Supervised Machine Learning

Elliott Ash, Malka Guillot, Philine Widmer

ETH Zürich | SICSS Zürich 2021

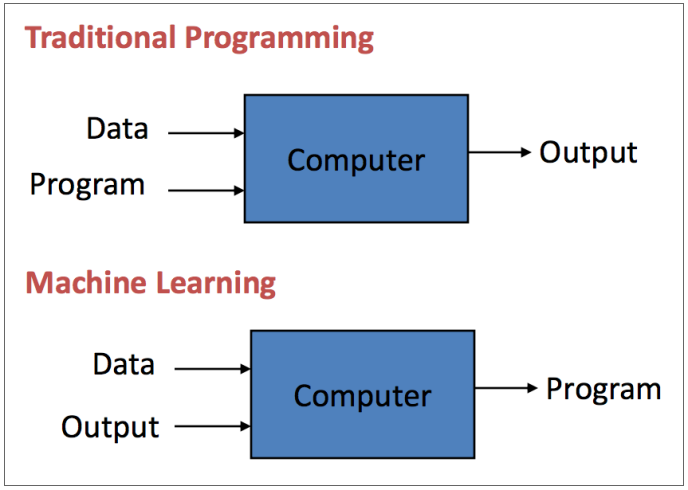
Table of Contents

≡

Prologue

≡

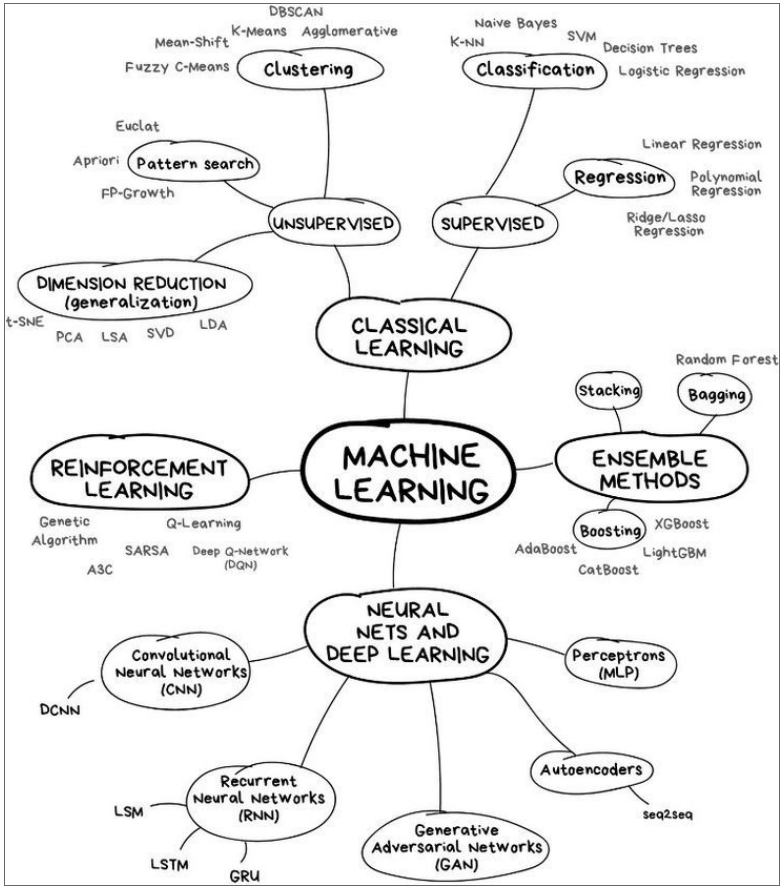
Econometrics vs. Machine Learning



- Classical computer programming: humans input the rules and the data, and the computer provides answers.
- Supervised ML: humans input the data and the answer, and the computer learns the rules.

