Big Data for Public Policy

Regressions

Malka Guillot

ETH Zürich | 860-0033-00L

Turn on recording

Table of contents

≡

Prologue

≡

Coming back on the homework

≡

Last week

- Model accuracy

Today

- Regression analysis (continuous outcome)
- Our first ML project!

Reference:

- JWHT, chap 3, 6.2
- Geron, chapter 2

Next week

- Classification (binary outcome)

≡

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

≡

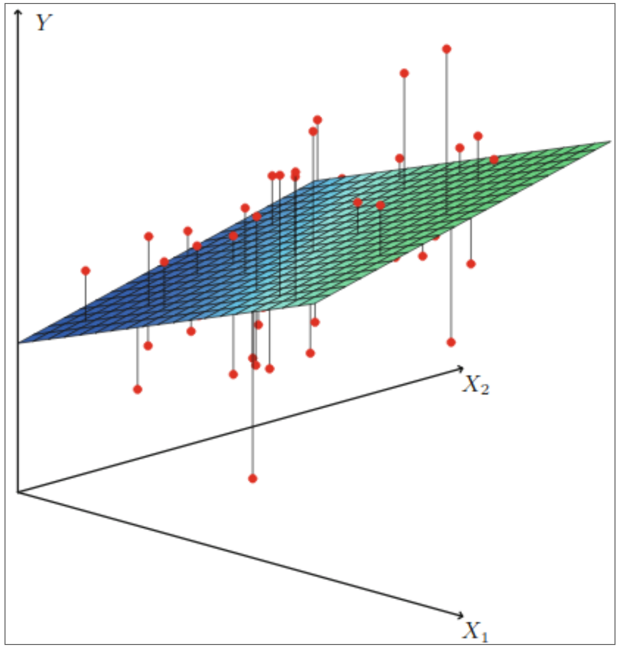
Estimation by Ordinary Least Squares

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

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

Most ML models do not have a a colsed form solution

≡


Y
X2
X1

≡

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

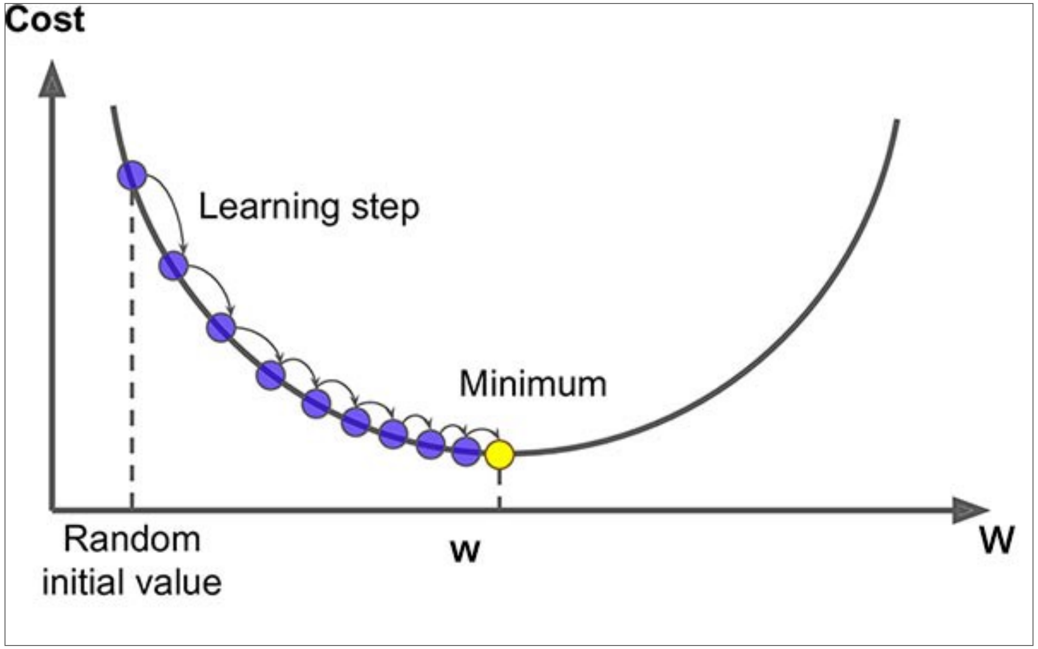
≡

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.
$\rightarrow$ Minimum in 0

