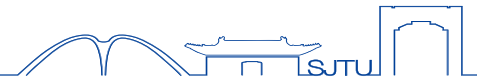
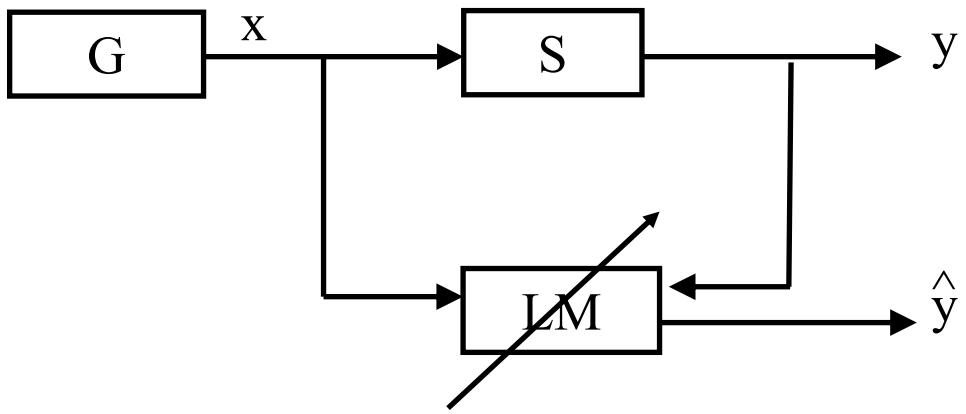
Overview of Supervised Learning



Contents

Notation

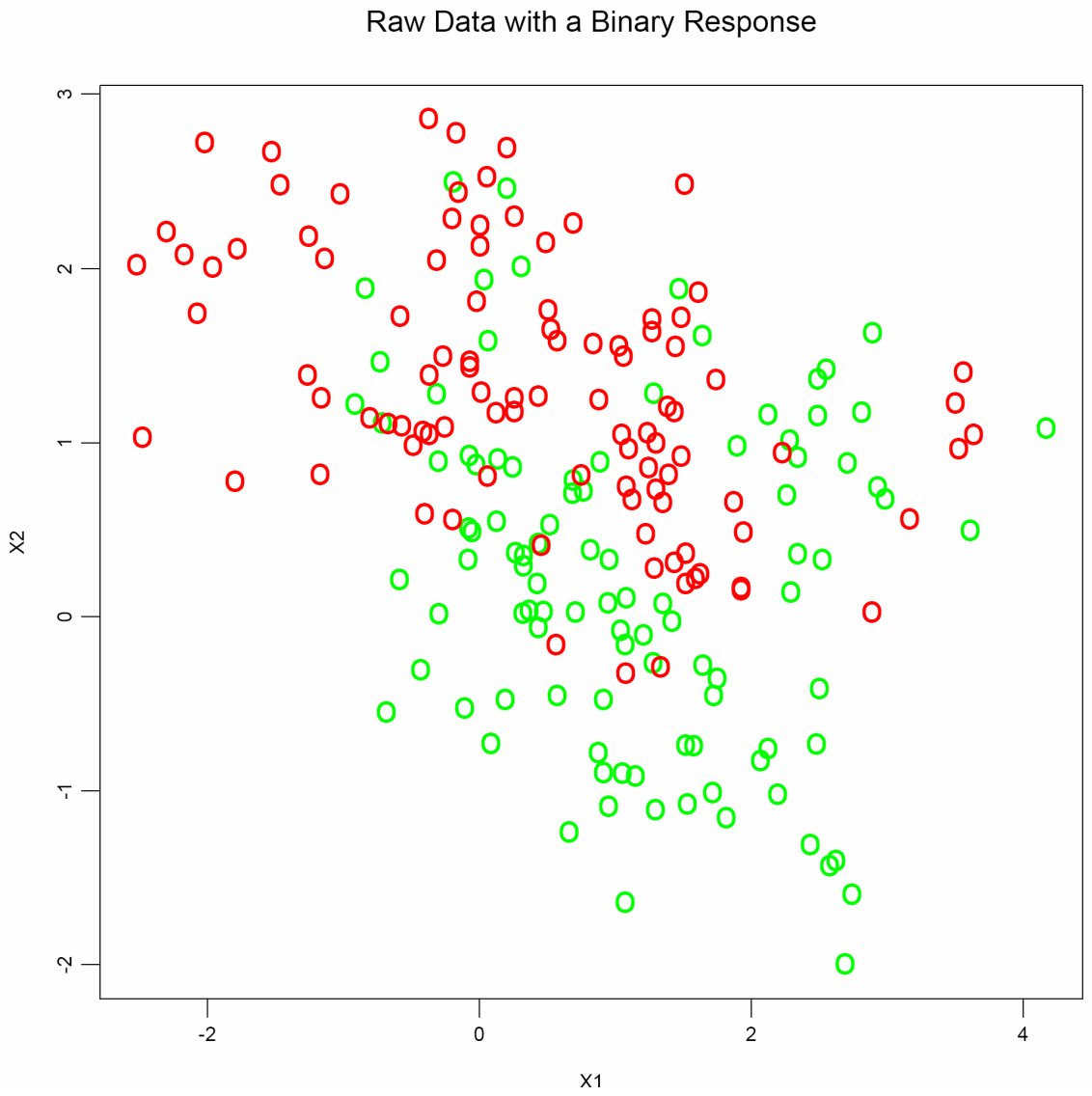
- X -- **inputs, feature vector, predictors**, independent variables. Generally X will be a vector of *p* values. Qualitative features are coded in X.
  - Sample values $D = \{x_i\}_{i=1}^{N}$, $x_i \in \mathrm{R}^p$ is i-th sample
- Y -- output, response, dependent variable.
  - Typically a scalar, can be a vector, of real values.
- G -- **a qualitative response**, taking values in a discrete set $\mathcal{G}$.
  e.g. G={ survived, died }.



