Statistical Learning– MATH 6333
Set 2 (Overview of Supervised Learning)

Tamer Oraby

UTRGV

tamer.oraby@utrgv.edu

* Last updated August 31, 2021

Supervised Learning

Training data: $\mathcal{T} = \{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i) : i = 1, 2, \ldots, n\}$

Model: $y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \epsilon_i$ for $i = 1, 2, \ldots, n$, and the errors ϵ_i are iidrv with mean 0 and are independent of the X's.

The function f, in its wide sense, could be parametric or non-parametric.

Goal of Sup.L.: To estimate f by $\hat{f}$ using a Loss function L. May involve validation step.

Testing: To compare the predictions $\hat{y}_j = \hat{f}(x_{j1}, x_{j2}, \ldots, x_{jp})$ of testing data $\{(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j) : j = 1, 2, \ldots, m\}$ to y_j's.

Ultimate Goal (Generalization): To make predictions $\hat{f}(x_*)$ for new inputs $x_* = (x_{*1}, x_{*2}, \ldots, x_{*p})$.

Supervised Learning

Training data: $\mathcal{T} = \{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i) : i = 1, 2, \ldots, n\}$

Model: $y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \epsilon_i$ for $i = 1, 2, \ldots, n$, and the errors ϵ_i are iidrv with mean 0 and are independent of the X's.

The function f, in its wide sense, could be parametric or non-parametric.

Goal of Sup.L.: To estimate f by $\hat{f}$ using a Loss function L. May involve validation step.

Testing: To compare the predictions $\hat{y}_j = \hat{f}(x_{j1}, x_{j2}, \ldots, x_{jp})$ of testing data $\{(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j) : j = 1, 2, \ldots, m\}$ to y_j's.

Ultimate Goal (Generalization): To make predictions $\hat{f}(x_*)$ for new inputs $x_* = (x_{*1}, x_{*2}, \ldots, x_{*p})$.

Supervised Learning

Training data: $\mathcal{T} = \{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i) : i = 1, 2, \ldots, n\}$

Model: $y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \epsilon_i$ for $i = 1, 2, \ldots, n$, and the errors ϵ_i are iidrv with mean 0 and are independent of the X's.

The function f, in its wide sense, could be parametric or non-parametric.

Goal of Sup.L.: To estimate f by $\hat{f}$ using a Loss function L. May involve validation step.

Testing: To compare the predictions $\hat{y}_j = \hat{f}(x_{j1}, x_{j2}, \ldots, x_{jp})$ of testing data $\{(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j) : j = 1, 2, \ldots, m\}$ to y_j's.

Ultimate Goal (Generalization): To make predictions $\hat{f}(x_*)$ for new inputs $x_* = (x_{*1}, x_{*2}, \ldots, x_{*p})$.

Supervised Learning

Training data: $\mathcal{T} = \{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i) : i = 1, 2, \ldots, n\}$

Model: $y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \epsilon_i$ for $i = 1, 2, \ldots, n$, and the errors ϵ_i are iidrv with mean 0 and are independent of the X's.

The function f, in its wide sense, could be parametric or non-parametric.

Goal of Sup.L.: To estimate f by $\hat{f}$ using a Loss function L. May involve validation step.

Testing: To compare the predictions $\hat{y}_j = \hat{f}(x_{j1}, x_{j2}, \ldots, x_{jp})$ of testing data $\{(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j) : j = 1, 2, \ldots, m\}$ to y_j's.

Ultimate Goal (Generalization): To make predictions $\hat{f}(x_*)$ for new inputs $x_* = (x_{*1}, x_{*2}, \ldots, x_{*p})$.

Supervised Learning

Training data: $\mathcal{T} = \{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i) : i = 1, 2, \ldots, n\}$

Model: $y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \epsilon_i$ for $i = 1, 2, \ldots, n$, and the errors ϵ_i are iidrv with mean 0 and are independent of the X's.

The function f, in its wide sense, could be parametric or non-parametric.

Goal of Sup.L.: To estimate f by $\hat{f}$ using a Loss function L. May involve validation step.

Testing: To compare the predictions $\hat{y}_j = \hat{f}(x_{j1}, x_{j2}, \ldots, x_{jp})$ of testing data $\{(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j) : j = 1, 2, \ldots, m\}$ to y_j's.

Ultimate Goal (Generalization): To make predictions $\hat{f}(x_*)$ for new inputs $x_* = (x_{*1}, x_{*2}, \ldots, x_{*p})$.

Supervised Learning

Training data: $\mathcal{T} = \{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i) : i = 1, 2, \ldots, n\}$

Model: $y_i = f(x_{i1}, x_{i2}, \ldots, x_{ip}) + \epsilon_i$ for $i = 1, 2, \ldots, n$, and the errors ϵ_i are iidrv with mean 0 and are independent of the X's.

The function f, in its wide sense, could be parametric or non-parametric.

Goal of Sup.L.: To estimate f by $\hat{f}$ using a Loss function L. May involve validation step.

Testing: To compare the predictions $\hat{y}_j = \hat{f}(x_{j1}, x_{j2}, \ldots, x_{jp})$ of testing data $\{(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j) : j = 1, 2, \ldots, m\}$ to y_j's.

Ultimate Goal (Generalization): To make predictions $\hat{f}(x_*)$ for new inputs $x_* = (x_{*1}, x_{*2}, \ldots, x_{*p})$.

Parametric vs Non-Parametric

Models

Parametric: f has a functional form and a fixed number of parameters.

$\Uparrow$ Easy to interpret. $\Downarrow$ But more complex models can lead to overfitting. $\Downarrow$ $\hat{f}$ may be very different than true f.

Models

Parametric: f has a functional form and a fixed number of parameters.

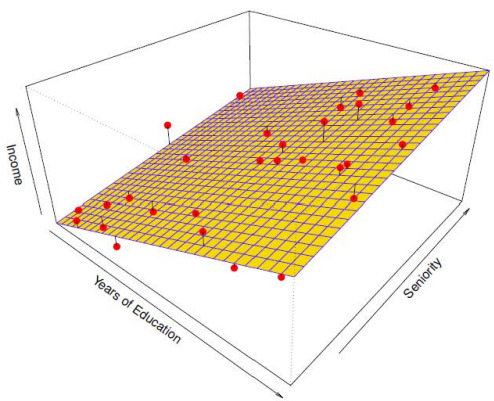
$\Uparrow$ Easy to interpret. $\Downarrow$ But more complex models can lead to overfitting. $\Downarrow$ $\hat{f}$ may be very different than true f.

Example (Multiple linear regression)

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

where the parameters $\beta_0, \beta_1, \beta_2 \ldots, \beta_p$ are estimated using the method of ordinary least squares to give

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$$



e.g.,

$$\text{income} = \hat{\beta}_0 + \hat{\beta}_1 \text{years of education} + \hat{\beta}_2 \text{seniority}$$

Models

Parametric: f has a functional form and a fixed number of parameters.

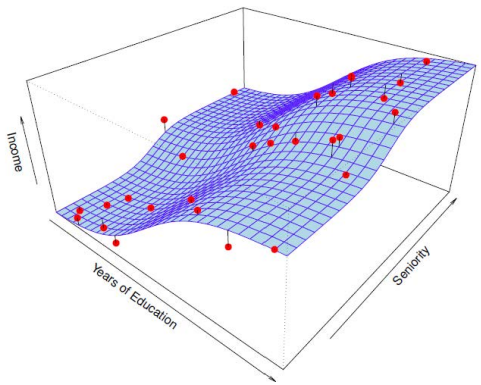
$\Uparrow$ Easy to interpret. $\Downarrow$ But more complex models can lead to overfitting. $\Downarrow$ $\hat{f}$ may be very different than true f.

Example (Non-linear regression)

$$f(X) = c/(1 + \exp(-\beta_0 - \beta_1 X_1 - \beta_2 X_2 - \cdots - \beta_p X_p))$$

where the parameters $c, \beta_0, \beta_1, \beta_2 \ldots, \beta_p$ are estimated using the method of least squares to give

$$\hat{Y} = \hat{c}/(1 + \exp(-\hat{\beta}_0 - \hat{\beta}_1 X_1 - \hat{\beta}_2 X_2 - \cdots - \hat{\beta}_p X_p))$$



e.g., income $=$

$$\frac{\hat{c}}{1 + \exp(-\hat{\beta}_0 - \hat{\beta}_1 \text{years of education} - \hat{\beta}_2 \text{seniority})}$$

Models

Parametric: f has a functional form and a fixed number of parameters.

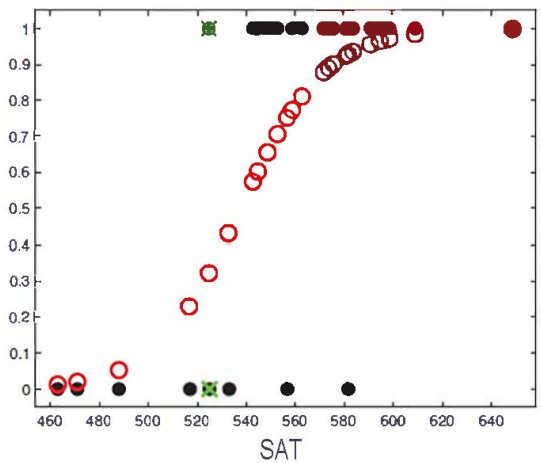
$\Uparrow$ Easy to interpret. $\Downarrow$ But more complex models can lead to overfitting. $\Downarrow$ $\hat{f}$ may be very different than true f.

Example (Logistic regression (classification))

$$Y|X \sim \text{Bernoulli}(1/(1+\exp(-\beta_0 - \beta_1 X_1 - \beta_2 X_2 - \cdots - \beta_p X_p)))$$

where the parameters $\beta_0, \beta_1, \beta_2 \ldots, \beta_p$ are estimated using the method of maximum likelihood to give

$$P(\hat{Y} = 1|X) = 1/(1+\exp(-\hat{\beta}_0 - \hat{\beta}_1 X_1 - \hat{\beta}_2 X_2 - \cdots - \hat{\beta}_p X_p))$$



$$P(pass|SAT) = \frac{1}{1+\exp(-\hat{\beta}_0 - \hat{\beta}_1 \text{SAT})}$$

Linear regression and method of ordinary least squares

- Let the $N \times p$ matrix X be given by

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix}$$

where

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}$$

for $i = 1, 2, \ldots, N$.

- In a vector form: $f(x_i) = x_i^T \beta$ and the steepest uphill direction is $f'(x) = \beta$

Linear regression and method of ordinary least squares

- Let the $N \times p$ matrix X be given by

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{pmatrix}$$

  where

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}$$

  for $i = 1, 2, \ldots, N$.