≡

The Machine learning landscape



Today

- Focus on regressions

Not covered

- Classification
- Advanced ML methods: XGboost notebook
- Unsupervised learning slides, notebook

≡

Reference:

- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning.*, book, chap 3, 6.2
- Géron, A., *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. book, chapter 2, and notebooks

≡

Modeling theory and accuracy measurement

≡

Mean Squared Error (MSE)

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

- Regression setting: the mean squared error is a metric of how well a model fits the data.
- But it's in-sample.
- What we are really interested in is the out-of-sample fit!

≡

Measuring fit (1)

- We would like $(y_0 - \hat{f}(x_0))^2$ to be small for some (y_0, x_0), not in our training sample $(x_i, y_i)_{i=1}^n$.
- Assume we had a large set of observations (y_0, x_0) (a test sample),
- then we would like a low $Ave(y_0 - \hat{f}(x_0))^2$
- i.e a low average squared prediction error (test MSE)

≡

Measuring fit (2)

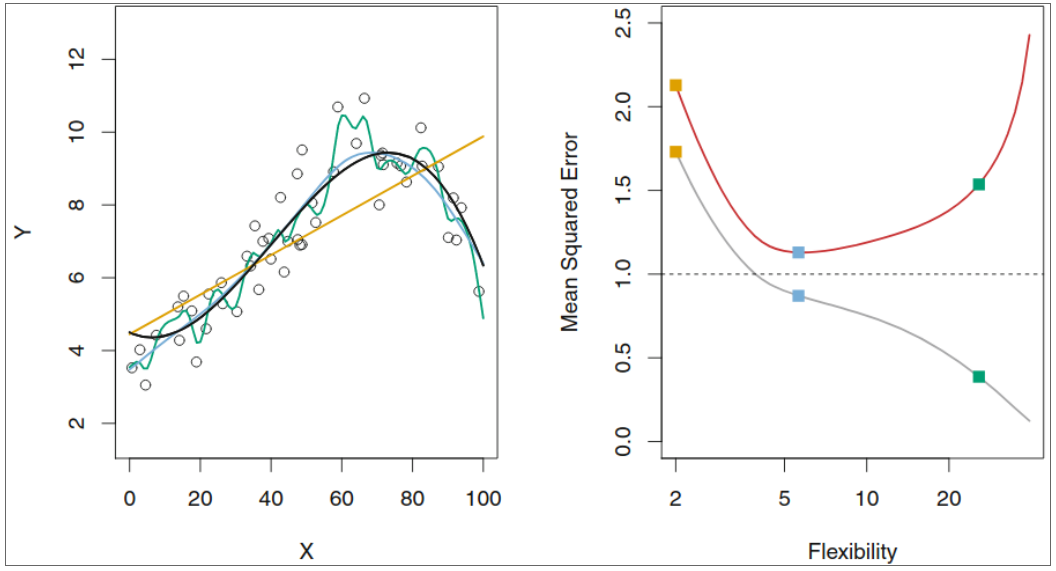
To estimate model fit we need to partition the data:

1. Training set: data used to fit the model
   - Training MSE: how well our model fits the training data.
2. Test set: data used to test the fit
   - Test MSE: how well our model fits new data

We are most concerned in minimizing test MSE

≡

Training MSE, test MSE and model flexibility



Red (grey) curve is test (train) MSE

Increasing model flexibility tends to decrease training MSE but will eventually increase test MSE

≡

Overfitting

- As model flexibility increases, training MSE will decrease, but the test MSE may not.
- When a given method yields a small training MSE but a large test MSE, we are said to be overfitting the data.
- (We almost always expect the training MSE to be smaller than the test MSE)
- Estimating test MSE is important, but requires training data...

≡

How to choose training and test set?