Problem – Toy Model

- 200 points generated in IR^2 from a unknown distribution; 100 in each of two classes $\mathcal{G}$={ GREEN, RED }.

- Can we build a rule to predict the color of the future points?



Linear regression

- Code Y=1, if G=RED, else Y=0.
- We model Y as a linear function of X:

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j = X^T \hat{\beta}$$

- Obtain β by least squares, by minimizing the quadratic criterion:

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

- Given an $N \times p$ model matrix X and a response vector Y,

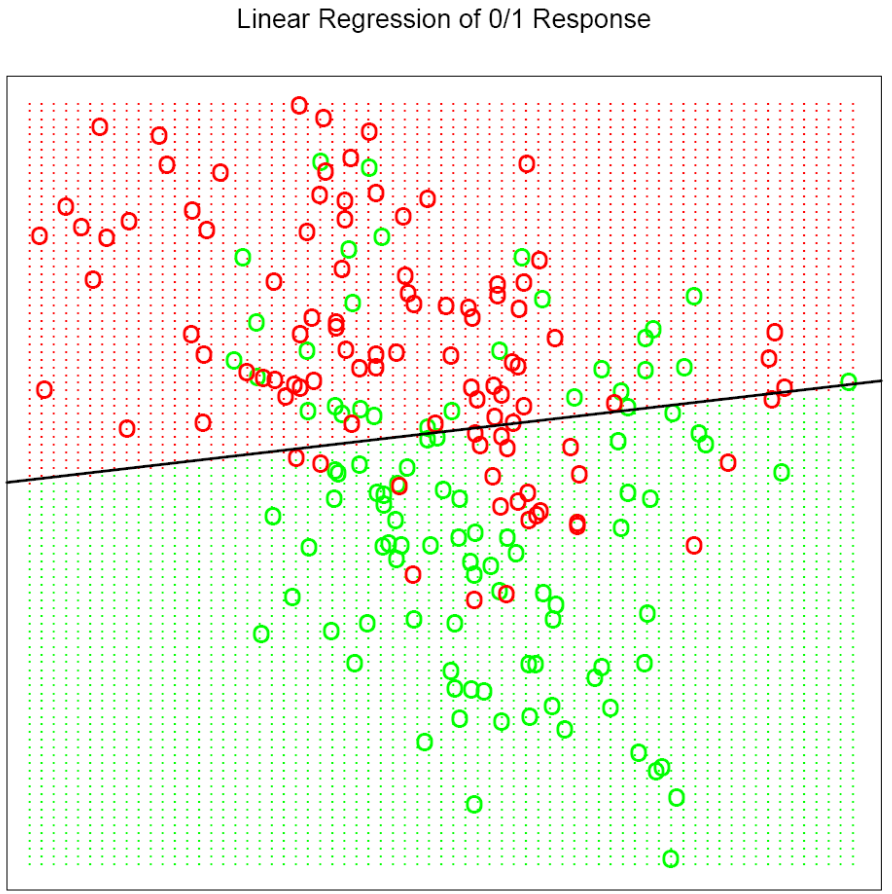
$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Linear regression

