Homework 8

Advanced Methods for Data Analysis (36-402)

Due Friday March 29, 2019, at 3:00 PM

See the syllabus for general instructions regarding homework. Note that you should **always show all your work** and submit **exactly two files** (Rmd or R file as *code*, and knitted or scanned/merged pdf or html as *writeup*). Make sure that everything you submit is readable.

Problem 1: Degrees of freedom for linear smoothers

Suppose that $Y_i = \mu(x_i) + \epsilon_i$, where ϵ_i are uncorrelated and have mean 0, but each row has its own variance σ_i^2. Consider modifying the definition of degrees of freedom to $\sum_{i=1}^{n} \frac{Cov[Y_i, \hat{Y}_i]}{\sigma_i^2}$. Show that this still equals to $tr(W)$ for a linear smoother with influence matrix W.

$$\begin{aligned}
\text{Cov}\left[Y_i, \hat{Y}_i\right] &= \text{Cov}\left[Y_i, \sum_{j=1}^{n} w_{ij} Y_j\right] \\
&= \sum_{j=1}^{n} w_{ij} \,\text{Cov}\left[Y_i, Y_j\right] \\
&= w_{ii} \mathbb{V}\left[Y_i\right] = \sigma_i^2 w_{ii}
\end{aligned}$$

$$df(\widehat{\mu}) = \sum_{i=1}^{n} \frac{\text{Cov}\left[Y_i, \hat{Y}_i\right]}{\sigma_i^2} = \sum_{i=1}^{n} \frac{\sigma_i^2 w_{ii}}{\sigma_i^2} = \sum_{i=1}^{n} w_{ii} = tr(W)$$

Problem 2: Degrees of freedom for kernels and splines (housing data revisited)

For this problem, we will use the `housing` data that you have used in HWs 2 and 5. Merge the traning and test data sets into a single data set.

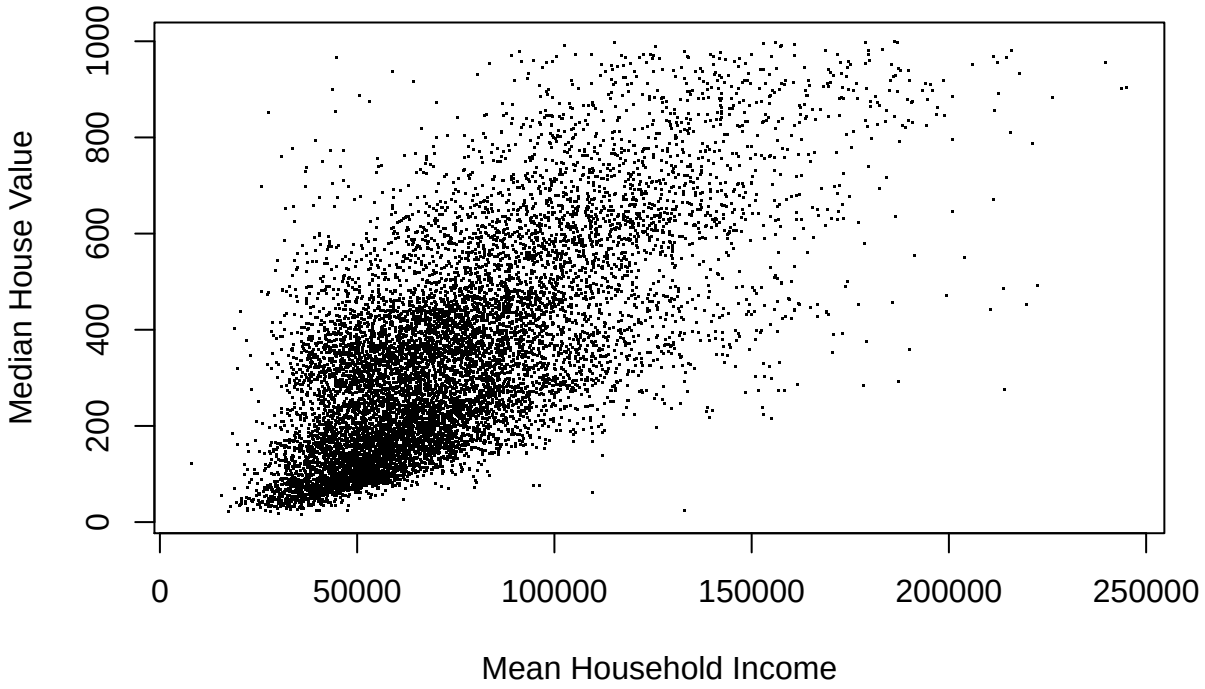
We will be fitting models in which we try to predict median house value from mean household income. We will use both splines and normal-kernel regressions with a variety of tuning parameters while we attempt to use "effective degrees-of-freedom" to make the comparisons on a common scale.