- In a vector form: $f(x_i) = x_i^T \beta$ and the steepest uphill direction is $f'(x) = \beta$

Linear regression and method of ordinary least squares

- In the methods of least squares we find β the minimizes the Residual Sum of Squares

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$
$$= (y - X\beta)^T (y - X\beta)$$

- By differentiation and setting equal to zero

$$X^T (y - X\beta) = 0$$

- If $X^T X$ is non-singular, then

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

Linear regression and method of ordinary least squares

- In the methods of least squares we find β the minimizes the Residual Sum of Squares

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - x_i^T\beta)^2$$
$$= (y - X\beta)^T(y - X\beta)$$

- By differentiation and setting equal to zero

$$X^T(y - X\beta) = 0$$

- If X^TX is non-singular, then

$$\hat{\beta} = (X^TX)^{-1}X^Ty$$

Linear regression and method of ordinary least squares

- In the methods of least squares we find β the minimizes the Residual Sum of Squares

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 \\ &= (y - X\beta)^T (y - X\beta) \end{aligned}$$

- By differentiation and setting equal to zero

$$X^T (y - X\beta) = 0$$

- If $X^T X$ is non-singular, then

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

A classification problem using linear regression

Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- After fitting that training data of $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to a linear regression model, we find $\hat{Y} = X^T \hat{\beta}$
- Decision rule:

$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid line is called the decision boundary $\{x : x^T \hat{\beta} = 0.5\}$

Blue =0 and Orange =1

A classification problem using linear regression

Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- After fitting that training data of $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to a linear regression model, we find $\hat{Y} = X^T \hat{\beta}$
- Decision rule:

$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid line is called the decision boundary $\{x : x^T \hat{\beta} = 0.5\}$

Blue =0 and Orange =1

A classification problem using linear regression

Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- After fitting that training data of $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to a linear regression model, we find $\hat{Y} = X^T \hat{\beta}$
- Decision rule:

$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid line is called the decision boundary $\{x : x^T \hat{\beta} = 0.5\}$

Blue =0 and Orange =1

A classification problem using linear regression

Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- After fitting that training data of $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to a linear regression model, we find $\hat{Y} = X^T \hat{\beta}$
- Decision rule:

$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid line is called the decision boundary $\{x : x^T \hat{\beta} = 0.5\}$

Blue =0 and Orange =1

Models

Non-parametric: f has no functional form and the number of parameters increases with n.

$\Uparrow$ Very flexible since they don't follow a certain form. $\Downarrow$ It has so many parameters that require Big Data. $\Downarrow$ It can also suffer from overfitting.

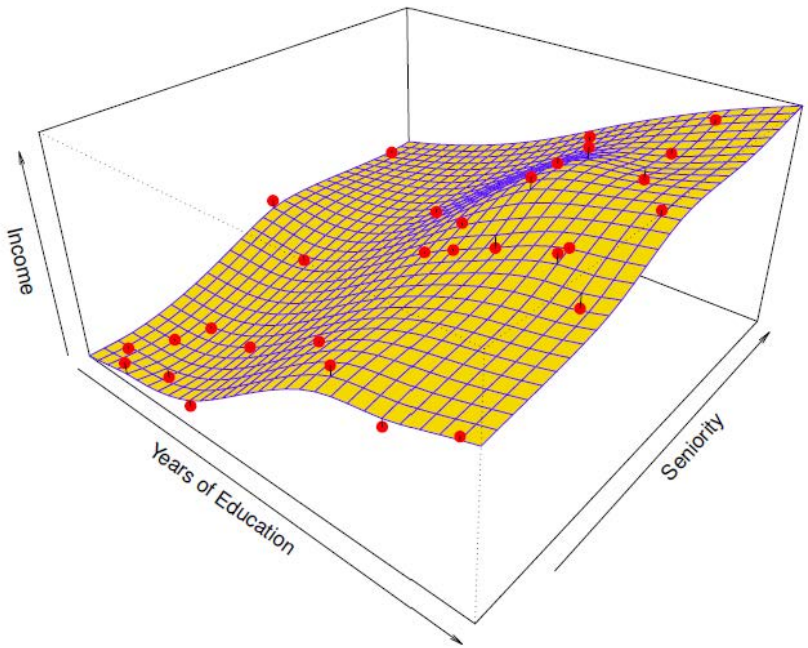
Models

Non-parametric: f has no functional form and the number of parameters increases with n.

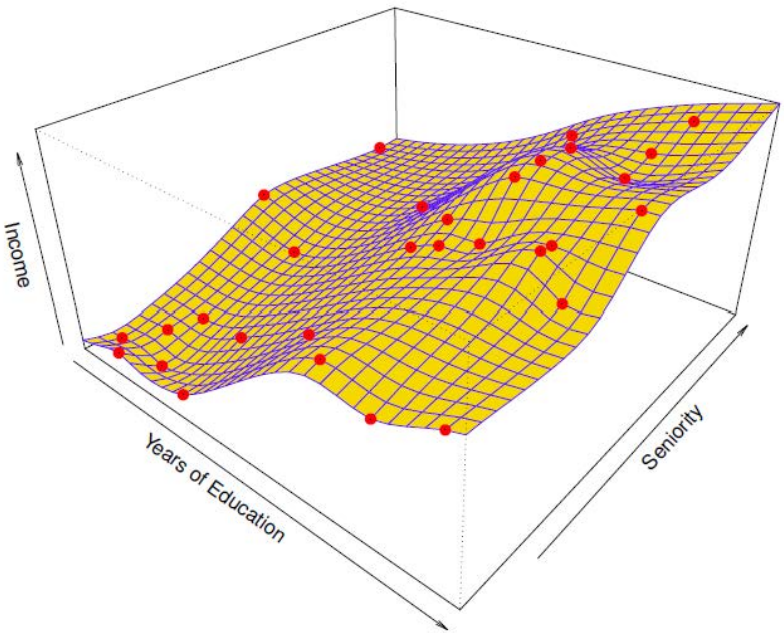
$\Uparrow$ Very flexible since they don't follow a certain form. $\Downarrow$ It has so many parameters that require Big Data. $\Downarrow$ It can also suffer from overfitting.

Example (Splines)

Smooth thin-plate Spline



Rough thin-plate Spline



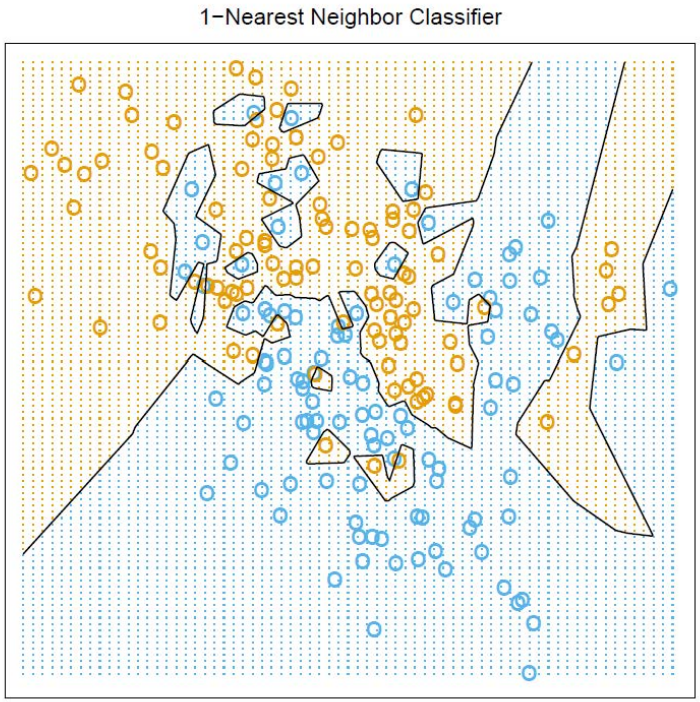
Models

Non-parametric: f has no functional form and the number of parameters increases with n.

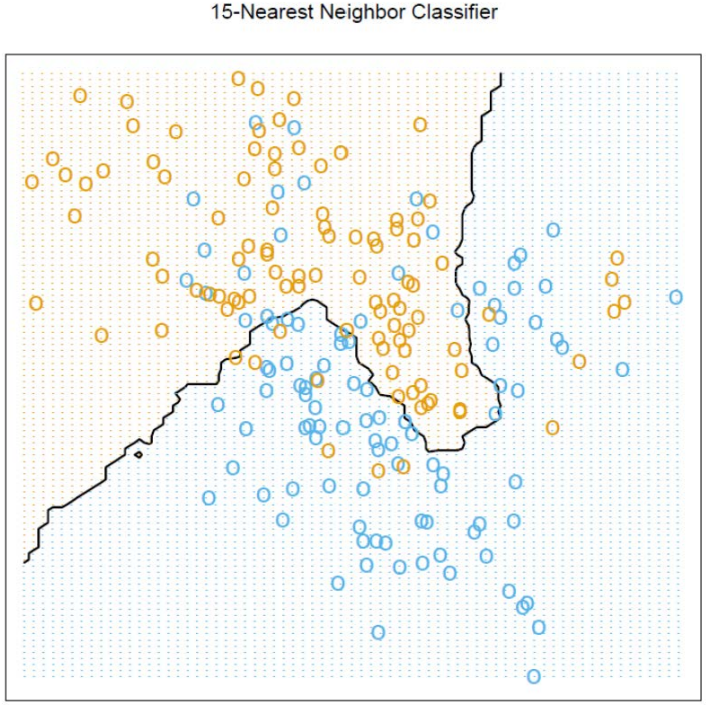
$\Uparrow$ Very flexible since they don't follow a certain form. $\Downarrow$ It has so many parameters that require Big Data. $\Downarrow$ It can also suffer from overfitting.

Example (K-nearest neighbor (K-NN))

1-NN (Voronoi tessellation)



K-NN



k-Nearest Neighbor (KNN)

- Let $N_k(x)$ be the set of closest k inputs x_i to the input x
- Closest ... using a metric, e.g., Euclidean distance
- Then,

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

k-Nearest Neighbor (KNN)

- Let $N_k(x)$ be the set of closest k inputs x_i to the input x
- Closest ... using a metric, e.g., Euclidean distance
- Then,

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

k-Nearest Neighbor (KNN)

- Let $N_k(x)$ be the set of closest k inputs x_i to the input x
- Closest ... using a metric, e.g., Euclidean distance
- Then,

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

A classification problem using K-NN

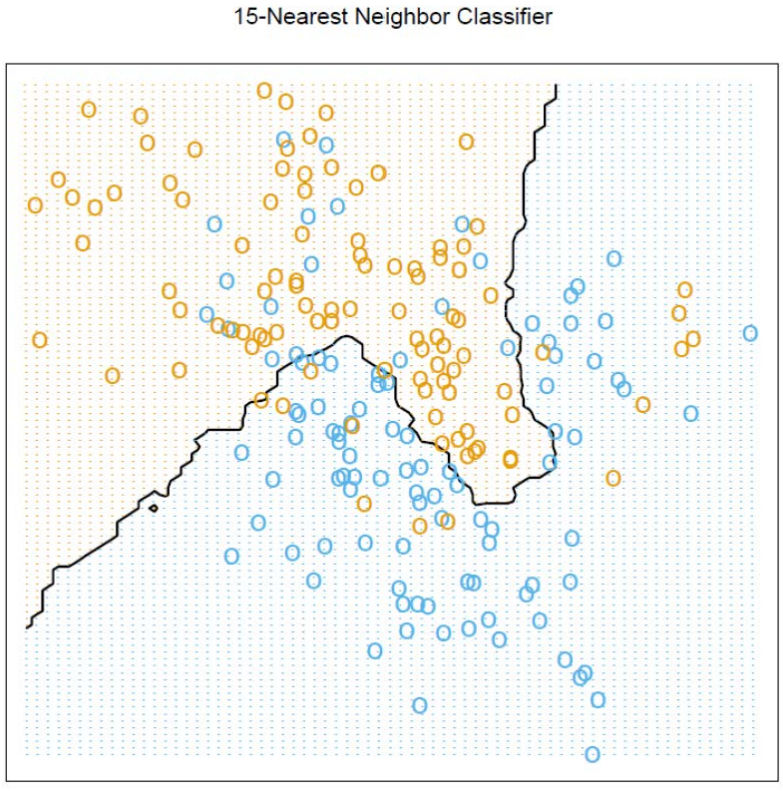
Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- At $k = 15$, using $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to find the average $\hat{Y}$ of the 15 closest 0's and 1's
- Decision rule:

$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid curve is the decision boundary found using the decision rule for a fine mesh of inputs in the plane.