- Prediction at a future point x_0 is $\hat{Y}(x_0) = x_0^T \hat{\beta}$. Also

$$\hat{G}(x_0) = \begin{cases} \text{RED} & \text{if } \hat{Y}(x_0) > 0.5, \\ \text{GREEN} & \text{if } \hat{Y}(x_0) \leq 0.5. \end{cases}$$

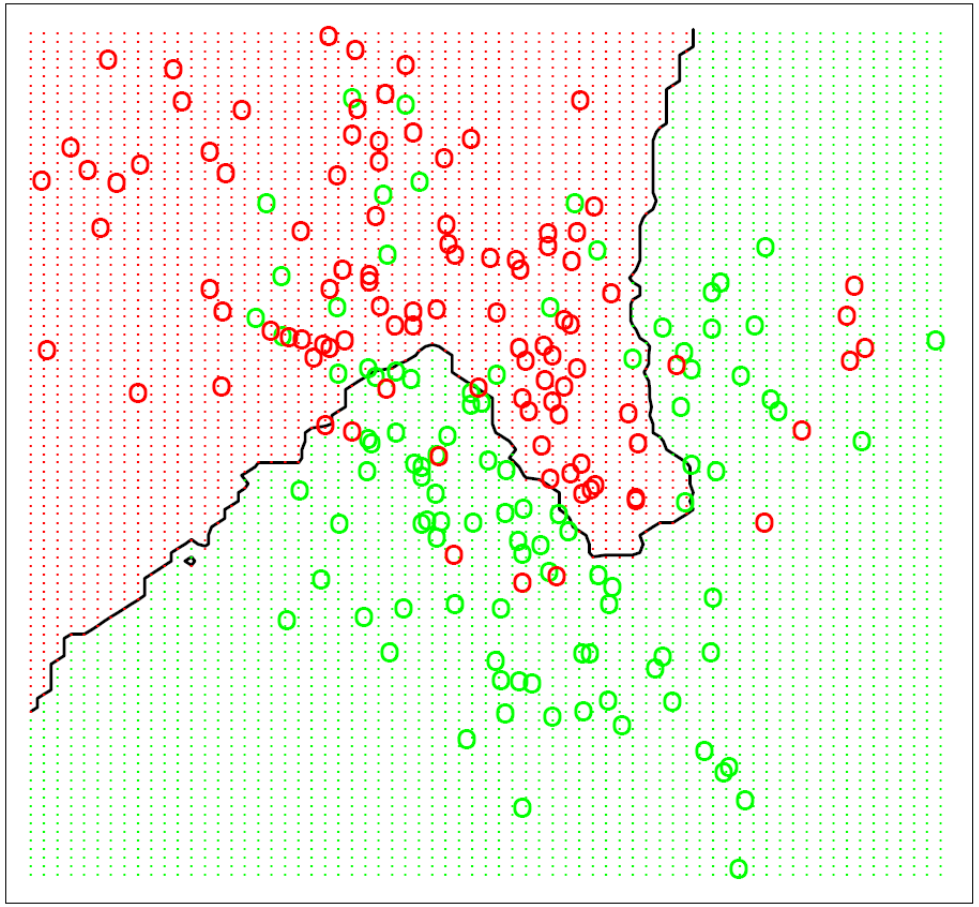
- The *decision boundary* is $\{X | X^T \hat{\beta} = 0.5\}$ is linear



K-Nearest Neighbors

- KNN method：The same classification examples in two dimensions as in the right Figure. The classes are coded as a binary variable (GREEN=0, RED=1).
- The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.

