Machine Learning for policy evaluation: an introduction

Jérémy Do Nascimento Miguel*

*BSE, Univ. Bordeaux, jdnmiguel@u-bordeaux.fr

Master APP - EADD; Univ. Bordeaux - Fall 2023

About this course

- Basic overview of ML methods and utilization for policy evaluation
- Course Material:
  - Rely mostly on Hastie et al. (2009)
  - See the course web-page on my github
- See syllabus for more details
- Do not hesitate to reach out

About this course: objectives

- Understand what machine learning does and does not do
    - Lasso is not only useful for cowboys but cannot interpret its coefficients
- Familiarization with recent Machine Learning applications in policy evaluation exercises:
    - In python

Course

1. Introduction to Machine Learning
2. More about Supervised Learning
3. Estimating average treatment effects with many control variables
4. Unpack the treatment effect heterogeneity
5. Other applications: remote sensing or text analysis

Overview

Introduction to Machine Learning

The basics

Overfitting

Re-sampling methods

Big data?

- Data can be tall (x observations) or **wide** (x covariates)
- With the "Big data revolution" we have more and more data collected (e.g., cell phone data, scanner data) but this is quite different in traditional surveys
- Most of the data we are collecting in development are quite wide $\Rightarrow$ which covariates are relevant for our analysis??

$\Rightarrow$ Machine learning can help to extract and analyze the relevant information

What I mean by Machine Learning?

1. **Supervised learning**
   - If y is continuous: use x to predict y (e.g., y is wheat price, x are farmers, wheat, market characteristics)
   - If y is categorical: use x to classify y (e.g., y is cooperative membership, x are farmers characteristics)
2. Unsupervised learning
   - Only have x
   - Can we predict *unobserved* y clusters?

$\Rightarrow$ Focus on supervised learning: interested in the conditional distribution of y given some covariates x

Overview

Introduction to Machine Learning

The basics

Overfitting

Re-sampling methods

Basic terminology

The Machine Learning literature uses a different terminology for the objects of interest

- **Training**: the model is not estimated but trained
- **Training Sample**: the sample used to estimate the function or parameters
- **Testing Sample**: the sample used to predict the outcome
- **Features**: X, regressors, covariates, or predictors
- **Response**:Y, outcome
- **Supervised Learning**: when we observe, and use, both the X and Y
- **Unsupervised Learning**: we only observe the X, and try to group them into clusters
- **Classification Problems**: unorderered discrete response problems

Start from the statistical model

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

with Y=outcome; $X = (X_1, ..., X_p)$ = predictors; ϵ = mean-zero error independent of X

- This is a *statistical model* meaning that nothing is causal here.
- f represents the *systematic* relationship between x and y
- Objective: estimate f using information X provides about Y, **but** do not care about f itself ($\approx$ *blackbox*)

Why should we estimate $f(X)$

1. Prediction: $\hat{Y} = \hat{f}(X)$; objective is to min. the following function

$$MSE = \frac{1}{n} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2$$

2. Inference:
    - Which variables are associated with the outcome
    - Relationship between the outcome and each predictor

How should we estimate $f(X)$? parametric vs non-parametric

Parametric methods: involve two steps

1. Assumption about the parametric shape for $f(X)$ (e.g., linear, log, etc.):

$$f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

   Note: parametric means that the function depends on a finite # parameters, $p + 1$

2. Estimate resulting from (1)

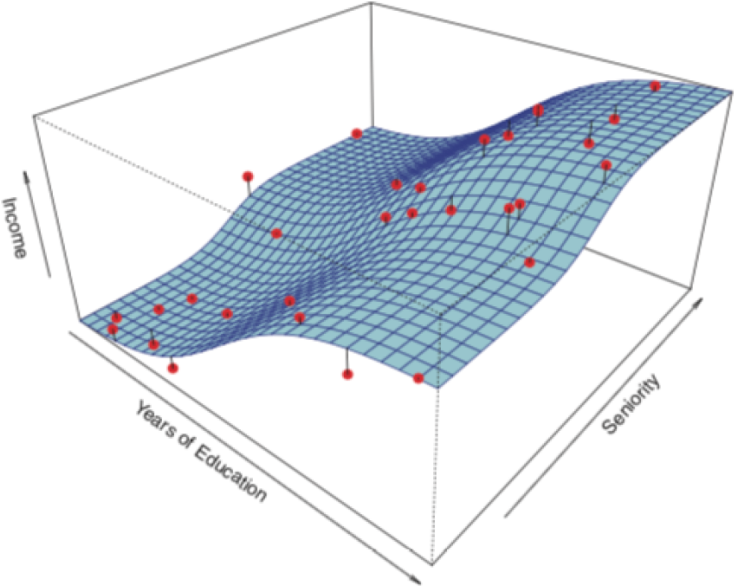
$$\hat{Y} = \hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p$$