Blue =0 and Orange =1



A classification problem using K-NN

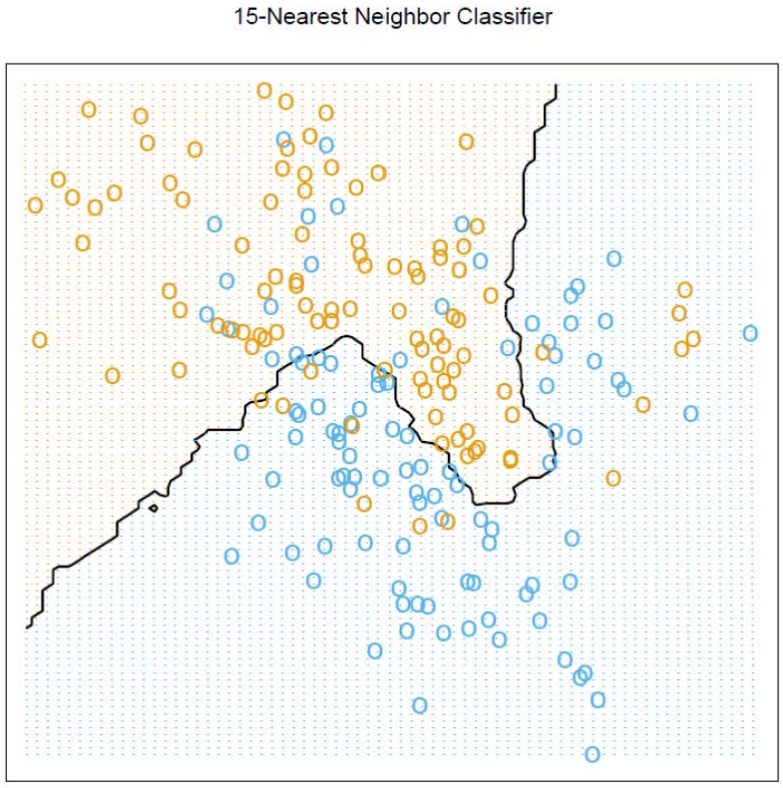
Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- At $k = 15$, using $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to find the average $\hat{Y}$ of the 15 closest 0's and 1's
- Decision rule:

$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid curve is the decision boundary found using the decision rule for a fine mesh of inputs in the plane.

Blue =0 and Orange =1



A classification problem using K-NN

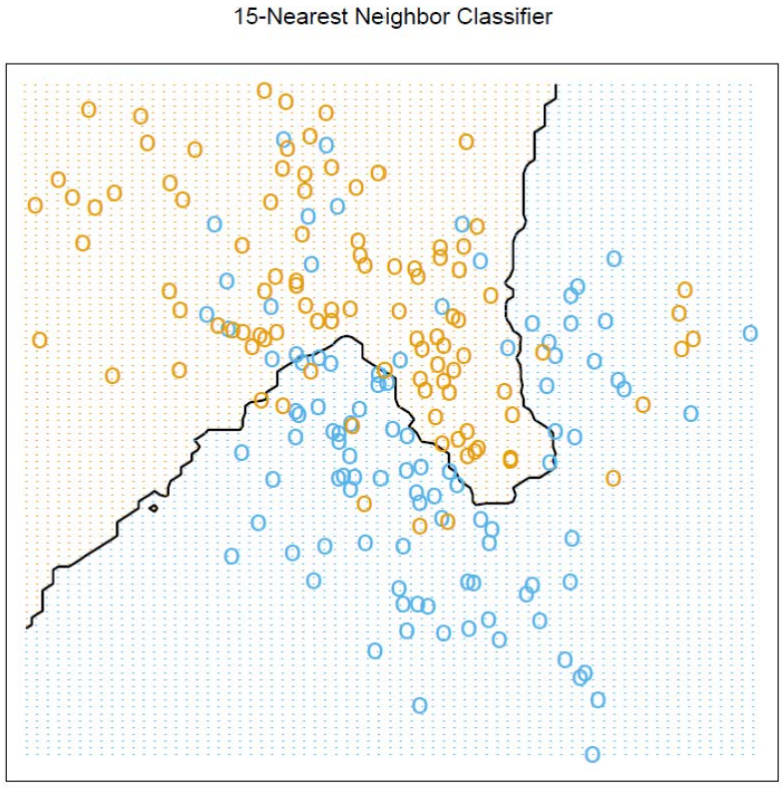
Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- At $k = 15$, using $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to find the average $\hat{Y}$ of the 15 closest 0's and 1's
- Decision rule:

$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid curve is the decision boundary found using the decision rule for a fine mesh of inputs in the plane.

Blue =0 and Orange =1



A classification problem using K-NN

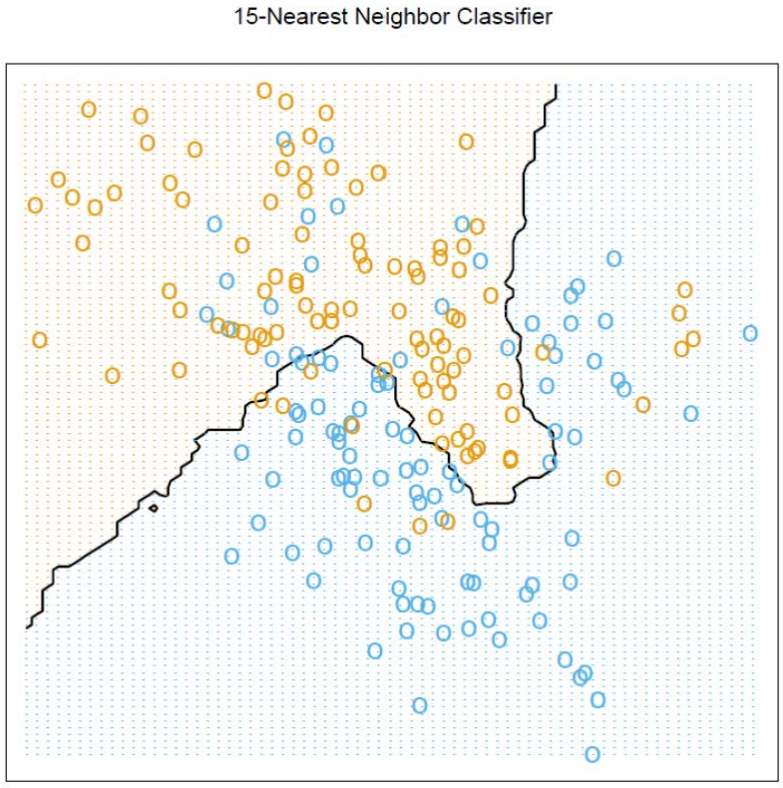
Example

- Class $G \in \mathcal{G} = \{0, 1\}$
- At $k = 15$, using $\{(x_{i1}, x_{i2}, Y_i = G_i) : i = 1, \ldots, 100\}$ to find the average $\hat{Y}$ of the 15 closest 0's and 1's
- Decision rule:

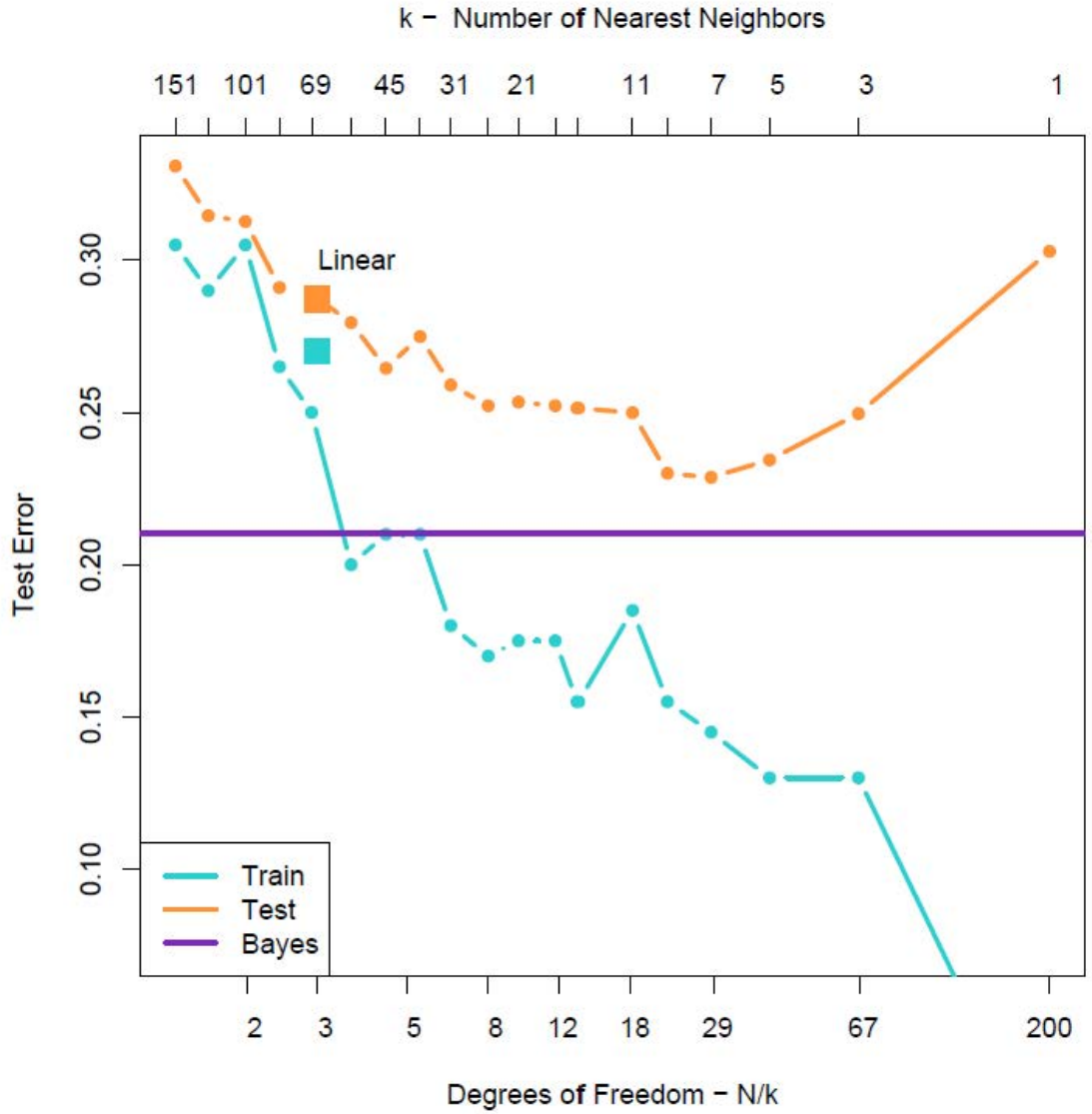
$$\hat{G} = \begin{cases} \textit{Orange} & \text{if } \hat{Y} > 0.5, \\ \textit{Blue} & \text{if } \hat{Y} \leq 0.5. \end{cases}$$

- The solid curve is the decision boundary found using the decision rule for a fine mesh of inputs in the plane.