$$f(x_j)=\frac{1}{k}\sum_{x_i\in N_k(x_j)} y_i,\quad N_k(x_j)是x_j的k个近邻集合$$



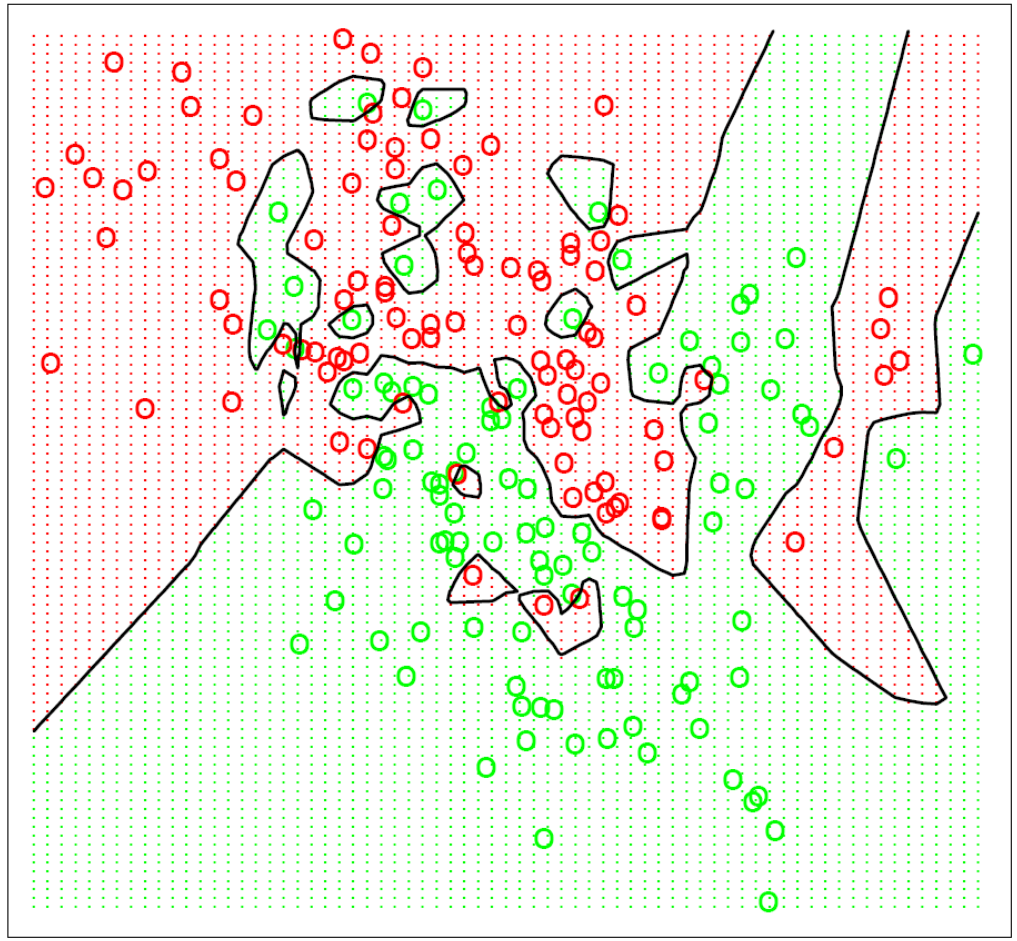
K-Nearest Neighbors

- KNN method using different size of ***k***.
- The same classification example are coded as a binary variable (GREEN=0, RED=1), and then predicted by

  1-nearest-neighbor classification.

