# 1. Introduction to Machine Learning

Jesper Armouti-Hansen

University of Cologne

December 10, 2018

## Course content

1. Introduction to Machine Learning
2. Linear methods for Classification
3. Model Assessment and Selection
4. Linear methods for Regression
5. Moving beyond linearity
6. Tree-based methods
7. Support Vector Machines

Format: mix of lectures, hands-on sessions, and case studies

## Course website

jeshan49.github.io/eemp2/

## Today

- Lecture:
    - Motivation

- Tutorial:
    - Python basics
    - Introduction to NumPy
    - Getting started with Pandas
    - Plotting and visualization

## What is Machine Learning (ML)?

**General Definition**

[Machine Learning is the] field of study that gives computers the ability to **learn** without being explicitly programmed.

- (Arthur Samuel, 1959)

**More specific...**

A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**.

- (Tom Mitchell, 1997)

## Types of ML

**Supervised Learning**

- The task of learning a function that maps an input to an output based on example input-output pairs
- The learning method learns from a training sample consisting of a set of input-output obervations
- Depending on the nature of the output, the learning method should generalize from the training data to unseen data in order to either
  - give a good prediction of the output for new input if the output is quantitative
  - give a good classification of the output for new input if the output is qualitative
- Examples:
  - classifiy emails into "spam" or "ham" based on the content
  - predict the price of a house given a set of features

## Unsupervised Learning

- We observe inputs but no outputs
- We can seek to understand the relationship between the variables or between the observations
- For example, in a market segmentation study we might observe multiple characteristics for potential customers and attempt to see if customers can be clustered into groups based on these characteristics
- If we have high-dimensional data, we might use unsupervised methods to project our inputs onto a lower dimensional space
- This can be a pre-processing step for a supervised learning method

## Reinforcement Learning

- The learning system, called an agent, can observe the environment, select and perform actions, and get a reward/punishment in return
- The agent then learns the best strategy in order to get the most reward over time
- The strategy defines what action the agent should choose when it is in a given situation
- Example: Deep-Q video

## Online vs. Batch Learning

- In batch learning, the learning method is incapable of learning incrementally
  - It must be trained using all available data
- Suppose we launch a method into production. Then, it is going to apply what it has already learned, but does not learn anymore: *offline learning*
- Thus, if we wish to update our model, we need to train a whole new version of it from scratch

- In online learning, you train the system incrementally by feeding it small data instances sequentially
- For example, if one wants a model that classifies spam/ham email, then it is probably a good option to have a model which can adapt to changes
- Online learning algorithms can also be used to train systems on huge data sets that cannot fit in one machine's memory
- An important parameter of the online learning algorithm is the learning rate
  - high: system will adapt quickly to new data, but quickly forget older
  - low: slow learning of new data, but will be less sensitive to noise in the new data

## Some Statistical Decision Theory

Let $X \in \mathbb{R}^p$ denote a real valued random vector

- i.e. the vector of inputs, features, predictors, or independent variables

Let $Y \in \mathbb{R}$ be a real valued random output

- i.e. the response, target, or independent variable

Let $Pr(X, Y)$ be their joint probability distribution

- We seek a function $f(X)$ for predicting $Y$ given values of the input $X$
- For this, we require a loss function $L(Y, f(X))$ that penalizes errors in prediction
- We will use the squared loss $(Y - f(X))^2$

Hence, we seek the function $f(X)$ which minimizes

$$\begin{aligned} ESE(f) &= E[(Y - f(X))^2] & (1) \\ &= E[E[(Y - f(X))^2|X]] & (2) \end{aligned}$$

The solution is

$$f(x) = E[Y|X = x] \tag{3}$$

the conditional expectation, known as the regression function

- Thus, the best prediction of $Y$ at any point $X = x$ is the conditional mean
- Note: $\varepsilon = Y - f(X)$ is the irreducible error
  - even if we knew $f(x)$, we would still make errors in prediction, since at each $X = x$ there is typically a distribution of possible $Y$ values.

Suppose we have estimated $f(x)$ with $\hat{f}(x)$ at the point $x$. Then we have

$$\begin{aligned} E[(Y - \hat{f}(X))^2 | X = x] &= E[(f(X) + \varepsilon - \hat{f}(X))^2 | X = x] \\ &= \underbrace{[f(x) - \hat{f}(x)]^2}_{\text{reducible}} + \underbrace{Var[\varepsilon]}_{\text{irreducible}} \end{aligned}$$

We focus on techniques for estimating $f$ with the aim of minimizing the reducible error.

