10702/36702 Statistical Machine Learning, Spring 2008: Homework 4 Solutions

May 8, 2008

1 [20 points], (Jingrui)

Let $X = (X_1, \ldots, X_d)^T$ where each $X_j \in \{0, 1\}$. Consider the loglinear model

$$\log p(x) = \beta_0 + \sum_{j=1}^{d} \beta_j X_j + \sum_{j<k}^{d} \beta_{jk} X_j X_k + \cdots + \sum_{j<k<l}^{d} \beta_{jkl} X_j X_k X_l + \cdots +$$

Suppose that $\beta_A = 0$ whenever $\{1, 2\} \subset A$. Show that

$$X_1 \amalg X_2 | X_3, \ldots, X_d.$$

★ **SOLUTION:** Since $\beta_A = 0$ whenever $\{1, 2\} \subset A$, the loglinear model can be written as follows.

$$\log p(x) = \sum_{1 \in A, 2 \notin A} \beta_A \prod_{j \in A} X_j + \sum_{2 \in A, 1 \notin A} \beta_A \prod_{j \in A} X_j + \sum_{1 \notin A, 2 \notin A} \beta_A \prod_{j \in A} X_j$$

Therefore,

$$p(x) = f_1(x_1, x_3, \ldots, x_d) f_2(x_2, x_3, \ldots, x_d)$$

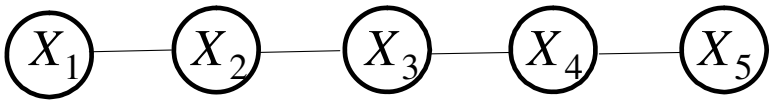
and

$$p(x_1, x_2 | x_3, \ldots, x_d) = f_1'(x_1, x_3, \ldots, x_d) f_2'(x_2, x_3, \ldots, x_d)$$

So $X_1 \amalg X_2 | X_3, \ldots, X_d$.

2 [20 points], (Jingrui)

Consider the graph: Assume all variables are binary. One loglinear model that is consistent with this graph



is

$$\log p(x) = \beta_0 + 5(X_1 X_2 + X_2 X_3 + X_3 X_4 + X_4 X_5)$$

Simulate $n = 100$ random vectors from this distribution. Fit the model

$$\log p(x) = \beta_0 + \sum_{j} \beta_j X_j + \sum_{j<k} \beta_{jk} X_j X_k$$

using maximum likelihood. Report your estimators. Use forward/backward model selection with BIC to choose a submodel. Compare the selected model to the true model.

Intercept	X_1	X_2	X_3	X_4	X_5	$X_1 \times X_2$	$X_1 \times X_3$
-24.04	-17.19	-17.19	-17.19	-17.19	-17.19	11.46	11.46
$X_1 \times X_4$	$X_1 \times X_5$	$X_2 \times X_3$	$X_2 \times X_4$	$X_2 \times X_5$	$X_3 \times X_4$	$X_3 \times X_5$	$X_4 \times X_5$
11.46	11.46	11.46	11.46	11.46	11.46	11.46	11.46

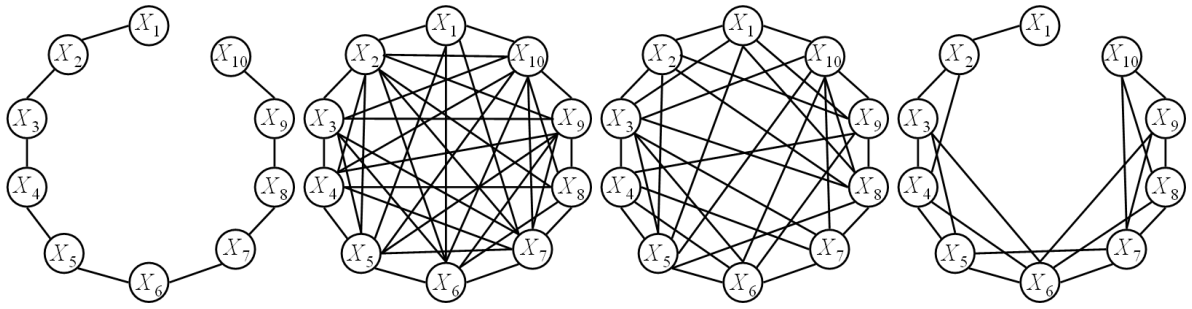
★ **SOLUTION:** Fitting the full model using maximum likelihood, we get the following coefficients:

Using backward model selection with BIC, the selected submodel is $\log p(x) = -126.63 + 26.25X_1 + 26.25X_2 + 26.25X_3 + 26.25X_4 + 26.25X_5$. Based on the true model, we can see that the probability of $X_i = 1$, $i = 1, 2, 3, 4, 5$ is 0.9864352. And the generated vectors are all 1s, which explains why both the full model and the selected model are far from the true model. This problem can be alleviated by sampling more vectors from the true model.

