Machine Learning in Asset Pricing, Ch 1-2.3

Kevin Mei

January 27, 2022

Overview

Chapter 1 – Introduction

Chapter 2 - Supervised Learning

- Regression methods
- Hyperparameter tuning

Chapter 3 - Supervised Learning in Asset Pricing (next time)

Overview of terminology

TABLE 1.1
Terminology in ML and Statistics

ML	Statistics
Training, Learning	Estimation
Learner, Algorithm	Model, Estimator
Features	Covariates, Explanatory Variables, Independent Variables, Predictors
Target, Label, Output	Dependent Variable
Example, Instance	Data Point, Observation

Sparsity in asset pricing

We have many potential characteristics, *J* (many features, *j* = *1…J*)

- If J > N, then OLS doesn't work

Ad hoc sparsity in empirical asset pricing

Ad hoc sparsity in theoretical asset pricing

- Enormity of variables can go into forming rational expectations
- Process of learning about parameters of the data-generating process is important for asset pricing

Goals of this textbook

Focus on supervised learning in asset pricing

- Find a function that maps features into labels

$$y_i = f(x_i)$$

Limitations – focused on economics and its restrictions

- Not focused on computational load nor a survey of ML tools
- Mostly focused on cross-sectional returns

Model selection in supervised learning

Wolpert (1996) – no free lunch; no one learning algorithm over another

Review of linear regression methods

- Ordinary least squares (OLS) regression
- Ridge regression (Hoerl and Kennard (1970))
- Lasso (Tibshirani (1996))
- Elastic net (Zou and Hastie (2005))

Nonlinear regression methods: random forests, neural networks

OLS and its discontents

N observations; *K* features

Assumes the function is linear

$$y_i = f(x_i) + \varepsilon_i$$

$$y_i = x_i' g + \varepsilon_i$$

Effectively we have an objective to minimize the sum of squared errors

$$\min_g (y - Xg)'(y - Xg)$$

$$\hat{g} = (X'X)^{-1} X'y$$

$$\hat{y} = X\hat{g}$$

Unreliable where *K* is not small relative to *N* (or even *K*>*N*);

In-sample R^2 may be large but OOS struggles

Methods vary by objective functions

OLS objective function:

$$\min_g (y - Xg)'(y - Xg)$$

$$\hat{g} = (X'X)^{-1}X'y$$

Ridge regression

square L^2-norm penalty

$$\min_g \left[\frac{1}{N}(y - Xg)'(y - Xg) + \gamma g'g\right]$$

$$\hat{g} = (X'X + \gamma I_K)^{-1}X'y$$

Lasso

L^1-norm penalty

$$\min_g \left[\frac{1}{N}(y - Xg)'(y - Xg) + \gamma \sum_{j=1}^{K} |g_j|\right]$$

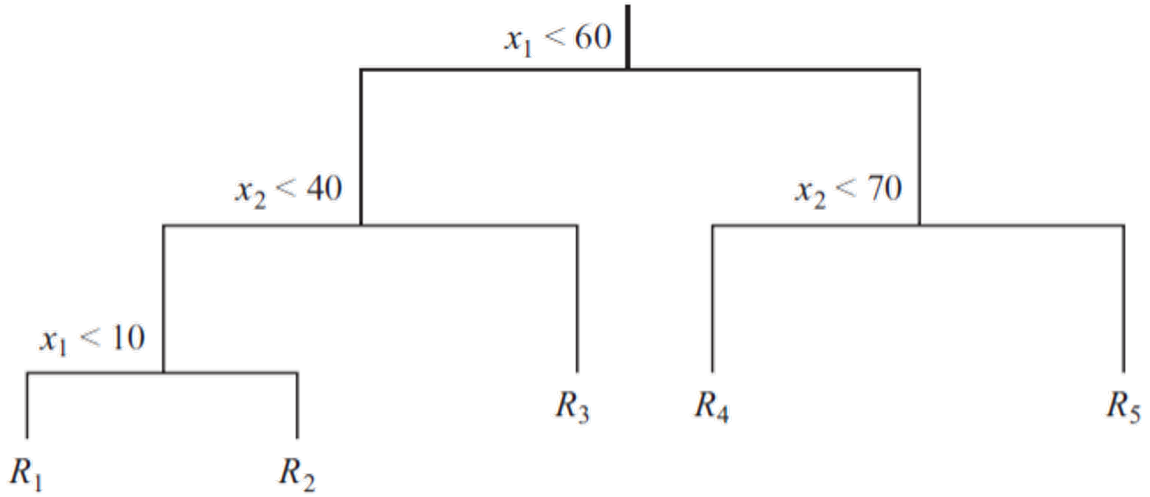
Elastic net

$$\min_g \left[\frac{1}{N}(y - Xg)'(y - Xg) + \gamma_1 \sum_{j=1}^{K} |g_j| + \gamma_2 g'g\right]$$

