Class Notes

Statistical Computing & Machine Learning

Class 2

Review

- Supervised vs unsupervised We will be doing only supervised learning until late in the course
- Supervised learning:
  - We have a set of cases, $i = 1, 2, 3, \ldots, n$, called the **training data**.
  - For each case, we have an input $\mathbf{X}_i$ consisting potentially of several variables measured on that case.
    - The subscript i in $\mathbf{X}_i$ means that we have one $\mathbf{X}$ for each case. The boldface $\mathbf{X}$ means that the input can be multi-dimensional, that is, consisting of multiple variables.
  - For each case, we have an output Y_i.
  - We want to learn the overall pattern of the relationship between the inputs $\mathbf{X}$ and the outputs Y, not just for our n training cases, but for potential cases that we have not encountered yet. These as yet unencountered cases are thought of as the **testing data**.
  - We are going to represent the pattern with a **function** $\hat{f} : \mathbf{X} \rightarrow Y$.
  - Sometimes I'll use the word **model** instead of function. A model is a representation for a purpose. A function is a kind of representation. So some models involve functions. That's the kind we'll focus on in this course. I say "model" out of habit, but it's a good habit that reminds us that the **purpose** of the function is important and we should know what that purpose is when we undertake learning.
- Regression vs classification
  Different kinds of functions. A classifier has output as a categorical level. A regression has output as a number.
- Prediction vs inference
  - Two different kinds of purpose. There may well be different kinds of functions best suited to each purpose.
- Accuracy vs interpretability
  We always want models to be accurate. Whether we need to be able to interpret the model depends on our overall purpose.

- Reducible error vs irreducible error
  It's good to know how accurate our models can get. That gives a goal for trying out different types of models to know when we don't need to keep searching.

Today's theory: accuracy, precision, and bias

Figure 2.10

In constructing a theory, it's good to have a system you can play with where you know exactly what is going on: e.g. a simulation.

The dark blue line in the left panel is a function the authors created for use in a simulation:

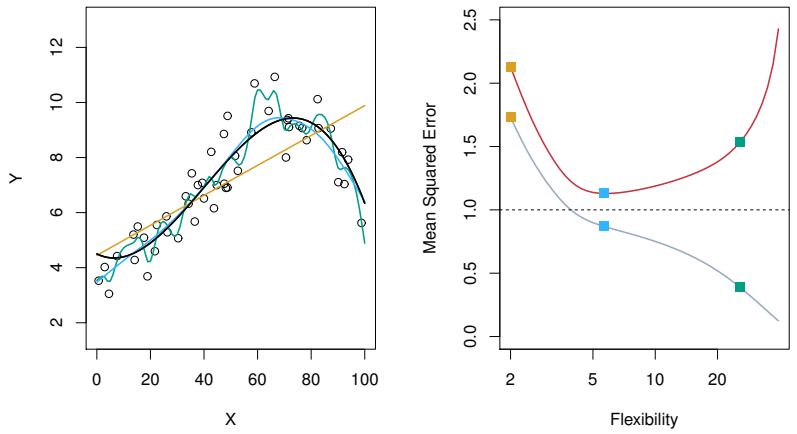

Figure 2.9 from ISL

The dots are data they generated from evaluating the function at a few dozen values of x and adding noise to each result.

The difference between the dots' vertical position and the function value is the *residual*, which they are calling the *error*. The mean square error MSE is

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

- Take this notation apart. What's n? What's i?
- Suppose that $f(x)$ were constant. In that situation, what kind of statistical quantity does this resemble?
- In actual practice, we don't know $f(x)$. (Indeed, it's a matter of philosophy whether this is an $f(x)$ — it's a kind of Platonic form.) Here we know $f(x)$ because we are playing a game: running a simulation.

Looking again at the left panel in Figure 2.9, you can see three different functions that they have fitted to the data. It's not important right now, but you might as well know what these model architectures are:

1. Linear regression line (orange)
2. Smoothing splines (green and light blue). A smoothing spline is a functional form with a parameter: the *smoothness*. The green function is less smooth than the light blue function.
3. That smoothness measure can also be applied to the linear regression form

Each of these three functions were fitted to the data. Another word for fitted is *trained*. As such, we use the term *training error* for the difference between the data points and the fitted functions. Also, because the functions are not the Platonic $f(x)$, they are written $\hat{f}(x)$.

For each of the functions, the training MSE is

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

Right panel of the graph is something completely different: both the axes are different than in the left panel.

- x-axis: the smoothness of the functions. This is labelled *flexibility*.
- The three x positions correspond to the smoothness of the three models. This is measured as the effective *number of parameters* of the function.

  Why does the straight-line function have a smoothness of 2?
- y-axis: the MSE
  - The dots connected by the gray curve show the *training* MSE of the three models.
  - The dots connected by the orange curve show the *testing* MSE of the three models.
  - The continuous curves were constructed by calculating the MSE for many more values of smoothness than shown in the left panel.
- How did they measure the *training* MSE?

Another example: A smoother simulated $f(x)$.