3 [20 points], (Jingrui)

Let $X \sim N(0, \Sigma)$ where $X = (X_1, \ldots, X_p)^T$, $p = 10$. Let $\Theta = \Sigma^{-1}$ and suppose that $\Theta(i, i) = 1$, $\Theta(i, i-1) = .5$, $\Theta(i-1, i) = .5$ and $\Theta(i, j) = 0$ otherwise. Simulate 50 random vectors and use the glasso to estimate the covariance matrix. Compare your estimated graph to the true graph.

★ **SOLUTION:** The true graph and the estimated graphs with different values of λ are shown as follows.



(a) Truth Graph (b) $\lambda = 0.05$ (c) $\lambda = 0.01$ (d) $\lambda = 0.5$

From these figures, we can see that the larger λ is, the more penalization on the L1 norm of the edges, and the fewer edges we have in the estimated graphs.

4 [20 points], (Robin)

Generate $n = 100$ observations from the model:

$$Y = m(X) + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$,$\sigma = 0.1$,

$$X \sim \text{Uniform}([0, 1]^{10})$$

and

$$m(x) = cos(5\pi x_1) + 5x_2^2.$$

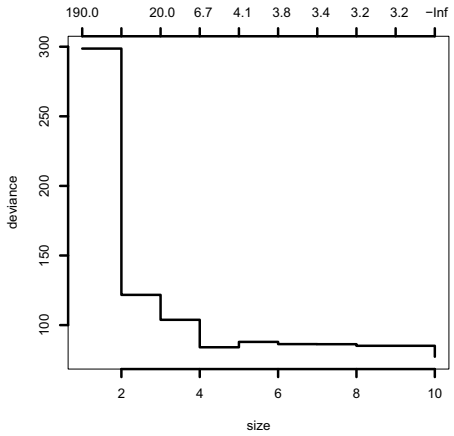
Note that x is 10-dimensional but $m(x)$ only depends on x_1 and x_2. Estimate m using (i) multivariate kernel regression, (ii) additive model, (iii) regression tree. For each estimator, report

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{m}(X_i) - m(X_i))^2.$$

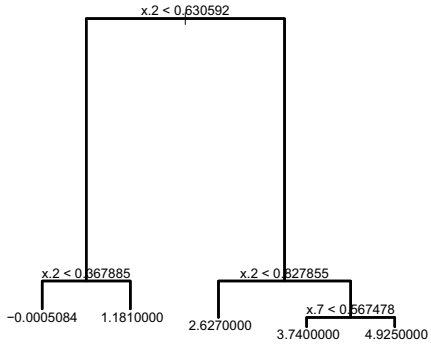
★ **SOLUTION:** This code is provided by Maxim Makatchev

```
#generate data
n=100
p=10
x = matrix(runif(p*n), nrow=n, ncol=p)
mx = cos(5*pi*x[,1]) + 5*x[,2]^2
e=rnorm(n = n, mean=0, sd=0.1)
y=mx+e

xtest = matrix(runif(p*n), nrow=n, ncol=p)
mxtest = cos(5*pi*x[,1]) + 5*x[,2]^2
e=rnorm(n = n, mean=0, sd=0.1)
ytest=mxtest+e


#training bias
H = seq(0.1, 3, length=50)
out = Kernreg(y, x, H, x)
#training bias
sum((out\$f-mx)^2/n)
#[1] 0.2056952
out\$h
#[1] 0.3367347

#test bias
ftest = kernreg(y, x, out\$h, xtest)
#test bias
sum((ftest-mxtest)^2/n)
#[1] 3.850887


######## 4b additive model
training.data = data.frame(x=x, y=y)
names(training.data) = c(paste("x", 1:10, sep=""),"y")
test.data = data.frame(x=xtest, y=ytest)
names(test.data) = c(paste("x", 1:10, sep=""),"y")

library(gam)

out = gam(y ~  lo(x1) + lo(x2) + lo(x3) + lo(x4) + lo(x5) +
               lo(x6) + lo(x7) + lo(x8) + lo(x9) + lo(x10),data=training.data)
#out1=step(gam(y ~  lo(x1) + lo(x2) + lo(x3) + lo(x4) + lo(x5) +
#               lo(x6) + lo(x7) + lo(x8) + lo(x9) + lo(x10),data=training.data),
#               data = training.data, k = log(n))
print(summary(out))

#returned model
#(Intercept) 1.0
#lo(x1)      1.0     2.4 12.3288 9.211e-06 ***
#lo(x2)      1.0     2.3 20.7006 3.446e-08 ***
#lo(x3)      1.0     2.3  0.9850 0.3873147
#lo(x4)      1.0     2.3  8.3140 0.0003579 ***
#lo(x5)      1.0     2.3  0.6713 0.5350998
```

```
#lo(x6)      1.0     2.2  0.4991 0.6280062
#lo(x7)      1.0     2.7  1.6557 0.1899930
#lo(x8)      1.0     2.4  1.0052 0.3839098
#lo(x9)      1.0     2.4  0.6429 0.5593328
#lo(x10)     1.0     2.6  0.5483 0.6242972

y_pred_train = predict(out)
##train bias
sum((y_pred_train - mx)^2/n)
#[1] 0.2096737
y_pred_test = predict(out,newdata=test.data)
#test bias
sum((y_pred_test - mxtest)^2/n)
#[1] 4.113105

########## 4c regression tree
library(tree)
out = tree(y ~ x,data=training.data)
print(summary(out))
plot(out,type="u",lwd=3)
text(out)

#training bias
sum((out\$y - mx)^2)/n
#[1] 0.009276874

postscript(``prob4_cv.ps'')
#Regression tree:
out=tree(formula = y ~ x, data = training.data)

#Variables actually used in tree construction:
#[1] "x.2" "x.4"
#Number of terminal nodes:  6
#Residual mean deviance:  0.3916 = 36.81 / 94

cv=cv.tree(out)
plot(cv,lwd=3)
dev.off()

postscript(``prob4_pruned.ps'')
newtree=prune.tree(out, best=5)
print(summary(newtree))
#Number of terminal nodes:  5
#Residual mean deviance:  0.4219 = 40.08 / 95
plot(newtree,lwd=3)
text(newtree,cex=1)
dev.off()

y_pred_train=predict.tree(newtree)
##train bias
sum((y_pred_train - mx)^2/n)
#[1] 0.3941942
```



