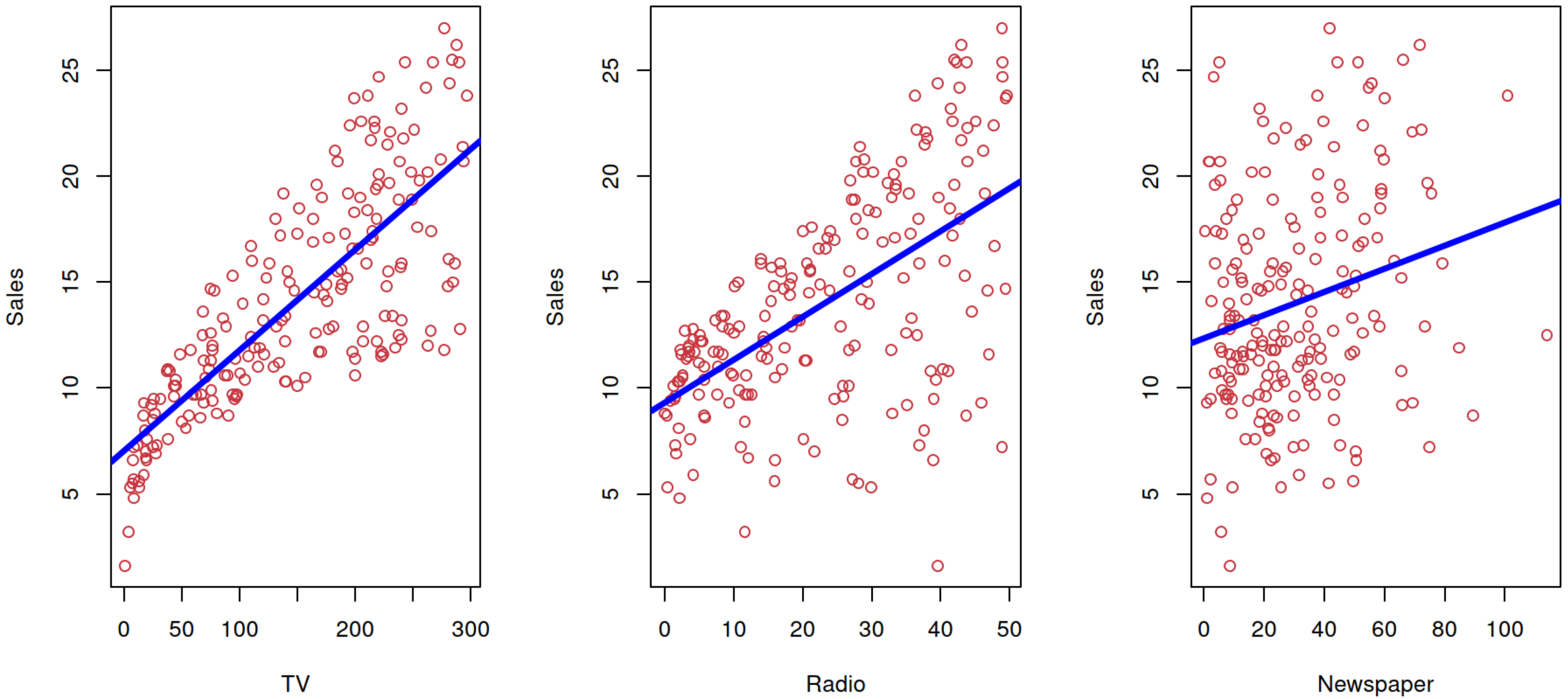
Statistical Learning

Szu-Chi Chung

Department of Applied Mathematics, National Sun Yat-sen University

What is Statistical Learning?

y=某產品在200家商店的銷售量



- Shown are *Sales* vs. *TV*, *Radio* and *Newspaper*, with a blue linear-regression line fit separately to each
- Can we predict Sales using these three? Perhaps we can do better using a model $Sales \approx f(TV, Radio, Newspaper)$

Notation

- Here, *Sales* is a response, dependent variable, or target that we wish to predict. We generically refer to the response as Y
- *TV* is a feature, independent variable, input, or predictor; we name it X_1. Likewise, name *Radio* as X_2, and so on
  - We can refer to the input vector collectively as

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

- Now, we write our model as

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

where ϵ captures measurement errors and other discrepancies and has a mean of zero

Notation

- Vectors are represented as a column vector

$$X_1 = \begin{pmatrix} X_{11} \\ X_{21} \\ \vdots \\ X_{n1} \end{pmatrix}$$

- We will use n to represent the number of distinct data points or observations
- We will let p denote the number of variables that are available for predictions
  - A general *design matrix* or input matrix can be written as an $n \times p$ matrix

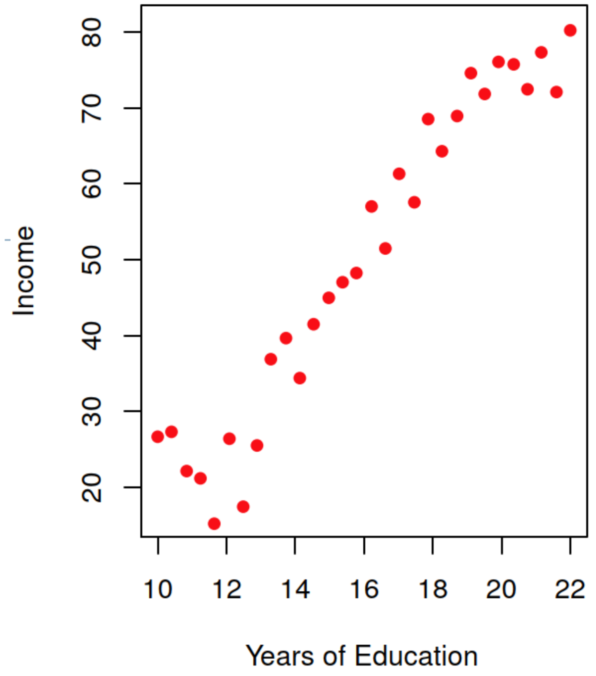
$$\begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$$

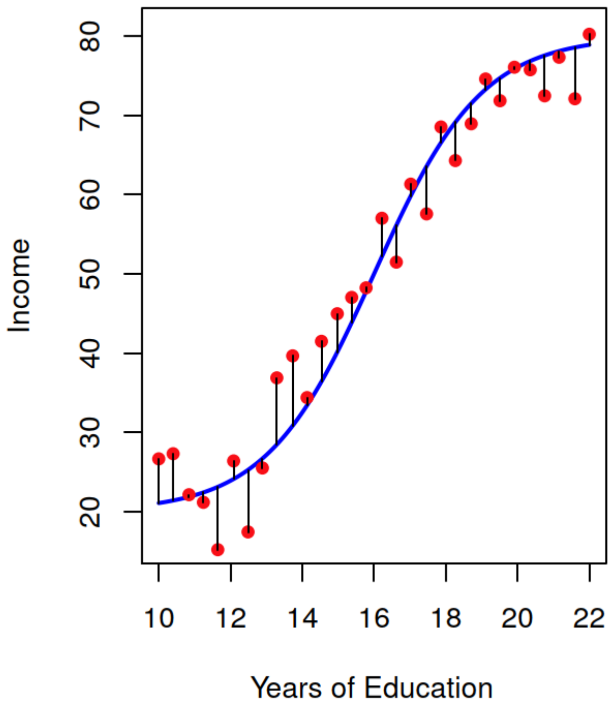
  - Y is usually a scalar in our example; if we have n observations, it can be written as

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

What is $f(X)$ good for?

- With a good f, we can make predictions of Y at new points $X = x$
  - We can understand which components of $X = (X_1, X_2, \ldots, X_p)$ are important in explaining Y and which are irrelevant. e.g., *Seniority* and *Years of Education* have a big impact on *Income*, but *Marital Status* typically does not
  - Depending on the complexity of f, we may be able to understand how each component X_j of X affects Y
- In essence, statistical learning refers to *a set of approaches for estimating f*





Why estimating f

- Prediction: In many situations, a set of inputs X are readily available, but the output Y cannot be easily obtained; we can then use $\hat{f}$ as follows

$$\hat{Y} = \hat{f}(X)$$

  - In this setting, $\hat{f}(X)$ is often treated as a black box
  - There will be reducible and irreducible error
    - *Reducible error* can be potentially improved by using the most appropriate statistical learning technique to estimate f
    - *Irreducible error* may contain unmeasured variables that are useful in predicting Y: since we don't measure them, f cannot use them for its prediction. It may also include unmeasurable variation
  - We will focus on the part of the reducible error

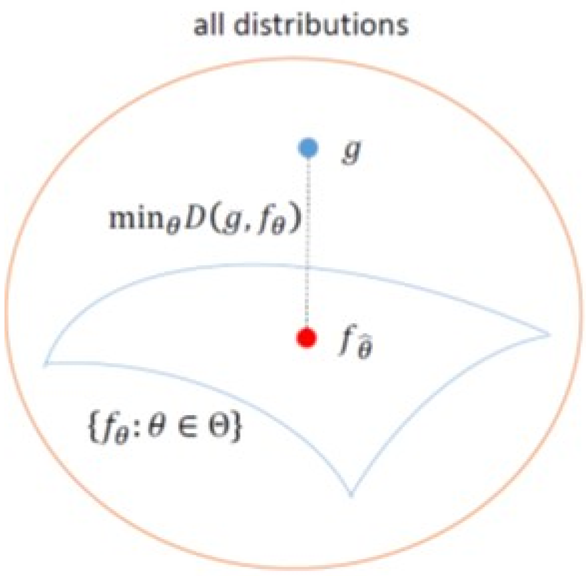
Why estimating f

- Inference: We are often interested in understanding the association between Y and $X_1, \ldots, X_p$. In this situation, we wish to estimate f, but our goal is not necessarily to make predictions for Y
  - Which predictors are associated with the response?
  - What is the relationship between the response and each predictor?
  - Can the relationship between Y and each predictor be adequately summarized using a linear equation, or is the relationship more complicated?
- We will see a number of examples that fall into the prediction setting, the inference setting, or a combination of the two

How to estimating f

- g is the distribution of data that is unknown
  - We have *training set* $\{(x_1, y_1), \dots, (x_n, y_n)\}$

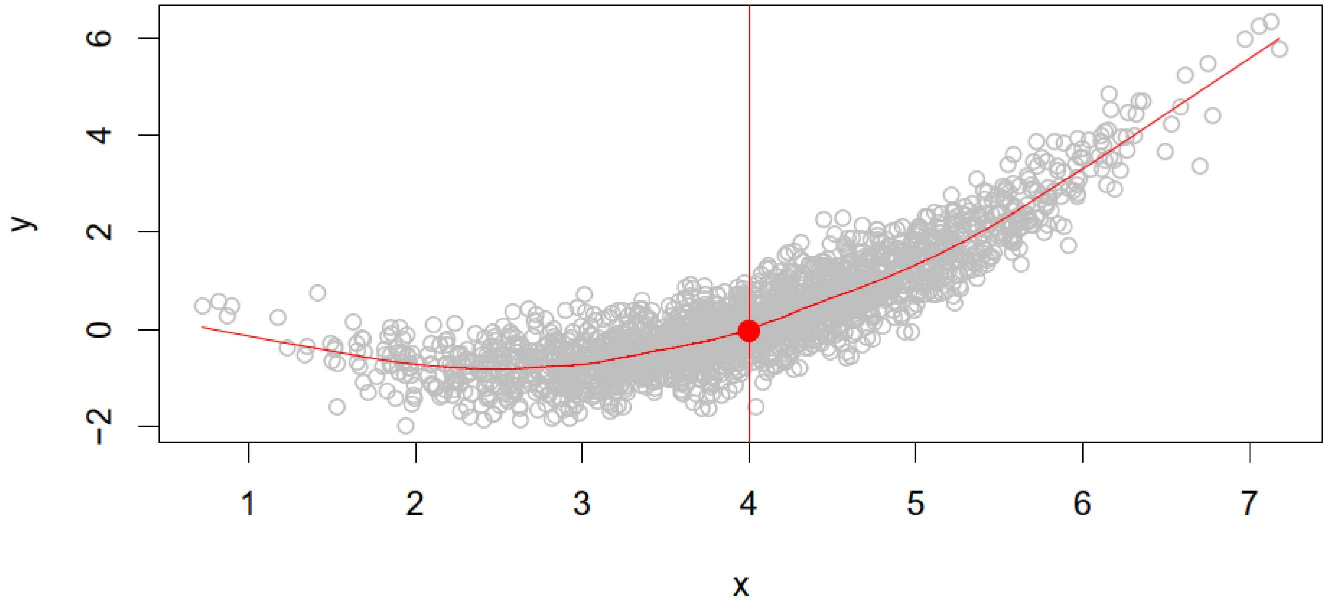
1. Choose a model f_θ
   - Parametric
     - Explicit assumption
     - Estimating a fixed set of parameters by *fitting* or *training*
   - Non-parametric
     - No explicit assumption
     - Need a large number of observations
2. Choose a quality measure (objective function) for fitting
   - Mean square error (Likelihood)…
3. Optimization (fitting) to choose the best θ
   - Calculus to find close form solution, gradient descent, expectation-maximization…