   with $\hat{\beta}_0, \ldots, \hat{\beta}_p$ the OLS estimates

Pros: easy to estimate ; cons: risk of mispecification

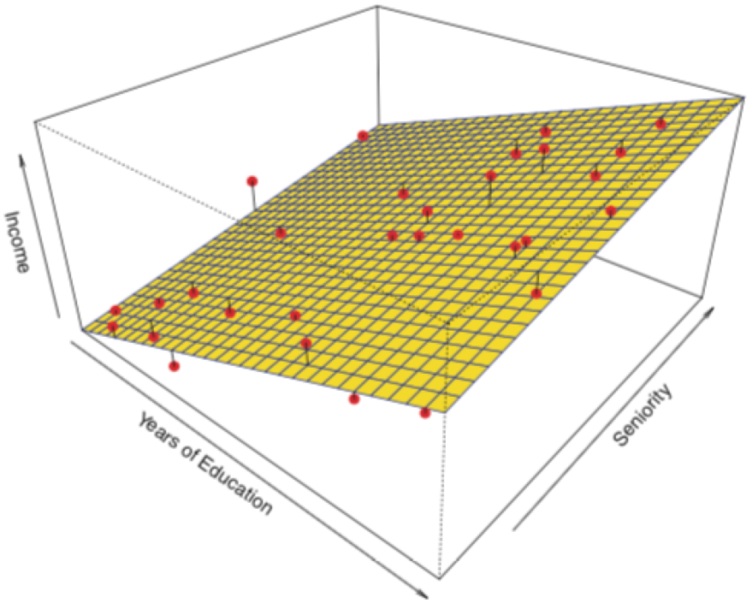
$\uparrow$ flexibility? $\uparrow$ # parameters to estimate wich $\uparrow$ risk of overfitting

True function

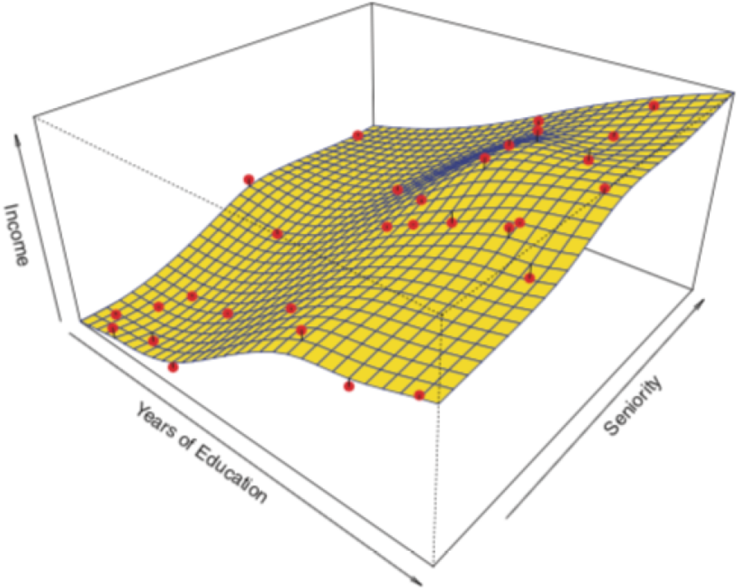


From Hastie et al. (2009)

