Lecture 6: Machine Learning
Paradigma Predictivo

Big Data and Machine Learning en el Mercado Inmobiliario
Educación Continua

Ignacio Sarmiento-Barbieri

Universidad de los Andes

September 2, 2021

Agenda

Modelo Monocéntrico

- Planteamos un modelo monocentrico
- Nos hablaba de la importancia de ciertos elementos que nos van a dar el valor de la tierra (determinar su uso)
  - Distancia al Centro
  - Costos de Transporte
  - Amenidades
  - Costos de construcción (ignoramos por ahora)

Precio de Propiedades

- Podemos pensar una casa como un bien diferenciado: Z que tiene distintas características $(z_1, z_2, \ldots, z_n)$

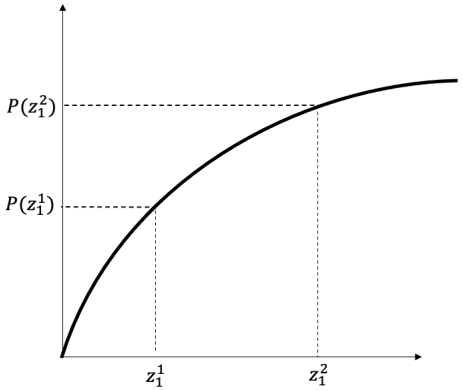
$$Z = (z_1, z_2, \ldots, z_n) \tag{1}$$

- Este producto diferenciado se vende en un mercado competitivo, y la interacción de compradores y vendedores determinan el precio del bien

$$P(Z) = f(z_1, z_2, \ldots, z_n) \tag{2}$$

- El precio de equilibrio para cada variedad del bien diferenciado (por ejemplo una casa en particular) es una función de los atributos de la misma.
- Valuación hedónica (Rosen, 1974)

Precio de Propiedades



Estadística Clásica vs Paradigma Predictivo

$$y = f(X) + u \tag{3}$$

- Estadística Clásica
  - Inferencia
  - $f()$ "correcta" el interes es en entender como y afecta X
  - modelos surge de la teoria/experimentos
  - Interés es en test de hipótesis (std. err., ci's)
- Maquina de Aprender
  - Interés es predecir y
  - El $f()$ correcto es el que predice mejor
  - Modelo?

Estadística Clásica vs Paradigma Predictivo

- Working model is

$$y = f(X) + u \tag{4}$$

- Linear regression is the "work horse" of econometrics and (supervised) machine learning.

$$y = X\beta + u \tag{5}$$

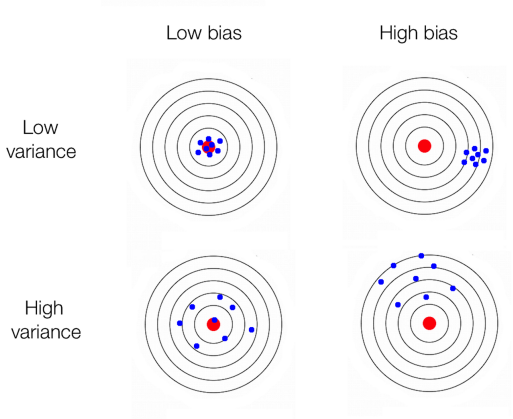
  - All the interest is on β
  - Gauss-Markov Theorem says that under classical assumptions it is BLUE

Mean Square Error

$$MSE(\beta) = E(\hat{\beta} - \beta)^2 \tag{6}$$
$$= E(\beta - E(\hat{\beta}))^2 + Var(\hat{\beta}) \tag{7}$$

- Intuitively, the result says that how wrong is the estimate (MSE) depends on:
  - how uncentered it is (bias) and
  - how dispersed it is around its center (variance).

Mean Square Error



Source: https://tinyurl.com/y3xlh87o

Prediction Error

- Now suppose that the goal is to predict Y with another random variable $\hat{Y}$.
- The *prediction error* is defined as:

$$Err(\hat{Y}) \equiv E\left(Y - \hat{Y}\right)^2 \tag{8}$$

- Conceptually the prediction error is equal to the MSE
  - MSE compares a RV ($\hat{\beta}$) with a parameter (β)
  - $Err(\hat{Y})$ involves two RV

Prediction Error

Then

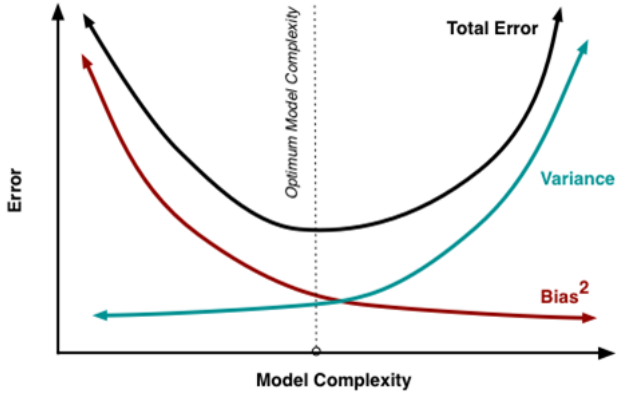
$$
\begin{aligned}
Err(\hat{Y}) &= E(Y - \hat{f})^2 && (9)\\
&= MSE(\hat{f}) + \sigma^2 && (10)\\
&= Bias^2(\hat{f}) + V(\hat{f}) + \sigma^2 && (11)
\end{aligned}
$$

Two parts:

- the error from estimating f with $\hat{f}$. (*reducible*)
- the error from not being able to observe u. (*irreducible*)

This is an important result, predicting Y properly we need a good estimate of f.