Blue =0 and Orange =1



Misclassifications



Training set of 200 points and test set of size 10,000. Effective number of parameters is $N/k > p$.

Flexibility vs Interpretability

Loss Functions

Loss Functions

It measures how far is the predict value $f(X)$ from the actual value Y, on the population level.

- Squared Error Loss (SEL) (when Y is continuous)

$$L(Y, f(X)) = (Y - f(X))^2$$

- Absolute Error Loss (AEL) (when Y is continuous)

$$L_1(Y, f(X)) = |Y - f(X)|$$

- 0-1 Loss (0-1L) (when Y is categorical)

$$L(Y, f(X)) = I(Y \neq f(X))$$

Loss Functions

It measures how far is the predict value $f(X)$ from the actual value Y, on the population level.

- Squared Error Loss (SEL) (when Y is continuous)

$$L(Y, f(X)) = (Y - f(X))^2$$

- Absolute Error Loss (AEL) (when Y is continuous)

$$L_1(Y, f(X)) = |Y - f(X)|$$

- 0-1 Loss (0-1L) (when Y is categorical)

$$L(Y, f(X)) = I(Y \neq f(X))$$

Loss Functions

It measures how far is the predict value $f(X)$ from the actual value Y, on the population level.

- Squared Error Loss (SEL) (when Y is continuous)

$$L(Y, f(X)) = (Y - f(X))^2$$

- Absolute Error Loss (AEL) (when Y is continuous)

$$L_1(Y, f(X)) = |Y - f(X)|$$

- 0-1 Loss (0-1L) (when Y is categorical)

$$L(Y, f(X)) = I(Y \neq f(X))$$

Statistical Decision Theory

Statistical Decision Theory

Find f that minimizes the expected prediction error $EPE(f)$. If Y is continuous:

$$\begin{aligned} EPE(f) &= E[L(Y, f(X))] \\ &= \int_{\mathbb{R}^{p+1}} L(y, f(x)) P(x, y) dx dy \\ &= \int_{\mathbb{R}^{p+1}} L(y, f(x)) P(y|X = x) P(x) dx dy \\ &= \int_{\mathbb{R}^p} \underbrace{\left[\int_{\mathbb{R}} L(y, f(x)) P(y|X = x) dy \right]}_{E_{Y|X}(L(Y, f(X))|X=x)} P(x) dx \end{aligned}$$

If Y is categorical: the inner integral is a sum over all possible categories.
Thus, generally

$$f(x) = \text{argmin}_c E_{Y|X}(L(Y, c)|X = x)$$

.

Statistical Decision Theory

- If Y is continuous and we use SEL then f is given by

$$f(x) = \text{argmin}_{c \in \mathbb{R}} E_{Y|X}((Y-c)^2|X=x) = E(Y|X=x)$$

which is a regression function.

- If Y is continuous and we use AEL then f is given by

$$f(x) = \text{argmin}_{c \in \mathbb{R}} E_{Y|X}(|Y-c||X=x) = \text{Median}(Y|X=x).$$

- If Y is categorical and we use 0-1L then f is given by

$$\begin{aligned} f(x) &= \text{argmin}_{c \in \mathcal{G}} E_{Y|X}(I(Y \neq c)|X=x) \\ &= \text{argmin}_{c \in \mathcal{G}} P_{Y|X}(Y \neq c|X=x) \\ &= \text{argmin}_{c \in \mathcal{G}} \left(1 - P_{Y|X}(Y=c|X=x)\right) \\ &= \text{argmax}_{c \in \mathcal{G}} P_{Y|X}(Y=c|X=x) \end{aligned}$$

which is Bayes classifier.

Statistical Decision Theory

- If Y is continuous and we use SEL then f is given by

$$f(x) = \text{argmin}_{c \in \mathbb{R}} E_{Y|X}((Y-c)^2|X=x) = E(Y|X=x)$$

which is a regression function.

- If Y is continuous and we use AEL then f is given by

$$f(x) = \text{argmin}_{c \in \mathbb{R}} E_{Y|X}(|Y-c||X=x) = \text{Median}(Y|X=x).$$

- If Y is categorical and we use 0-1L then f is given by

$$\begin{aligned} f(x) &= \text{argmin}_{c \in \mathcal{G}} E_{Y|X}(I(Y \neq c)|X=x) \\ &= \text{argmin}_{c \in \mathcal{G}} P_{Y|X}(Y \neq c|X=x) \\ &= \text{argmin}_{c \in \mathcal{G}} \left(1 - P_{Y|X}(Y=c|X=x)\right) \\ &= \text{argmax}_{c \in \mathcal{G}} P_{Y|X}(Y=c|X=x) \end{aligned}$$

which is Bayes classifier.

Statistical Decision Theory

- If Y is continuous and we use SEL then f is given by

$$f(x) = \text{argmin}_{c \in \mathbb{R}} E_{Y|X}((Y-c)^2|X=x) = E(Y|X=x)$$

which is a regression function.

- If Y is continuous and we use AEL then f is given by

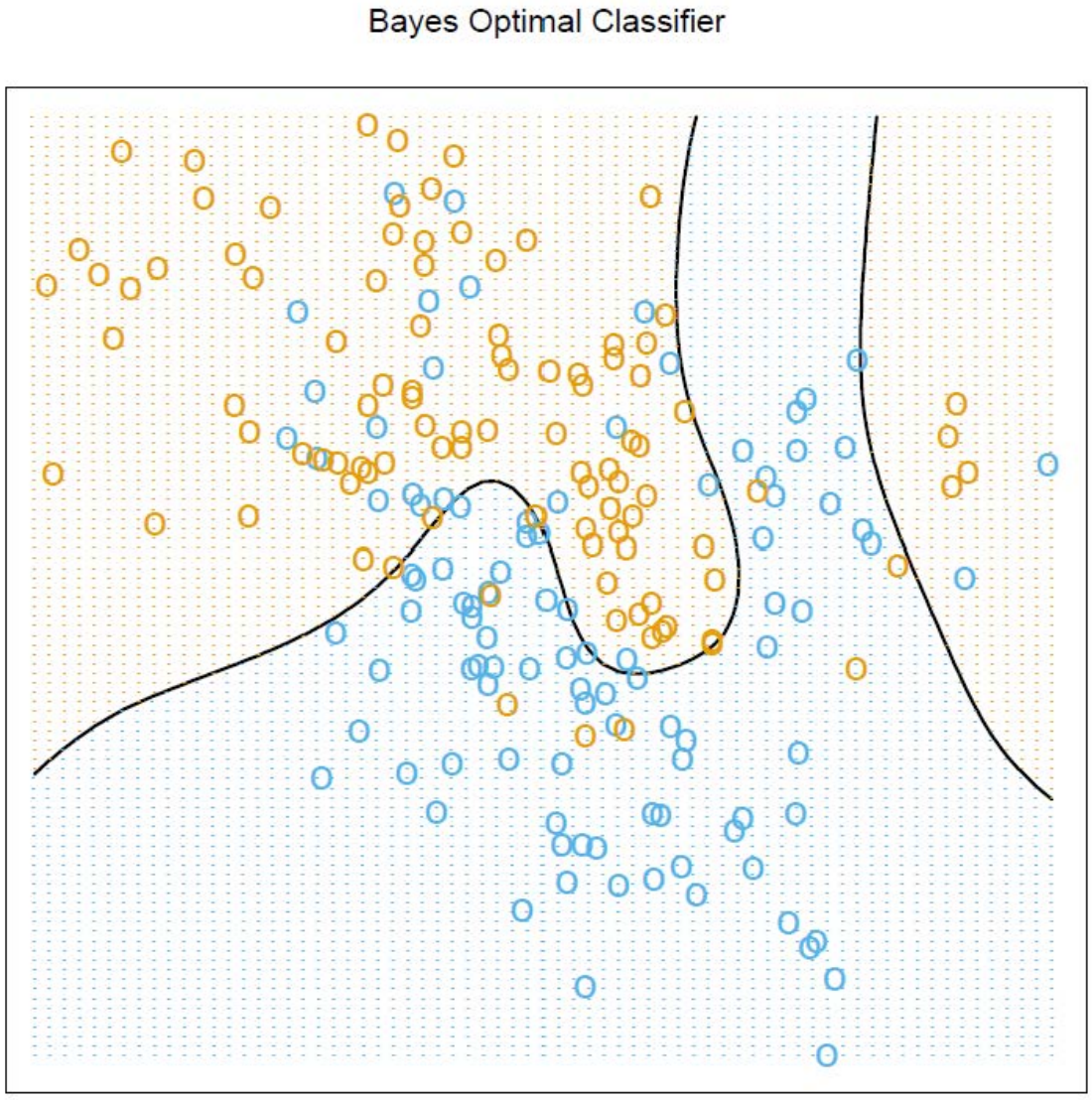
$$f(x) = \text{argmin}_{c \in \mathbb{R}} E_{Y|X}(|Y-c||X=x) = \text{Median}(Y|X=x).$$

- If Y is categorical and we use 0-1L then f is given by

$$\begin{aligned} f(x) &= \text{argmin}_{c \in \mathcal{G}} E_{Y|X}(I(Y \neq c)|X=x) \\ &= \text{argmin}_{c \in \mathcal{G}} P_{Y|X}(Y \neq c|X=x) \\ &= \text{argmin}_{c \in \mathcal{G}} \left(1 - P_{Y|X}(Y=c|X=x)\right) \\ &= \text{argmax}_{c \in \mathcal{G}} P_{Y|X}(Y=c|X=x) \end{aligned}$$

which is Bayes classifier.

A classification problem using Bayes classifier



Statistical Decision Theory

So based on SEL where

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

- In regression, if we approximate $f(x) = x^T\beta$ then we get through the optimization step $\beta = [E(XX^T)]^{-1}E(XY)$ and expected values could be replaced by sample averages
- In K-NN, it would be

$$\hat{f}(x) = Average(y_i|x_i \in N_k(x))$$

and due to SLLN when $k, N \to \infty$ and $k/N \to 0$, then $\hat{f}(x) \to E(Y|X = x)$, a.s.

Statistical Decision Theory

So based on SEL where

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

- In regression, if we approximate $f(x) = x^T\beta$ then we get through the optimization step $\beta = [E(XX^T)]^{-1}E(XY)$ and expected values could be replaced by sample averages
- In K-NN, it would be

$$\hat{f}(x) = Average(y_i|x_i \in N_k(x))$$

and due to SLLN when $k, N \to \infty$ and $k/N \to 0$, then $\hat{f}(x) \to E(Y|X = x)$, a.s.