Supervised vs Unsupervised learning

- Supervised Learning problem
  - In the regression problem, Y is quantitative (e.g., price, blood pressure)
  - In the classification problem, Y takes values in a finite, unordered set (survived/died, digit 0-9, cancer class of tissue sample)
  - We have training data $(x_1, y_1), \ldots, (x_n, y_n)$. These are observations (examples, instances) of these measurements
- Unsupervised Learning problem
  - No outcome variable, just a set of predictors (features) measured on a set of samples
  - Objective is fuzzier - find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation
- Semi-supervised learning problem
  - Only for m of the observations $(m < n)$ that we have the response

The regression problem



- Is there an ideal $f(X)$? In particular, what is a good value for $f(X)$ at any selected value of X, say $X = 4$? There can be many Y values at $X = 4$. A good value is

$$f(4) = E(Y|X = 4)$$

  - $E(Y|X = 4)$ means the expected value (average) of Y given $X = 4$. This ideal $f(x) = E(Y|X = x)$ is called the regression function

The regression function $f(x)$

- Also defined for vector X; e.g.

$$f(x) = f(x_1 x_2, x_2) = E(Y\,|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$

  - The ideal or optimal predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y|X = x)$ is the function that minimizes $E[(Y - f(X))^2|X = x]$ over all functions f at all points $X = x$
  - $\epsilon = Y - f(x)$ is the irreducible error — i.e., 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

- For any estimate $\hat{f}(x)$ of $f(x)$, we have

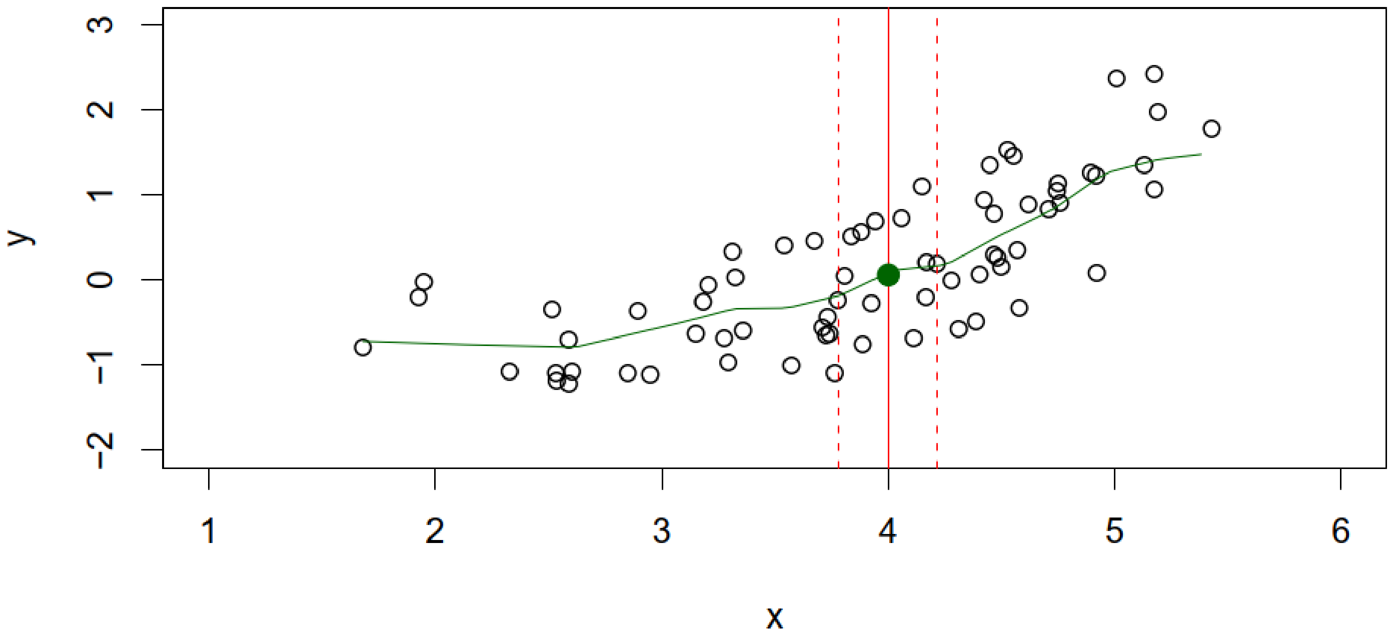
$$E\left[\left(Y - \hat{f}(x)\right)^2 \middle| X = x\right] = E[f(x) + \epsilon - \hat{f}(x)]^2 = [f(x) - \hat{f}(x)]^2 + Var(\epsilon)$$

How to estimate f

- Typically, we have few if any data points with $X = 4$ exactly
  - So that we cannot compute $E(Y|X = x)$!
  - Relax the definition and let

$$\hat{f}(x) = Ave(Y|X \in N(x))$$

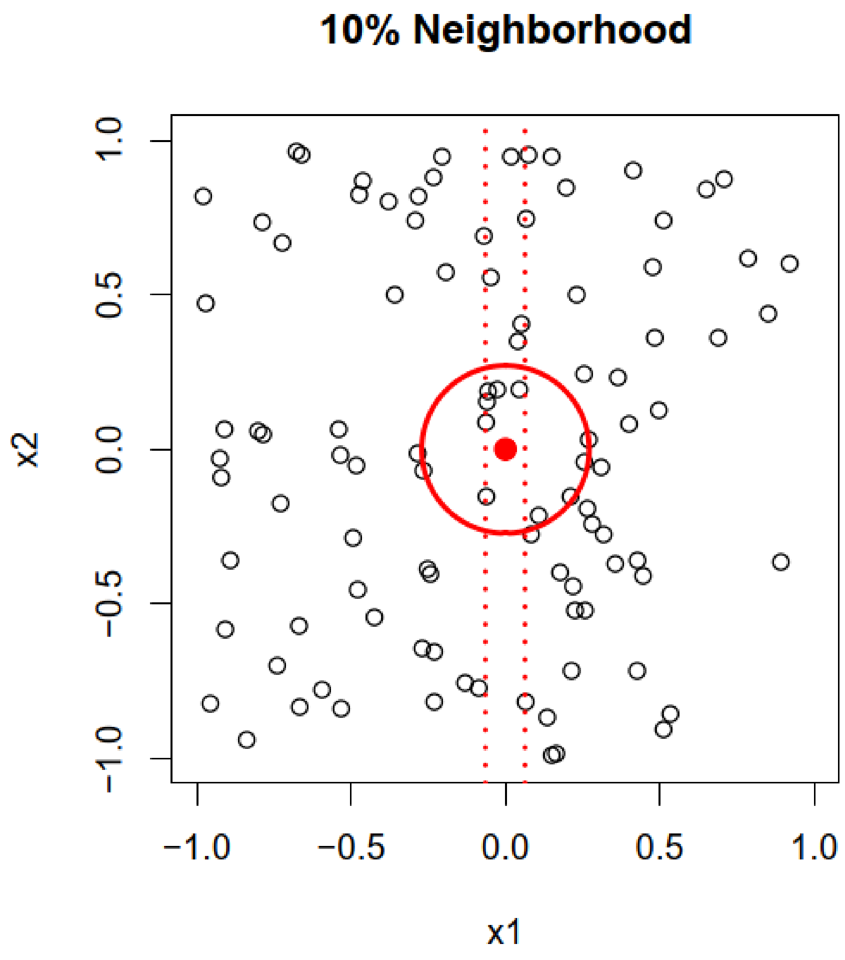
where $N(x)$ is some neighborhood of x.

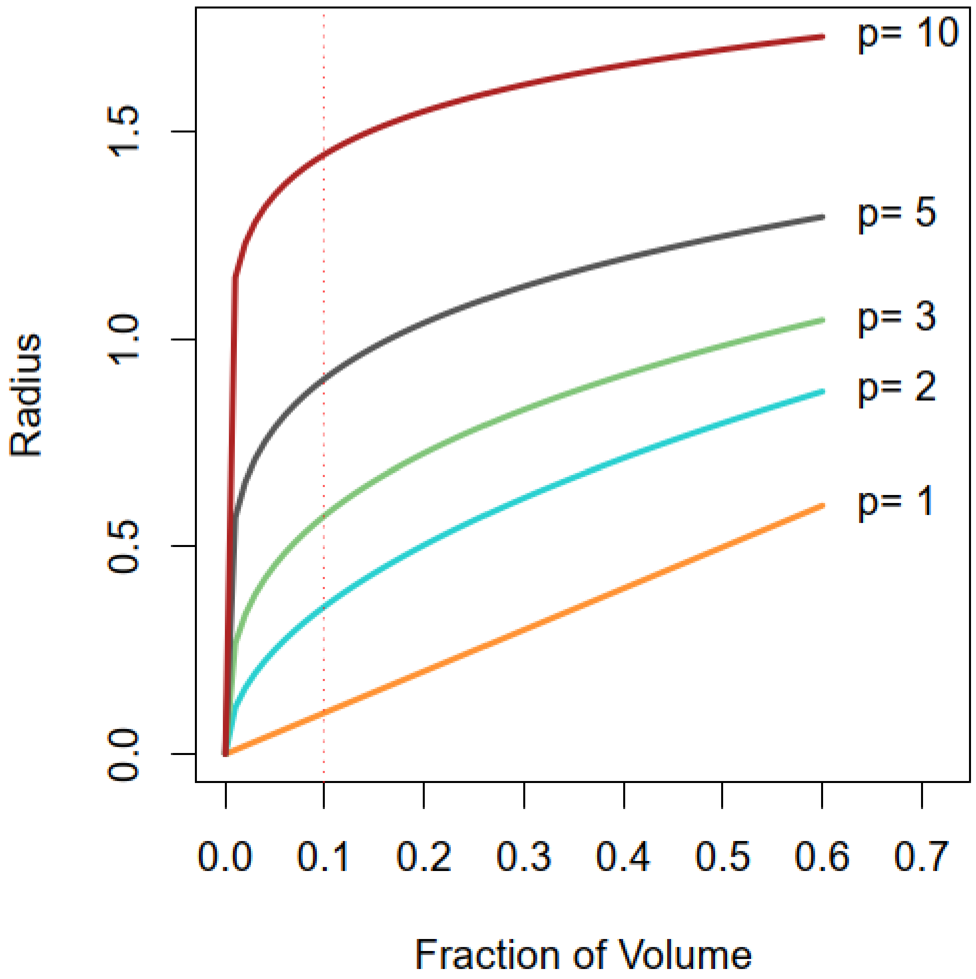


The curse of dimensionality

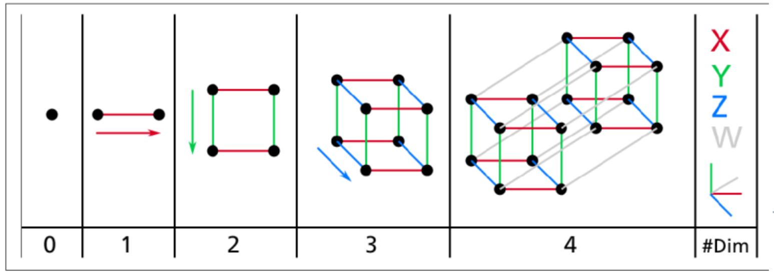
- Nearest neighbor averaging can be good for small p ($p \leq 4$) and large n
  - We will discuss smoother versions, such as kernel and spline smoothing later in the course
- Nearest neighbor methods can be lousy when p is large. Reason: the <u>curse of dimensionality</u>. Nearest neighbors tend to be far away in high dimensions.
  - We need to get a reasonable fraction of the n values of y_i to average to bring the variance down — e.g., 10%
  - A 10% neighborhood in high dimensions need no longer be local, so we lose the spirit of estimating $E(Y|X = x)$ by local averaging

The curse of dimensionality





The curse of dimensionality



https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

p	1	2	3	4	5	6
(a) Ball with radius R	2R	πR^2	$\frac{4}{3}\pi R^3$	$\frac{\pi^2}{2}R^4$	$\frac{8\pi^2}{15}R^5$	$\frac{\pi^3}{6}R^6$
(b) Volume of hypercube 2^p	2	4	8	16	32	64
$r=(a)/(b)$	R	$\frac{\pi R^2}{4}$	$\frac{\pi R^3}{6}$	$\frac{\pi^2 R^4}{32}$	$\frac{\pi^2 R^5}{60}$	$\frac{\pi^3 R^6}{384}$

$r=\frac{\pi^{\frac{p}{2}}}{2^p\Gamma(\frac{p}{2}+1)}R^p$, it turns out that if we want to cover a fraction of r of the hypercube, we will need a ball with a radius $\frac{2}{\pi^{\frac{1}{2}}}[r\Gamma(\frac{p}{2}+1)]^{\frac{1}{p}}$ (note that $\Gamma(\frac{p}{2}+1)\sim\sqrt{\pi p}\left(\frac{p}{2e}\right)^{\frac{p}{2}}$)