≡

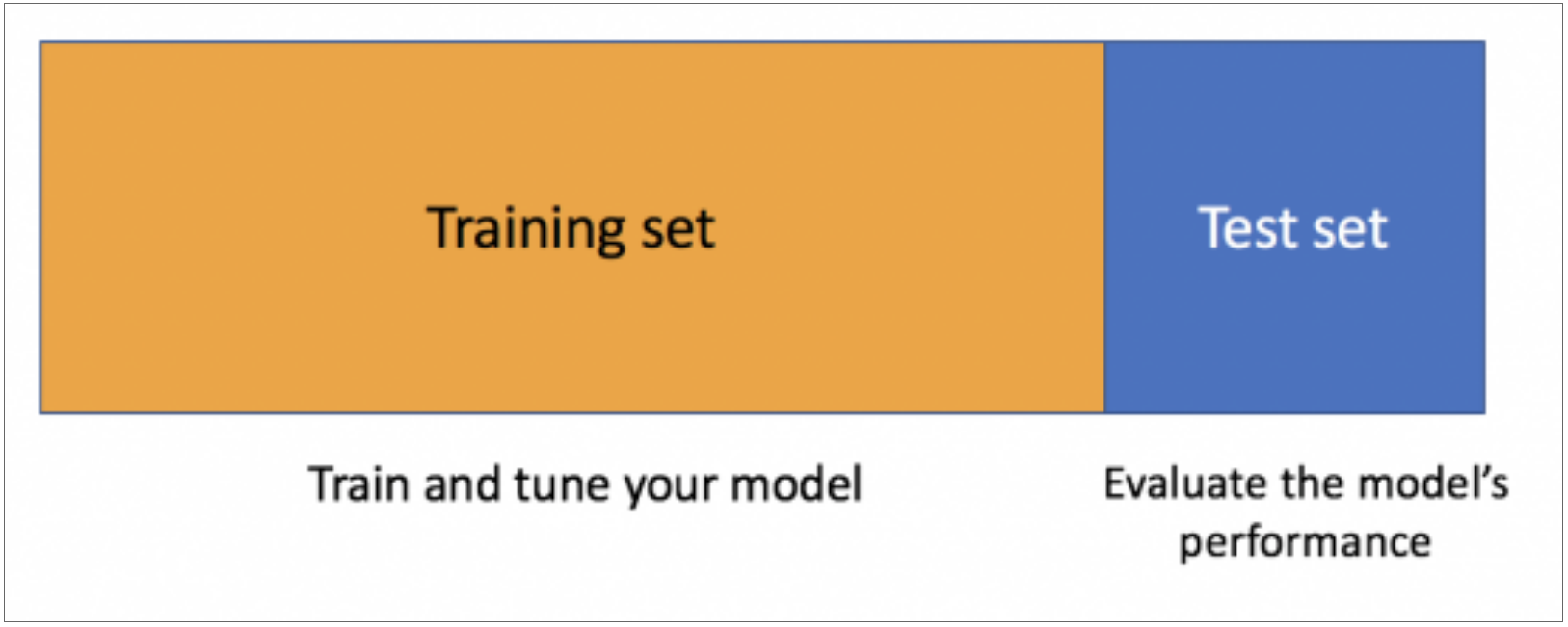
Resamling methods

Estimate the test error rate by
holding out a subset of the training observations from the fitting process,
+ then applying the statistical learning method to those held out observations

≡

Validation set approach

- Randomly divide labeled data randomly into two parts: training and test (validation) sets.



≡

Two concerns

- Arbitrariness of split
- Only use parts of the data for estimation
  $\longrightarrow$ we tend to overestimate test MSE because our estimate of $f(x)$ is less precise

