Topics covered in this lecture

- K-nearest neighbors
- Lasso
- Decision Trees

There is no reading for these lectures. Just notes. However, I have copied a number of online websites that may help to the course schedule.

There are some extra notes on the website (prepared by a former PhD student), which provide more examples for those who are interested.
In 216 and 217, we have (mostly) evaluated empirical relationships using parametric models.

Parametric models almost surely have some form of model misspecification, but are helpful in that the techniques to analyze the models are well sussed-out and interpretations of the model are fairly straightforward (i.e., take a derivative).

In new data science lingo, we are "supervising" the data with a model.

In this last few lectures, we have been more flexible with our modeling choices.

- More flexible models that are non-parametric
- Resampling procedures to conduct inference and choose smoothing parameters

Practically, much of the new data science literature, learning and otherwise, isn’t all that different from what we’re doing already.

The main difference is the choice of non-parametric model, and the goal is to improve prediction.
Modern data science (cont.)

- Whether you adopt new techniques or old techniques is usually a function of your research objective.

- In economics, we often wish to understand the mechanisms behind behaviors, as opposed to the collection of attributes that lead to behaviors.
  - Example: Knowing that graduates from Harvard are more likely to own a new house than graduates of Cabrillo college might be interesting from a marketing perspective, but it tells us nothing of why this is the case.
  - If we are constructing policy, we want to know why. That is the big difference between modern data science and econometrics as I see it (even though in principle the techniques are very similar).
  - To be sure, the techniques can be complementary.

- In this lecture, we will study three techniques:
  - **K-Nearest Neighbors**: Similar people do similar things.
  - **LASSO**: A common technique for model selection.
  - **Decision Trees**: Individuals adopt a heuristic to make choices.
K-Nearest Neighbors

- K-Nearest Neighbors is extremely similar to the Nadaraya-Watson binned estimator
  - In NW, we take a bandwidth of $h$ on either side of a given $x$, and average the behaviour within the region to generate a prediction for $y$.
  - This technique can be extended to more than one dimension of $x$ by using a measure of "Euclidean Distance".

- K-Nearest Neighbors (KNN) also measures average (or modal) behavior around a particular point.
  - Instead of a fixed distance of $h$ around a particular $x$, KNN, uses $k$ nearest neighboring observations to measure behavior.

- The key inputs to a basic KNN model
  - The choice of $k$ (obviously)
  - The distance function
  - The outcome variable (e.g. unemployment)
  - The input variables (which will be used to determine who is nearest)
Euclidean Distance is a common measure of distance.

In P dimensions, Euclidean distance of two observations, $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$, and $x_j = (x_{j1}, x_{j2}, \cdots, x_{jp})$, is:

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{p} (x_{il} - x_{jl})^2}$$

In one dimension, it is just absolute distance.

In two dimensions, this is basically the Pythagorean theorem.

Other distance functions exist, but we’ll just use Euclidean distance.
K-Nearest Neighbors - Outcomes

- In the NW estimator, we averaged outcomes within the bandwidth.
  - Eg. Average real wage
  - Averages might be weighted by a kernel function
- In data science jargon, outcomes can also be "classifications"
  - Unemployed, part-time, employed, out of workforce
  - Classifications are hard to average
- For KNN, the prediction is:
  - Average value if outcome is numeric
  - The modal value if outcome is a classification (this is called "majority rule" in data science lingo)
- Similar to $h$ being chosen by cross-validation in NW, $k$ can be chosen by a similar technique for KNN.
R example: K-Nearest Neighbors

- **Load the necessary libraries**
  
  ```r
  library(caret)
  library(foreign)
  ```

- **Load and clean data**

  ```r
d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")
d<-subset(d,is.na(nilf)==FALSE)
d<-subset(d,is.na(educ)==FALSE)
d<-subset(d,is.na(age)==FALSE)
d<-subset(d,is.na(female)==FALSE)
  ```

- **Construct the "training" and "testing" samples:**

  ```r
  subtrain<-subset(d,year==2013&state=="CA")
  subtest<-subset(d,year==2013&state!="CA")
  ```

- **Run your model:**

  ```r
  model.knn <- train(nilf~age+educ+female, data = subtrain, method = "knn")
  ```
After the regression, check accuracy using the training sample

```r
val.pred <- predict(model.knn, subtrain)
```

Calculate the share of predictions that match the actual values in the training sample

```r
val.acc <- sum(val.pred == subtrain$nilf, na.rm=TRUE)/length(subtrain$nilf)
print(val.acc)
```