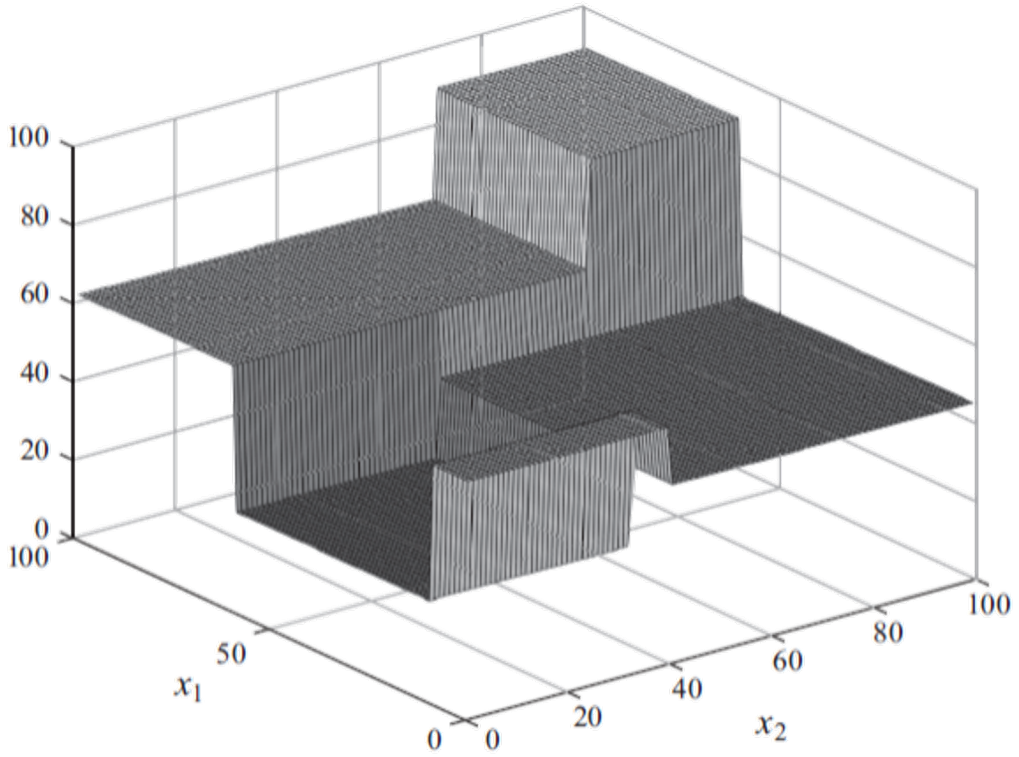
Regression trees are still regressions

Divide the feature space into *H* leaves; $\bar{y}_h$ is average of observations; minimize the penalized residual sum of squares within regions R_h :

$$\sum_{h=1}^{H} \sum_{x_i \in R_h} (y_i - \bar{y}_h)^2 + \gamma H$$



(a) Tree representation



(b) Fitted step function

Random forests are bagged trees

1) Allow “full sized” trees without pruning
2) Apply bootstrap aggregation, or bagging
   (i.e., randomly select $m<J$ features to grow a tree)

The constant m is now the tuning parameter

Averaging across trees is similar to kernel regressions; form a weighted average observations around $\boldsymbol{x}_i$ with more weight to closer observations

Neural networks and their underlying models

Highly nonlinear regressions

Suppose we have *J* covariates x_i as inputs; y_i as output

Hidden layers of *H* nodes

Single layer model is then: $f(x_i) = a_2 + w_2' g(a_1 + W_1 x_i)$

Non-linear activation function *g*, commonly use Recitfied Linear Unit

$$g(z) = max(0, z)$$

Neural networks layers approximate nonlinear *f*

Model with 2 layers would be shown as below (often have 10-20 layers)

$$f(x_i) = a_3 + w_3 g(a_2 + W_2 g(a_1 + W_1 x_i))$$

With large *H*, can approximate highly nonlinear functions arbitrarily well

Hornik, Stinchcombe, and White (1989)