≡

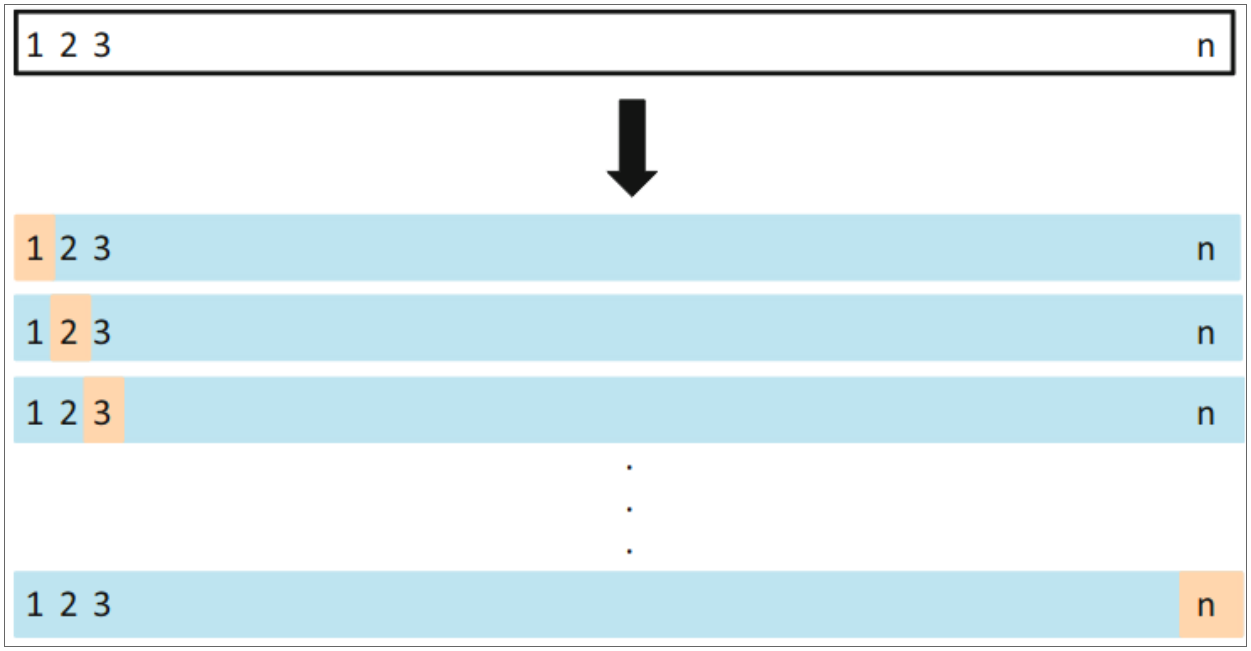
Leave-One-Out Cross-Validation (LOOC)

- Fit on $n-1$ training observations, and a prediction the Last
- Iterate n times
- Assess the average model fit across each test set.
  Estimate for the test MSE:

$$CV_n = \sum_{i=1}^{n} MSE_i$$

≡

Leave-One-Out Cross-Validation (LOOC)



- less bias than the validation set approach
- always yield the same results
- potentially too expensive to implement