See chapter 2 of Foundations of Data Science

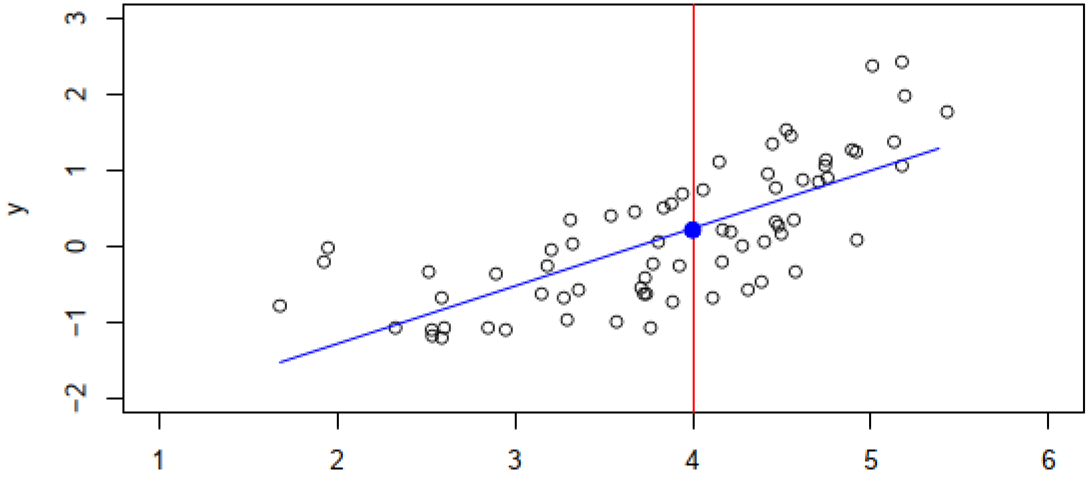
Parametric and structured models

- The linear model is an important example of a parametric model:

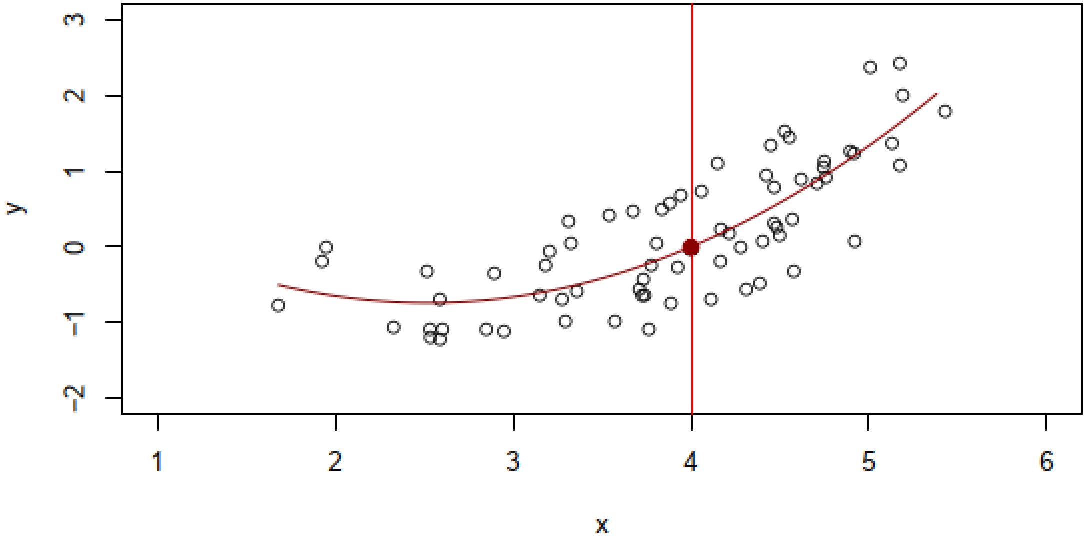
$$f_L(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

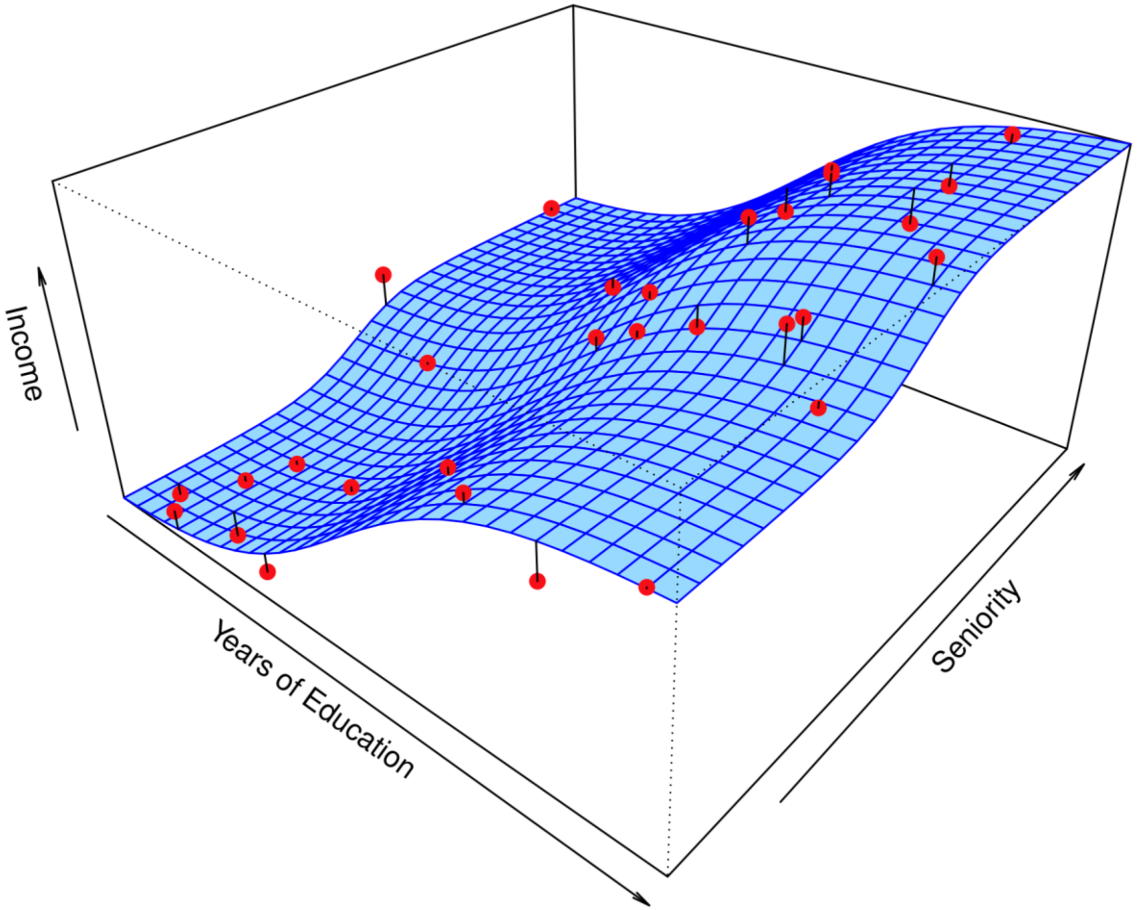
  - A linear model is specified in terms of $p + 1$ parameters $\beta_0, \beta_1, \ldots, \beta_p$
  - We estimate the parameters by fitting the model to training data
  - Although it is almost never correct, a linear model often serves as a good and interpretable approximation to the unknown true function $f(X)$

- A linear model $f_L(X) = \beta_0 + \beta_1 X$ gives a reasonable fit here



- A quadratic model $f_Q(X) = \beta_0 + \beta_1 X + \beta_2 X^2$ fits slightly better