Linear Estimate



Smooth non linear estimate



Assessing model accuracy

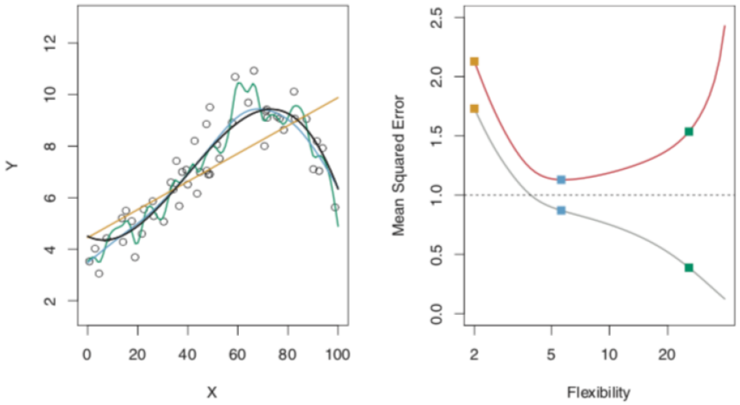
Measure how the predicted variable for an observation is close to the observed one:

$$MSE = \frac{1}{n}\sum_{i=1}^{N}(y_i - \hat{f}(x_i))^2$$

But this is the **in sample Mean Squared Errors** and used to fit the model. line We are interested in the **MSE on test data**: **out-of-sample-fit**

$$Ave(Y_0 - \hat{f}(X_0))^2$$

Training and Test MSE: an example



From Hastie et al. (2009)

Left: Data simulated from f (black). Three estimates of f: the linear regression line (orange), and two smoothing spline fits (blue and green). Right: Training MSE (grey), test MSE (red). Squares represent the training and test MSEs for the three fits shown in the left-hand panel

Overview

Introduction to Machine Learning

The basics

Overfitting

Re-sampling methods

Overfitting risk

Why don't we estimate the most flexible model we can?

- Maximizing flexibility leads to a low out-of-sample fit

- Maximal flexibility leaves all the noise in the prediction model: new observation with the same X has a different idiosyncratic noise, so the prediction is off

- Overfitting arises when a less flexible model would have yield a smaller MSE

Bias-variance trade-off

Overfitting is an example of the bias-variance trade-off. Let's consider the following expected test MSE for a given value x_0:

$$E[y_0 - \hat{f}(x_0)]^2 = \underbrace{Var(\hat{f}(x_0))}_{\text{Variance}} + \underbrace{[Bias(\hat{f}(x_0))]^2}_{\text{Bias}} + Var(\epsilon)$$

$\Rightarrow$ Need a method such as we have low variance and bias

- Variance: the extent by which $\hat{f}(x)$ would change if we estimated it using a different data set

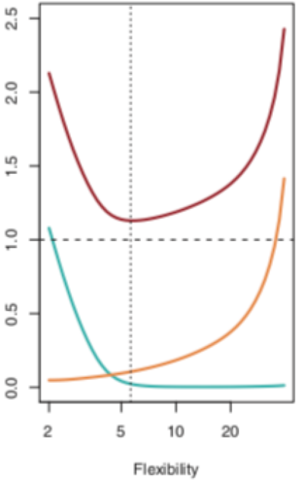
- Bias: error introduced by approximating a complex general function (i.e., real life problem) by a restricted functional forms. $\hat{f}(X)$ *always* under or over-predicts for some regions of X

- $\uparrow$ Flexibility reduces the bias but increases the variance

Practical problems: (i) how do we measure complexity? (ii) Do not observe $E[y_0 - \hat{f}(x_0)]^2$ and need large test set

Bias-variance trade off

$$\text{Test MSE} = \text{bias}^2 + \text{variance} + Var(\epsilon)$$



- Variance and bias $\downarrow$ as the flexibility increases
- Bias fall rapidly as flexibility increases $\Rightarrow$ sharp decrease in test MSE

$\Rightarrow$ This illustrates the bias-variance trade-off: easy to obtain a method with low variance but high bias

Overview

Introduction to Machine Learning

The basics

Overfitting

Re-sampling methods

Adding a complexity measure

The objective will be to minimize

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \text{s.t.} R(f) \leq c$$

where R(f) is a complexity measure (solve our first problem). This is basically how nonparametric estimation is working. Practical problem, how to chose c?