K-NN as a local method ...

In other words, K-NN (a local method) gives a consistent estimator of $f(x) = E(Y|X = x)$ as

$$\hat{f}(x) = Average(y_i|x_i \in N_k(x)) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

But, nearest neighbor (local) methods suffer from ...

K-NN as a local method ...

In other words, K-NN (a local method) gives a consistent estimator of $f(x) = E(Y|X = x)$ as

$$\hat{f}(x) = Average(y_i|x_i \in N_k(x)) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

But, nearest neighbor (local) methods suffer from ...

Curse of Dimensionality

Curse of Dimensionality

⊛ *Manifestation 1*: In high input dimension d, local methods fail in making accurate predictions.

- A coverage at a fraction f will be achieved via $s = f^{1/d}$ for each input's range. What happens when d is large and f is small?
- For instance, in $[0, 1]^{20}$,

f	.001	.01	.1
$s = f^{1/d}$	.71	.79	.89

- ⇓ But, faraway inputs become less and less relevant in predictions for the central input.

e.g. 3-dimensional space $[0, 1]^3$ with uniformly distributed inputs

Curse of Dimensionality

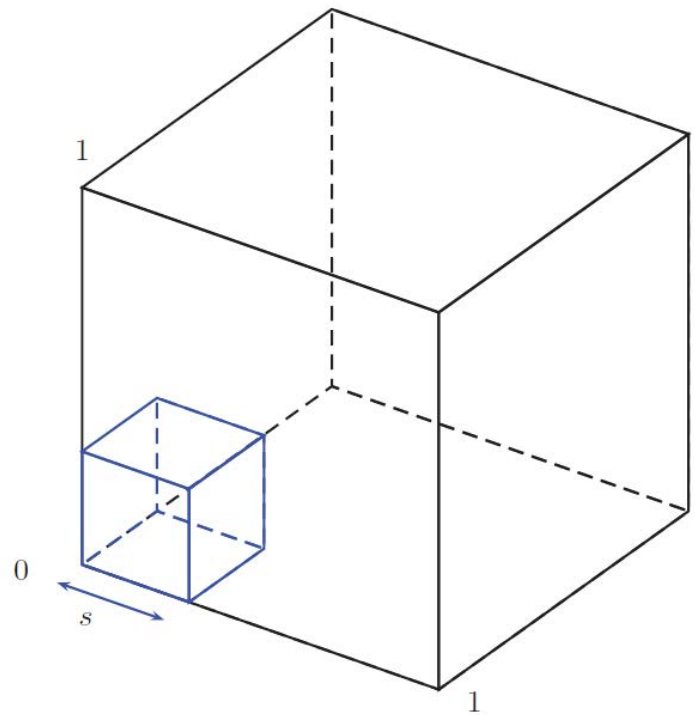
⊛ *Manifestation 1*: In high input dimension d, local methods fail in making accurate predictions.

- A coverage at a fraction f will be achieved via $s = f^{1/d}$ for each input's range. What happens when d is large and f is small?
- For instance, in $[0, 1]^{20}$,

f	.001	.01	.1
$s = f^{1/d}$	.71	.79	.89

- ⇓ But, faraway inputs become less and less relevant in predictions for the central input.

e.g. 3-dimensional space $[0, 1]^3$ with uniformly distributed inputs



Curse of Dimensionality

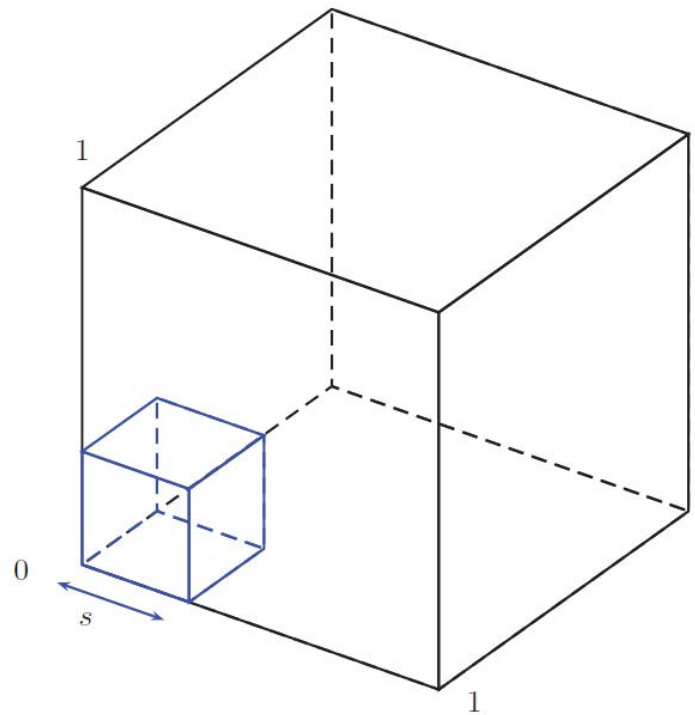
⊛ *Manifestation 1*: In high input dimension d, local methods fail in making accurate predictions.

- A coverage at a fraction f will be achieved via $s = f^{1/d}$ for each input's range. What happens when d is large and f is small?
- For instance, in $[0, 1]^{20}$,

f	.001	.01	.1
$s = f^{1/d}$	.71	.79	.89

- ⇓ But, faraway inputs become less and less relevant in predictions for the central input.

e.g. 3-dimensional space $[0, 1]^3$ with uniformly distributed inputs



Curse of Dimensionality

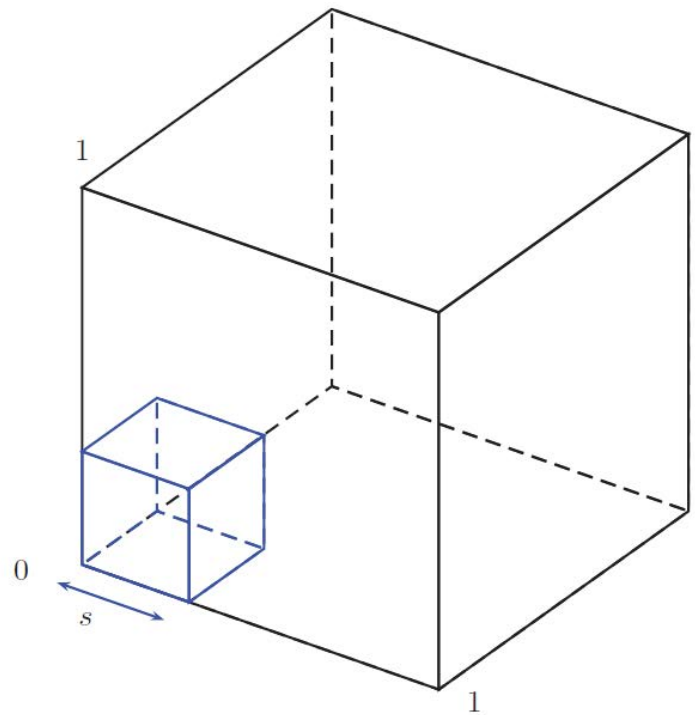
⊛ *Manifestation 1*: In high input dimension d, local methods fail in making accurate predictions.

- A coverage at a fraction f will be achieved via $s = f^{1/d}$ for each input's range. What happens when d is large and f is small?
- For instance, in $[0, 1]^{20}$,

f	.001	.01	.1
$s = f^{1/d}$	.71	.79	.89

- ⇓ But, faraway inputs become less and less relevant in predictions for the central input.

e.g. 3-dimensional space $[0, 1]^3$ with uniformly distributed inputs



Curse of Dimensionality

⊛ *Manifestation 2*: In high input dimension d, all *sample* data points are close to the boundary of the sample.

- The median distance from 0 to closest point is $r = (1 - 0.5^{1/N})^{1/d}$. What happens when d is large?
- For instance, for $d = 20$,

N	100	500
$r = (1 - 0.5^{1/N})^{1/d}$	.78	.72

- ⇓ But, using training points near the boundary makes predictions very difficult.

e.g. 3-dimensional space $\mathcal{B}_1$ with uniformly dist. N data points

Curse of Dimensionality

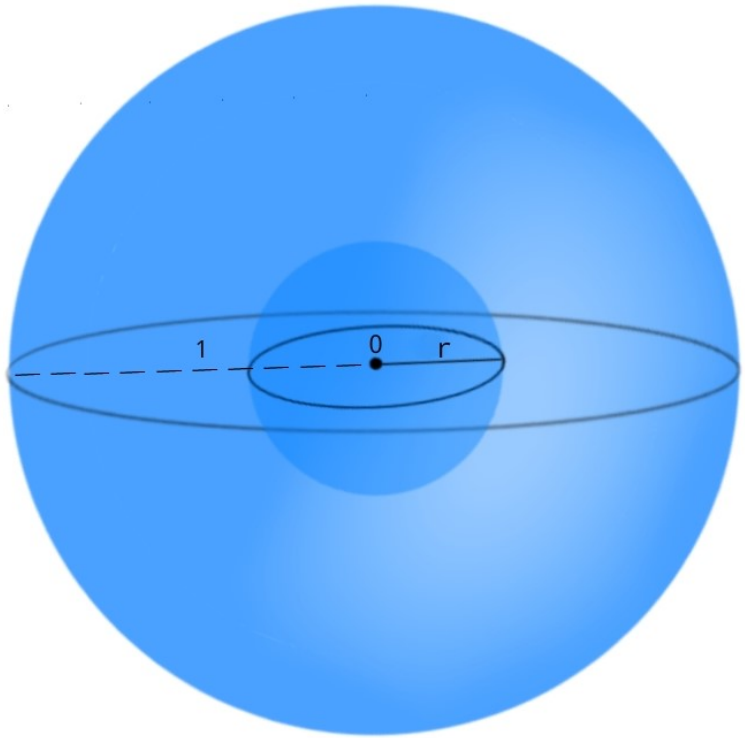
⊛ *Manifestation 2*: In high input dimension d, all *sample* data points are close to the boundary of the sample.

- The median distance from 0 to closest point is $r = (1 - 0.5^{1/N})^{1/d}$. What happens when d is large?
- For instance, for $d = 20$,

N	100	500
$r = (1 - 0.5^{1/N})^{1/d}$	.78	.72

- ⇓ But, using training points near the boundary makes predictions very difficult.

e.g. 3-dimensional space $\mathcal{B}_1$ with uniformly dist. N data points



Curse of Dimensionality

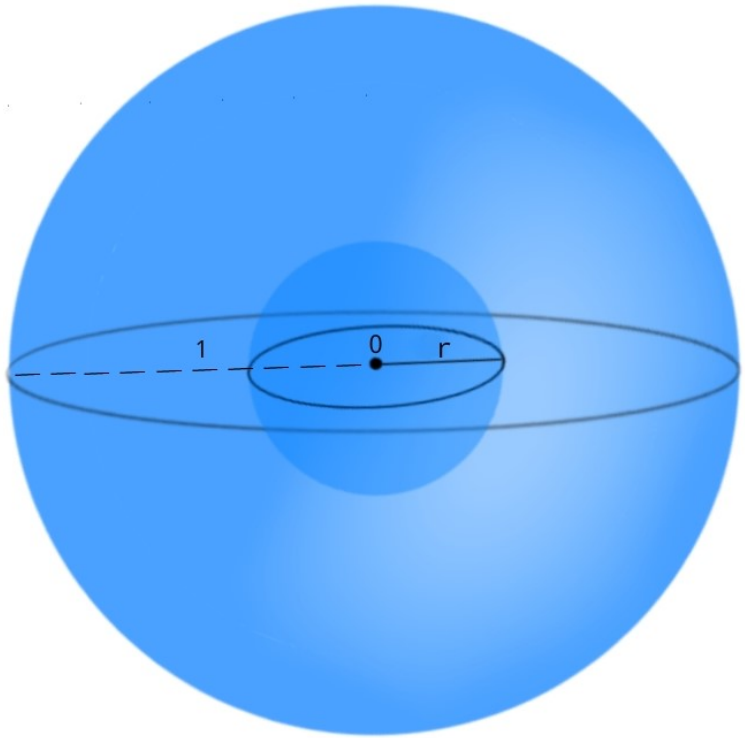
⊛ *Manifestation 2*: In high input dimension d, all *sample* data points are close to the boundary of the sample.