(e) CV score



(f) Pruned Tree

5 [20 points], (Robin)

You're goal is to compare several classifiers, namely: (i) logistic regression, (ii) additive model, (iii) k-nearest neighbors and (iv) classification trees. The data are the "iris data" which is a famous dataset. These data are already in R. Type:

```
data(iris)
names(iris)
print(iris)
pairs(iris)
boxplot(iris)
```

There are three species and the goal is to predict species from the four features. We will use the first 100 observations only so there are only two species. Construct the classifiers and compare the training error rates. For the additive model, plot the estimated functions.

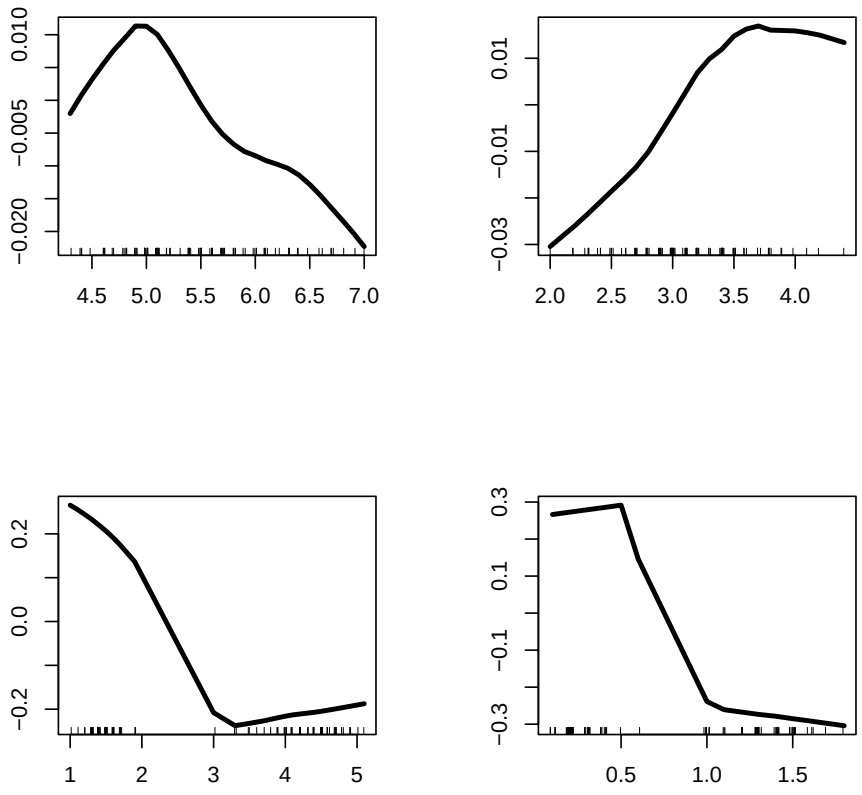
★ **SOLUTION:** This code is provided by Yu-Hsiang Chi