≡

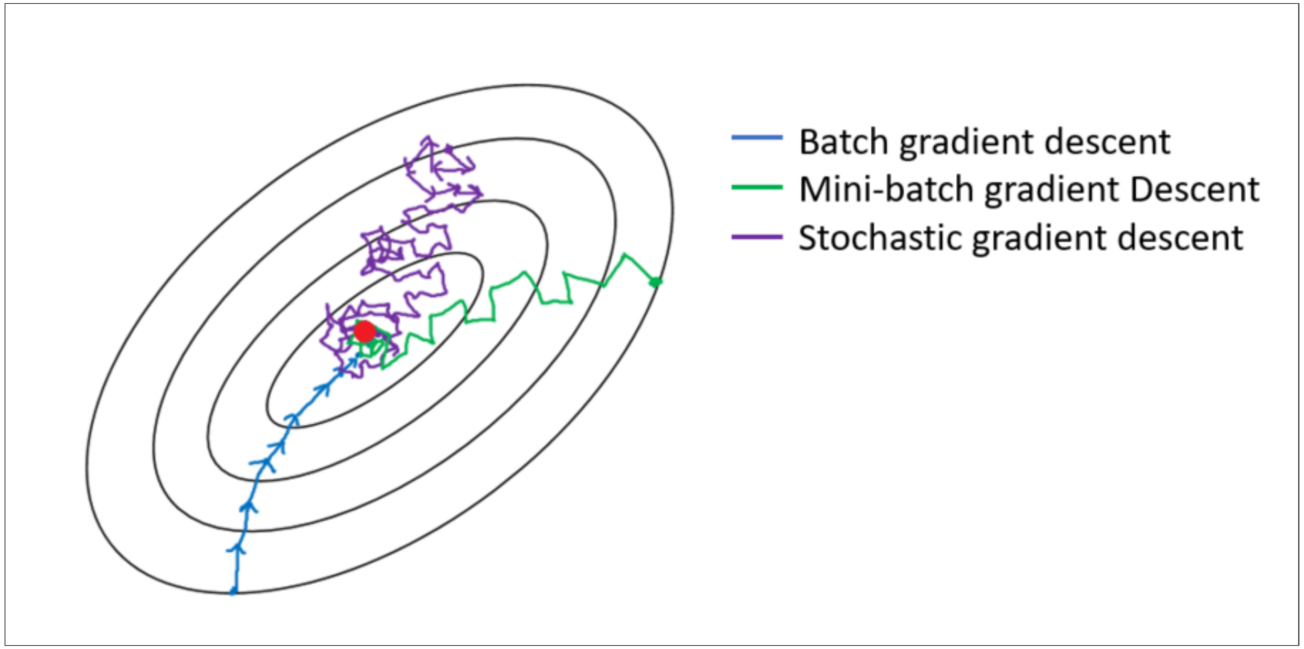
Stochastic Gradient Descent

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.
  $\rightarrow$ 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 $p >> N$

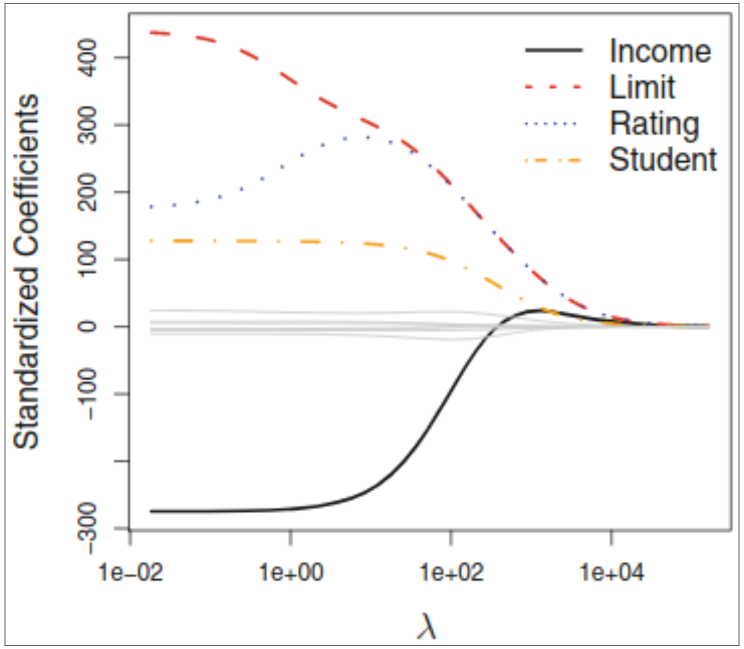
Trade-off, from the minimization of the sum of