- The median distance from 0 to closest point is $r = (1 - 0.5^{1/N})^{1/d}$. What happens when d is large?
- For instance, for $d = 20$,

N	100	500
$r = (1 - 0.5^{1/N})^{1/d}$	.78	.72

- ⇓ But, using training points near the boundary makes predictions very difficult.

e.g. 3-dimensional space $\mathcal{B}_1$ with uniformly dist. N data points



Curse of Dimensionality

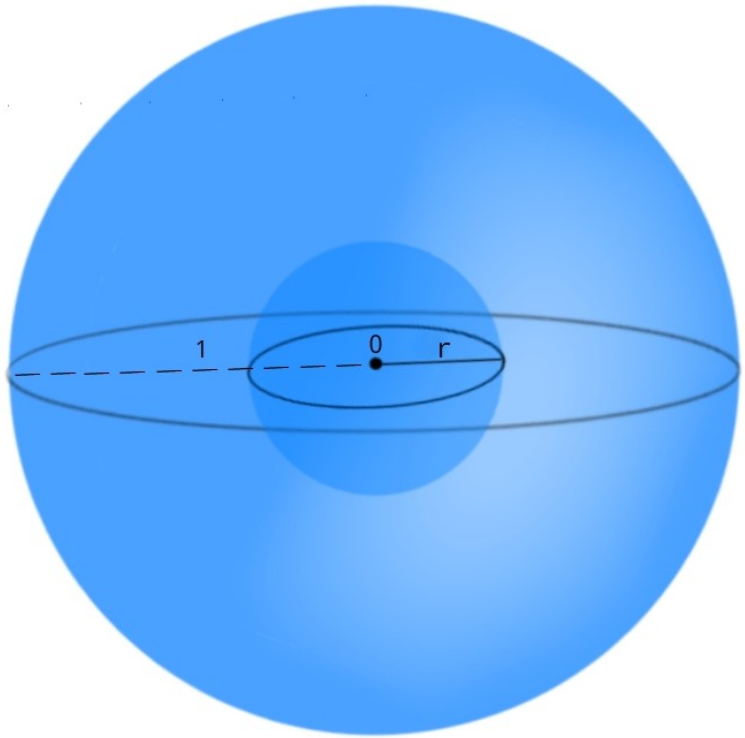
⊛ *Manifestation 2*: In high input dimension d, all *sample* data points are close to the boundary of the sample.

- The median distance from 0 to closest point is $r = (1 - 0.5^{1/N})^{1/d}$. What happens when d is large?
- For instance, for $d = 20$,

N	100	500
$r = (1 - 0.5^{1/N})^{1/d}$	.78	.72

- ⇓ But, using training points near the boundary makes predictions very difficult.

e.g. 3-dimensional space $\mathcal{B}_1$ with uniformly dist. N data points



Curse of Dimensionality

⊛ *Manifestation 3*: In high input dimension d, large number of data points N is required to populate the space.

- A sample size $N = m^d$ is required to populate the space so as to acquire a density of m. What happens when d is large?
- For instance, to achieve a sampling density of 100,

d	1	20
$N = m^d$	100^1	100^{20}

- ⇓ All feasible training samples sparsely populate the input space.

e.g. 3-dimensional space S_3 with $N = 6^3$ data points

Curse of Dimensionality

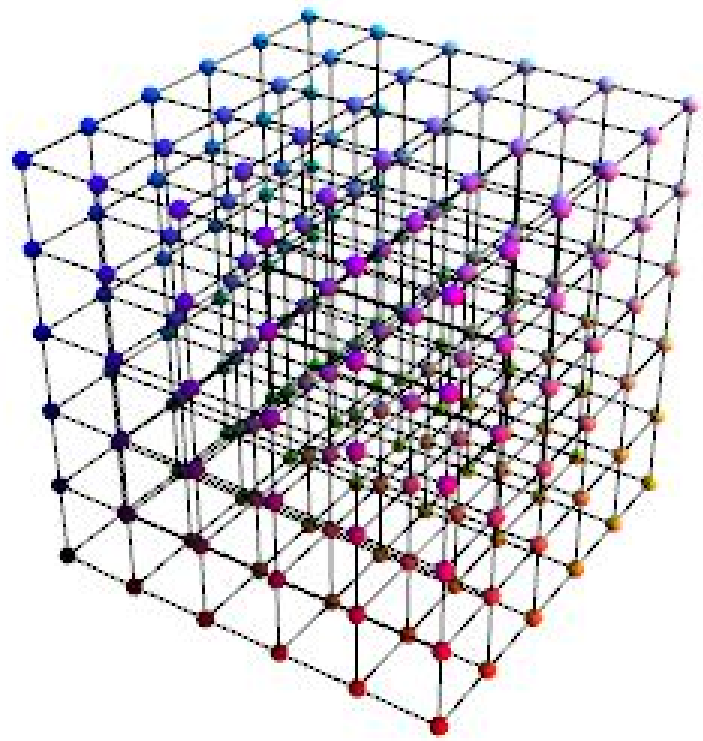
⊛ *Manifestation 3*: In high input dimension d, large number of data points N is required to populate the space.

- A sample size $N = m^d$ is required to populate the space so as to acquire a density of m. What happens when d is large?
- For instance, to achieve a sampling density of 100,

d	1	20
$N = m^d$	100^1	100^{20}

- ⇓ All feasible training samples sparsely populate the input space.

e.g. 3-dimensional space $\mathcal{S}_3$ with $N = 6^3$ data points



Curse of Dimensionality

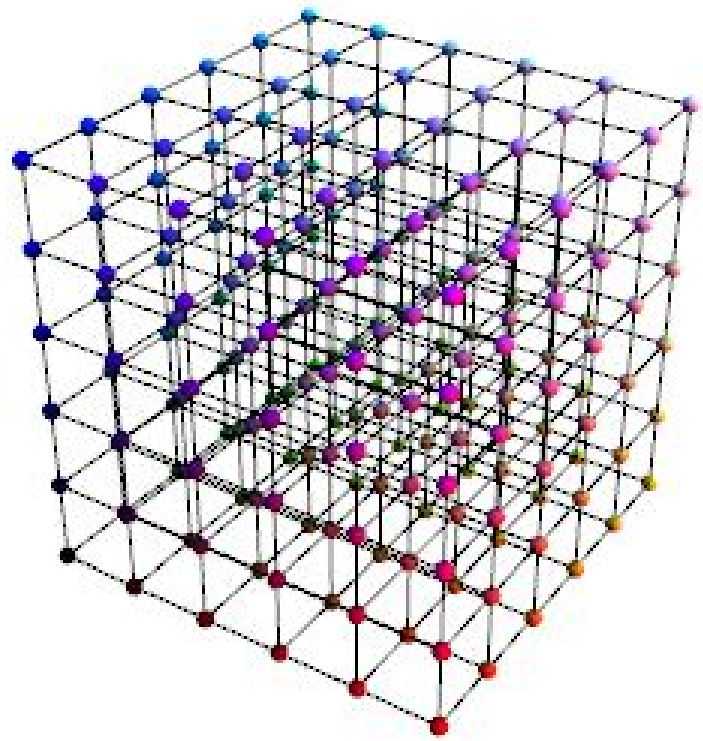
⊛ *Manifestation 3*: In high input dimension d, large number of data points N is required to populate the space.

- A sample size $N = m^d$ is required to populate the space so as to acquire a density of m. What happens when d is large?
- For instance, to achieve a sampling density of 100,

d	1	20
$N = m^d$	100^1	100^{20}

- ⇓ All feasible training samples sparsely populate the input space.

e.g. 3-dimensional space $\mathcal{S}_3$ with $N = 6^3$ data points



Curse of Dimensionality

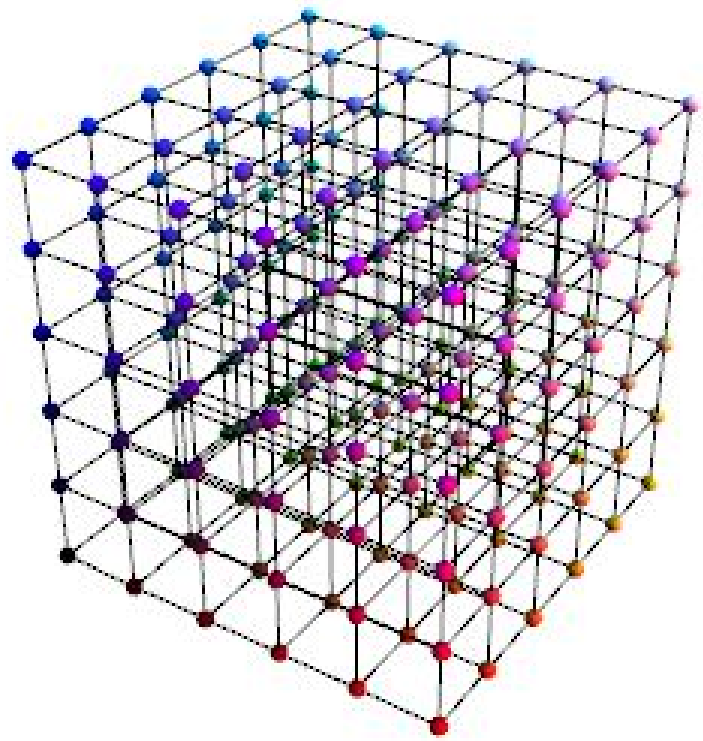
⊛ *Manifestation 3*: In high input dimension d, large number of data points N is required to populate the space.

- A sample size $N = m^d$ is required to populate the space so as to acquire a density of m. What happens when d is large?
- For instance, to achieve a sampling density of 100,

d	1	20
$N = m^d$	100^1	100^{20}

- ⇓ All feasible training samples sparsely populate the input space.

e.g. 3-dimensional space $\mathcal{S}_3$ with $N = 6^3$ data points



Measuring the Quality of Fit

Quality of Fit

To compare the estimated response $\hat{y}_i = \hat{f}(x_{i1}, x_{i2}, \ldots, x_{ip})$ of the **training** data $\{(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i) : j = 1, 2, \ldots, N\}$ to the true response y_i's.

