Machine Learning Crash Course: 1. The Basics

Carlo Zanella

September 11, 2018

This course

This is a crash course. It aims to lay down the basic thinking behind Machine Learning and how it differs from econometrics. We cover some of the foundational methods in the field. Although the focus is not on the methods, we do cover some state-of-the art learning algorithms. However, we leave deep learning and neural networks to more advanced courses.

Of course, to get our hands on as quickly as possible, this course also serves as an introduction Machine Learning in R.

This course is largely based on the book "An Introduction to Statistical Learning with Applications in R" by Garet J., Witten D., Hastie T., and Tibshirani R.

Roadmap

Contents:

1. Wednesday Morning — Basics
2. Wednesday Afternoon — Overfitting and Regularization
3. Thursday Morning — Classification
4. Thursday Afternoon — Tree-based methods
5. Friday Morning — Some Practice & Wrap-Up

Structure

- I will use Slides to introduce theory and methods
- Slides will be pushed to pp4rs/2018-uzh-course-material/21-machine-learning/slides before the session
- We will together go through code notebooks to apply what we learn
- And then there will be time for exercises for you to try stuff on your own and ask any questions

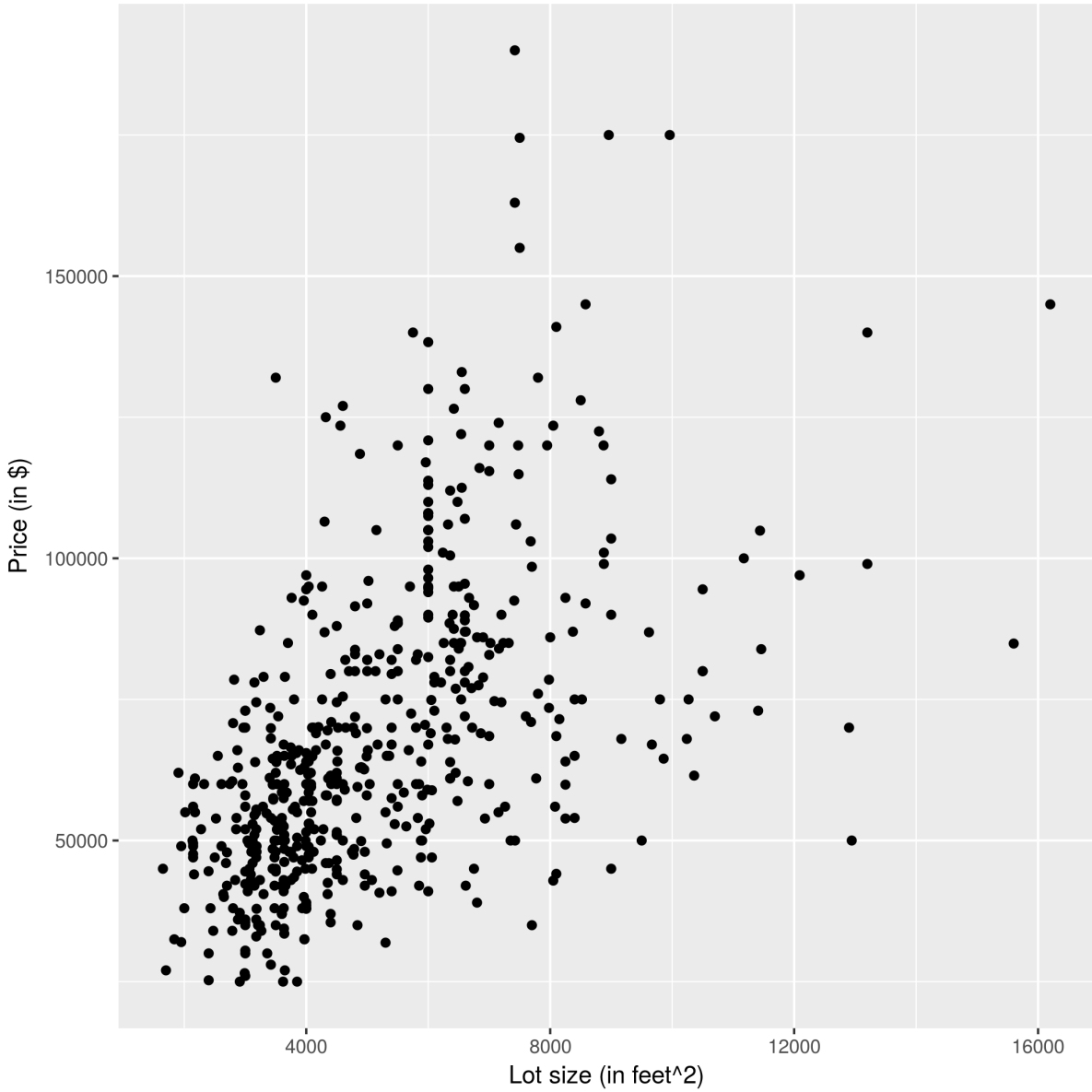
Getting set up

- For this course you need a couple of R packages.
- They should ☺ be easy and quick to install
- We will make an extended break where you will have the time to install everything.