1. RSS
2. shrinkage penalty: decreases with β_j

$\rightarrow$ relative importance given by λ

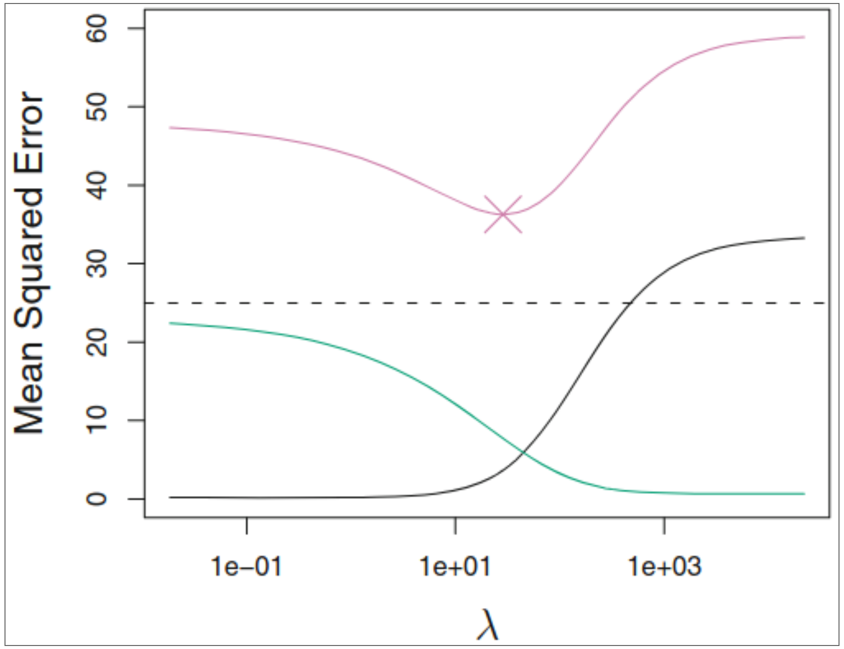
≡

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

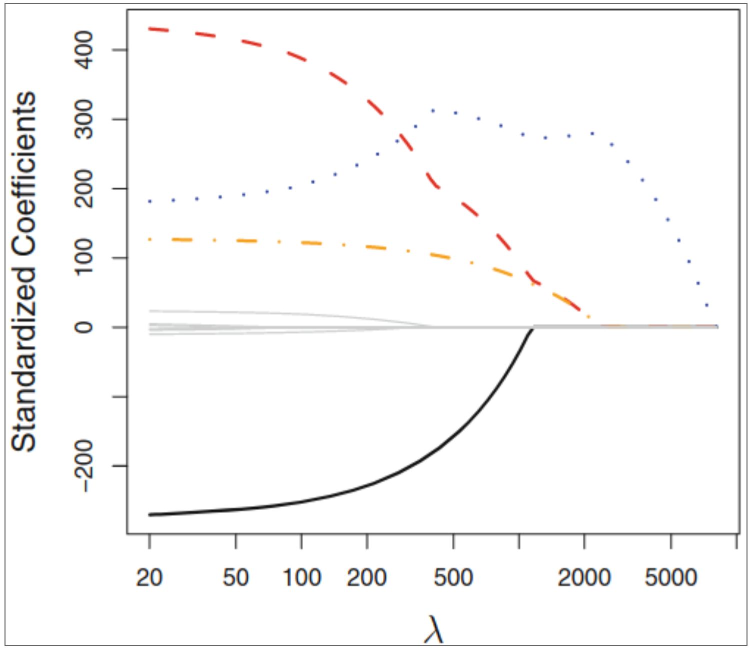
≡

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!