The irreducible error provides an upper bound on the accuracy, but is almost always unknown.

## Estimating $f$ - kNN regression

With our training data, we might attempt to directly apply the concept of $E[Y|X = x]$ by asking, at each point $x$, for the average $y_i$.

Since we rarely have sufficient data points to do this, we can settle for:

$$\hat{f}(x) = Ave(y_i|x_i \in N_k(x)) \tag{4}$$

We perform two approximations in this approach:

- Expectation is approximated by averaging over sample data;
- Conditioning at a point is relaxed to conditioning on some region "close" to the target point

The $k$ is called a *hyper* or *tuning parameter* - A parameter not learned from the learning procedure

# The Curse of Dimensionality

The kNN approach can work quite well for *small* $p$ and large $N$

As $p$ gets large, the nearest neighbors tend to be far away:

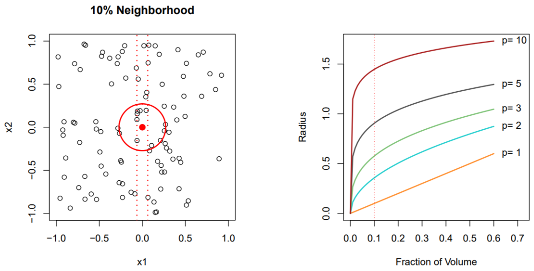

Figure: The radius of a sphere needed to capture a fraction of the volume of the data for different dimensions $p$ (See ESL p. 23)

## Estimating $f$ - Linear regression

How does linear regression fit into this framework?

- We simply assume that the regression function $f(x)$ is approximately linear in its arguments

$$f(x) \approx x^T \beta \tag{5}$$

- Plugging this linear model for $f(x)$ into ESE in (1) and differentiating we can solve for $\beta$:

$$\beta = [E[XX^T]]^{-1} E[XY] \tag{6}$$

- the least squares solution:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{7}$$

  amounts to replacing the expectation by averages over the training data

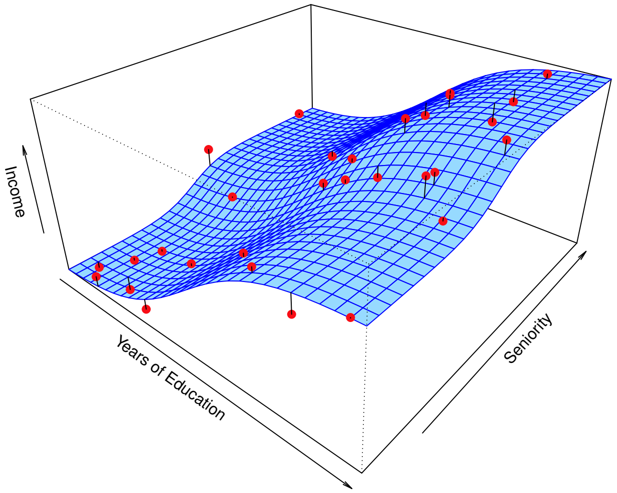

Figure: Simulated data: Income as a function of years of education and seniority. The blue surface represents the true underlying relationship. Red dots indicate observed values for 30 individuals (See ISLR p. 18)

## Parametric vs. Non-parametric Methods

Our goal: Apply a learning method in order to estimate $f$ such that, for any observation $(X, Y)$, we have $Y \approx \hat{f}(X)$.

Broadly speaking, we can decompose the learning methods into one of the following groups:

1. Parametric methods
2. Non-parametric methods

The parametric methods share the following two step approach:

1. We make an assumption about the functional form of $f$
   - For example, if we assume $f$ is linear, we only need to estimate $p+1$ coefficients as opposed to an arbitrary $p$-dimensional function
2. Based on our chosen functional form, we choose a procedure to fit the model
   - For example, for the linear model, we (may) use the least squares procedure.

The method is parametric as it reduces the problem down to estimating a set of parameters.

Potential disadvantages?