Part a

Plot `Median_house_value` versus `Mean_household_income`.

```
library(np)
htest=read.csv("housetest.csv",header=T)
htrain=read.csv("housetrain.csv",header=T)
housing=rbind(htrain,htest)
```

```
plot(housing$Mean_household_income,housing$Median_house_value,pch=".",
xlab="Mean Household Income",ylab="Median House Value")
```



Part b

First, fit several spline modes with different smoothing parameters. We will use `smooth.spline` with the `df` version of the smoothing parameter. (This is the same as the `effective degrees-of-freedom` that was defined in lecture.)

Use `df` parameter values from 2 to 27 in steps of 1. For each `df`, run five-fold cross-validation on the full data set, computing the estimated MSE of prediction on each held-out fold. Compute the sample average of the five estimated MSE of prediction values for each `df`. Also, for each `df`, compute an estimated standard error for the average by computing the sample standard deviation of the five estimated MSE of prediction values divided by $\sqrt{5}$. Plot the sample average of the five estimated MSE of prediction values against `df`, and find the `df` value that provides the lowest average. Compare the differences between the plotted averages to the estimated standard errors of the averages. Say why this comparison either does or does not inspire confidence that we have really found the `df` that provides the best out-of-sample prediction.

First, we initialize the `df` parameter values and create the folds. These same folds will be used again for the kernel regressions. The remaining discussion will follow the requested plot.

```
dfss=2:27

k=5
thefolds=sample(rep(1:k,length=nrow(housing)),replace=F)

predss=matrix(NA,k,length(dfss))

for(fold in 1:k){
      train=housing[thefolds!=fold,]
      test=housing[thefolds==fold,]
      for(j in 1:length(dfss)){
```