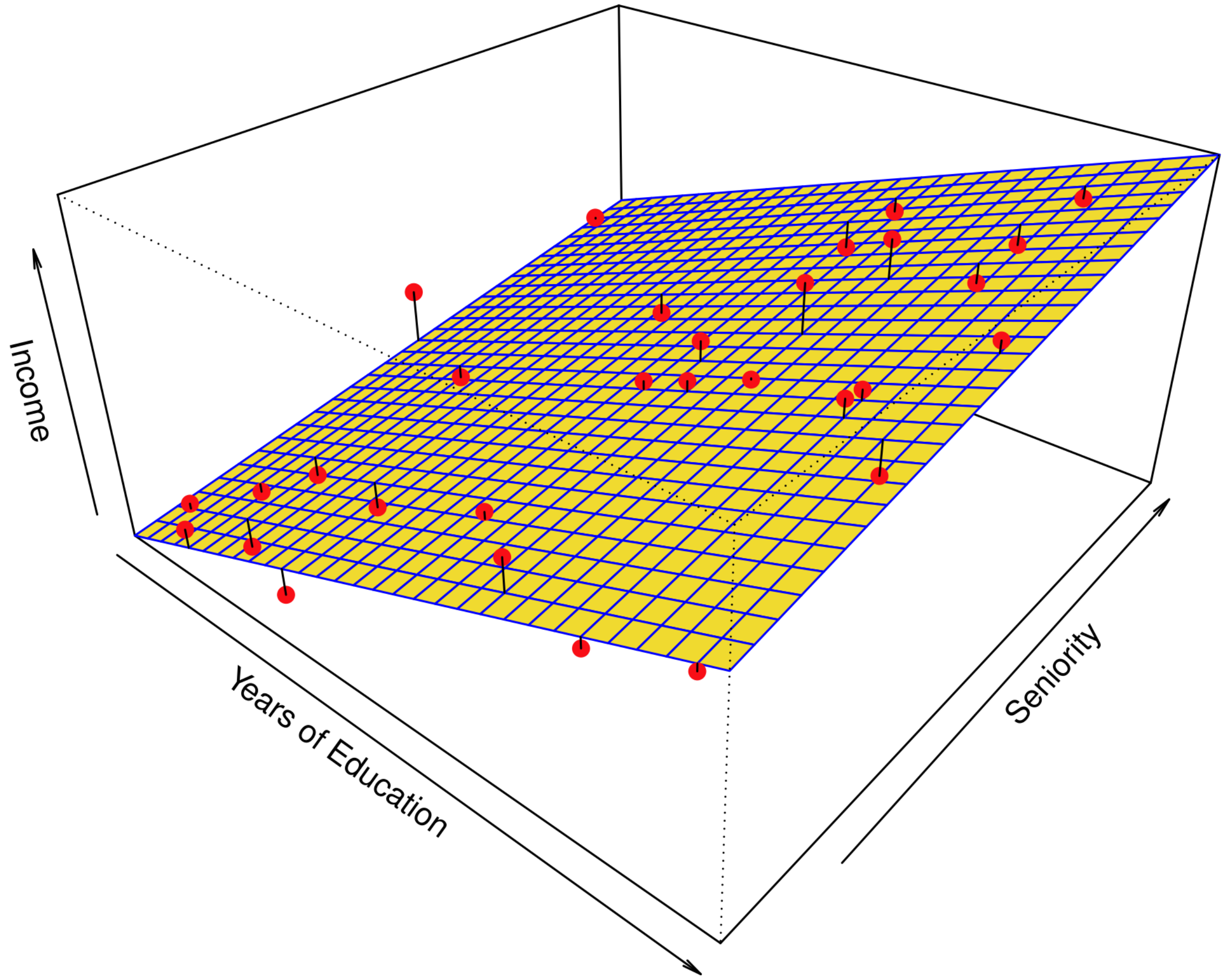




- Simulated example. Red points are simulated values for income from the model

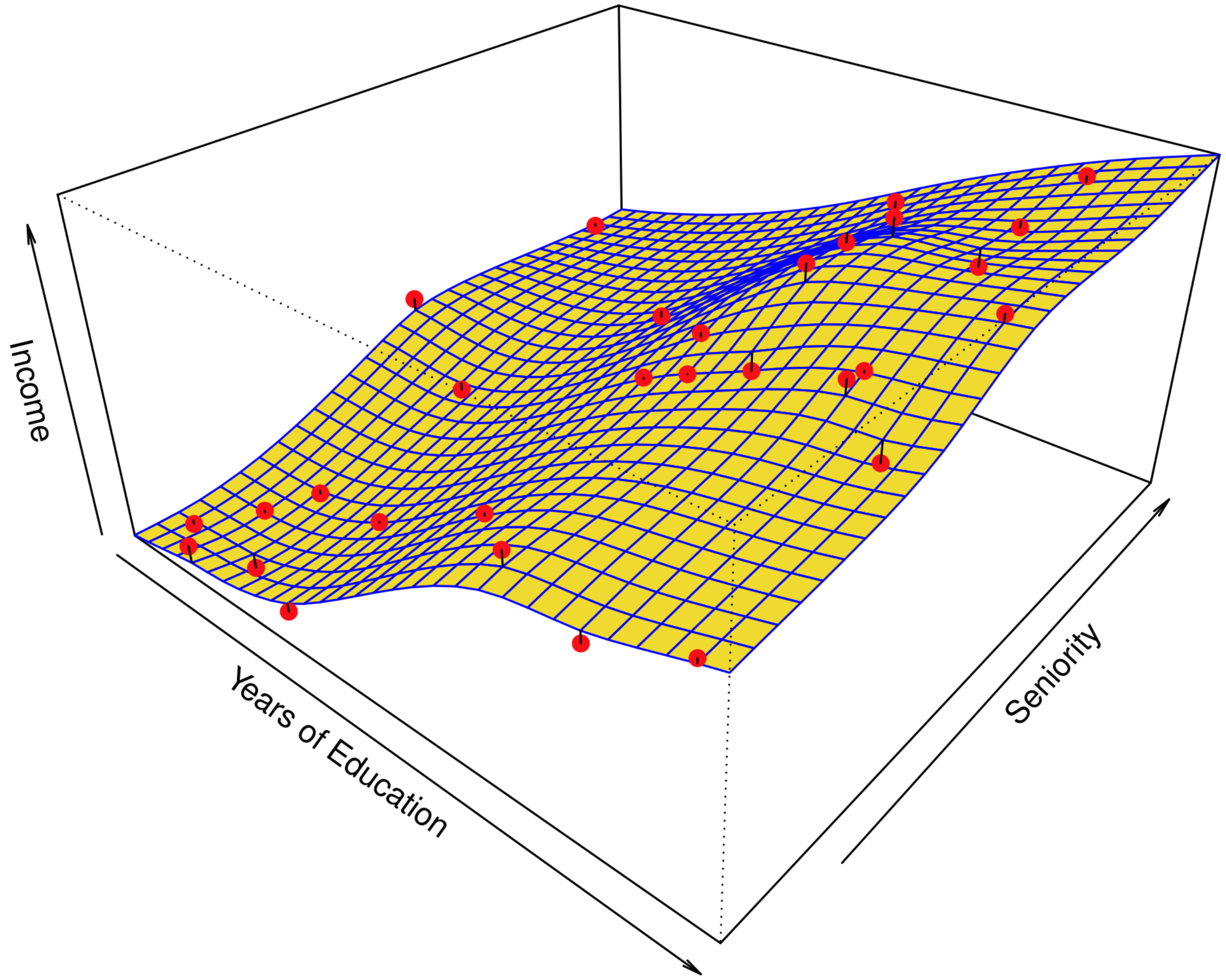
$$income = f(education, seniority) + \epsilon$$

f is the blue surface

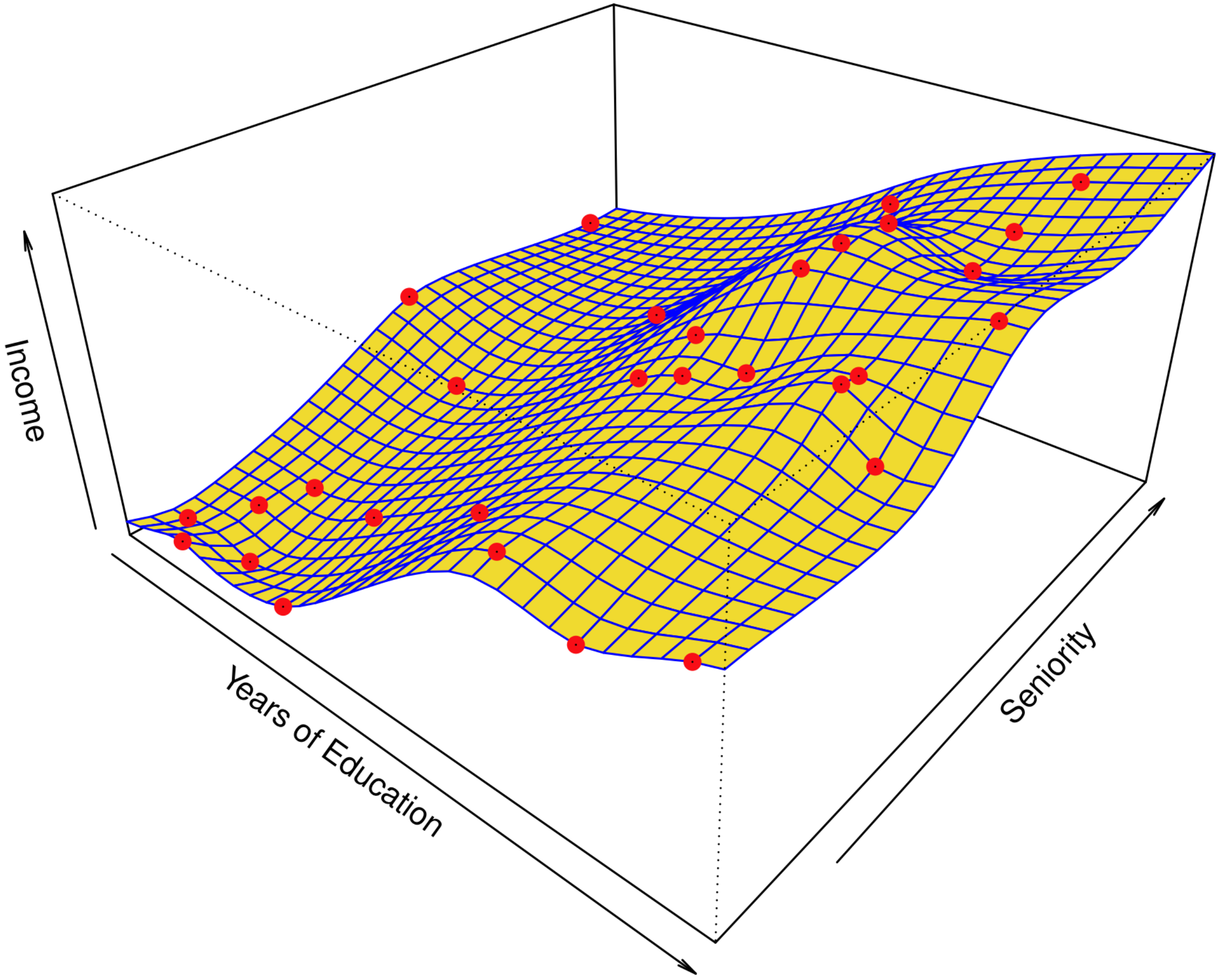


- Linear regression model fit to the simulated data

$$\hat{f}_L(education, seniority) = \hat{\beta}_0 + \hat{\beta}_1 \times education + \hat{\beta}_2 \times seniority$$



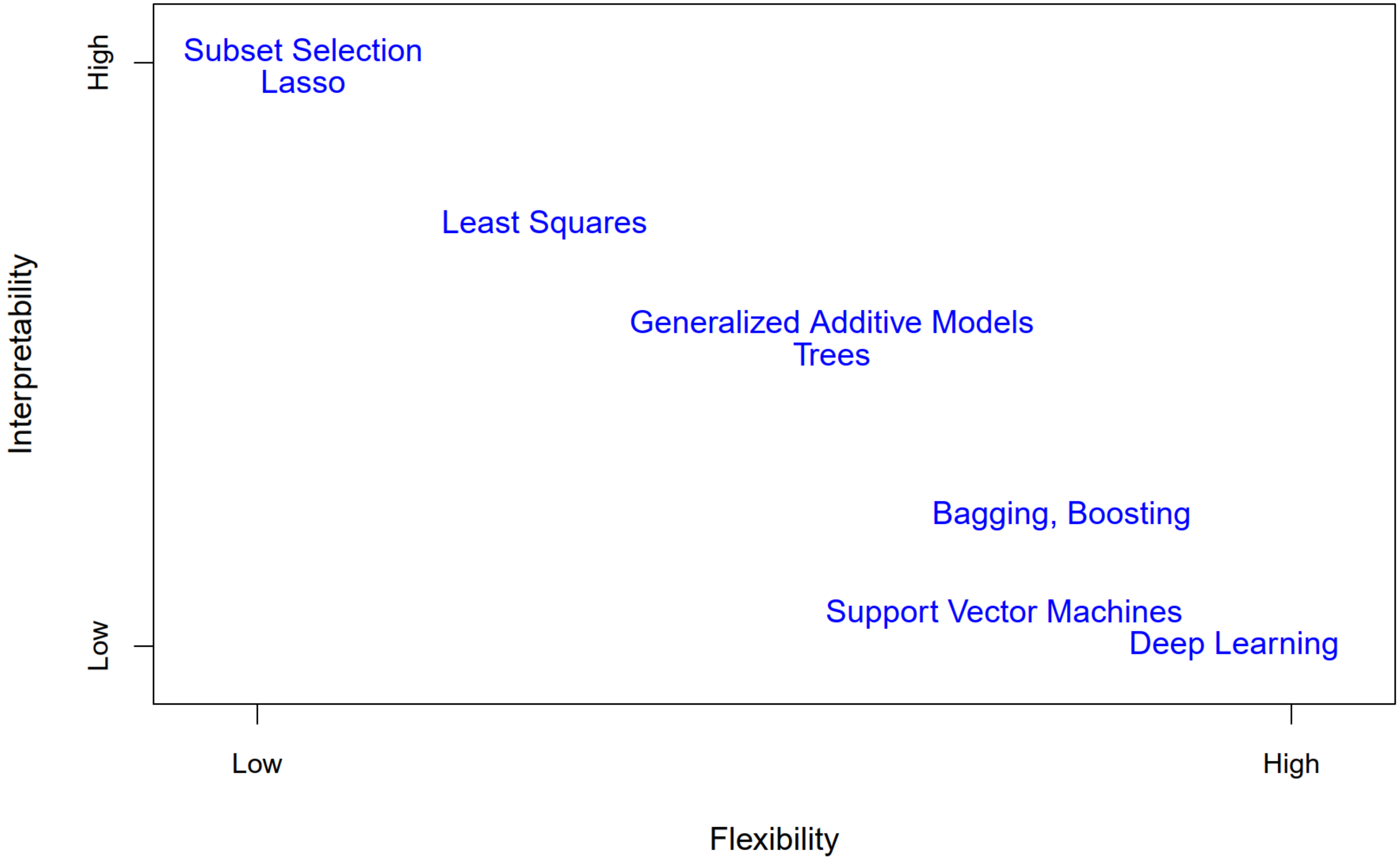
- More flexible regression model $\hat{f}_S(education, seniority)$ fit to the simulated data. Here we use a technique called a *thin-plate spline* to fit a flexible surface. We control the roughness of the fit (chapter 7)



- Even more flexible *spline regression model* $\hat{f}_s(education, seniority)$ fit to the simulated data. Here the fitted model makes no errors on the training data! Also known as *overfitting*

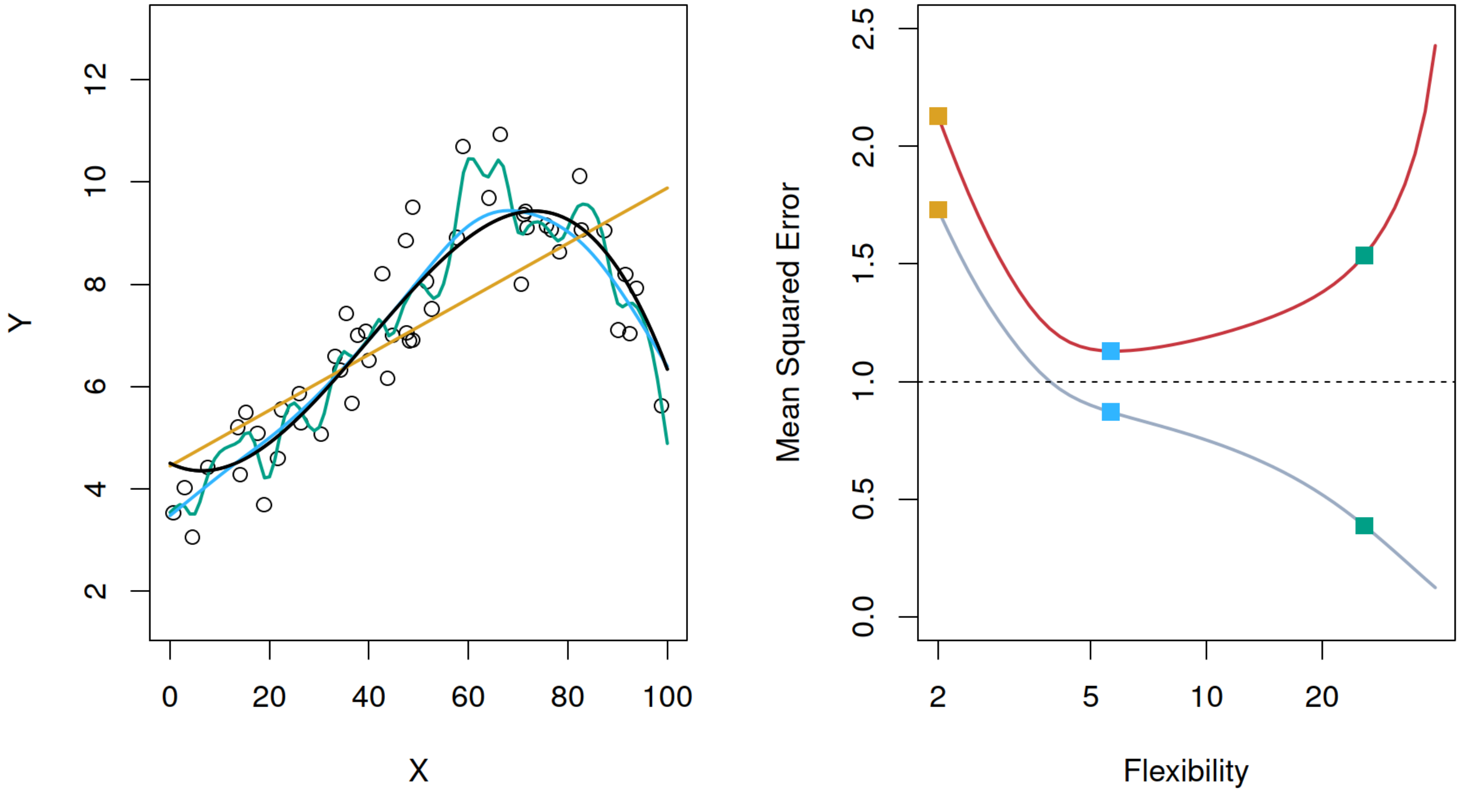
Some trade-offs

- Prediction accuracy versus interpretability
  - Linear models are easy to interpret; thin-plate splines are not
- Good fit versus over-fit or under-fit
  - How do we know when the fit is just right?
- Parsimony versus black-box
  - We often prefer a simpler model involving fewer variables over a black-box predictor involving them all