- Machine learning use cross validation to estimate test MSE and then pick c

Spliting samples

We are splitting our data set into:

1. Training sample: data used for fitting the model

2. Testing sample (*hold-out or validation set*): the out-of-sample "sample" from our actual data. Use the model fitted with 1. to predict $\hat{y}$ in this sample

- Resampling means that we are repeatedly drawing sample from 1. to refit the model on each sample to get new information

- Most used are cross-validation and bootstrap. They allow us to evaluate the model's performance (model assessment) and to select the level of flexibility (model selection)

$\Rightarrow$ Solve our practical problems

Validation-set approach

Easiest way to obtain a testing and training sample: randomly split the dataset into two.

Testing error rate $\approx$ Training error rate

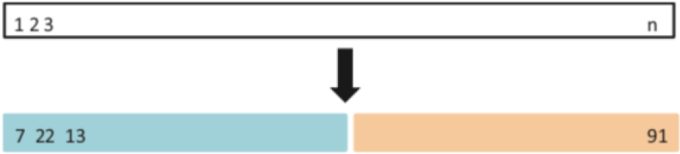
However, most of the time we have access to small data, then a single random split may yield to idiosyncratic differences between the samples/

Drawbacks:

- Testing error rate is **highly variable** depending on which observations
- Use only a subset of the observation to fit the model, while many statistical models perform better wither larger sample size $\Rightarrow$ overestimate the test MSE

No cross validation

Take $(x_i, y_i)_{i=1}^n$ and split it into a training and a validation set.



From Hastie et al. (2009)

Pick $c \Rightarrow$ Obtain $\hat{f}^c(x)$ on *training* set $\Rightarrow$ Compute MSE on validation set $\Rightarrow$ Loop over c and choose c that minimizes MSE on validation set

Cross-validation: Leave-one-out cross-validation (LOOCV)

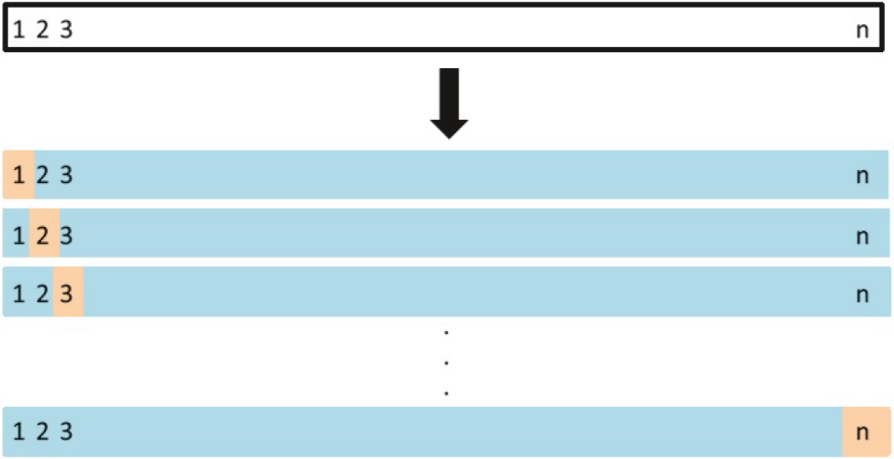
Set of n data points is repeatedly split into two parts: (i) training set with $(n-1)$ obs.; (ii) testing with 1 obs. leaved out from (i).

- How does it work?
  - Using a model fitted without observation i, $\hat{y}_i$ is the leave-out-one prediction of i
  - Compute $MSE_i = (y_i - \hat{y}_i(x_i))^2$
  - Repeat it for all n observations, leaving one observation out each time
  - Average each test error to obtain the sample test MSE wich is:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