≡

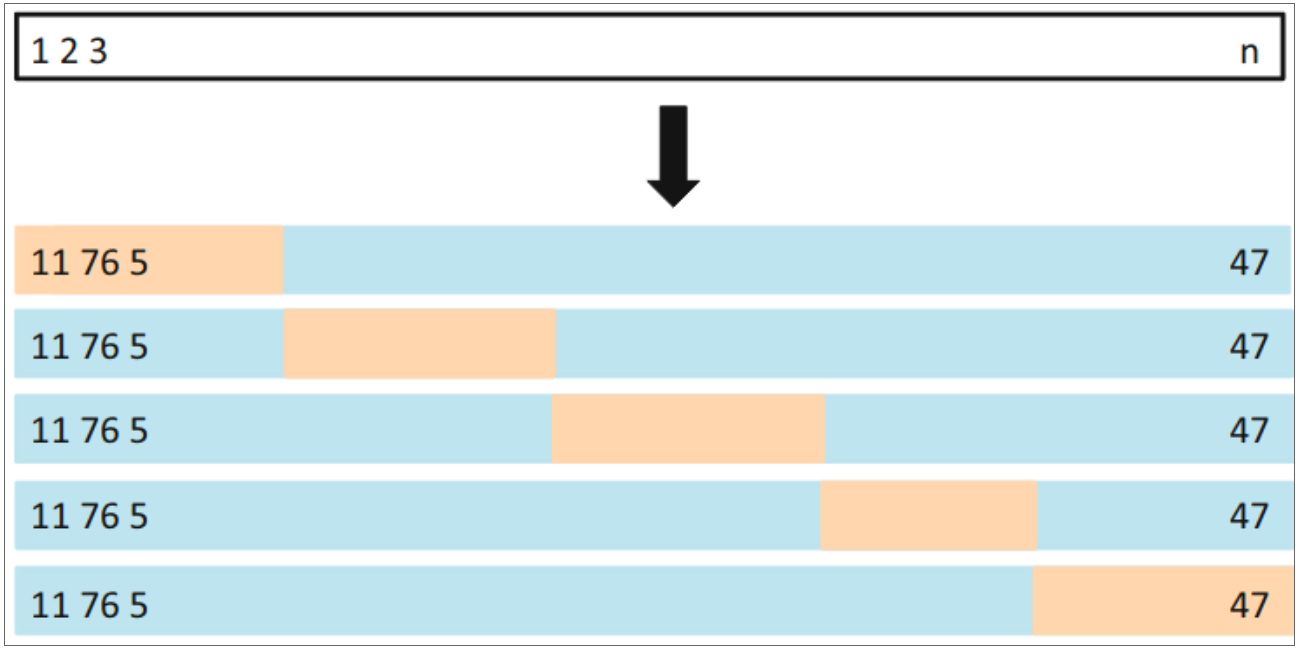
k-fold Cross-validation

- Leave-One-Out Cross-Validation with $k = 1$
- Randomly dividing the data into the set of observations into k groups
- 1st fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds
- Iterate k times
  Estimate for the test MSE:

$$CV_k = \sum_{i=1}^{k} MSE_i$$

≡

k-fold Cross-validation



$\Rightarrow$ Arguably the contribution to econom(etr)ics: Cross-validation (to estimate test MSE)!

Linear Regression as a Predictive Model

≡

Linear Regression

$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$

$=$ one of the simplest algorithms for doing supervised learning

A good starting point before studying more complex learning methods

≡

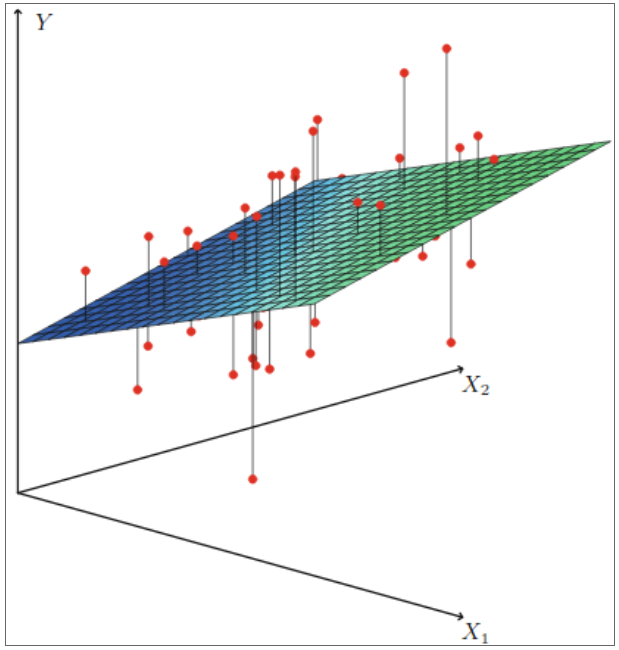
Estimation by Ordinary Least Squares

$RSS = \text{Residual sum of squares} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

Minimizing RSS gives a closed form solution for the $\hat{\beta}_1, \cdots \hat{\beta}_p$