Prediction Error



Source: https://tinyurl.com/y4lvjxpc

A very interesting discussion in a recent Twitter thread by Daniela Witten: https://twitter.com/daniela_witten/status/1292293102103748609?s=20

Prediction Error

- Model $y = X\beta + u$
- $\hat{y} = \hat{X}\beta$ is the prediction
- The estimated prediction error is

$$\hat{Err}(\hat{Y}) = \sum (y_i - \hat{y}_i)^2 \tag{12}$$

- Common alternatives involve: the mean or the square root
- In Econometrics

$$\hat{Err}(\hat{Y}) = \sum_{i=1}^{n} e_i^2 \tag{13}$$

- $R^2 = 1 - \frac{Err(\hat{Y})}{TSS}$
- OLS minimizes $\hat{Err}(\hat{Y})$ and maximizes $R^2 \rightarrow$ minimizing the predictive error is to maximize fit in the sample

Prediction Error

Challenge:

- The goal of machine learning is *out of sample* prediction
- Minimize the prediction error outside of the sample
- OLS designed to minimize inside the sample
- Predicting well in sample doesn't mean that it would work outside
- There are estimators that work very well in sample but very badly outside (Overfit)

more on this later

Train and Test Samples

- Problem with OLS (and other estimators)
  - Minimize the prediction error outside of the sample
  - OLS designed to minimize inside the sample
- A workaround: split the data
  - Training sample: to build/estimate/train the model
  - Test sample: to evaluate its performance

Train and test samples

- From a strictly classical perspective splitting is a good idea
  - Makes sense if training data is iid from the population, even works if it is iid conditional on X
  - Two problems with this idea:
    - The first one is that given an original data set, if part of it is left aside to test the model, less data is left for estimation (leading to less efficiency).
    - A second problem is how to decide which data will be used to train the model and which one to test it. (more on how cross validation helps later)

Train and test samples

The *estimated prediction error* is defined as

$$\hat{Err}(\hat{Y}) = \sum_{i \in Test\ Sample} \left(Y_i - \hat{Y}_i\right)^2 \tag{14}$$

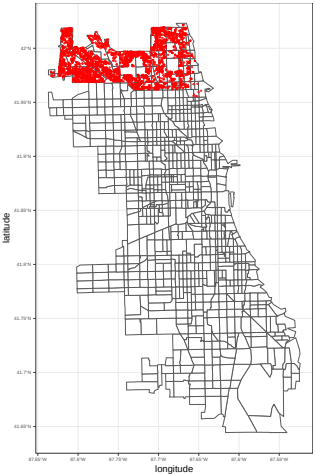
- $i \in Test\ Sample$ refers to all the observations in the test sample.
- $Test\ Sample \cup Training\ Sample = Full\ Sample$

- Note that:
  - No obvious way on how to partition this
  - In some cases is exogenously given. `Kaggle Competition, Netflix Challenge`
  - This idea is almost inexistent (or trivial) in classical econometrics

Example: Predicting House Prices in R

- `matchdata` in the *McSpatial package* for R.
- 3,204 sales of SFH Far North Side of Chicago in 1995 and 2005.
- This data set includes 18 variables/features about the home,
  - price sold
  - number of bathrooms, bedrooms,
  - latitude and longitude,
  - etc.
- in R:

```
require("McSpatial")#loads the package
data(matchdata) #loads the data
?matchdata # help/info about the data
```



Example: Predicting House Prices in R

- Train and Test samples
- 30% / 70% split

```
set.seed(101010) #sets a seed
matchdata <- matchdata %>%
                      mutate(price=exp(lnprice),
                             #transforms log prices to standard prices
                             holdout= as.logical(1:nrow(matchdata)
                             %in% sample(
                             nrow(matchdata), nrow(matchdata)*.7))
                             #generates a logical indicator divides train and test set
                             )

test<-matchdata[matchdata$holdout==T,]
train<-matchdata[matchdata$holdout==F,]
```

Example: Predicting House Prices in R

- Naive approach: model with no covariates, just a constant
- $y = \beta_0 + u$

```
model1<-lm(price~1,data=train)
summary(model1)
```

```
##
## Call:
## lm(formula = price ~ 1, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -258018 -127093  -24018   92732  598482
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   284018       4782   59.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 148300 on 961 degrees of freedom
```