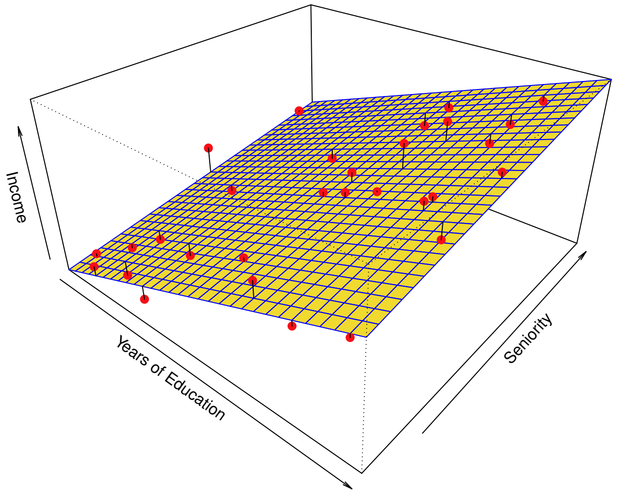

Figure: A linear model fit by least squares:
$\hat{income} = \beta_0 + \beta_1 * education + \beta_2 * seniority$ (See ISLR p. 22)

Non-parametric methods:

- No explicit assumptions about the functional for of $f$
- Attempts to give an estimate of $f$ close to observed data points subject to pre-specified constraints

Advantage:

- Avoids making wrong functional form assumptions about $f$

Disadvantage:

- Since the estimation problem is not reduced down to a set of parameters, a very large number of observations is required to obtain an accurate estimate.

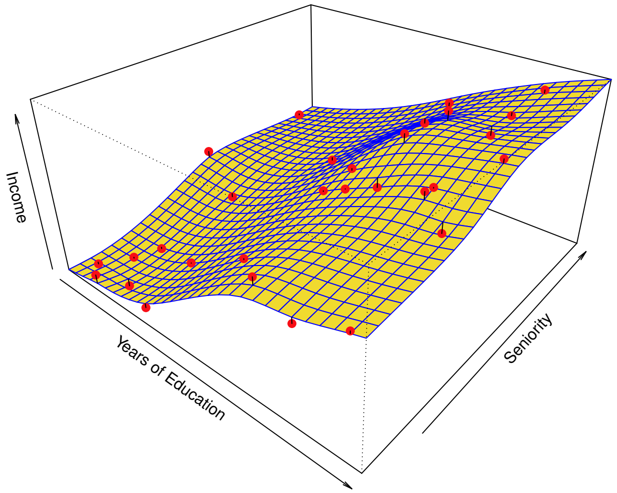

Figure: A smooth thin-plate spline fit to the same income data (See ISLR p. 23)

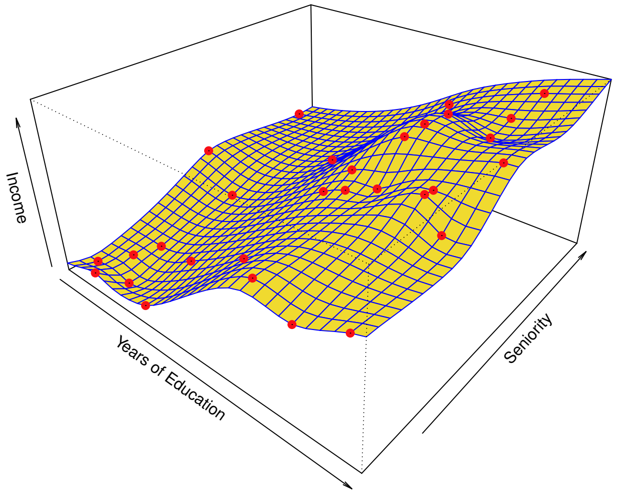

Figure: A rough thin-plate spline fit to the same income data (See ISLR p. 24)

## Prediction Accuracy vs. Model Interpretability

Why would we ever choose to use more restrictive models instead of a very flexible approach?

- As discussed, we might not have enough data
- If our goal includes model interpretability
  - Some of the models become so complex that understanding how any individual predictor is associated with the response becomes difficult
- We might overfit with highly flexible methods
- We often prefer a simple model involving fewer variables over a black-box model involving them all