Most ML models do not have a a closed form solution

≡


Y
X_2
X_1

Extensions of the Linear Model

Going further model's assumptions:

- the additive: the effect of changes in a predictor X_j on the response Y is independent of the values of the other predictors
- linearity: the change in the response Y due to a one-unit change in X_j is constant

≡

Interactions

- Adding interacted variable can help
- Should respect the hierarchy principle:
  - if an interaction is included, the model should always include the main effects as well

≡

Non Linearity

- Include transformed versions of the predictors in the model
  $\Rightarrow$ Including polynomials in X may provide a better fit

≡

Linear Models: pros and cons

- Pros:
  - Interpretability
  - Good predictive performance
  - Accuracy measures for
    - coefficient estimates (standard errors and confidence intervals)
    - the model
- Cons:
  - When $p > n$
  - Tend to over-fit training data.
  - Cannot handle multicollinearity.

≡

Generalization of the Linear Models

- Classification problems: logistic regression, support vector machines
- Non-linearity: nearest neighbor methods
- Interactions: Tree-based methods, random forests and boosting
- Regularized fitting: ridge regression and lasso

≡

Regularized Regressions

≡

Why Regularization?

- Solution against over-fitting
- Allow High-Dimensional Predictors
  - $p >> n$: OLS no longer has a unique solution
  - x_i "high-dimensional" i.e. very many regressors
    - pixels on a picture

≡

Adding a Regularization Term to the Loss Function $L(.)$

$$\hat{\beta} = argmin_{\beta} \frac{1}{n} \sum_{i=1}^{n} L(h(x_i, \beta), y_i) + \lambda R(\beta)$$

- $R(\beta)$ = regularization function
  - $R(\beta) = \sum_{i=1}^{n} p(\beta_i)$ for $p(.)$ the penalty function
- λ is a hyperparameter where higher values increase regularization

≡

Different Penalty Functions $p()$

- Ridge (L2): $p(\beta_j) = \beta_j^2$
- LASSO (L1): $p(\beta_j) = |\beta_j|$
- Elastic Net: $p(\beta_j) = \alpha|\beta_j| + (1-\alpha)\beta_j^2$
- Subset selection: $p(\beta_j) = 1\{\beta_j \neq 0\}$

≡

How to Solve Without a Closed-form Solution?

Gradient Descent

Gradient descent measures the local gradient of the error function, and then steps in that direction.
$\longrightarrow$ Minimum in 0

Stochastic Gradient Descent

1. Picks a random instance in the training set
2. Computes the gradient only for that single instance

- Pro: SGD is much faster to train,
- Cons: bounces around even after it is close to the minimum.
  $\longrightarrow$ Compromise: mini-batch gradient descent, selects a sample of rows (a “mini-batch”) for gradient compute

≡

Varients of Gradient Descent

Ridge Regression

$min_\beta \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

Where

- $\lambda > 0$ = penalty parameter
- covariates can be high-dimensionnal $p >> N$

Trade-off, from the minimization of the sum of

1. RSS
2. shrinkage penalty: decreases with β_j

$\rightarrow$ relative importance given by λ

≡

Ridge Regression: shrinkage to 0

Ridge: Variance-Bias Trade-Off

Squared bias (black), variance (green), [test] MSE (red)

Ridge vs. Linear Models

- when outcome and predictors are close to having a linear relationship, the OLS will have low bias but potentially high variance
  - small change in the training data $\rightarrow$ large change in the estimates
  - worse with p close tp n
  - if $p > n$, OLS do not have a unique solution

  $\rightarrow$ ridge regression works best in situations where the least squares estimates have high variance

≡

LASSO

Overcome an important drawback of Ridge (all p predictors are included in the final model)
LASSO proposes a method to build a model which just includes the most important predictors.
Better for interpretability than Ridge!