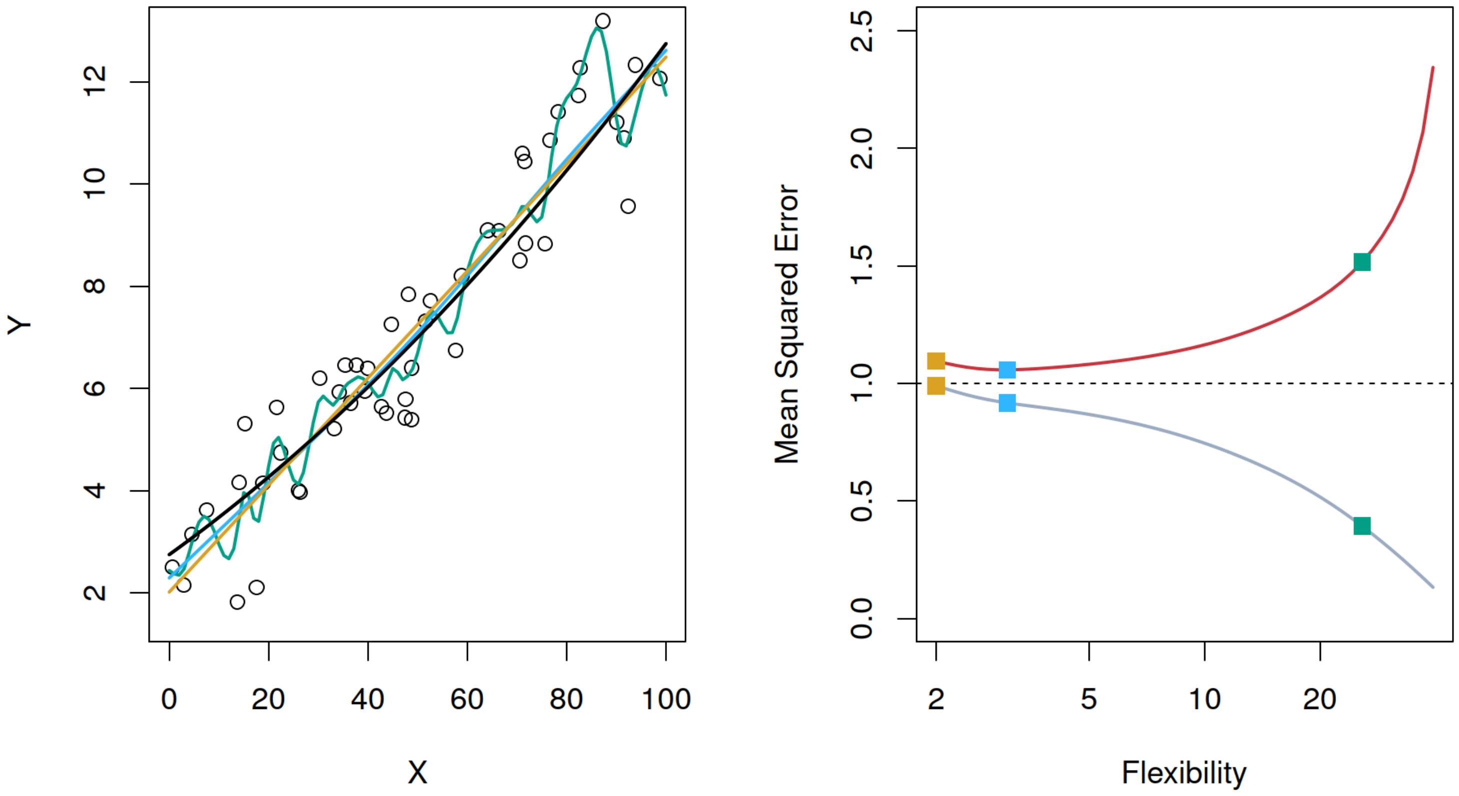

Subset Selection
Lasso
Least Squares
Generalized Additive Models
Trees
Bagging, Boosting
Support Vector Machines
Deep Learning
Interpretability
High
Low
Flexibility
Low
High

Assessing Model Accuracy

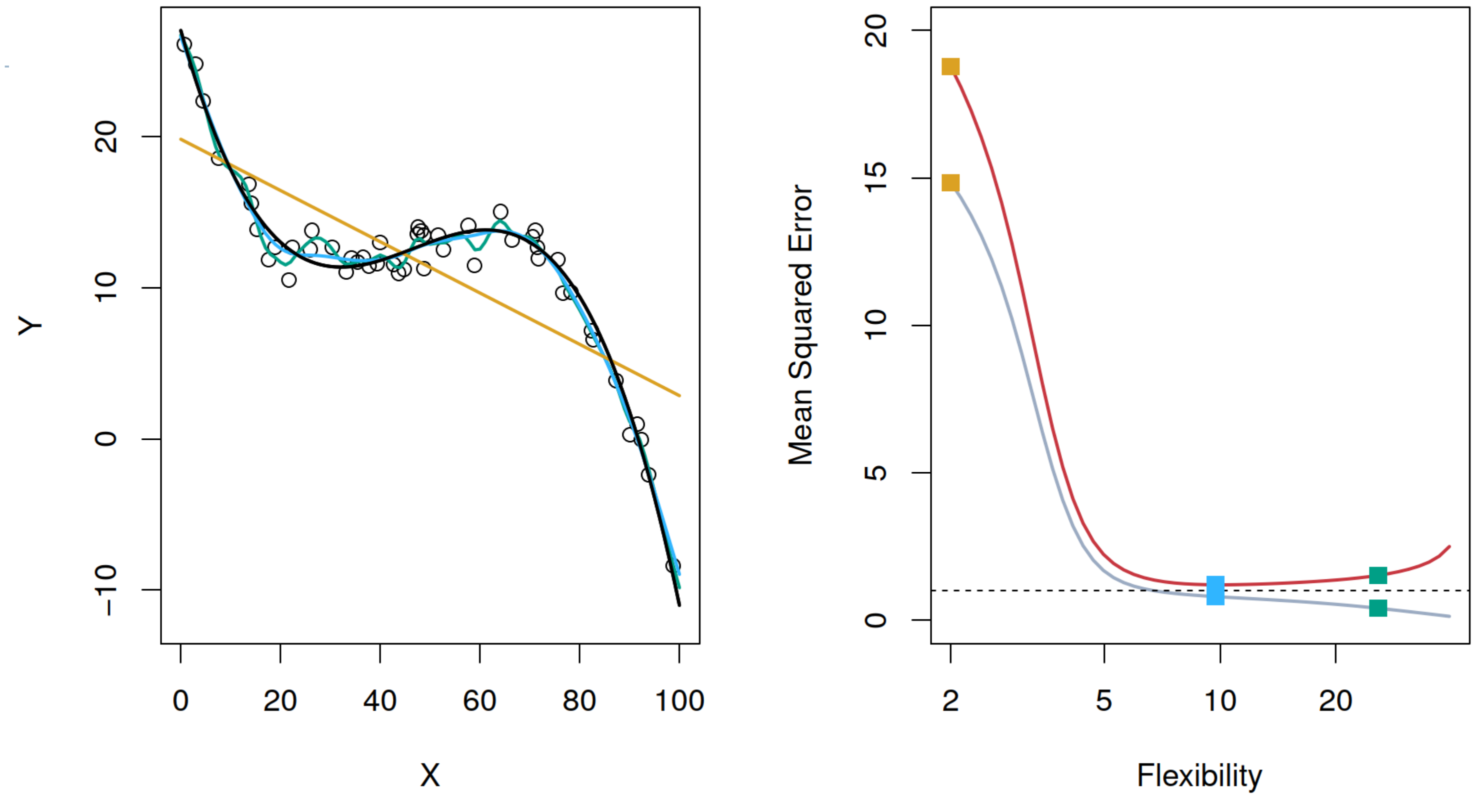
- Suppose we fit a model $\hat{f}(x)$ to some training data $Tr = \{x_i, y_i\}, i = 1 \dots n$, and we wish to see how well it performs
  - We could compute the average squared prediction error over Tr:
    $$MSE_{Tr} = Ave_{i \in Tr}[y_i - \hat{f}(x_i)]^2$$
- This may be biased toward more overfit models
  - Instead, we should, if possible, compute it using fresh test data $Te = \{x_i, y_i\}, i = 1 \dots m$,
    $$MSE_{Te} = Ave_{i \in Te}[y_i - \hat{f}(x_i)]^2$$



- The black curve is truth. Red curve on the right is MSE_{Te}, grey curve is MSE_{Tr}. Orange, blue and green curves/squares correspond to fits of different flexibility



- Here, the truth is smoother, so the smoother fit and linear model do really well



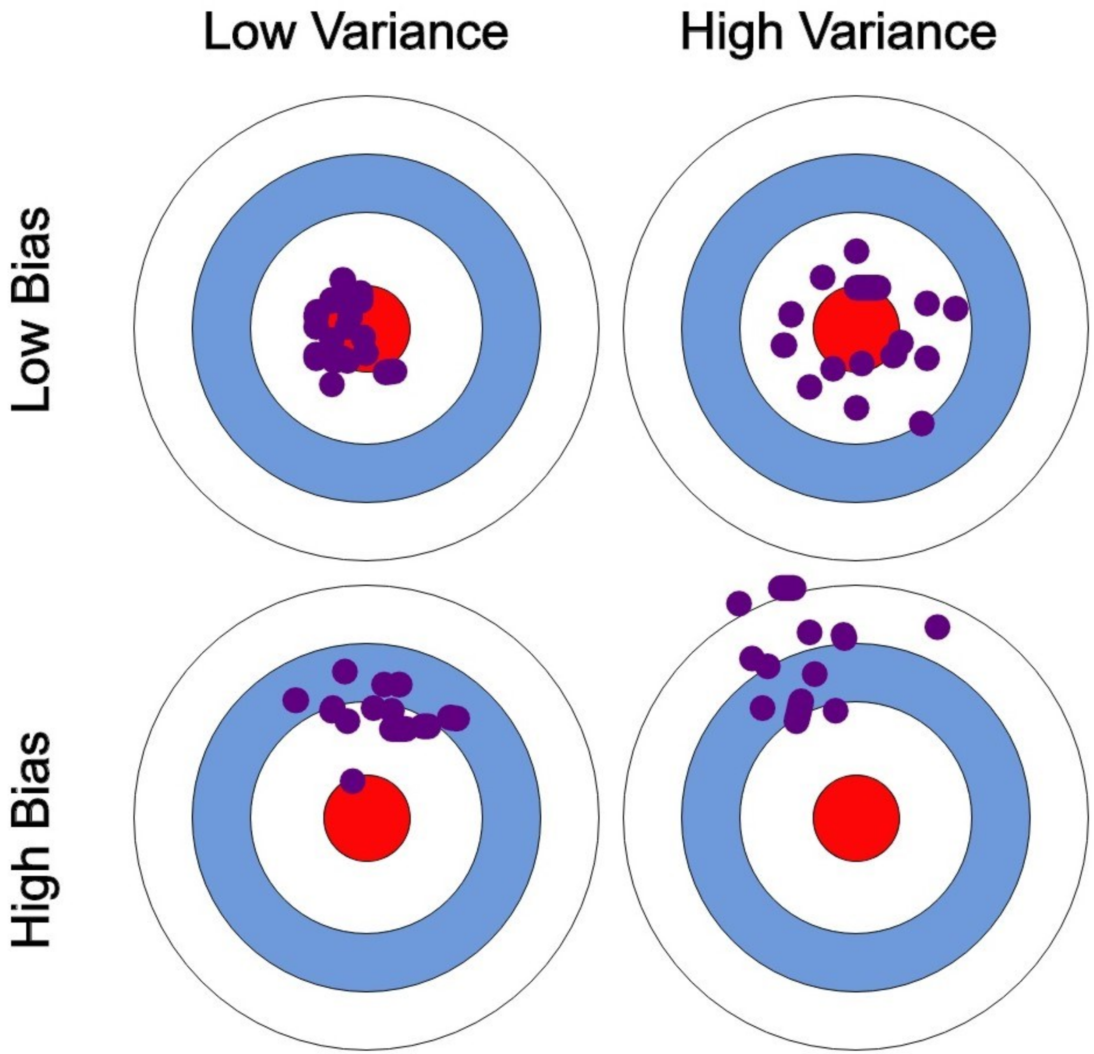
- Here, the truth is wiggly and the noise is low, so the more flexible fits do the best
  - [Proof of testing error is usually larger than the training error]()