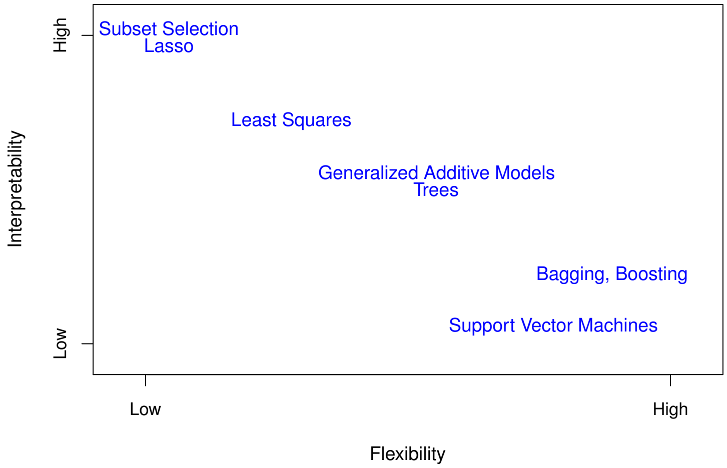

Figure: Tradeoff between flexibility and interpretability, using different learning methods (See ISLR p. 25)

## Assessing Model Accuracy

Why introduce many different learning approaches?

- No Free Lunch: No one methods dominates all others over all possible data sets

Hence an important task is deciding on the best model for a given data set. To decide on a method, we need a metric to evaluate the quality of the fit

- We could compute the average squared prediction error over *Tr*:

$$MSE_{Tr} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(x_i))^2 \tag{8}$$

- This may be biased toward more overfit models.

- What we would like to know is how our model, in general, will perform on new data
- If we are in a data-rich environment, we could have a designated hold-out or test set $Te = \{x_i, y_i\}_1^M$ to estimate it:

$$MSE_{Te} = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{f}(x_i))^2 \tag{9}$$

  i.e., the test squared prediction error
- What if we don't have a large test set?
  - Perhaps we could use the training MSE!

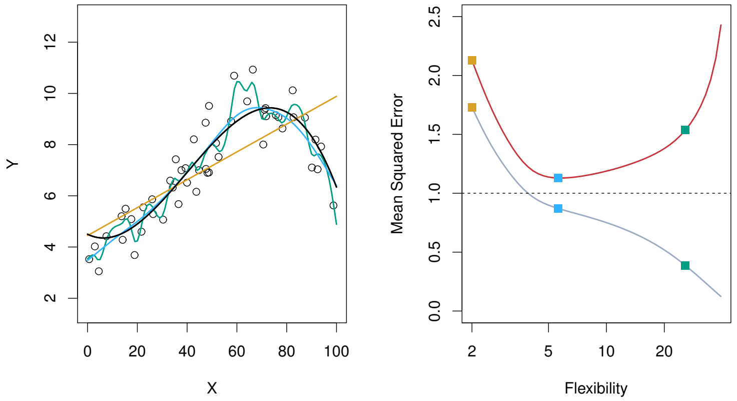

Figure: Simulated data: true *f* (black), linear regression line (orange), and two smoothing splines (blue and green) (See ISLR p. 31)

## A more "linear" example:

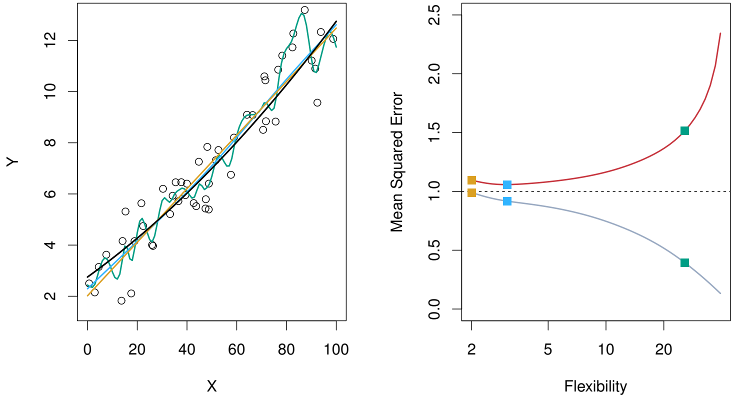

Figure: Simulated data: true $f$ (black), linear regression line (orange), and two smoothing splines (blue and green) (See ISLR p. 33)