This is computational costly because it requires n regressions

LOOCV



From Hastie et al. (2009)

k-fold Cross-Validation

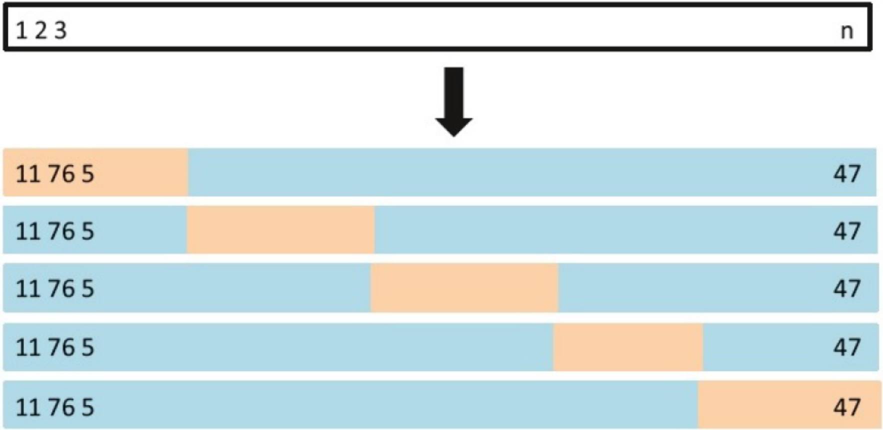
Instead of leaving-out one observation, we randomly **divide our data into k-groups** of $\approx$ similar size

- One fold is the testing sample, the remaining $k-1$ folds are the training sample
- Fit a model without fold I: $\hat{y}_I$
- Compute $MSE_I = \sum_{j \in I} (y_j - \hat{y}_I(x_j))^2$
- Repeated k times, holding each fold I out successively
- Average each test error to obtain the sample test MSE wich is:

$$CV_{(k)} = \frac{1}{n} \sum_{I=1}^{k} MSE_I$$

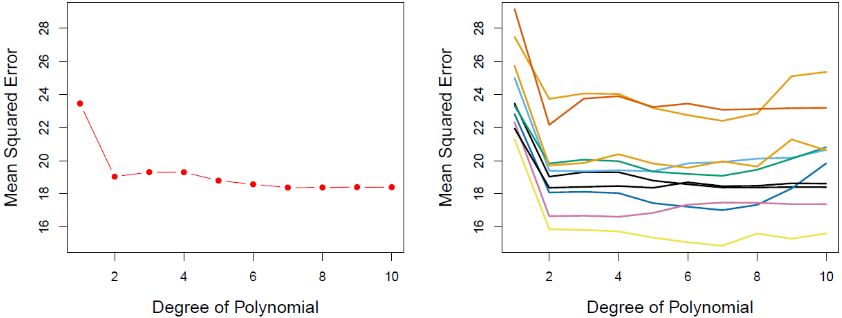
This requires only k regressions. Usually $k = 5$ or $k = 10$

5-fold cross-validation



From Hastie et al. (2009)

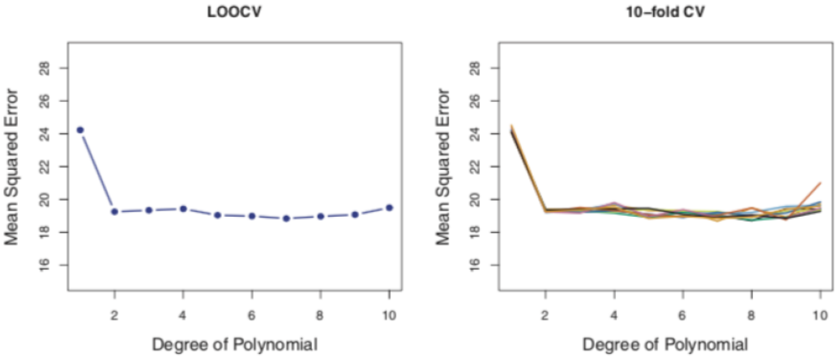
Test error estimates using LOOCV



From Hastie et al. (2009)

On the left: validation error estimates for a single split into; On the right: repeated 10 times each time with a different random split $\Rightarrow$ large variability in test MSE

Test error estimates: LOOCV vs k-fold CV



From Hastie et al. (2009)

On the left: validation error estimates for a single split into; On the right: 10-fold CV run 9 times, each with a different random split of the data into 10 parts $\Rightarrow$ small variability in test MSE

References

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.