≡

Lasso Coefficients

Lasso: Variance-Bias Trade-Off

Squared bias (black), variance (green), [test] MSE (red)

≡

Constrained Regression

The minimization problem can be written as follow:

$$\sum_{i=1}^{n}(y_i - x_i'\beta)^2 \text{ s.t. } \sum_{j=1}^{p} p(\beta_j) \leq s,$$

Where

- Ridge: $\sum_{j=1}^{p} \beta_j^2 < s \rightarrow$ equation of a circle
- Lasso: $\sum_{j=1}^{p} |\beta_j| < s \rightarrow$ equation of a diamond

Constraint Regions

Lasso Ridge

Elastic Net = Lasso + Ridge

$$MSE(\beta) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2$$

$\lambda_1, \lambda_2 =$ strength of L1 (Lasso) penalty and L2 (Ridge) penalty

≡

Selecting Elastic Net Hyperparameters

- Elastic net hyperparameters should be selected to optimize out-of-sample fit (measured by mean squared error or MSE).
- "Grid search"
  - scans over the hyperparameter space ($\lambda_1 \geq 0, \lambda_2 \geq 0$),
  - computes out-of-sample MSE for all pairs (λ_1, λ_2) ,
  - selects the MSE-minimizing model.

≡

Evaluating Regression Models: R^2

MSE is good for comparing regression models, but the units depend on the outcome variable and therefore are not interpretable

Better to use R^2 in the test set, which has same ranking as MSE but it more interpretable.

≡

ML & Causal Inference

≡

Double Machine Learning to Adjust for Confounders

- If the treated group and comparison group differ only by a set of observable characteristics $\longrightarrow$ "control" for these variables to obtain causal estimates.
- But what if we have 1000 covariates? $\longrightarrow$ Machine learning can help.

≡

Double ML: Setup

$Y = \beta D + g(X) + \epsilon$

- low-dimensional treatment D, high-dimensional set of (observed) confounders X.
  - OLS regression without adjusting for confounders will be biased for $\hat{\beta}$
  - can we just include them in the regression as linear covariates?
    - will not adjust correctly due to potential non-linearities.
    - will probably fail to converge due to high dimensionality / collinearity / overfitting

≡

Double ML method

1. Learn Y given X, $\hat{Y}(X)$, using any ML method
2. Learn D given X, $\hat{D}(X)$, using any ML method
3. Form residuals $\tilde{Y} = Y - \hat{Y}(X)$ and $\tilde{D} = D - \hat{D}(X)$
4. Regress $\tilde{Y}$ on $\tilde{D}$ to learn $\hat{\beta}$.

≡

Double ML method - Cross-Fitting

Split into samples A and B, 50% of data each, to prevent overfitting:

- Fit (1) and (2) on sample A, then predict (3) and regress (4) on sample B, to estimate $\hat{\beta}_A$
- vice versa: fit (1)/(2) on sample B, and predict/regress (3)/(4) on sample A, to learn a second estimate for $\hat{\beta}_B$.
- average them to get a more efficient estimator: $$\hat{\beta}=\frac{1}{2}(\hat{\beta}*{A}+|hat{|beta}*{B})$$.

≡

Final Thoughts

≡

Selecting the Tuning Parameter By Cross-Validation

1. Choose a grid of λ values
2. Compute the CV error for each lambda
3. Select the tuning parameter value for which the CV error is smallest
4. Re-fit the model using all available observation and the best λ

≡

Data Prep for Machine Learning

- See Geron Chapter 2 for pandas and sklearn syntax:
  - imputing missing values.
  - feature scaling (coefficient size depends on the scaling)
  - encoding categorical variables.
- Best practice
  - reproducible data pipeline
  - standardize coefficients

≡

Other Supervised Machine Learning Methods

- Forward Selection,
- Backward Selection
- Trees and Forests
- Neural Networks
- Boosting
- Ensemble Methods

≡

“Essentially, all models are wrong, but some are useful” -- George Box

≡