In regression, we use the mean squared error

$$\text{training } MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

But truly, quality of prediction of (new) testing data points is more important.

Quality of Fit

In case of absence of such testing data we minimize the training

$$\text{training } MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

If we know the actual f, then the testing

$$\text{testing } MSE = \frac{1}{m}\sum_{j=1}^{m}(f(x_j) - \hat{y}_j)^2$$

Quality of Fit

In case of absence of such testing data we minimize the training

$$\text{training } MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

If we know the actual f, then the testing

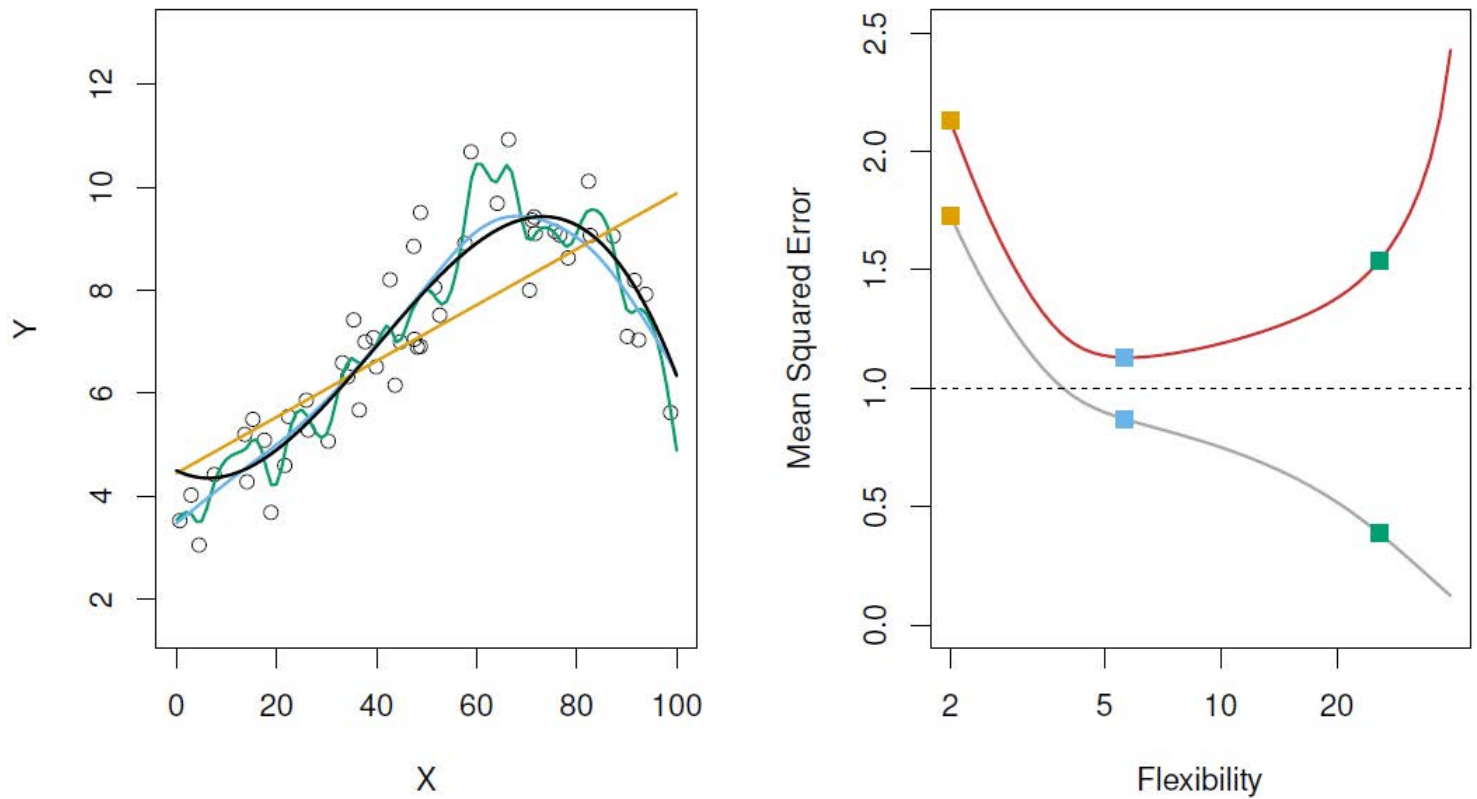
$$\text{testing } MSE = \frac{1}{m}\sum_{j=1}^{m}(f(x_j) - \hat{y}_j)^2$$

Training *vs* Testing MSE

Example (Case of overfitting)

-Left: Black curve is the true function, orange line is a fitted linear regression, and blue and green are two differently smoothed splines.

-Right: Gray curve is training MSE, and red curve is testing MSE. Dashed line is the irreducible error $Var(\epsilon) = 1.0$.

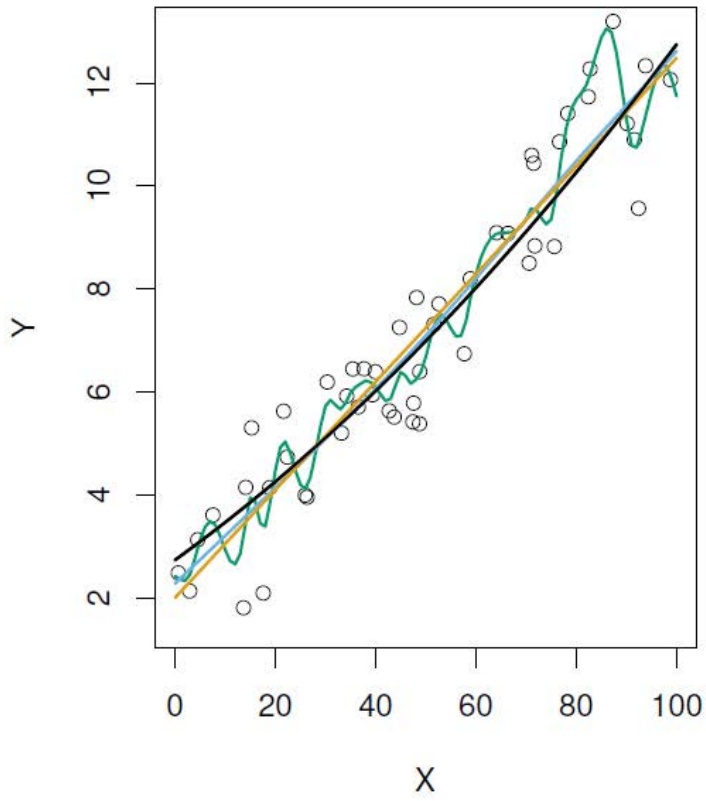


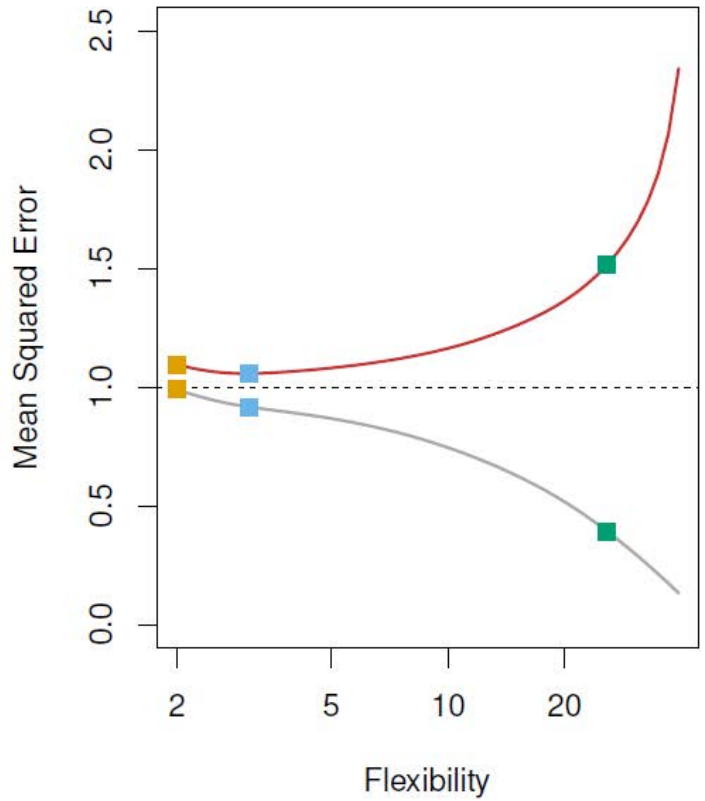
Training *vs* Testing MSE

Example (Linear regression and splines)

-Left: Black curve is the true function, orange line is a fitted linear regression, and blue and green are two differently smoothed splines.

-Right: Gray curve is training MSE, and red curve is testing MSE. Dashed line is the irreducible error $Var(\epsilon) = 1.0$.

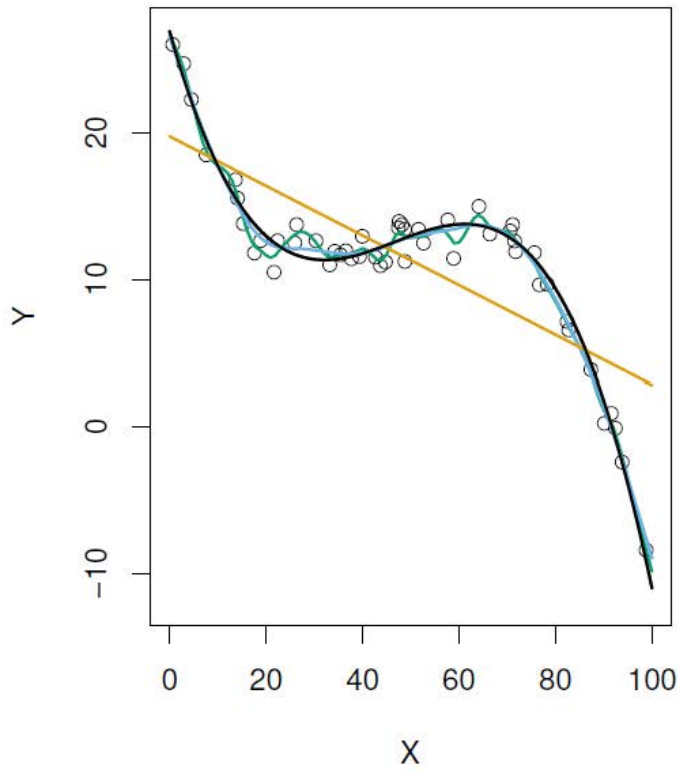


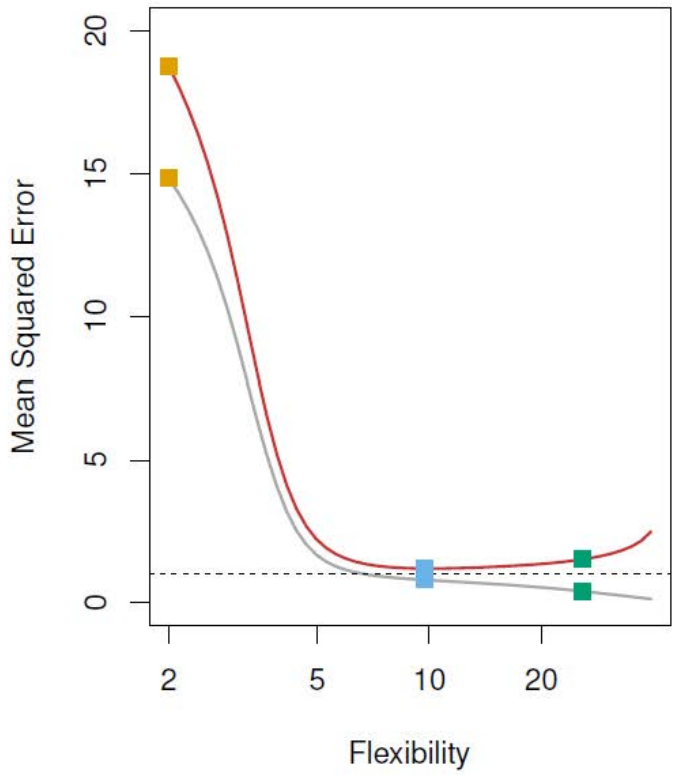


Training *vs* Testing MSE

Example (Linear regression and splines)

-Left: Black curve is the true function, orange line is a fitted linear regression, and blue and green are two differently smoothed splines.
-Right: Gray curve is training MSE, and red curve is testing MSE. Dashed line is the irreducible error $Var(\epsilon) = 1.0$.





