mspe.vec)
}
```

Applying them on 100 random 1:1 splits on the data we get the following average mean squared prediction errors:

```
> set.seed(10222014)
> nrep=100
> outmat = matrix(c("Linear","All subsets linear","LASSO","Ridge",
+          geterr(Boston, lmod, nrep=nrep),
+          geterr(Boston, lmod.as, nrep=nrep),
+          geterr.glmnet(as.matrix(Boston), alpha=1, nrep=nrep),
+          geterr.glmnet(as.matrix(Boston), alpha=0, nrep=nrep)),
+        ncol=2, byrow=F)
> outmat[,2] = format(as.numeric(outmat[,2]), digits=4)
> outmat
     [,1]                 [,2]
[1,] "Linear"             "25.01"
[2,] "All subsets linear" "24.75"
[3,] "LASSO"              "24.35"
[4,] "Ridge"              "24.62"
```

Looks like there isn't much difference between the models in terms of prediction.

Note When combining model selection and cross-validation, it is not appropriate to do model selection first using the full data, then do cross-validation using training-test splits. The right thing to do here is to select the best model for each split, then do prediction on the reduced models. This is known as **Two-deep Cross Validation**. In this exercise you don't need to worry about this, but keep this in mind when you face a similar situation in future.