Now do the same with the testing sample

```r
pred <- predict(model.knn, subtest)
accuracy <- sum(pred == subtest$nilf, na.rm=TRUE)/length(subtest$nilf)
print(accuracy)
```

By comparing "acc" and "accuracy", we can compare how well the model does within sample and out of sample.
The results look pretty poor. So, let’s redefine our outcome variable as non-numeric:

```r
d$nilf2 <- ifelse(d$nilf==1, "Out of Labor Force", "In Labor Force")
```

Re-construct the "training" and "testing" samples:

```r
subtrain <- subset(d, year==2013 & state=="CA")
subtest <- subset(d, year==2013 & state!="CA")
```

Run the model:

```r
model.knn2 <- train(nilf2 ~ age + educ + female, data = subtrain, method = "knn")
```

And compare accuracy:

```r
val.pred <- predict(model.knn2, subtrain)
val.acc <- sum(val.pred == subtrain$nilf2, na.rm=TRUE)/length(subtrain$nilf2)
pred <- predict(model.knn2, subtest)
accuracy <- sum(pred == subtest$nilf2, na.rm=TRUE)/length(subtest$nilf2)
print(val.acc)
print(accuracy)
```
R example: KNN with more than two outcomes

- Labor force models often distinguish between labor force participation, and if so, employment and unemployment
- Augmenting our models to account for this:
  
  ```
  d$nilf3 <- ifelse(d$nilf == 1, "Out of Labor Force", ifelse(d$empl == 0, "Unemployed", "Employed"))
  ```
- Re-construct the "training" and "testing" samples:
  
  ```
  subtrain <- subset(d, year == 2013 & state == "CA")
  subtest <- subset(d, year == 2013 & state != "CA")
  ```
- Run the model:
  
  ```
  model.knn3 <- train(nilf3 ~ age + educ + female, data = subtrain, method = "knn")
  ```
- And compare accuracy:
  
  ```
  val.pred <- predict(model.knn3, subtrain)
  val.acc <- sum(val.pred == subtrain=nilf3, na.rm=TRUE)/length(subtrain=nilf3)
  pred <- predict(model.knn3, subtest)
  accuracy <- sum(pred == subtest=nilf3, na.rm=TRUE)/length(subtest=nilf3)
  print(val.acc)
  print(accuracy)
  ```
Model selection is an important issue in econometrics

- We have a choice of how many variables to include.
- Including more variables must make predictions better (weakly), but may reduce precision.

The LASSO:

- "Least Absolute Shrinkage and Selection Operator"

Suppose that we have $N$ observations, $P$ potential explanatory variables

The Lasso Problem:

$$\min_{\beta_p} \sum_{i=1}^{N} \left( y_i - \sum_{p=1}^{P} \beta_p x_{ip} \right)^2$$  \hspace{1cm} (1)

$$s.t. \sum_{p=1}^{P} |\beta_p| < \lambda$$  \hspace{1cm} (2)

- (1) is the OLS problem.
- (2) constrains the total absolute size of all coefficients
The LASSO (cont.)

- We’ll study LASSO by estimating a third degree spline predicting labor force participation:

\[
\min_{\beta_p} \sum_{i=1}^{N} \left( \text{nilf}_i - \sum_{p=0}^{3} \beta_p \text{age}_i^p - \sum_{a \in A} \beta_a (\text{age}_i - c_a)^3 \mathbf{1}(\text{age}_i > c_a) \right)^2
\]

\[
\text{s.t. } \sum_{p=0}^{3} |\beta_p| + \sum_{a \in A} |\beta_a| < \lambda
\]

where \( a \in A \) identifies as set of age knots, \( c_a \)

- \( \lambda \) can be chosen by cross-validation. Let’s first look at the procedure

- Load the required libraries and the org data

```r
library(lars)
library(foreign)
d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")
d<-subset(d,is.na(nilf)==FALSE & is.na(age)==FALSE & year==2013)
sd<-d[,c("nilf","age")]
sd<-sd[order(sd$age),]
```
The LASSO (cont.)