Bias-Variance Trade-off

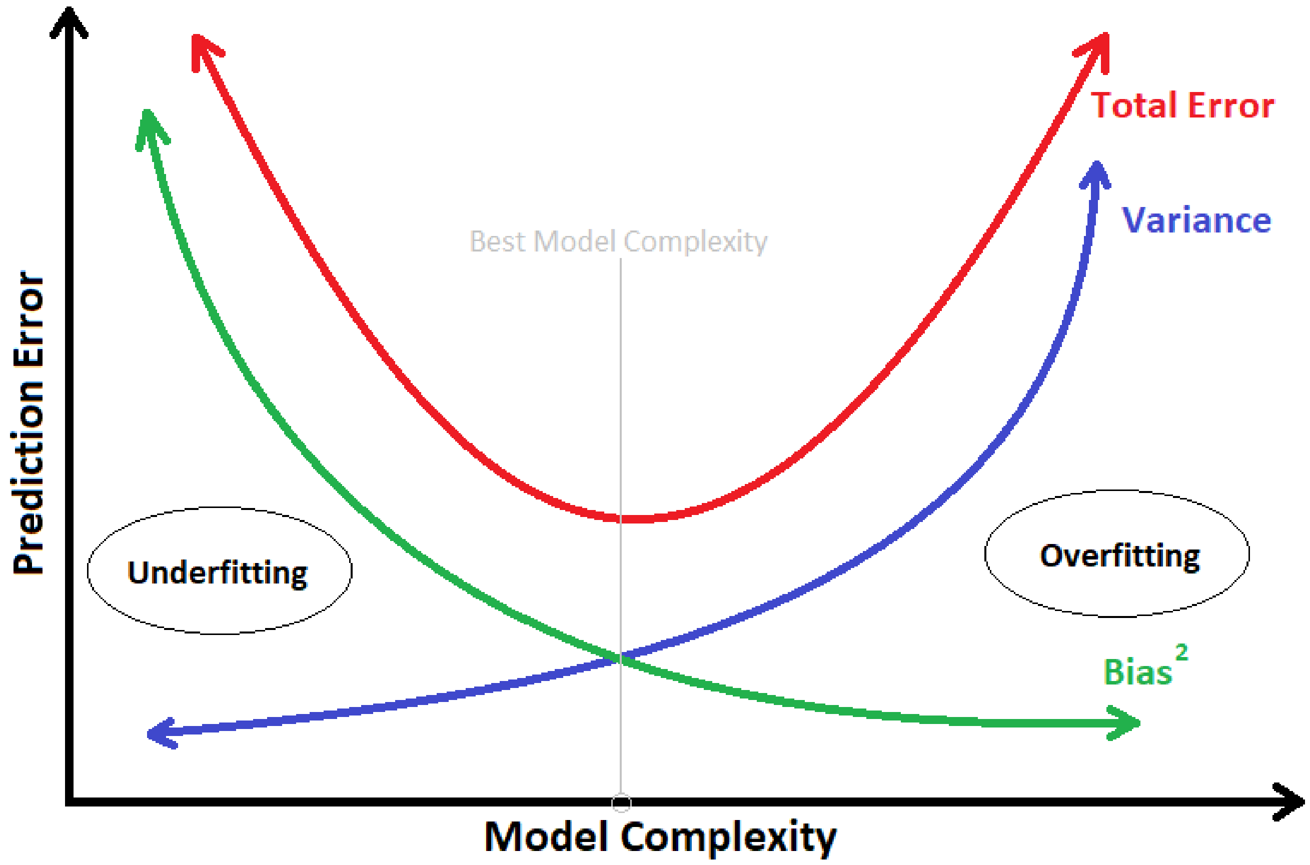
- Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \epsilon$ (with $f(x) = E(Y|X = x)$), then

$$E\left[\left(y_0 - \hat{f}(x_0)\right)^2\right] = Bias_{Tr}[\hat{f}(x_0, Tr)]^2 + Var_{Tr}\left[\hat{f}(x_0, Tr)\right] + \mathrm{Var}(\epsilon)$$

- The expectation averages over the variability of y_0 as well as the variability in Tr. Note that $Bias_{Tr}\left[\hat{f}(x_0, Tr)\right] = E\left[\hat{f}(x_0, Tr)\right] - f(x_0)$
  - Typically, as the *flexibility* of $\hat{f}$ increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off*
  - Proof of the decomposition

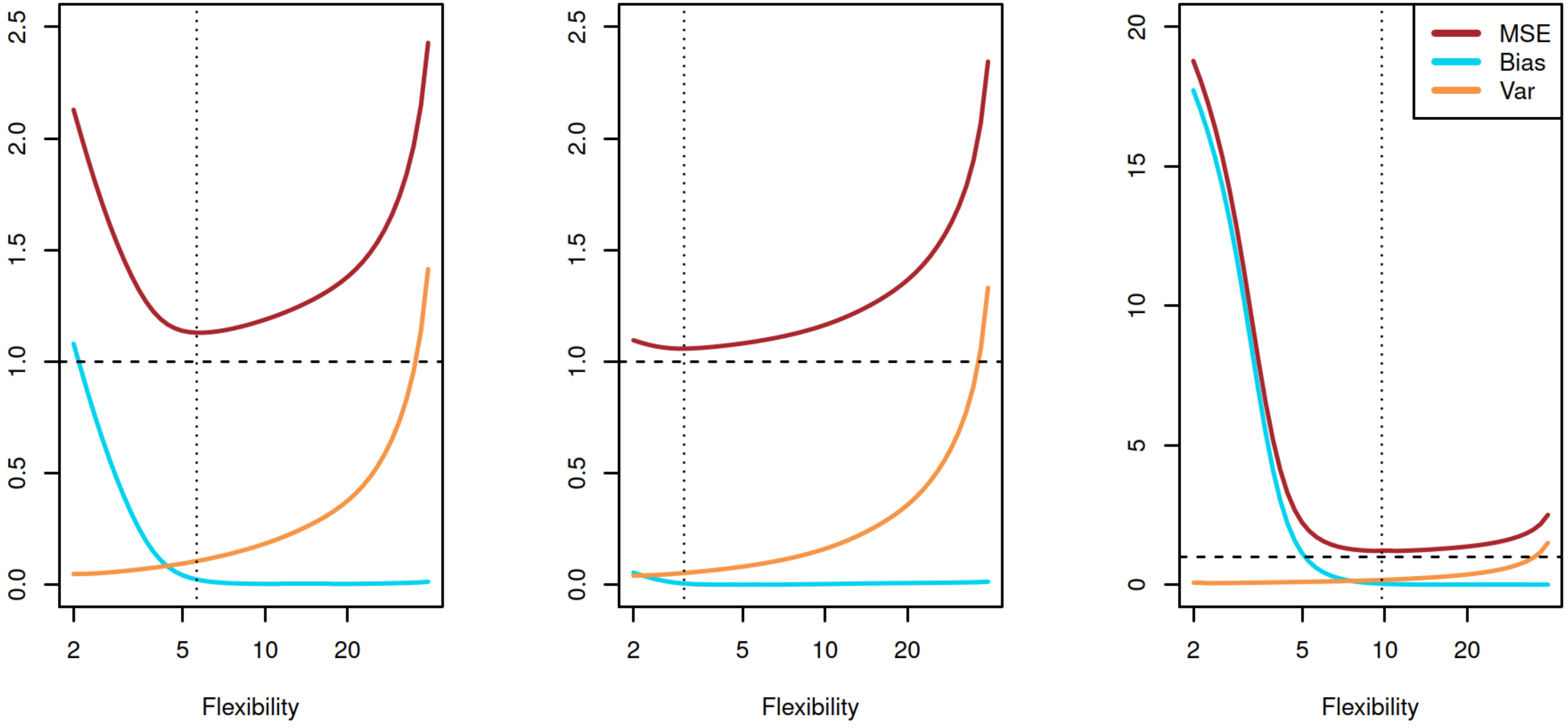


https://nvsyashwanth.github.io/machinelearningmaster/bias-variance/



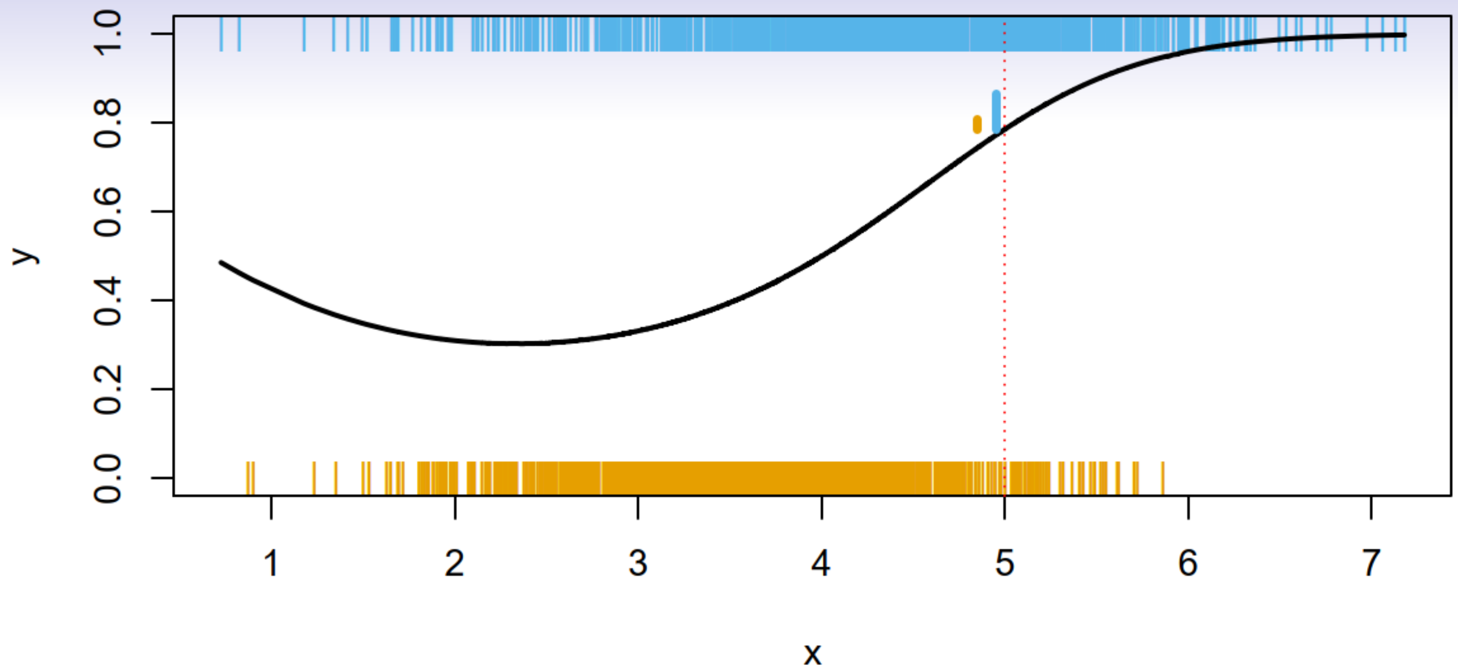
https://jason-chen-1992.weebly.com/home/-bias-variance-tradeoff

Bias-variance trade-off for the three examples



Classification Problems

- Here the response variable Y is qualitative — e.g. email is one of $C = (spam, ham)$ $(ham = good\ email)$, digit class is one of $C = \{0, 1, \ldots, 9\}$. Our goals are to:
  - Build a classifier $C(X)$ that assigns a class label from C to a future unlabeled observation X
  - What is an optimal classifier?
  - Understand how flexibility affects the classification