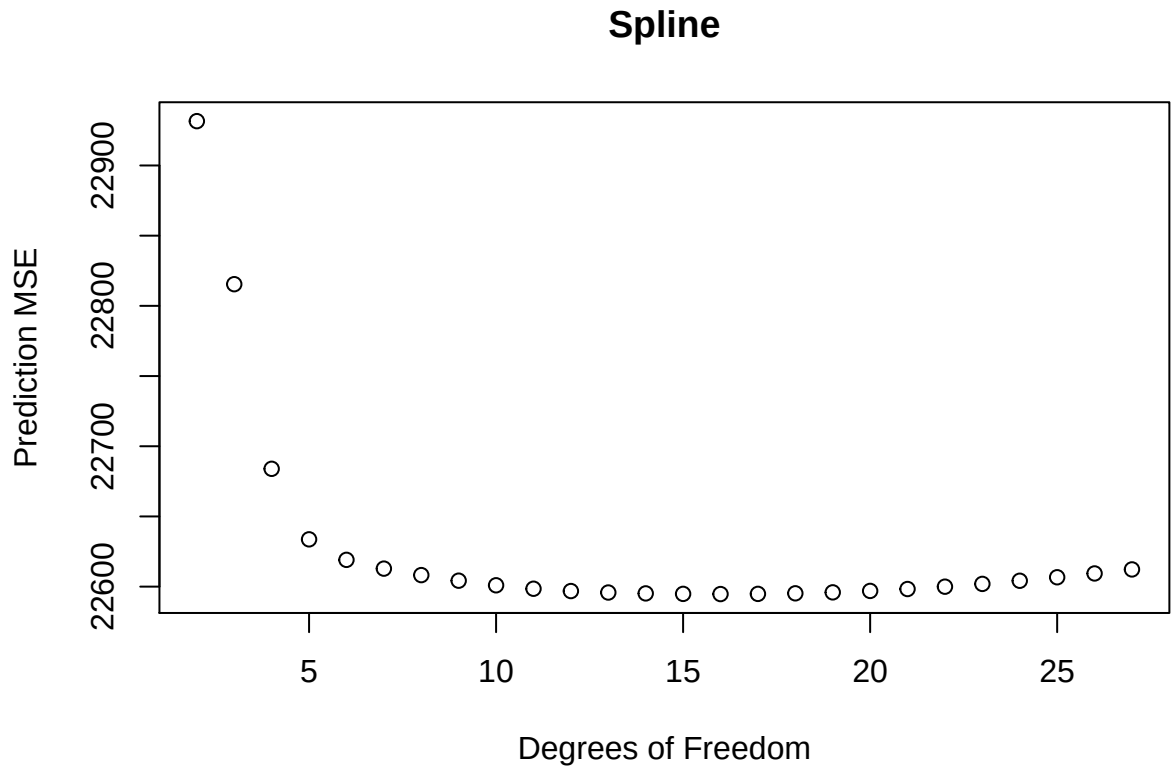
```
    ssfit=smooth.spline(train$Median_house_value~
  train$Mean_household_income,df=dfss[j])
    predss[fold,j]=mean((test$Median_house_value-
  predict(ssfit,x=test$Mean_household_income)$y)^2)
    }
}

predssa=apply(predss,2,mean)
predssa
```

```
##  [1] 22931.57 22815.44 22683.92 22633.68 22619.07 22612.82 22608.21
##  [8] 22604.22 22600.94 22598.49 22596.84 22595.80 22595.16 22594.82
## [15] 22594.70 22594.83 22595.23 22595.92 22596.94 22598.29 22599.97
## [22] 22601.93 22604.18 22606.67 22609.37 22612.25
```

```
sdss=apply(predss,2,sd)/sqrt(k)
sdss
```

```
##  [1] 327.5279 320.7695 316.2679 314.8346 313.9318 312.9840 311.8413
##  [8] 310.4740 308.9169 307.2614 305.6145 304.0691 302.6666 301.4573
## [15] 300.4419 299.6078 298.9425 298.4319 298.0558 297.7974 297.6386
## [22] 297.5625 297.5566 297.6055 297.6984 297.8291
```

```
plot(dfss,predssa,xlab="Degrees of Freedom",
          ylab="Prediction MSE",main="Spline")
```




```
dfssm=dfss[predssa==min(predssa)]
dfssm
```

```
## [1] 16
```

The bandwidth that provides the lowest estimated prediction MSE is 16.

Notice that the estimated prediction MSE's vary by a few hundred from the best to the worst `df`, but the estimated standard errors are all around 300.
It would not be surprising to find that we got a different "best" `df` with a different set of folds. As long as we stay away from the really small `df` values, the estimated prediction MSE doesn't change by much as a percentage.

Part c

Next, perform a similar analysis for kernel regression with a normal (Gaussian) kernel. This time, there is no `df` option for the smoothing parameter. Instead, use the `bws` parameter to set the bandwidth to each of the values from 4000 to 7000 in steps of 300. (That should be 11 different values, for a sanity check.) Make sure that you use the same five folds for five-fold cross-validation as you used in part (b). The comparisons will not be fair if you use different folds. Use the `npreg` function in `library(np)` to fit the kernel regressions. In the instructions from part (b), replace `df` by `bws` and repeat all of the analyses for the kernel regression models. (This calculation is time-consuming.)