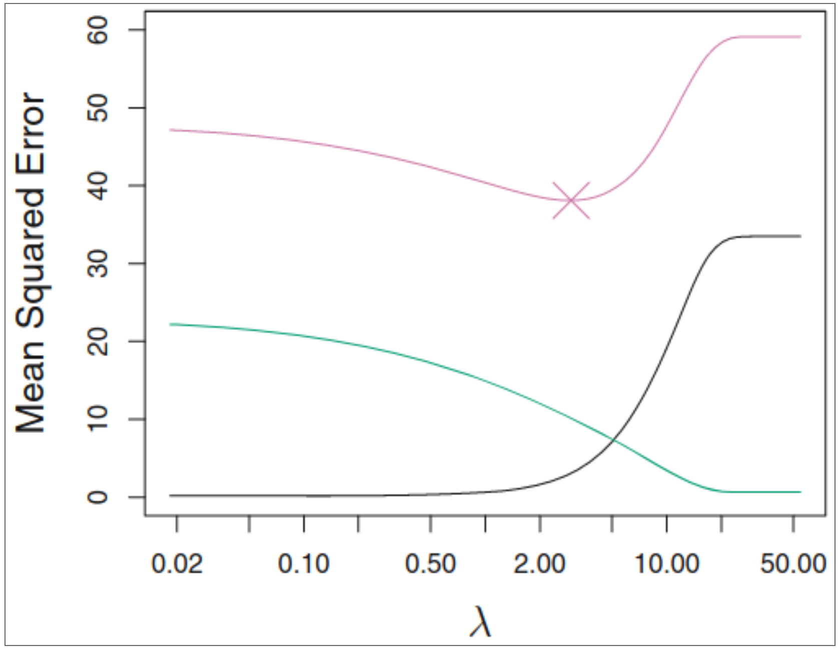
≡

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:

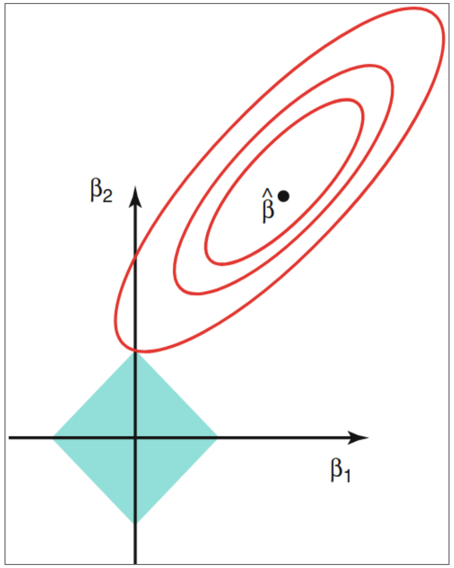
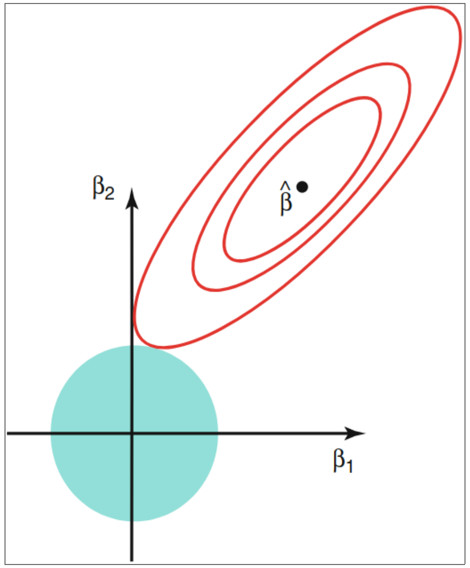
$$\sum_{i=1}^{n}(y_i - x_i'\beta)^2 \text{ s.t. } \sum_{j=1}^{p} p(\beta_j) \leq s,$$

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.

≡

Final Toughts

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*

≡

Turn off recording

≡