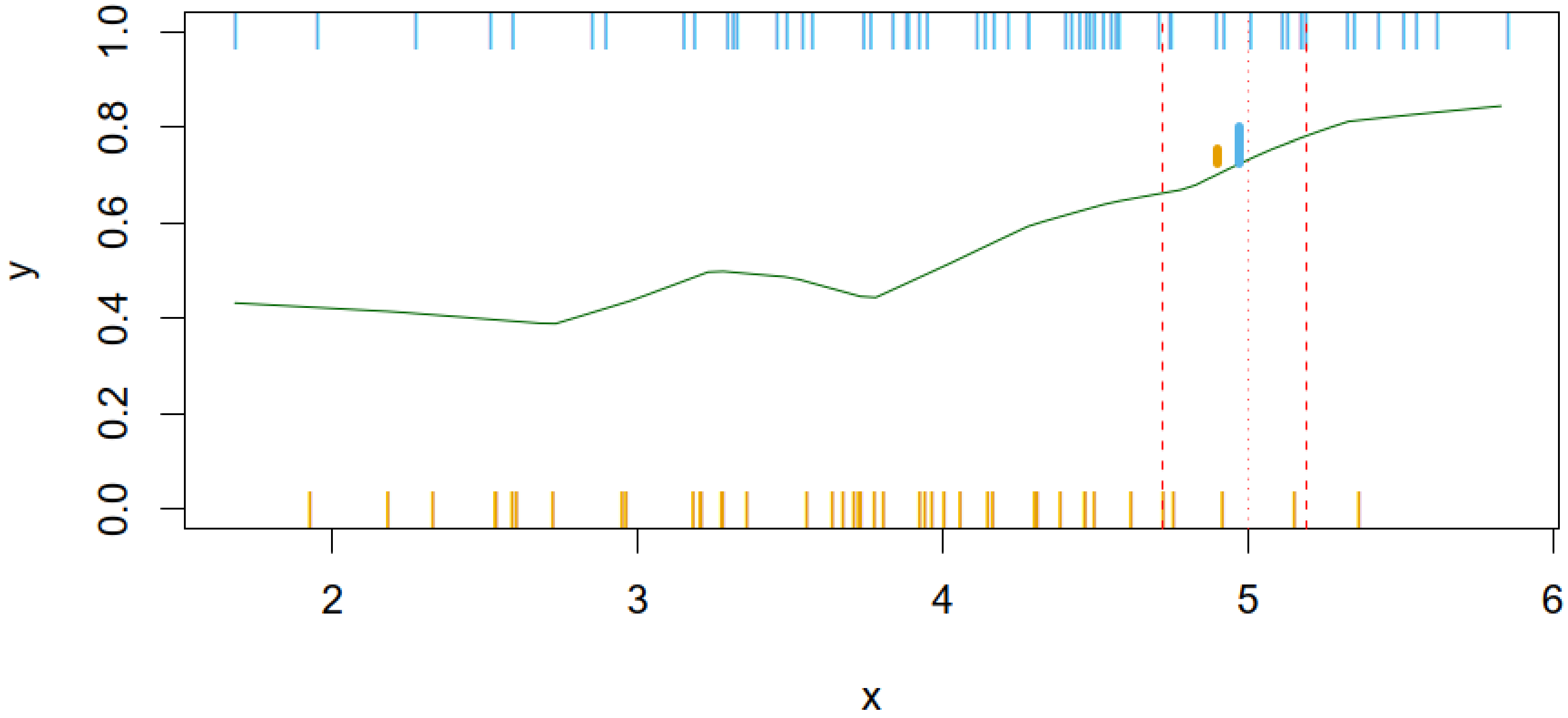


The orange/blue marks indicate the response Y, either 0 or 1

- Is there an ideal $C(X)$? Suppose the K elements in C are numbered $1, 2, \ldots, K$. Let

$$p_k(x) = \Pr(Y = k | X = x), k = 1, 2, \ldots, K.$$

  - These are the *conditional class probabilities* at x; e.g., see the little barplot at $x = 5$. Then the Bayes optimal classifier at x is $C(x) = j$ if $p_j(x) = \max\{p_1(x), p_2(x), \ldots, p_K(x)\}$



- Nearest-neighbor averaging can be used as before. It also breaks down as the dimension grows. However, the impact on $\hat{C}(x)$ is less than on $\hat{p}_k(x), k = 1, \ldots, K$

Classification: some details

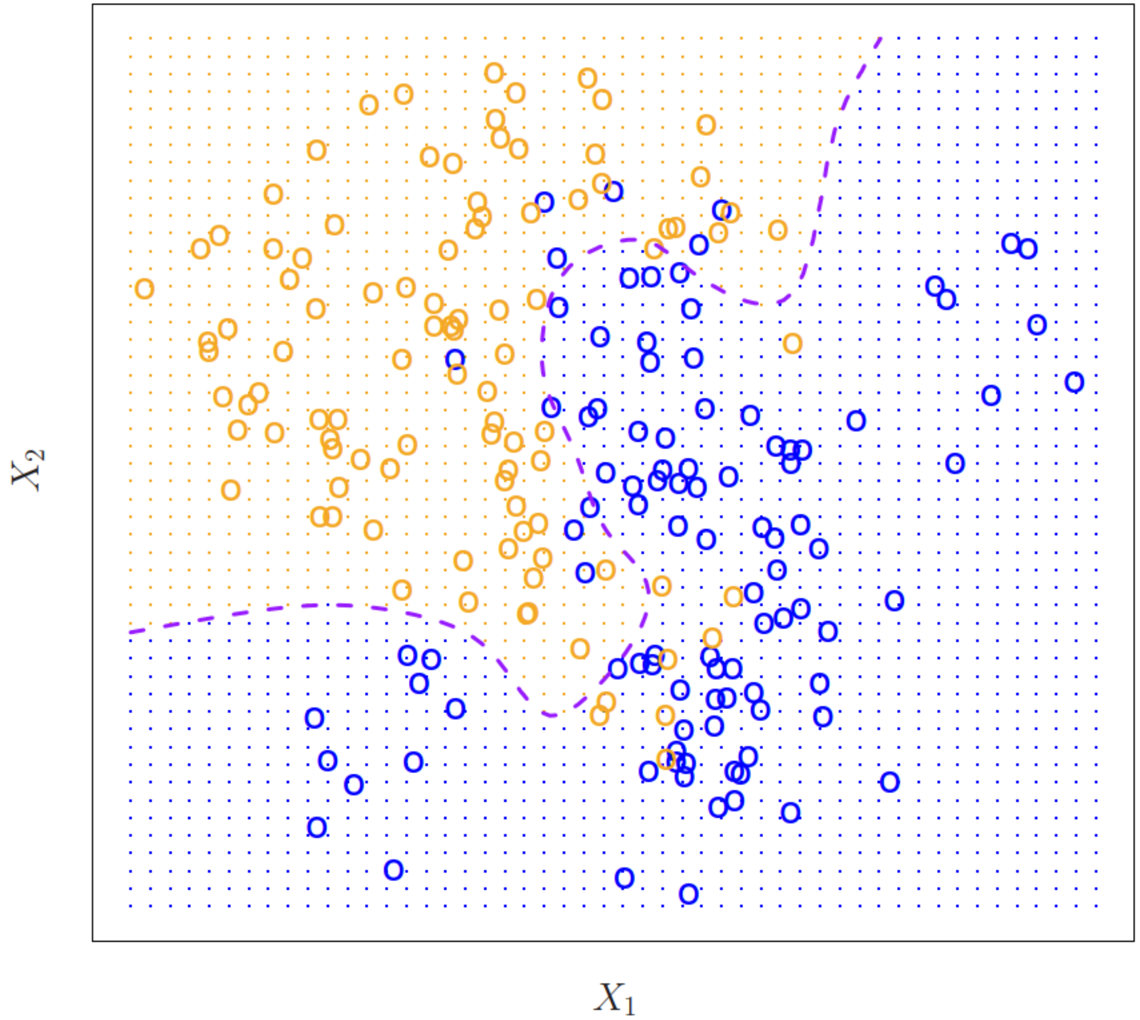
- Typically, we measure the performance of $\hat{C}(x)$ using the misclassification *error rate*:

$$Err_{Te} = Ave_{i \in Te} I[y_i \neq \hat{C}(x_i)]$$

  - The Bayes classifier (using the true $\hat{p}_k(x)$) has the smallest error (in the population)
  - Support-vector machines build structured models for $C(x)$
  - We will also build structured models for representing the $p_k(x)$. e.g., Logistic regression, generalized additive models

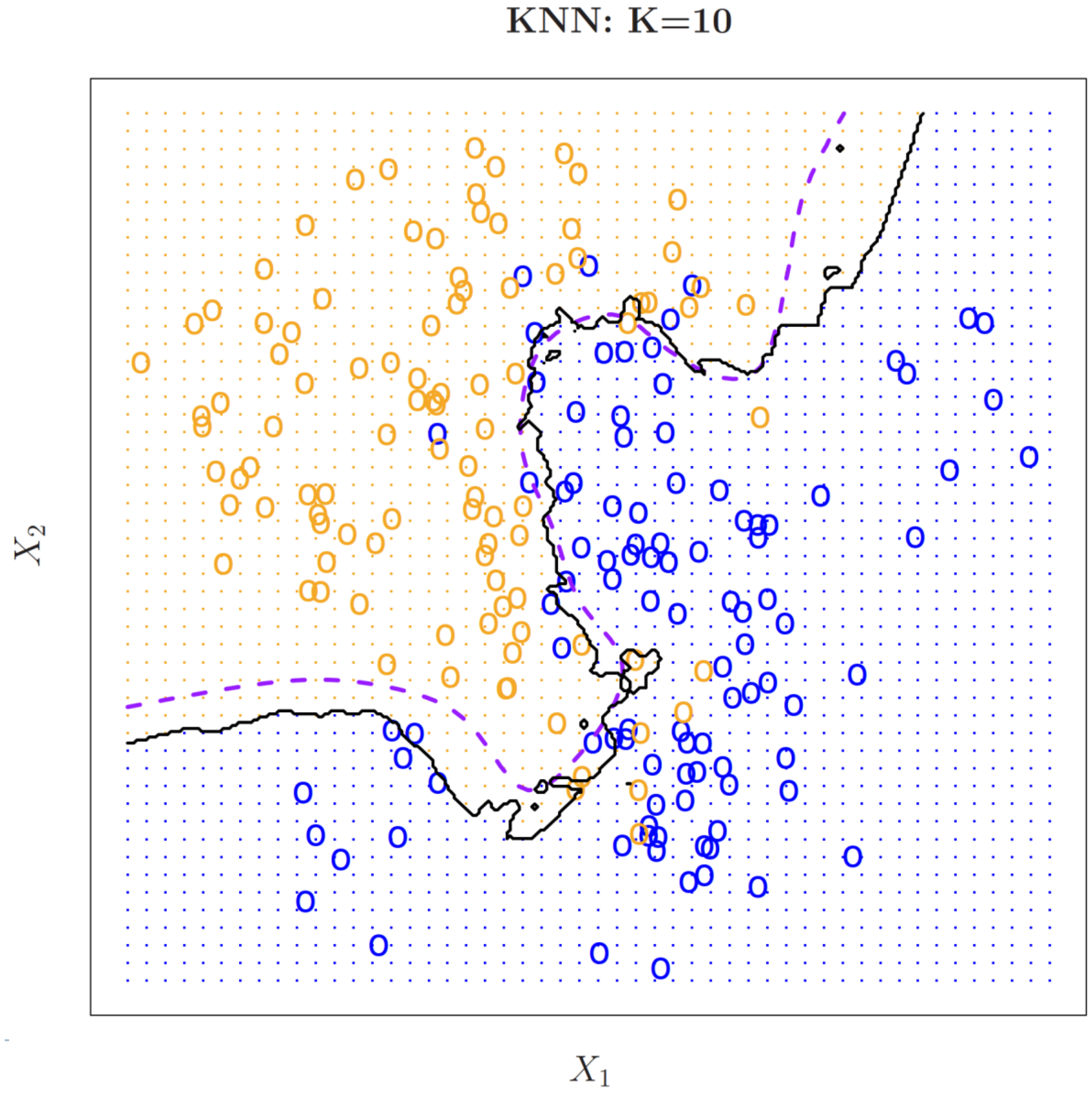
Example: K-nearest neighbors in two dimensions

- The Bayes classifier produces the lowest possible test error rate, called the Bayes error rate
  - $1 - max_j \Pr(Y = j | X = x_0)$