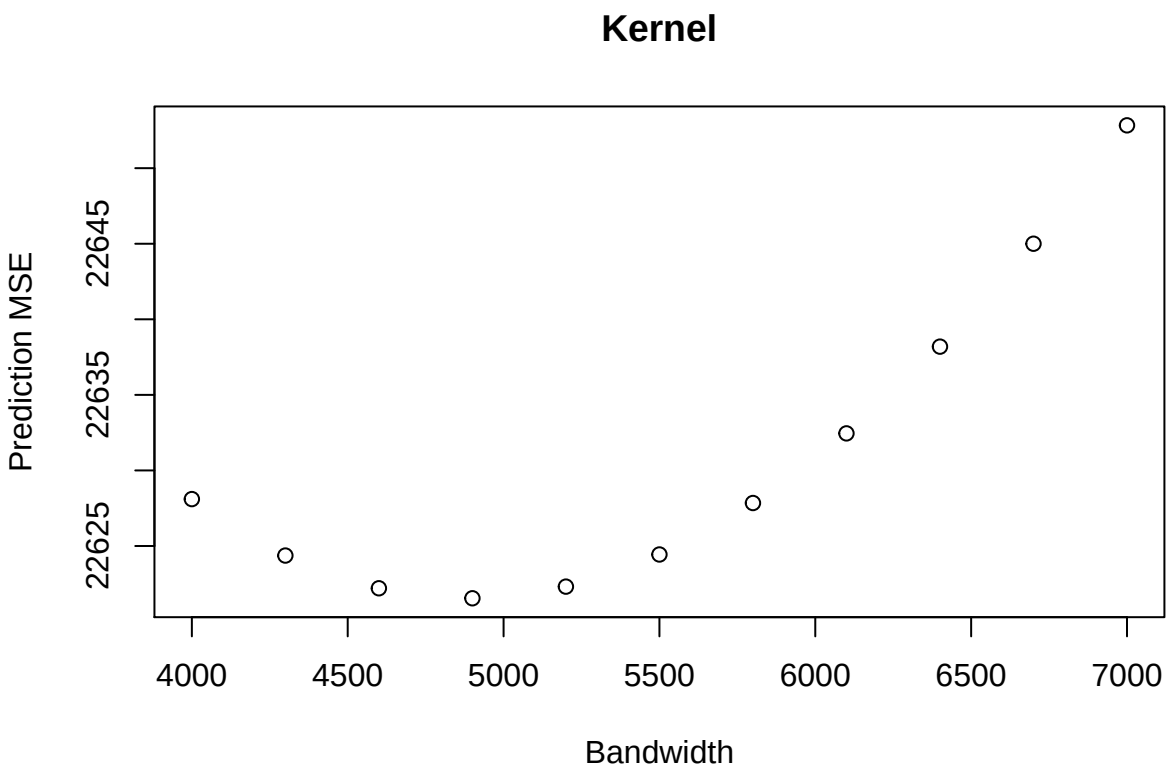
```
bws=4000+3000*c(0:10)/10

predkr=matrix(NA,k,length(bws))

for(fold in 1:k){
      train=housing[thefolds!=fold,]
      test=housing[thefolds==fold,]
      for(j in 1:length(bws)){
      krfit=npreg(Median_house_value~Mean_household_income,data=train,
    bws=bws[j])
      predkr[fold,j]=mean((test$Median_house_value-
    predict(krfit,newdata=test))^2)
      }
}

predkra=apply(predkr,2,mean)
predkra
```

```
##  [1] 22628.10 22624.37 22622.20 22621.54 22622.31 22624.44 22627.84
##  [8] 22632.45 22638.19 22645.00 22652.83
```

```
sdkr=apply(predkr,2,sd)/sqrt(k)
sdkr
```

```
##  [1] 314.3207 314.9570 315.6856 316.4865 317.3502 318.2756 319.2675
##  [8] 320.3352 321.4895 322.7419 324.1023
```

```
plot(bws,predkra,xlab="Bandwidth",
           ylab="Prediction MSE",main="Kernel")
```




```
bwkrm=bws[predkra==min(predkra)]
bwkrm
```

```
## [1] 4900
```

The bandwidth that provides the best estimated prediction MSE is 4900. Once again, the estimated standard errors are all around 314.3 or more, while the differences between the estimated MSE's are all less than 100. We would not be surprised to get a different "best" bandwidth with a different choice of folds.