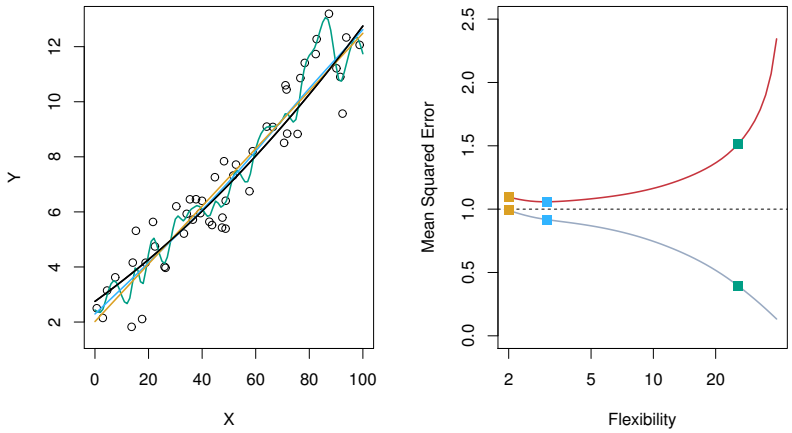

Figure 2.10 from ISL

- What's different between the right panel of 2.9 and that of 2.10?

What's the "best" of these models?

When examining training MSE, the more flexible model has the smaller MSE. This answer is pre-ordained, regardless of the actual shape of the Platonic $f(x)$.

In traditional regression, we use ANOVA or *adjusted$ R^2 to help avoid this inevitability that more complicated models will be closer to the training data.

Another approach to this is to use *testing* MSE rather than training MSE. So pick the model with flexibility at the bottom of the U-shaped testing MSE curve.

Why is testing MSE U-shaped?

- Bias: how far $\hat{f}(x)$ is from $f(x)$
- Variance: how much $\hat{f}$ would vary among different randomly selected possible training samples.

In traditional regression, we get at the variance by using confidence intervals on parameters. The broader the confidence interval, the higher the variation from random sample to random sample. These confidence intervals come from normal theory or from bootstrapping. Bootstrapping is a simulation of the variation in model fit due to training data.

Bias decreases with higher flexibility.

Variance tends to increase with higher flexibility.
Irreducible error is constant.

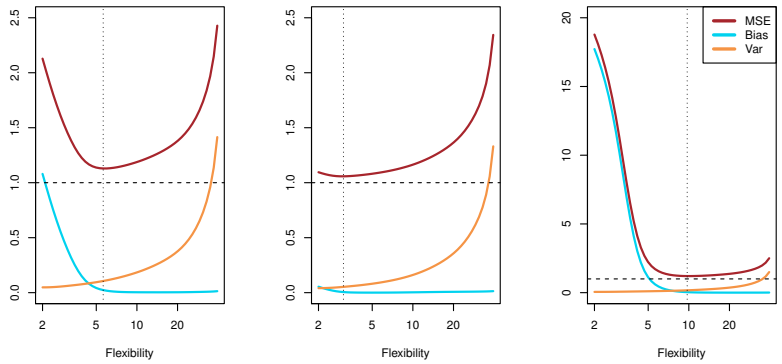

Figure 2.12 from ISL

Measuring the variance of independent sources of variation

Simulation:

Add three different sources of variation. The width of the individual sources is measured by the standard deviation `sd=`.

```
require(mosaic)
n <- 1000
sd(rnorm(1000, sd = 3) + rnorm(1000, sd = 1) +
    rnorm(1000, sd = 2))
```

```
## [1] 3.648619
```

- Divide into small groups and
  - construct a theory about how the variation in the individual components relates to the variation in the whole.
  - test whether your theory works for other random distributions, e.g. `rexp()`

Equation 2.7

$$E(y - \hat{f}(x))^2 = \text{Var}(\hat{f}(x)) + [\text{Bias}(\hat{f}(x))]^2 + \text{Var}(\epsilon)$$

Breaks down the total "error" into three independent sources of variation:

1. How y_i differs from $f(x_i)$. This is the irreducible noise: ϵ
2. How $\hat{f}(x_i)$ (if fitted to the testing data) differs from $f(x_i)$. This is the *bias*.
3. How the particular $\hat{f}(x_i)$ fitted to the training data differs from the $\hat{f}(x_i)$ that would be the best fit to the testing data.

$$\underbrace{E(y - \hat{f}(x))^2}_{\text{Total error}} = \underbrace{\text{Var}(\hat{f}(x))}_{\text{source 3.}} + \underbrace{[\text{Bias}(\hat{f}(x))]^2}_{\text{source 2.}} + \underbrace{\text{Var}(\epsilon)}_{\text{source 1.}}$$

Programming Basics I: Names, classes, and objects

Names

Composed of letters, numbers, _ and ..

- Don't use . — it's a bad habit. But plenty of people do. - Can't lead with a number. - Capitalization counts. - Unquoted (... almost always)

Objects

Information (bits) in a particular format.

- Different formats for different purposes. - The format is the `class()` or `mode()`. `mode` is more basic than `class`.

Assignment: Give a name to an object

Classes

Different classes represent different kinds of things. They typically have different operations that are relevant.

- 1-dimensional homogeneous collections of numbers or of character strings. These collections are called *vectors*
  - 1-dimensional means you need only one index to refer to a specific element
  - `length()`, which can be zero.
  - Operations, e.g. `sum()`, `mean()`, `max()` ...
- 2-dimensional homogeneous collections of numbers or of character strings. These collections are called *matrices*
  - 2-dimensional means you need two indices to refer to a specific element.
  - `dim()`
  - Operations, e.g., `t()`, `colSums()`, `rowSums()`, `%*%`, ...
- 1-dimensional heterogeneous collections: *lists*
- Data frames
- Functions

In-class programming activity

Day 2 activity