- Question: What is the best choice of ***k***

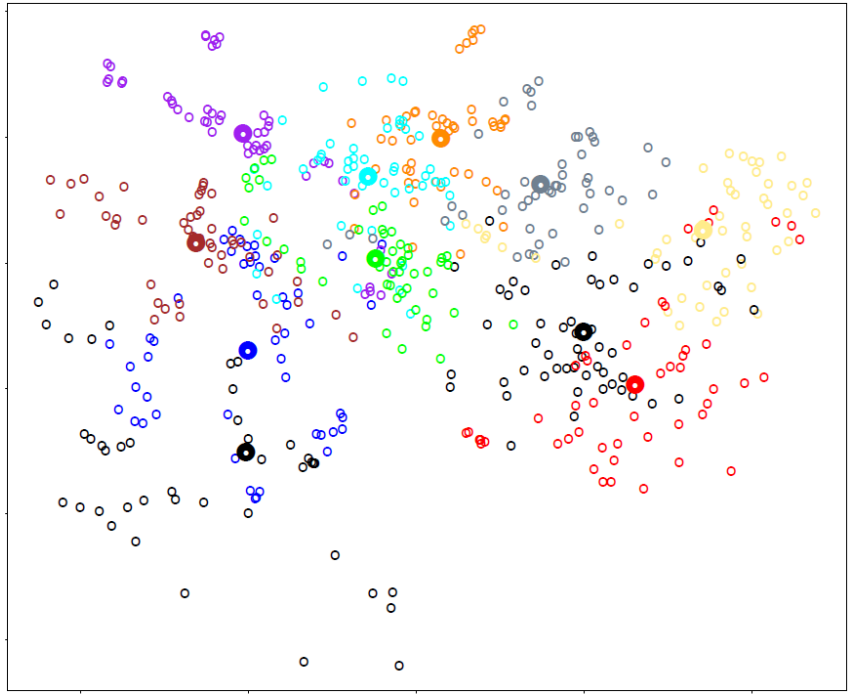
$$f(x_j) = \frac{1}{k} \sum_{x_i \in N_k(x_j)} y_i$$



Linear regression vs. k-NN

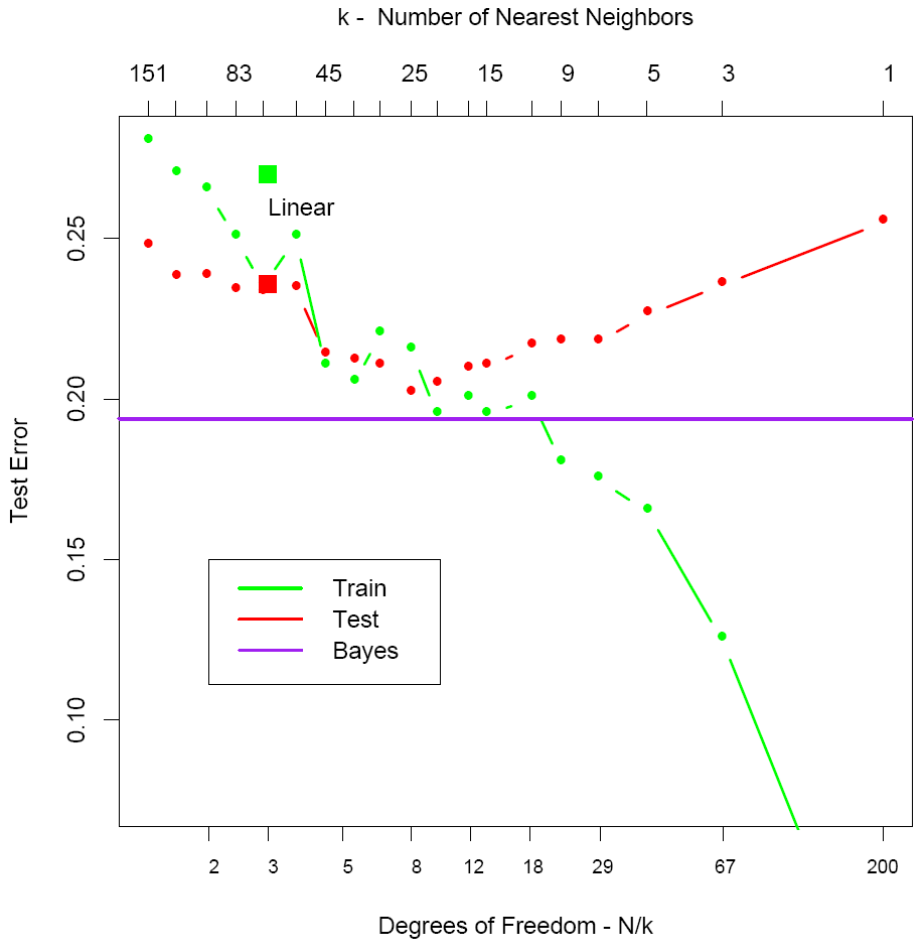
Linear regression vs k-nearest neighbors?