## A more "non-linear" example:

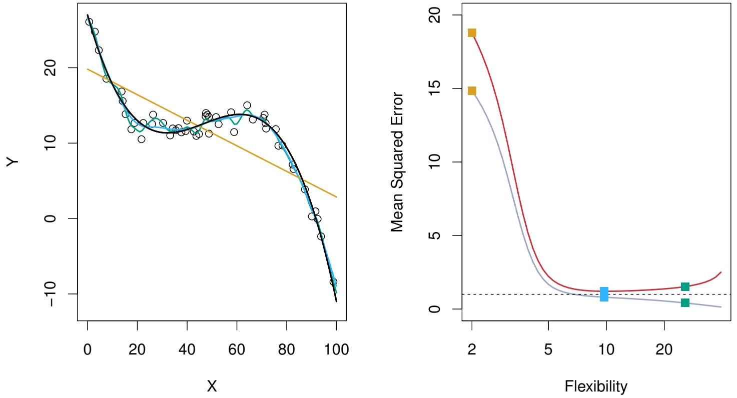

Figure: Simulated data: true *f* (black), linear regression line (orange), and two smoothing splines (blue and green) (See ISLR p. 33)

## The Bias-Variance Tradeoff

We saw that the test MSE tends to be U-shaped

- The shape is the result of two competing forces
- More formally, given $x_0$, the expected squared prediction error is given by

$$\begin{aligned}
E[(y_0 - \hat{f}(x_0))^2] &= E[y_0^2 + (\hat{f}(x_0))^2 - 2y_0\hat{f}(x_0)] \\
&= E[y_0^2] + E[(\hat{f}(x_0))^2] - E[2y_0\hat{f}(x_0)] \\
&= Var[y_0] + E[y_0]^2 + Var[\hat{f}(x_0)] + E[\hat{f}(x_0)]^2 \\
&- 2f(x_0)E[\hat{f}(x_0)] \\
&= Var[\varepsilon] + \underbrace{Var[\hat{f}(x_0)]}_{\text{variance of } \hat{f}} + \underbrace{[f(x_0) - E[\hat{f}(x_0)]]^2}_{\text{bias}^2 \text{ of } \hat{f}}
\end{aligned}$$

- $E[(y_0 - \hat{f}(x_0))^2]$ is the expected squared prediction error
  - i.e. the average test MSE if we repeatedly estimated $f$ using a large number of training sets, and tested each at $x_0$

A comparison of bias and variance in the three cases :

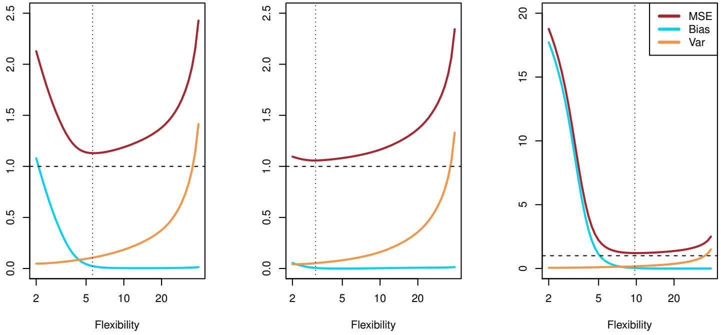

Figure: Squared bias (blue), variance (orange), $Var(\varepsilon)$ (dashed), and test MSE (red) for the three data sets (See ISLR p. 36)

## The Classification Setting

Now $Y$ is qualitative

- e.g. $\mathcal{C} = \{\text{spam, ham}\}$ or $\mathcal{C} = \{0, ..., 9\}$

We wish to build a classifier $C(X)$ that assigns a class label from $\mathcal{C}$ to a future unlabeled observation $X$

- Suppoe $\mathcal{C}$ contains $K$ elements numbered $1, ..., K$
- Let $p_k(x) = Pr(Y = k|X = x)$, $k = 1, ..., K$
- Suppose we knew the conditional probability of $Y$ given $X$
- Then, the *Bayes optimal classifier* at $x$ given by

$$C(x) = j \text{ if } p_j(x) = \max\{p_1(x), ..., p_K(x)\} \tag{10}$$

is optimal in the sense that it minimizes the expected one-zero loss:

$$E[\mathbb{I}(Y \neq C(X))] \tag{11}$$

## Estimating $C$ - kNN classification

The Bayes classifier produces the lowest possible test error rate, called *Bayes error rate*

$$1 - E[\max_j Pr(Y = j|X)] \tag{12}$$

We might attempt to apply kNN once again to

1. estimate the conditional distribution of $Y$ given $X$

$$\hat{p}_j(x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} \mathbb{I}(y_i = j) \tag{13}$$

2. classify a given observation to the class with highest estimated probability

$$\hat{C}(x) = j \text{ if } \hat{p}_j(x) = \max\{\hat{p}_1(x), ..., \hat{p}_K(x)\} \tag{14}$$

Thus, kNN gives an estimate for the conditional probabilities as well as for the decision boundary

Note that the curse of dimensionality applies here too!