What is Machine Learning?

- Arthur Samuel (1959): Machine Learning: Field of study that gives computers the ability to learn without explicitly being programmed.
- Usually learn a relationship between some variable of interest Y and explanatory variables $X = X_1, \ldots, X_N$.
- Two kinds of ML:
- Supervised learning: Give labelled data $(y_1, x_1), (y_2, x_2), \ldots$ to an algorithm. Task: to learn the relationship between Y and X.
  This is the most common form and what is the focus of this course.
- Unsupervised learning: We only know X but not Y. Task: infer some "structure" only using the X
- Example: clustering or cat videos `https://www.wired.com/2012/06/google-x-neural-network/`

Supervised Learning



Prediction, Prediction, Prediction!

- In Machine Learning we use features ("explanatory variables") to predict the outcome variable.
- For continuous outcome variables: regression problem
- For discrete outcome variables: classification problem
- Not interested in:
  - causality
  - bias of coefficients
  - significance of coefficients
- We only care about how "well" we can make out-of-sample predictions (or classifications).

More applications

- Spam filtering (classify "spam" vs "not spam")
- Default probability on credit card loans
- Text classification
- Predict effect of a drug based on patients' characteristics
- Recommender systems ("oh you bought X, you might like Y, Z, A, and Q!")
- ...

The Task of ML

- Again suppose we have variables Y and $X = X_1, \ldots$.
- Suppose there is some relationship of the general form

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

  where f is fixed but unknown and ϵ is a random error term.
- We have a set of observations $(y_1, x_1), (y_2, x_2), \ldots$
- Goal: estimate f in order to predict Y for unseen data on X

Questions of Machine Learning

1. How to estimate f?
2. How to evaluate the estimates we obtain?

Reducible vs irreducible error

- Writing

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

  allows for an irreducible error represented as ϵ.
- This may come from omitted variables or inherent uncertainties Y or X.
- Suppose we have an estimate $\hat{f}$ of f.
- Since $\hat{f}$ is not perfect this will add to the error in predicting Y.
- We call this the reducible error since we may improve our estimation of f.
- Let $\hat{Y} = \hat{f}(X)$ so that

$$\begin{aligned} MSE = E(Y - \hat{Y})^2 &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{Irreducible}} \end{aligned}$$

Assessing Model Accuracy

- Let's take OLS as an example.
- We already know that OLS can give as an estimate of f.
- By definition, OLS minimizes the in-sample mean squared error.
- However, what we are truly interested in is the out-of-sample error.
- This is called the test error.
- In the regression case: test MSE (mean squared error)
- Not the same as training MSE!

Assessing Model Accuracy

(Notebook “01 Assessing Model Accuracy”)

Cross-Validation

- Problem with previous approach: often we cannot easily get more data!
- We need a way to estimate the test error with the data we have.
- Carefully estimating the test error is one of the main tasks of the ML practitioner!
- Next we are looking at methods to estimate the test error.
- Such methods are usually called Cross-Validation

Validation Set approach

- The Validation Set approach randomly partitions the data into two parts called the training set and a validation set or hold-out set.
- The models are fitted on the training set and the test error is estimated on the validation set.
- For example we can randomly assign 75% of the data to the training set and 25% to the validation set.
- Advantage: very easy to implement
- Disadvantage: the obtained test MSE has a large variance

Leave-One-Out Cross-Validation (LOOCV)

- Let our data have n observations.
- Like in the validation set approach, this method involves splitting the data into two parts
- But here the validation set contains only one element, say (y_i, x_i).
- The model is fit for $n-1$ remaining observations and the test error is computed only for (y_i, x_i). Call it MSE_i.
- We can repeat this procedure for all the observations.
- Then we get n estimates for the test error.
- Our overall estimate is then simply the average of all estimates:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \text{MSE}_i.$$

k-fold Cross-Validation

- Leave-One-Out Cross-Validation turns out to be a special case of k-fold Cross-Validation.
- k-fold CV randomly splits the data into k folds (subsets of about equal size)
- We select one fold to be the hold-out set and fit the model to the other $k - 1$ folds.
- On the hold-out set we compute the test error.
- We repeat k times for all folds to get k estimates of the test error.
- Finally

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i.$$

as before.

Cross-Validation: Discussion

- k-fold CV with $k < n$ is computationally easier than LOOCV
- k CV is not deterministic: there is variability but it is lower than for validation set approach
- There is also a bias-variance trade-off for $CV_{(k)}$ at work here:
- LOOCV (high k): low bias (we use almost all observations), but high variance (average of highly correlated MSE_i)
- $k = 2$: high bias (we ignore half of the observations), low variance (the two estimates are uncorrelated because the data sets are distinct)
- Note: we mean the bias and the variance of the estimate of the test error ($CV_{(k)}$).
- For this reason 10-fold CV is often preferred to LOOCV and Validation Sets.

Cross-Validation

(Notebook "02 Cross-Validation")