First we expose the oracle. The density for each class was an equal mixture of 10 Gaussians. For the GREEN class, its 10 means were generated from a $N((1,0)^T, \mathbf{I})$ distribution (and considered fixed). For the RED class, the 10 means were generated from a $N((0,1)^T, \mathbf{I})$. The within cluster variances were $1/5$.



Linear regression vs. k-NN

- Figure 2.4: Misclassification curves for the simulation example above. Sample size: 10,000
- The red curves are test and the green are training error for k-NN classification.
- The results for linear regression are the bigger green and red dots at three degrees of freedom. The purple line is the optimal Bayes Error Rate.

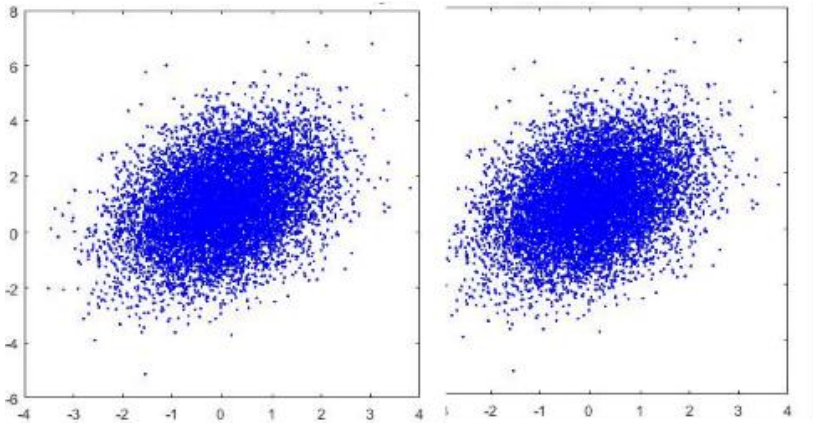


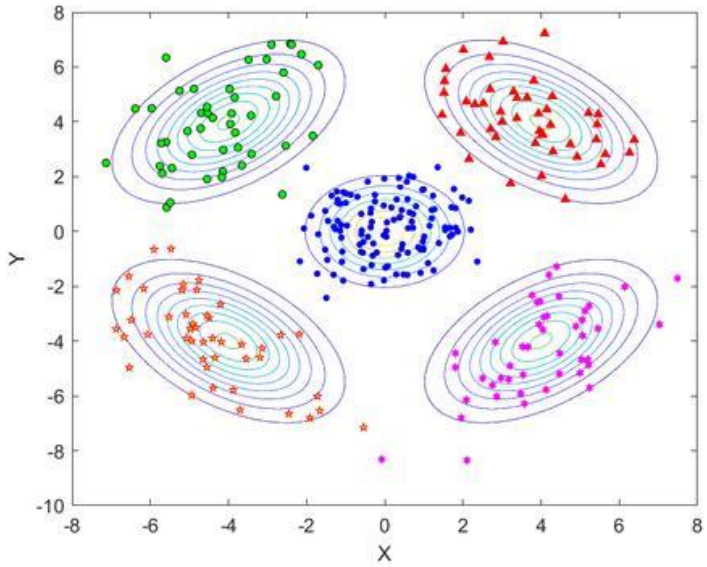
Possible scenarios

- **Scenario 1:** The data in each class are generated from a Gaussian distribution with uncorrelated components, same variances, and different means.
- **Scenario 2:** The data in each class are generated from a mixture of 10 gaussians in each class.

For Scenario 1, the linear regression rule is almost optimal (Chapter4).

For Scenario 2, it is far too rigid.





Statistical decision theory

Case 1: Quantitative output Y

Loss function $L(Y, f(X))$ for penalizing errors in prediction.