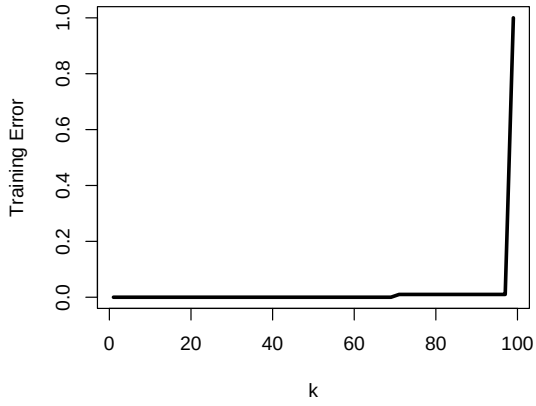
```
Logistic Regression: Training Error:  0
Additive Regression: Training Error:  0
K-NN Training Error: 0
Classification Tree Training Error: 0
```

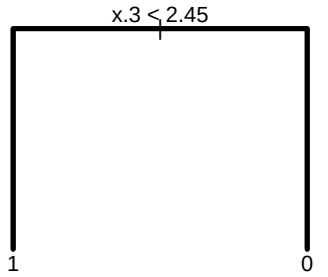
```
hat = function(p){
    n = length(p)
    y = rep(1,n)
    y[p < .5] = 0
    return(y)
}

# data from Iris dataset
n <- 100
x <- matrix(rep(0,n*4),n,4)
```

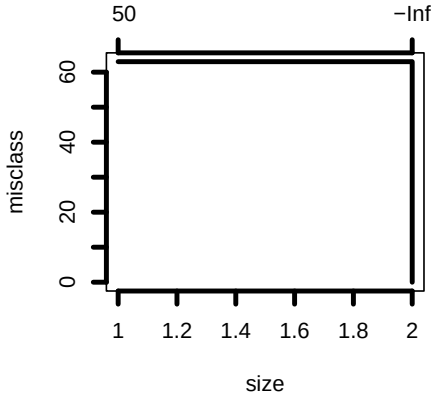
```
for(i in 1:4){
    x[,i] <- iris[1:n,i]
}

y <- matrix(rep(0,n),n,1)
for(i in 1:n){
    if(iris[i,5]=="setosa"){
       y[i,1] <- 1
    }
}
training.data <- data.frame(x=x,y=y)
names(training.data) = c(paste("x",1:4,sep=""),"y")

### logistic regression
out_lr   = glm(y~ .,family="binomial",data=training.data)
p     = predict(out_lr,type="response")
yhat_lr  = hat(p)
error_lr = mean(y != yhat_lr)
cat("Logistic Regression: Training Error: ",error_lr,"\n")

### additive model
library(gam)

out_add = gam(y~ lo(x1) + lo(x2) + lo(x3) + lo(x4), data = training.data)
print(summary(out_add))

yhat_add  = hat(predict(out_add))
error_add = mean(y != yhat_add)
cat("Additive Regression: Training Error: ",error_add,"\n")

postscript("prob5_fig1.ps");
par(mfrow=c(2,2))
plot(out_add,lwd=3,xlab="",ylab="")
dev.off();

### knn
library(class)
kmax = 50
error2 = rep(0,kmax)
index = rep(0,kmax)
for(i in 1:kmax){
    y_cv = knn.cv(x, y, k = (2*i-1))
    error2[i] = mean(y_cv != y)
    index[i] = 2*i-1
}

postscript("prob5_fig2.ps");
par(mfrow=c(1,1))
plot(index,error2,type="l",lwd=3,xlab="k",ylab="K-NN Training Error")
dev.off();


### Classification tree
library(tree)
```

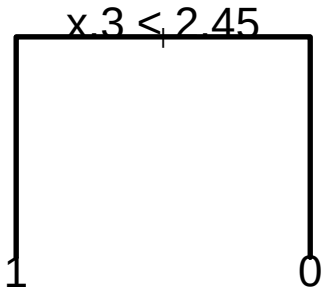
```
out_tree = tree(as.factor(y) ~ x,data=training.data)
print(summary(out_tree))

postscript("prob5_fig3.ps");
par(mfrow=c(2,2))
plot(out_tree,type="u",lwd=3)
text(out_tree)

cv = cv.tree(out_tree,method="misclass")
plot(cv,lwd=3)

newtree = prune.tree(out_tree,best=cv$size[which.min(cv$dev)],method="misclass")

print(summary(newtree))

# Misclassification error rate: 0 = 0 / 100

plot(newtree,lwd=3)
text(newtree,cex=2)
dev.off();
```




Training Error
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
k


x.3 < 2.45
1
0


50
−Inf
misclass
0
20
40
60
1
1.2
1.4
1.6
1.8
2
size


x.3 < 2.45
1
0