## KNN: K=10

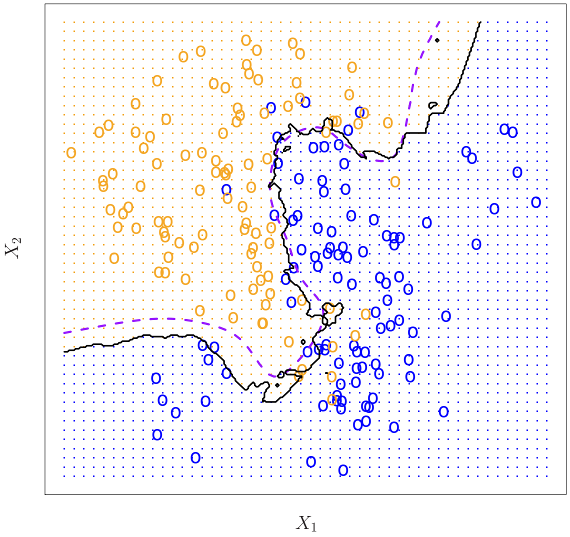

Figure: Bayes decision bounday (dashed), 10-NN decision boundary (black) (See ISLR p. 41)

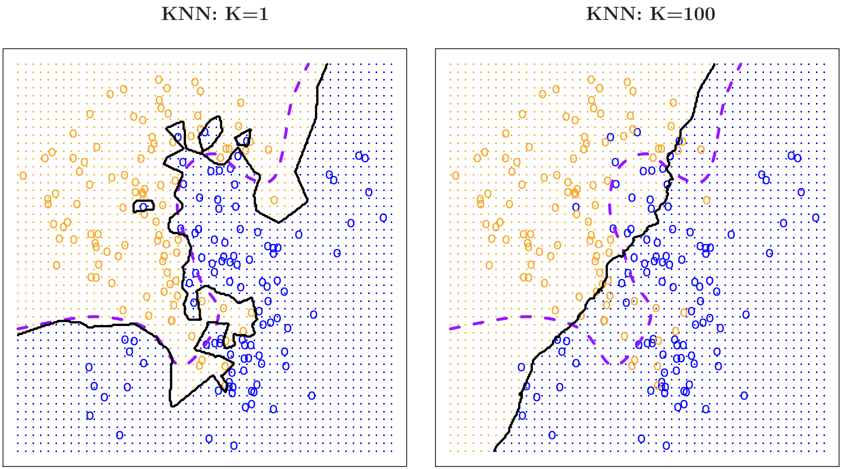

Figure: Bayes decision bounday (dashed), Left: 1-NN decision boundary (black), Right: 100-NN decision boundary (See ISLR p. 41)

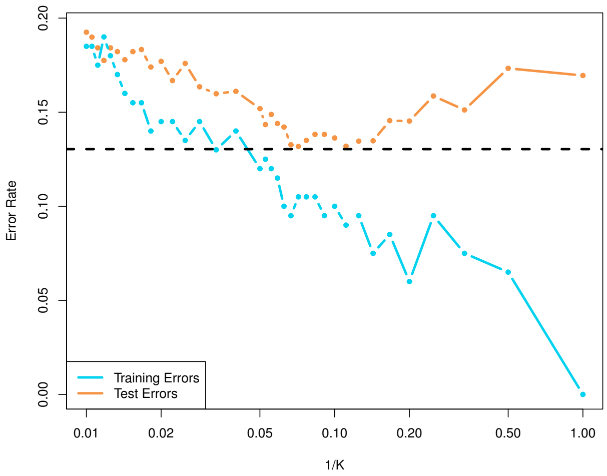

Figure: kNN training error rate (blue, 200 obs), test error rate (orange, 5,000 obs), Bayes error rate (dashed) (See ISLR p. 42)

- Some of the models we will consider build structured models for $C(x)$
  - e.g. support vector machines
- However, we will also build structured models for representing $p_k(x)$
  - e.g. logistic regression and generalized additive models

# Tutorial

## References

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). **Chapter 2**

Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). **Chapters 2**

McKinney, W. (2012). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. **Chapters 3, 4, 5, 6 & 7**

Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. **Chapters 1**