Most common and convenient loss is ***squared error loss*****:**

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

This leads us to a criterion for choosing f.

$$EPE(f) = E(Y - f(X))^2$$

Minimizin $EPE(f)$ leads to a solution:

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

Regression Function

$$
\begin{aligned}
EPE(f) &= E[Y - f(X)] \\
&= \int (y - f(x))^2 pr(dx, dy) \\
&= \int (y - f(x))^2 pr(dy \mid dx) pr(dx) \\
&= E_X E_{Y|X} \left([Y - f(X)]^2 \mid X \right)
\end{aligned}
$$

- 对EPE逐点极小化得： $f(x) = \arg\min_c E_{Y|X} \left([Y - c]^2 \mid X = x \right)$
- 极小解为：

$$
f(x) = E(Y \mid X = x)
$$

Case 2: Qualitative output G

- Suppose our prediction rule is $\hat{G}(X)$, and G and $\hat{G}(X)$ take values in $\mathcal{G}$.

- We have a different loss function for penalizing prediction errors $L(k,l)$ is the price paid for classifying an observation belonging to class $\mathcal{G}_k$ as $\mathcal{G}_l$.
- 0-1 loss function is a commonly used approach
- The expected prediction error is

$$EPE = E[L(G, \hat{G}(X))]$$

- Solution is

$$\hat{G}(x) = \text{argmin}_{g \in \mathcal{G}} \sum_{k=1}^{K} L\left(\mathcal{G}_k, g\right) P\left(\mathcal{G}_k \mid X = x\right)$$

Bayes Classifier

With the 0-1 loss function this simplifies to:

$$\hat{G}(x) = \mathcal{G}_k \text{ if } P\left(\mathcal{G}_k \mid x\right) = \max_{g \in \mathcal{G}} P(g \mid X)$$

This is known as the *Bayes classifier*. It just says that we should pick the class having maximum probability at the input x

Question: how do we construct the Bayes classifier for our simulation example?

- As $N, k \to \infty$, such that $k/N \to 0$, the K-nearest neighbor estimate $\hat{f}(x) \to E(Y \mid X = x)$ —— it is consistent

Question: Why not always use k-nearest neighbors ?

The Reason?

1. Due to the curse of high dimensional space?
2. The samples in high dimensional space are sparsely distributed. The K-nearest neighbors are actually **quite far** from x. Thus using the mean of KNN samples in high dimensional space is not able to capture the features of x

Curse of dimensionality

K-nearest neighbors can fail high dimensions, because it becomes difficult to gather observations close to a target point x_0

- near neighborhoods tend to be spatially large, and estimates are biased.
- reducing the spatial size of the neighborhood means reducing K, and the variance of the estimate increases.

- Most points are at the boundary
- Sampling density is proportional to $N^{1/p}$; if 100 points are sufficient to estimate a function in $\mathbb{R}^1$, 100^{10} are needed to achieve similar accuracy in $\mathbb{R}^{10}$