Example: Predicting House Prices in R

In this case our prediction for the log price is the average train sample average

$$\hat{y} = \hat{\beta}_0 = \frac{\sum y_i}{n} = m$$

```
coef(model1)
```

```
## (Intercept)
##    284017.6
```

```
mean(train$price)
```

```
## [1] 284017.6
```

Example: Predicting House Prices in R

- But we are concerned on predicting well our of sample,:

```
test$model1<-predict(model1,newdata = test)
with(test,mean((price-model1)^2))
```

```
## [1] 21935777917
```

- $\hat{A}Err(\hat{y}) = \frac{\sum((y-\hat{y})^2)}{n} = 2.1935778 \times 10^{10}$
- This is our starting point, Can we improve it?

Example: Predicting House Prices in R

- How to improve it?
  - One way is using econ theory as guide
  - hedonic house price function derived directly from the Rosen's theory of hedonic pricing
  - however, the theory says little on what are the relevant attributes of the house.
  - So we are going to explore the effects of adding house characteristics on our out $AErr(\hat{y})$
- The simple inclusion of a single covariate can improve with respect to the *naive* constant only model.

```
model2<-lm(price~bedrooms,data=train)
test$model2<-predict(model2,newdata = test)
with(test,mean((price-model2)^2))
```

```
## [1] 21695551442
```

Example: Predicting House Prices in R

- What about if we include more variables?

```
model3<-lm(price~bedrooms+bathrooms+centair+fireplace+brick,data=train)
test$model3<-predict(model3,newdata = test)
with(test,mean((price-model3)^2))
```

```
## [1] 21111169595
```

- Note that the *AErr* is once more reduced. If we include all?

```
model4<-lm(price~bedrooms+bathrooms+centair+fireplace+brick+
                 lnland+lnbldg+rooms+garage1+garage2+dcbd+rr+
                 yrbuilt+factor(carea)+latitude+longitude,data=train)
test$model4<-predict(model4,newdata = test)
with(test,mean((price-model4)^2))
```

```
## [1] 20191829518
```

- Is there a limit to this improvement? Can we keep adding features and complexity?

Example: Predicting House Prices in R

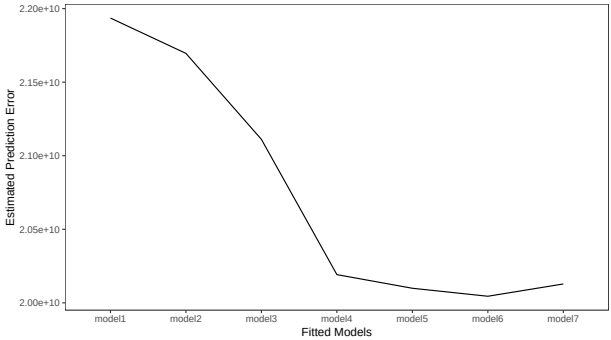
- Is there a limit to this improvement? Can we keep adding features and complexity?
- Let's try a bunch of models

```
model5<-lm(price~poly(bedrooms,2)+poly(bathrooms,2)+
                centair+fireplace+brick+
                lnland+lnbldg+rooms+
                garage1+garage2+dcbd+rr+
                yrbuilt+factor(carea)+poly(latitude,2)+
                poly(longitude,2),data=train)
test$model5<-predict(model5,newdata = test)

model6<-lm(price~poly(bedrooms,2)+poly(bathrooms,2)+centair+fireplace+brick+
                lnland+lnbldg+garage1+garage2+rr+
                yrbuilt+factor(carea)+poly(latitude,2)+poly(longitude,2),
                data=train)
test$model6<-predict(model6,newdata = test)

model7<-lm(price~poly(bedrooms,2)+poly(bathrooms,2)+centair+fireplace+brick+
                lnland+lnbldg+garage1+garage2+rr+
                yrbuilt+factor(carea)+poly(latitude,3)+poly(longitude,3),
                data=train)
test$model7<-predict(model7,newdata = test)
```

Example: Predicting House Prices in R



- Take aways from the example
- Classical econometrics set up, choosing between smaller and larger models
- More complexity, the prediction error keeps getting smaller.
- The choice of a model's complexity faces a bias/variance trade-off.
- Open question: how to find the optimal complexity level?

  (later on the course)

Review & Next Steps

- Urban Models
- Prediction vs Estimation
- Train and Test Samples
- Example in `R`

- **Next Class: More on ML, Complexity and the variance/bias trade off**

Further Readings

- Davidson, R., & MacKinnon, J. G. (2004). Econometric theory and methods (Vol. 5). New York: Oxford University Press.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.
- Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.
- Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.
- Rosen, S. (1974). Hedonic prices and implicit markets: product differentiation in pure competition. Journal of political economy, 82(1), 34-55.

Volvemos en 5 min con Python