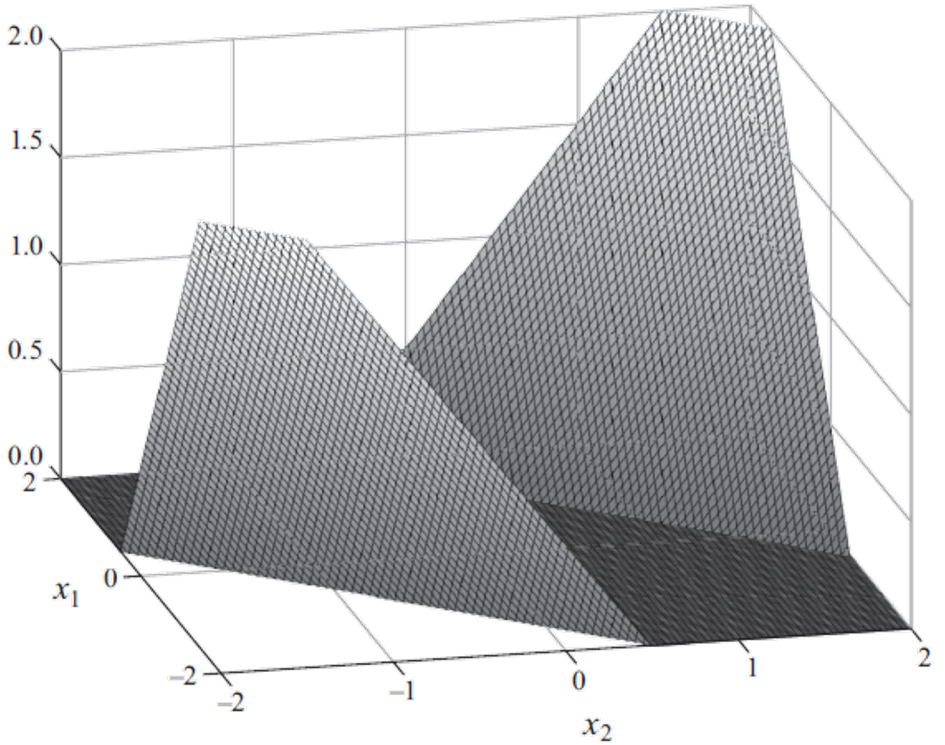


Figure 2.2. Neural network example: Interaction effects

Example: 1 layer, 2 nodes

$$f(x_i) = \max(0, -3/2 + x_{i,1} + x_{i,2}) + \max(0, -3/2 - x_{i,1} - x_{i,2})$$

Neural networks and their objective functions

Recall OLS objective function: $\min_g (y - Xg)'(y - Xg)$

Neural networks are similar, $\min_\theta \sum_{i=1}^{N} [y_i - f(x_i, \theta)]^2$

and can also have penalties $\min_\theta \sum_{i=1}^{N} [y_i - f(x_i, \theta)]^2 + \gamma\theta'\theta$

Solve these with stochastic gradient descent or quasi-Newton methods

AIC commonly used to tune hyperparameters

We estimate in-sample error $mse(\gamma) = \frac{1}{N}[y - X\hat{g}(\gamma)]'[y - X\hat{g}(\gamma)]$

In a linear model ($\hat{y} = Hy$), capture model complexity with effective number of parameters

$$d(\gamma) = tr[X(X'X + \gamma I_K)^{-1}X']$$

We can minimize the Akaike information criterion (AIC)

$$AIC(\gamma) = N \log mse(\gamma) + 2d(\gamma)$$

Requires errors ε to be iid Normal; or, specify the likelihood function

Cross-validation is also commonly used

In non-linear models, reliably calculating AIC can be difficult

Instead, use **cross-validation**, using model from training set in the validation set

- Stone (1977) showed that model comparisons based on CV or AIC are asymptotically equivalent

Construct (X_v, y_v) from the validation set

$$\hat{\gamma} = \arg\min[y_v - X_v\hat{g}(\gamma)]'[y_v - X_v\hat{g}(\gamma)]$$

$$\hat{\gamma} = \arg\min \frac{1}{k}\sum_{j=1}^{k}[y_{v(j)} - X_{v(j)}\hat{g}_{-v(j)}(\gamma)]'[y_{v(j)} - X_{v(j)}\hat{g}_{-v(j)}(\gamma)]$$

Cross-validation is also commonly used

K-fold the data sets

- $v(j) = 1, 2, \ldots, k$ denotes an index of the k folds that the data set has been divided into, while $-v(j)$ set not in the j-th fold

Tradeoff between larger and smaller *k*

- Small *k* → smaller training data; pessimistic bias of prediction error
- Large *k* → training sets may overlap; high variance of prediction error

Usually choose *k* much smaller than number of observations

Summary

Augment objective functions (e.g., ridge, lasso)

$$\min_g \left[\frac{1}{N}(y - Xg)'(y - Xg) + \gamma g'g\right] \quad \min_g \left[\frac{1}{N}(y - Xg)'(y - Xg) + \gamma \sum_{j=1}^{K} |g_j|\right]$$

Random forests and neural networks add more structure

$$\sum_{h=1}^{H} \sum_{x_i \in R_h} (y_i - \bar{y}_h)^2 + \gamma H \qquad \min_\theta \sum_{i=1}^{N} [y_i - f(x_i, \theta)]^2 + \gamma \theta'\theta$$

$$f(x_i) = a_2 + w_2' g(a_1 + W_1 x_i)$$

Hyperparameter tuning with AIC or cross-validation

$$AIC(\gamma) = N \log mse(\gamma) + 2d(\gamma)$$