- **Generate series terms**
  
  \[
  \text{sd}$\text{age}2 <- \text{sd}$\text{age}^2 \\
  \text{sd}$\text{age}3 <- \text{sd}$\text{age}^3 \\
  \]

- **Generate many spline terms and constant**
  
  \[
  \text{ages} <- \text{seq(from=18, to=70, by=2)} \\
  \text{for(a in ages)}{ \\
    \text{sd}$\text{newvar} <- \text{ifelse(sd}$\text{age} \geq \text{a}, (\text{sd}$\text{age-a})^3, 0) \\
    \text{names(sd)}[\text{ncol(sd)}] <- \text{paste("agespline","a",sep="\_")} \\
  } \\
  \text{sd}$\text{cons} <- 1 \\
  \]

- **Run a regression, a LASSO, and compare coefficients**
  
  \[
  \text{rhs} <- \text{sd} \\
  \text{rhs}$\text{nilf} <- \text{NULL} \\
  \text{rhs} <- \text{as.matrix(rhs)} \\
  \text{lhs} <- \text{as.matrix(sd}$\text{nilf)} \\
  \text{lm.reg} <- \text{lm(nilf~.,data=sd)} \\
  \text{lasso.reg} <- \text{lars(rhs, lhs, type="lasso", normalize=TRUE)} \\
  \]
The LASSO (cont.)

- Choose the optimal $\lambda$ via cross validation
  
  ```r
  CVlasso<-cv.lars(rhs,lhs,K=10,type="lasso",normalize=TRUE)
  str(CVlasso)
  ```

- Extract the optimal $s$ using "which.min" and "index"

  ```r
  opt<-CVlasso$index[which.min(CVlasso$cv)]
  predict(lasso.reg,s=opt,type="coef",mode="fraction")
  ```

- Plot LASSO predictions and compared with linear regression.

  ```r
  lassopredict<-{predict(lasso.reg,newx=rhs,s=opt,
                   type="fit",mode="fraction")$fit}
  lmpredict<-predict(lm.reg)
  plot(lassopredict sd$age,type='l',lwd=3)
  lines(lmpredict sd$age,lwd=3,col="red")
  ```
Decision Trees

- Decision Trees are a form of classification, and map nicely into a "heuristic" approach of decision making by individuals.

- An example: Buying a car
  - Car or Truck
    - Domestic or Foreign

- Decision Trees can also be used to categorize outcomes by defining thresholds

- Suppose the outcome is "employed"
  - White or Non-White
    - Education greater than X, or less than X

- These are very complex models, but they generally require (1) an order of "sub-trees", (2) splitting variables and (3) splitting points.

  - All three components can be chosen by cross-validation.

- The technique that is used for estimation is called "recursive partitioning".
R example: Decision Trees

Let’s evaluate employment outcomes as a function of education and demographics.

Load the required libraries

```r
library(rpart)
library(foreign)
```

Reload and prepare outcome variable

```r
d<-read.dta("/Users/acspearot/Data/CPSDWS/org_example.dta")
d<-subset(d,is.na(educ)==FALSE&is.na(age)==FALSE &is.na(female)==FALSE&is.na(nilf)==FALSE)
```

Take "lfstat", which is labor force status, and create a dichotomous variable for whether or not the respondent is employed

```r
d$lfstat2<-ifelse(d$lfstat="Employed","Employed","Not Employed")
```

Also, it will be easier if we create a gender factor variable:

```r
d$gender=ifelse(d$female==1,"female","male")
```
Just like with the KNN, create the training and testing samples

```
subtrain <- subset(d, year == 2013 & state == "CA")
subtest <- subset(d, year == 2013 & state != "CA")
```

Run the classification tree

```
tree <- rpart(lfstat2 ~ educ + wbho + gender, data = subtrain, method = "class")
```

Use plot and labeling functions from `rpart` to visualize the results

```
plot(tree, cex=1.5, branch=0, main="Decision Tree for Employment", margin=.05)
text(tree, cex=1.5, use.n=TRUE, minlength=0)
```

Convention on plots:

- To the left when condition is satisfied
- Counts at bottom are in order of aggregate frequency
Try again on the three outcome employment status model

```r
d$nilf3 <- ifelse(d$nilf==1, "Out of Labor Force", ifelse(d$empl==0, "Unemployed", "Employed"))
subtrain <- subset(d, year==2013 & state=="CA")
subtest <- subset(d, year==2013 & state!="CA")
```

Plot the results

```r
tree2 <- rpart(nilf3 ~ educ + wbho + gender, data = subtrain, method = "class")
plot(tree2, cex=1.5, branch=0, main="Decision Tree for Labor Force Status", margin=.05)
text(tree2, cex=1.5, use.n=TRUE, minlength=0)
```

Evaluate how the testing model works

```r
outcomes <- predict(tree2, subtest, type='class')
subtest$outcomes <- as.character(outcomes)
sum(subtest$outcomes==subtest$nilf3)/nrow(subtest)
```

Compare this with the KNN precision in the testing dataset.