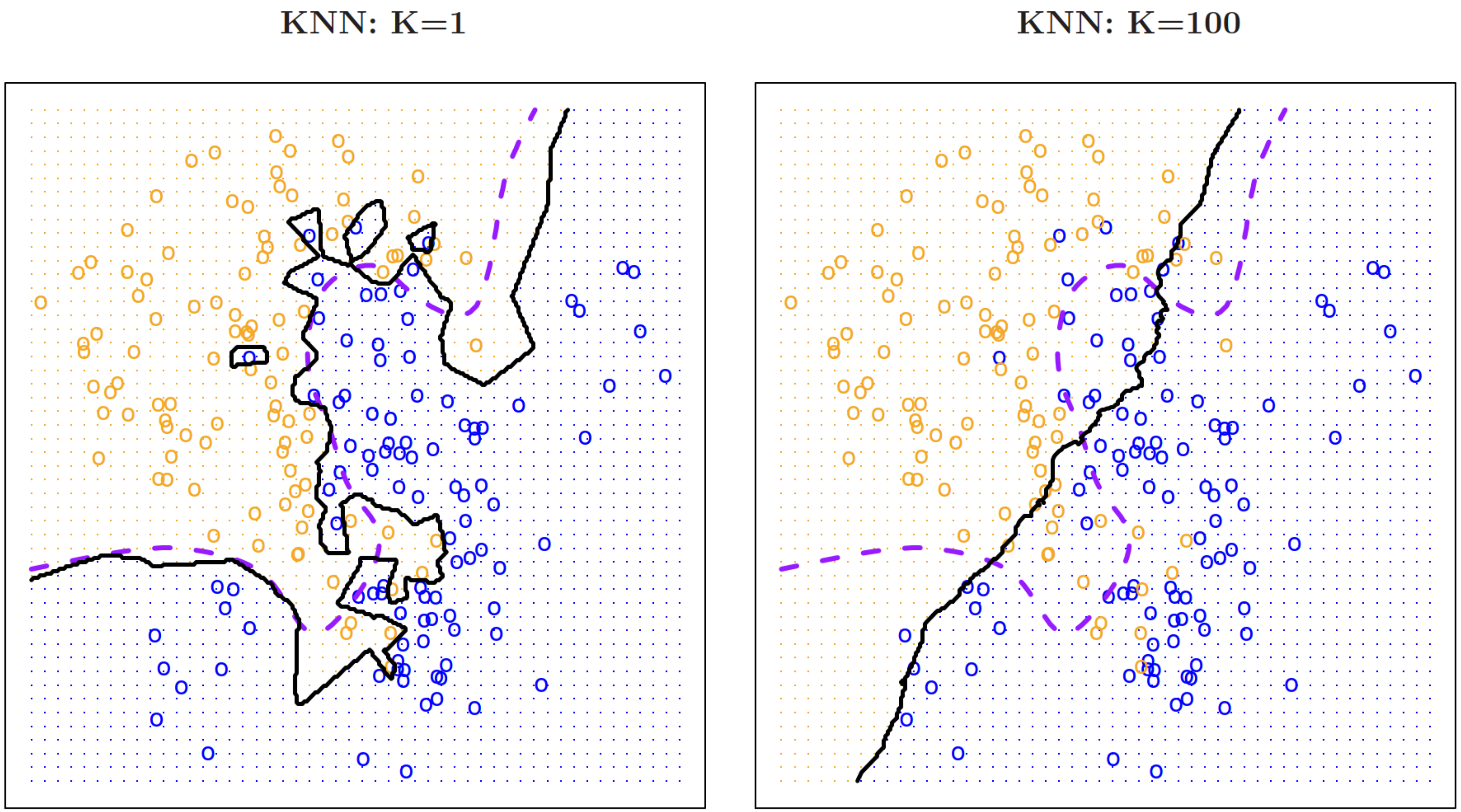


Example: K-nearest neighbors in two dimensions

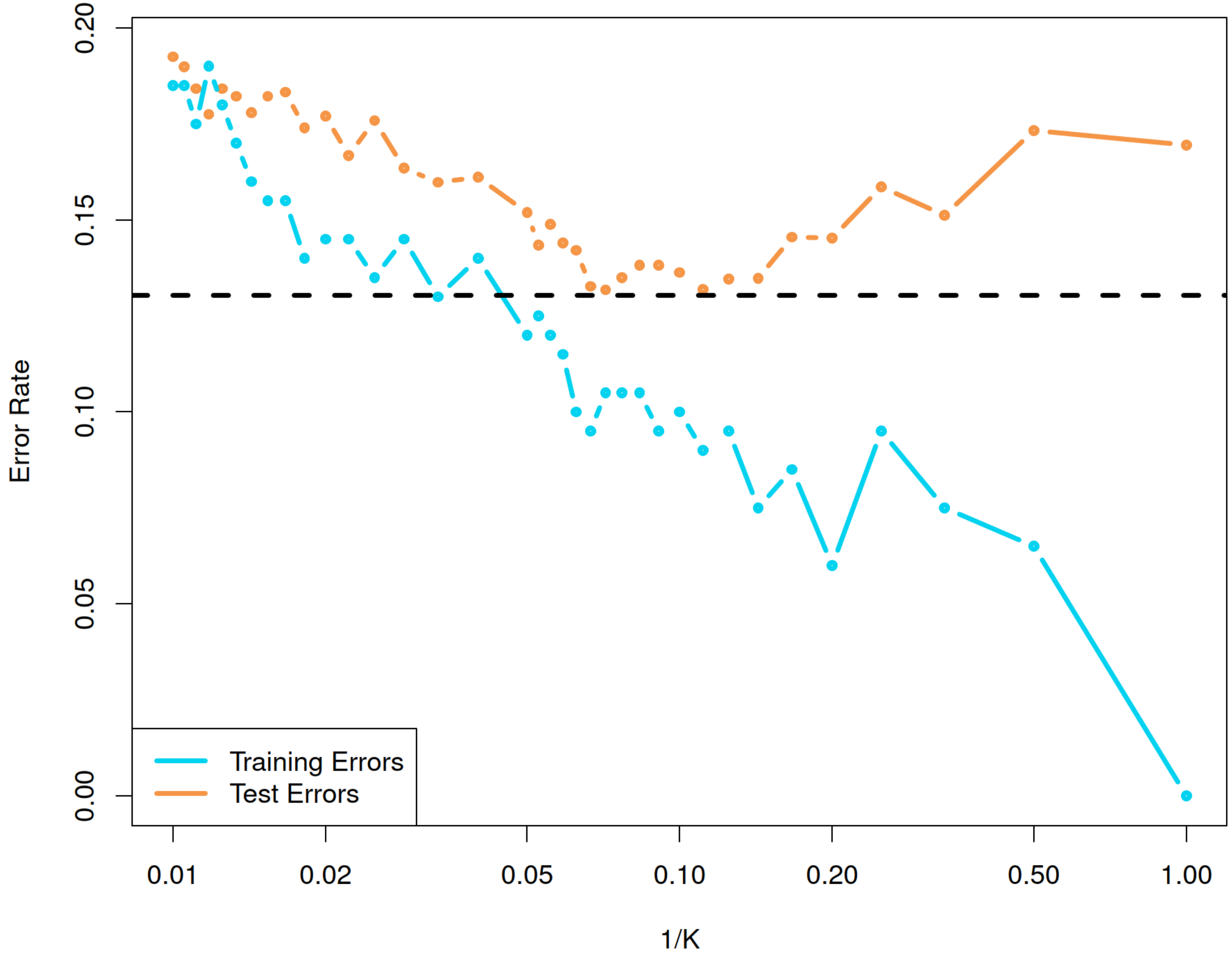
- K-nearest neighbors (KNN) classifier
  - $\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in Tr} I(y_i = j)$



Example: *K*-nearest neighbors in two dimensions



Example: *K*-nearest neighbors in two dimensions



Appendix

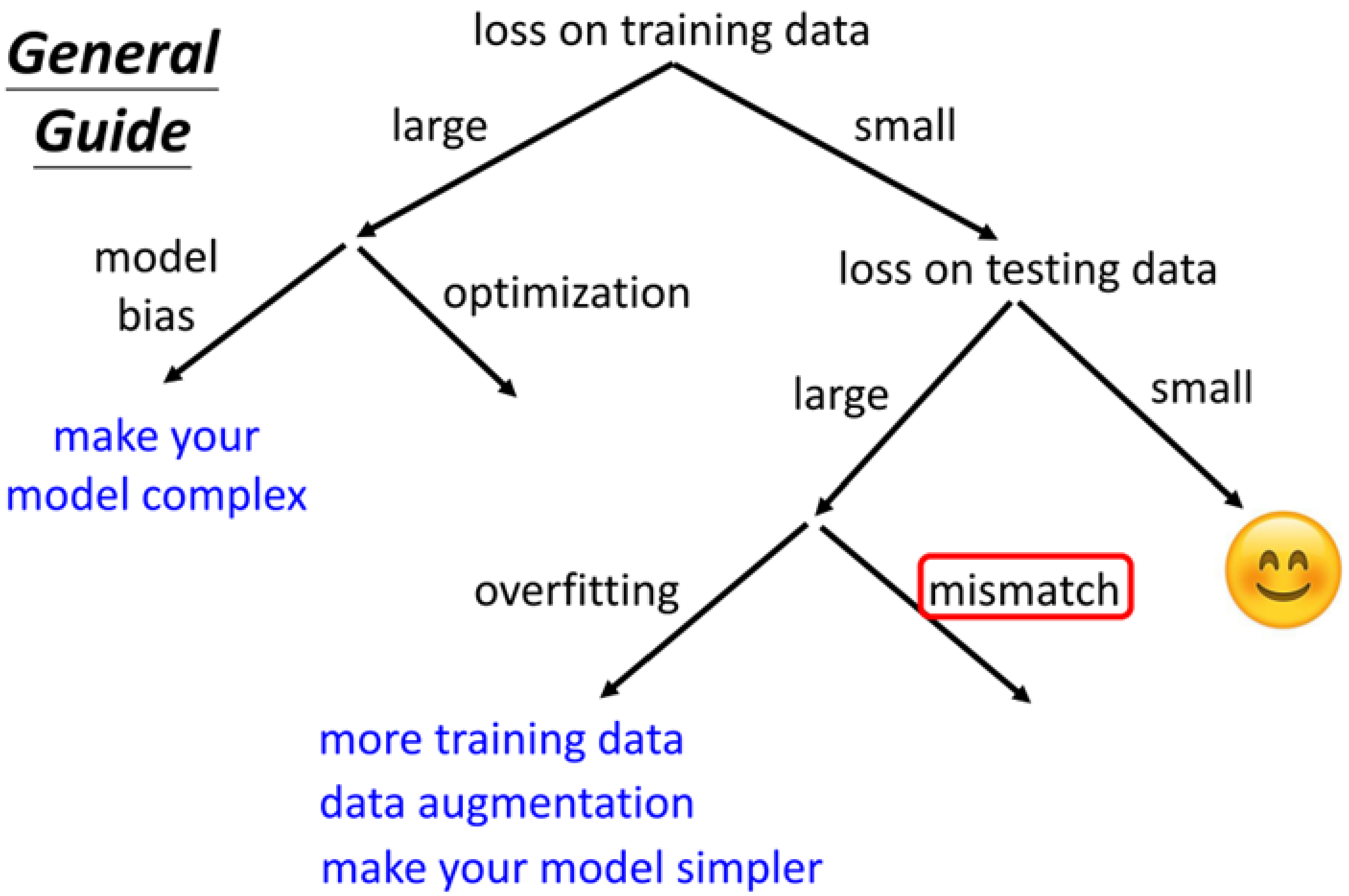
The Bias-variance tradeoff

- $f = f(x), \hat{f} = \hat{f}(x, Tr),\ Var(X) = E(X^2) - E[X]^2$
- $y = f + \epsilon \rightarrow E(y) = E(f) = f$ (f is deterministic, independent of Tr and $\hat{f}$ is independent of ϵ)
- $Var[y] = E\left[\left(y - E(y)\right)^2\right] = E[(y - f)^2] = E[\epsilon^2] = Var[\epsilon] + E[\epsilon]^2 = \sigma^2$
- $E\left[\left(y - \hat{f}\right)^2\right] = E\left[\left(f + \epsilon - \hat{f} + E[\hat{f}] - E[\hat{f}]\right)^2\right]$

$$= E\left[\left(f - E[\hat{f}]\right)^2\right] + \mathrm{E}[\epsilon^2] + E\left[\left(E[\hat{f}] - \hat{f}\right)^2\right] + 2\mathrm{E}[(f - E[\hat{f}])\epsilon] + 2\mathrm{E}[\epsilon(E[\hat{f}] - \hat{f})]$$
$$+ 2E[(E[\hat{f}] - \hat{f})(f - E[\hat{f}])] = \left(f - E[\hat{f}]\right)^2 + \mathrm{E}[\epsilon^2] + E\left[\left(E[\hat{f}] - \hat{f}\right)^2\right]$$
$$= Bias[\hat{f}]^2 + Var[\hat{f}] + \sigma^2$$

- $MSE = E_x[Bias_{Tr}[\hat{f}(x, Tr)]^2 + Var_D[\hat{f}(x, Tr)]] + \sigma^2$ (Taking expectation over x)

General Guide



https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/overfit-v6.pdf