Part d

For each bandwidth h, the effective degrees-of-freedom for a kernel regression is

$$df(h) = K(0) \sum_{i=1} \frac{1}{\sum_{j=1}^{n} K([x_i - x_j]/h)}.$$

(This calculation is time-consuming.) For each bandwidth (`bws`) in part (c), compute the corresponding effective degrees-of-freedom. Draw a single plot that contains the pairs

(effective degrees-of-freedom,estimated MSE of prediction)

for both the spline fits and the kernel fits, labeled well enough to be able to tell which are which. Comparing the differences between the estimated MSE's for splines and kernels to the estimated standard errors, say why this comparison does or does not inspire confidence that we can tell which method provides better out-of-sample prediction.

Below, you will see a function that computes the diagonal elements of the L "hat" matrix for a kernel regression and adds them up. The remainder of the discussion will appear after the requested plot.

```
kdf=function(x,h){
 n=length(x)
 s=0
```

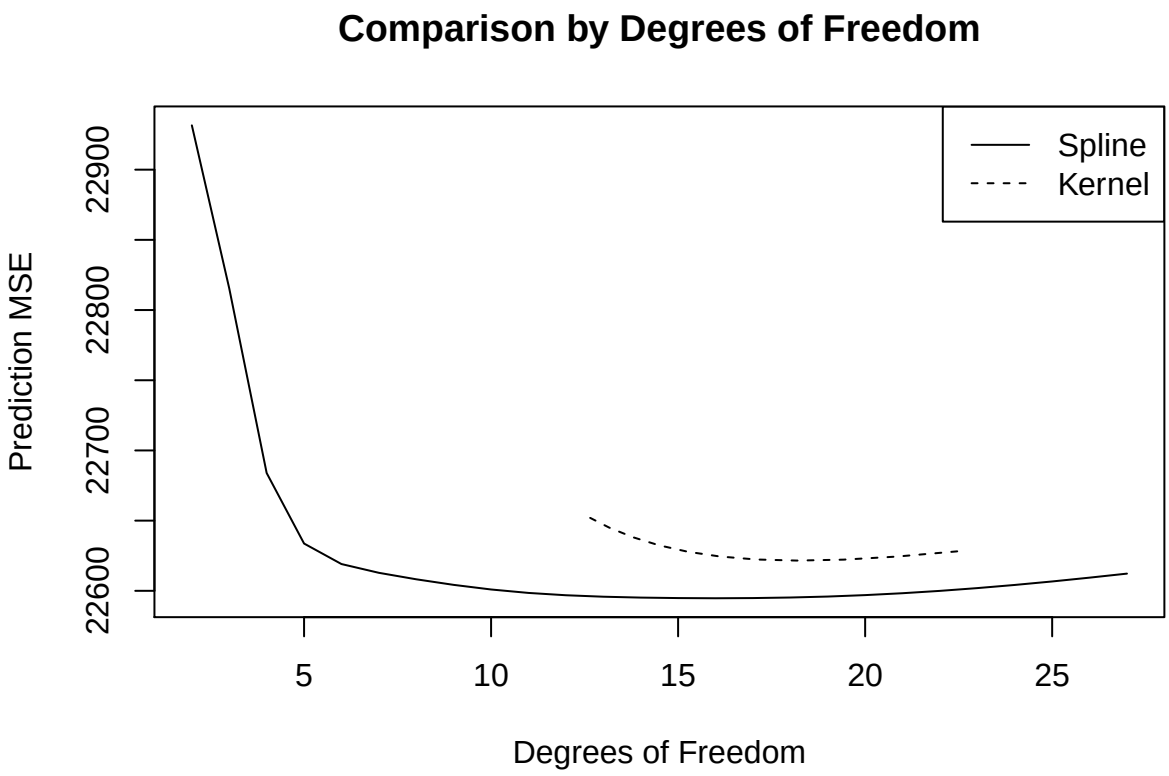
```
 for(j in 1:n){s=s+1/sum(dnorm((x-x[j])/h))}
   s*dnorm(0)
}

dfkr=NULL
for(j in 1:length(bws)){
       dfkr[j]=kdf(housing$Mean_household_income,bws[j])
}

signif(data.frame(bandwidth=bws,df=dfkr),digi=4)
```

```
##    bandwidth    df
## 1       4000 22.46
## 2       4300 20.84
## 3       4600 19.42
## 4       4900 18.18
## 5       5200 17.08
## 6       5500 16.12
## 7       5800 15.26
## 8       6100 14.48
## 9       6400 13.79
## 10      6700 13.16
## 11      7000 12.59
```

```
dfkrm=dfkr[predkra==min(predkra)]
dfkrm
```

```
## [1] 18.17743
```

```
thetop=max(c(predkra,predssa))
thebottom=min(c(predkra,predssa))
dfmin=min(c(dfkr,dfss))
dfmax=max(c(dfkr,dfss))
plot(c(dfmin,dfmax),c(thetop,thebottom),pch="",xlab="Degrees of Freedom",
main="Comparison by Degrees of Freedom",ylab="Prediction MSE")
lines(dfss,predssa,lty=1)
lines(dfkr,predkra,lty=2)
legend("topright",lty=1:2,legend=c("Spline","Kernel"))
```