Bias-Variance Trade-Off

Bias-Variance Trade-Off

The expected test MSE for a given input x_0

$$E[(y_0 - \hat{f}(x_0))^2] = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon)$$

- $Var(\hat{f}(x_0))$ is the variability in $\hat{f}$ which might change with the training data. It would increase if the method is highly flexible.
- $Bias(\hat{f}(x_0))$ is about how far is the fitted to the actual and so it decreases if the method is highly flexible.
- A good learning method requires the less of both and that is the trade-off.

Bias-Variance Trade-Off

The expected test MSE for a given input x_0

$$E[(y_0 - \hat{f}(x_0))^2] = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon)$$

- $Var(\hat{f}(x_0))$ is the variability in $\hat{f}$ which might change with the training data. It would increase if the method is highly flexible.
- $Bias(\hat{f}(x_0))$ is about how far is the fitted to the actual and so it decreases if the method is highly flexible.
- A good learning method requires the less of both and that is the trade-off.

Bias-Variance Trade-Off

The expected test MSE for a given input x_0

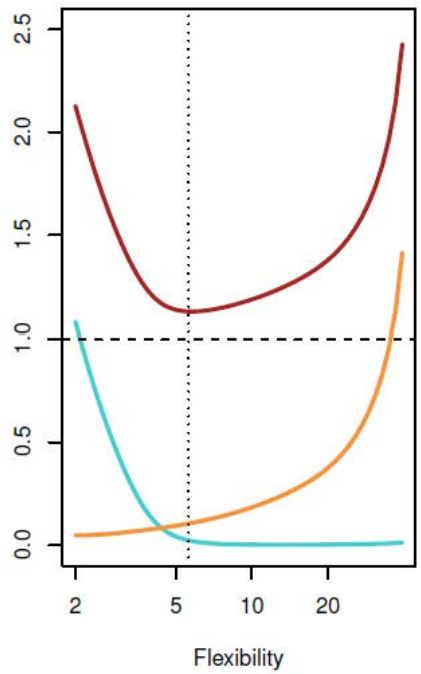
$$E[(y_0 - \hat{f}(x_0))^2] = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon)$$

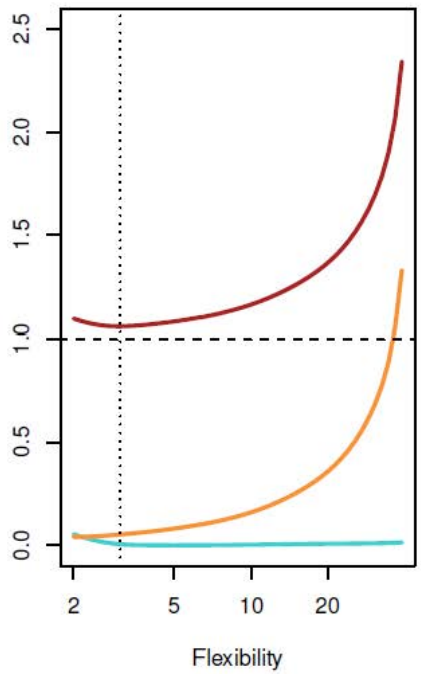
- $Var(\hat{f}(x_0))$ is the variability in $\hat{f}$ which might change with the training data. It would increase if the method is highly flexible.
- $Bias(\hat{f}(x_0))$ is about how far is the fitted to the actual and so it decreases if the method is highly flexible.
- A good learning method requires the less of both and that is the trade-off.

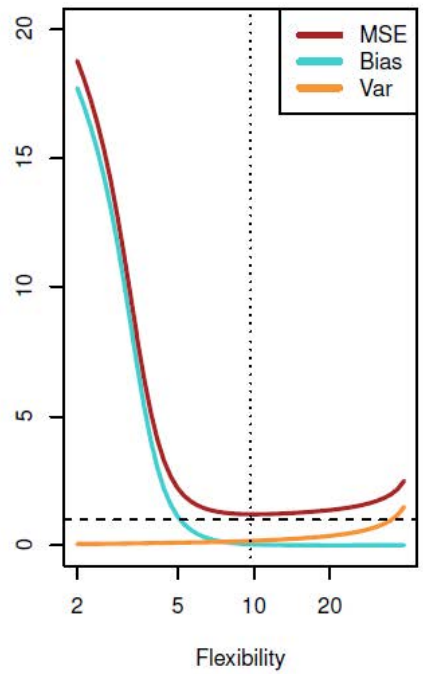
Bias-Variance Trade-Off

Example (Linear regression and nonlinear regression)

.







Testing Models

Testing Models

To compare the new predictions $\hat{y}_j = \hat{f}(x_{j1}, x_{j2}, \ldots, x_{jp})$ of **testing** data $\{(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j) : j = 1, 2, \ldots, m\}$ to y_j's. Use the testing error function

$$Err(f) = \frac{1}{m} \sum_{j=1}^{m} L(y_j, \hat{y}_j)$$

is an estimate of the prediction error $E(L(Y, \hat{f}(X)))$.

Cross-Validation

Cross-Validation Algorithms

⊗ *Cross-validation methods* are used when there is no access to external testing data other than the training data. And there is a need to select between different models. There are many of CV methods but the most common are:

- K-fold cross-validation
- Monte-Carlo cross-validation
- Generalized cross-validation

(Mostly from section 7.10and more to come later.)

Cross-Validation Algorithms

#1: K-fold cross-validation.

Step 1: Split the data into K parts. Call one of those parts, of N/K data points, a validation set $\mathcal{V}_i$ while the rest $K-1$ parts, of $N(1-1/K)$ data points, a training set $\mathcal{T}_i$.

Step 2: Use $\mathcal{T}_i$ to train the model and then use $\mathcal{V}_i$ to test it and calculate the prediction error

$$\frac{1}{N/K}\sum_{j\in\mathcal{V}_i} L(y_j, \hat{f}^{(-\mathcal{V}_i)}(X_j))$$

Step 3: Repeat step 2 for each i, for $i = 1, \ldots, K$.

Step 4: Finally, find the average of the resulting K errors

$$\begin{aligned} CV(\hat{f}) &= \frac{1}{K}\sum_{i=1}^{K}\frac{1}{N/K}\sum_{j\in\mathcal{V}_i} L(y_j, \hat{f}^{(-\mathcal{V}_i)}(X_j)) \\ &= \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{(-k(i))}(X_i)) \end{aligned}$$

Cross-Validation Algorithms

Example (5-fold cross-validation)

RUN 1	Validate	Train	Train	Train	Train
RUN 2	Train	Validate	Train	Train	Train
RUN 3	Train	Train	Validate	Train	Train
RUN 4	Train	Train	Train	Validate	Train
RUN 5	Train	Train	Train	Train	Validate

Example (leave-one-out cross-validation (LOOCV))

$K = N$-fold CV is called leave-one-out cross-validation. In that case, $\mathcal{T}_i = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N\}$ and $\mathcal{V}_i = \{x_i\}$ for $i = 1, \ldots, N$.

Cross-Validation Algorithms

#2: $\frac{1}{K}$100% Monte-Carlo cross-validation.

Step 1: Randomly select N/K data points from the whole set of N points. Call them a validation set $\mathcal{V}_i$ while the rest $N(1-1/K)$ data points are called a training set $\mathcal{T}_i$.

Step 2: Use $\mathcal{T}_i$ to train the model and then use $\mathcal{V}_i$ to test it and calculate the prediction error

$$\frac{1}{N/K}\sum_{j\in\mathcal{V}_i} L(y_j, \hat{f}^{(-\mathcal{V}_i)}(X_j))$$

Step 3: Repeat step 2 for a large number of times, say M.

Step 4: Finally, find the average of the resulting M errors.

$$CV(\hat{f}) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{N/K}\sum_{j\in\mathcal{V}_i} L(y_j, \hat{f}^{(-\mathcal{V}_i)}(X_j))$$

Cross-Validation Algorithms

Example (30% Monte-Carlo cross-validation)

For $N = 67$ and $\frac{1}{K}100\% = 30\%$, a number of $.3 \times N \approx 20$ randomly selected data points make a validation set and the rest are for training.

Cross-Validation Algorithms

#3: Generalized cross-validation (for LOOCV).

Step 1: Estimate the $N \times N$ matrix S through the linear fitting of

$$\hat{y} = Sy$$

Step 2: Find the effective number of parameters (or the effective degrees of freedom)

$$df(S) := trace(S) = \sum_{i=1}^{N} S_{ii}$$

Step 3: The generalized cross-validation is

$$GCV(\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{y_i - \hat{f}(x_i)}{1 - df(S)/N} \right]^2$$

Cross-Validation Algorithms

How does it work? With a tuning parameter α of the model f, define

$$CV(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{(-k(i))}(X_i; \alpha)).$$

or

$$GCV(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{y_i - \hat{f}(X_i; \alpha)}{1 - df(S)/N} \right]^2$$

We find

$$\hat{\alpha} = \text{argmin}_{\alpha} CV(\hat{f}, \alpha)$$

or

$$\hat{\alpha} = \text{argmin}_{\alpha} GCV(\hat{f}, \alpha)$$

Finally, re-run the training step for best-tuned model $f(x, \hat{\alpha})$ to fit all of the data.

Cross-Validation Algorithms

How does it work? With a tuning parameter α of the model f, define

$$CV(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{(-k(i))}(X_i; \alpha)).$$

or

$$GCV(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{y_i - \hat{f}(X_i; \alpha)}{1 - df(S)/N} \right]^2$$

We find

$$\hat{\alpha} = \text{argmin}_{\alpha} CV(\hat{f}, \alpha)$$

or

$$\hat{\alpha} = \text{argmin}_{\alpha} GCV(\hat{f}, \alpha)$$

Finally, re-run the training step for best-tuned model $f(x, \hat{\alpha})$ to fit all of the data.

End of Set 2