The values of effective degrees-of-freedom are decreasing as functions of the bandwidth, as we would have expected. Larger bandwidths make it harder to track the data. The effective degrees-of-freedom that correspond to the bandwidths that we used are between 12.59 and 22.46, much closer together than the ones that we used for fitting splines.

The effective degrees-of-freedom that corresponds to the bandwidth with smallest estimated prediction MSE is 18.2. Over the small range of effective degrees-of-freedom that we used, the spline fits are uniformly better than the kernel fits in terms of cross-validation error. The differences between the two are still small compared to the estimated standard errors, but they are about four times the size of the differences between the group of best spline fits.

Part e

Finally redraw the plot from part (a) and add

(i) a line for the spline fit corresponding to the best smoothing parameter value, and

(ii) a line for the kernel regression fit with the best smothing parameter value.

For the two lines, use a common set of 201 predictor values that is equally-spaced over the observed range of `Mean_household_income` and extending 5% at each end. Comment on how well or badly each of the two fitted curves seems to fit the data. Also offer a reason for why the two fitted curves behave so differently at and beyond the extremes of the predictor.

The redrawn plot appears below with the curves added. There appear to be two "clouds" of data points in the plot, one right above the two fitted curves and one right below. The curves are nearly identical throughout much of the range of the data, diverging from each other near the extremes as one would expect. Kernel regressions with a Gaussian kernel eventually flatten out at the response value corresponding to the most extreme observation at each end. Smoothing splines become linear with a slope that approximates the general direction in which the points are moving at each end.

```
therange=range(housing$Mean_household_income)
thelength=therange[2]-therange[1]
x0=therange[1]-.05*thelength+1.1*thelength*c(0:200)/200

plot(x0[c(1,201)],c(range(housing$Median_house_value)),pch="",
xlab="Mean Household Income",ylab="Median House Value")
points(housing$Mean_household_income,housing$Median_house_value,pch=".")
krmin=npreg(Median_house_value~Mean_household_income,data=housing,bws=bwkrm)
ssmin=smooth.spline(housing$Median_house_value~housing$Mean_household_income,
    df=dfssm)
lines(x0,predict(ssmin,x=x0)$y,lty=3)
lines(x0,predict(krmin,newdata=data.frame(Mean_household_income=x0)),lty=2)
legend("bottomright",lty=c(3,2),legend=c("Spline","Kernel"))
```