Curse of dimensionality

- The Volume of Cube of edge length l

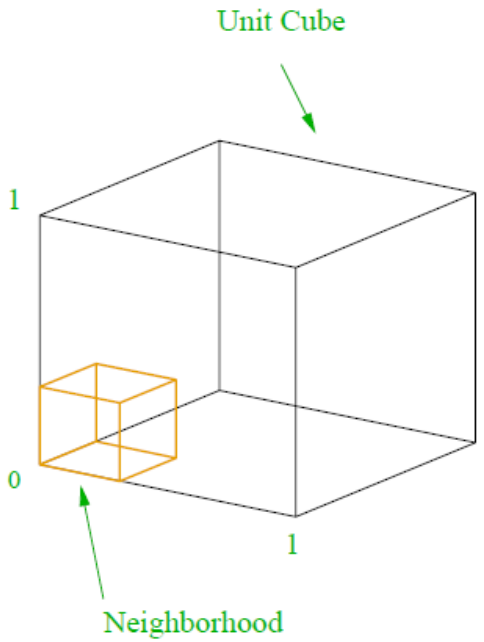
$$V = l^p$$

- For a fraction r of unit cube, the expected edge length

$$e_p(r) = r^{1/p}$$

$$e_{10}(0.01) = 0.63$$

$$e_{10}(0.1) = 0.8$$



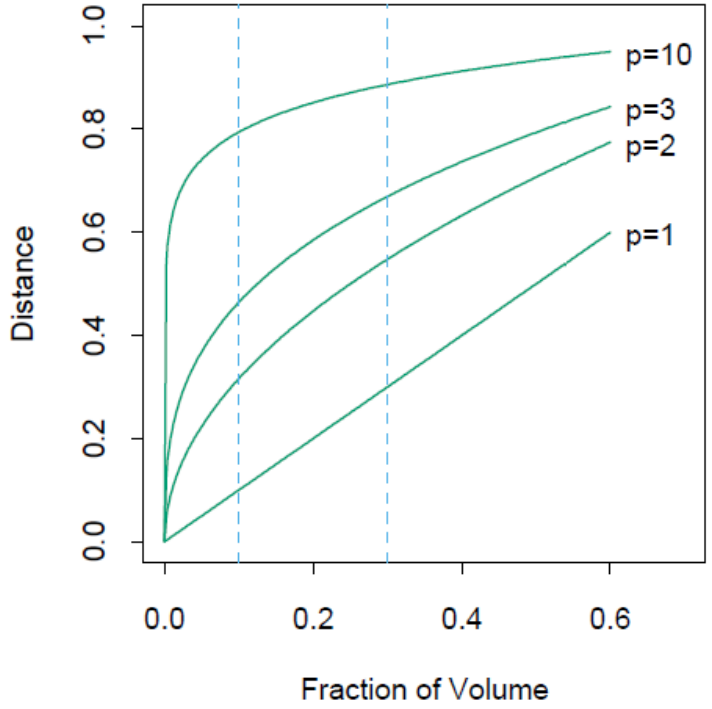


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.*

Local Methods in High Dim.

- The fewer observations we average, the higher is the variance of our fit.
- Another consequence of the sparse sampling in high dimensions is that all sample points are close to an edge of the unit ball.
- The median distance from the origin to the closest data point is given by the expression

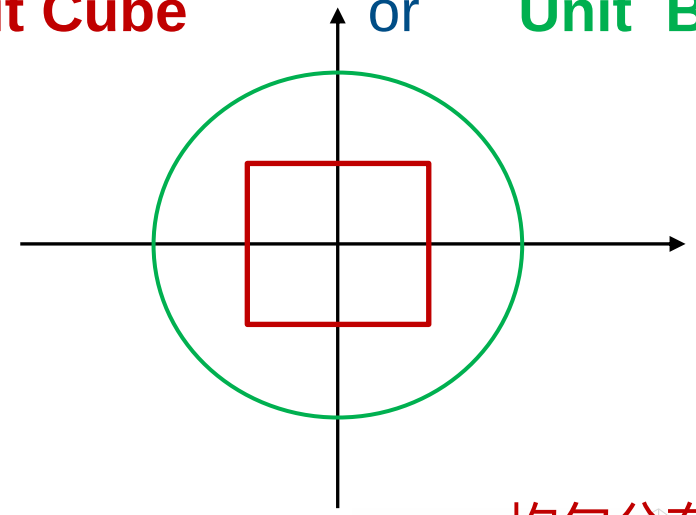
$$d(p,N) = \left(1 - \left(\frac{1}{2}\right)^{1/N}\right)^{1/p}$$

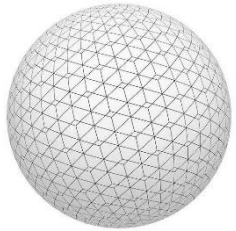
- For $N = 500$, $p = 10$, $d(p,N) \approx 0.52$, more than halfway to the boundary.

Data Distribution in High Dimensional Spaces

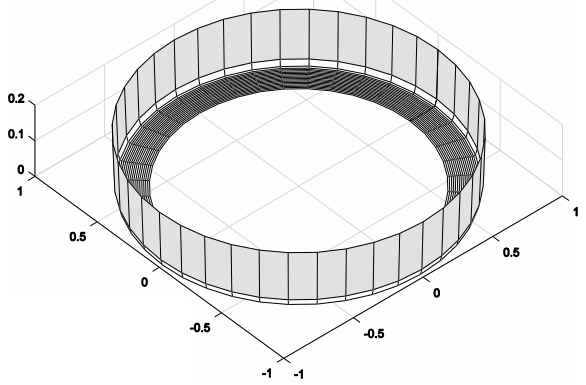
- **Problem 1:** Which one has bigger volume in high dimensional space ?

Unit Cube or **Unit Ball**



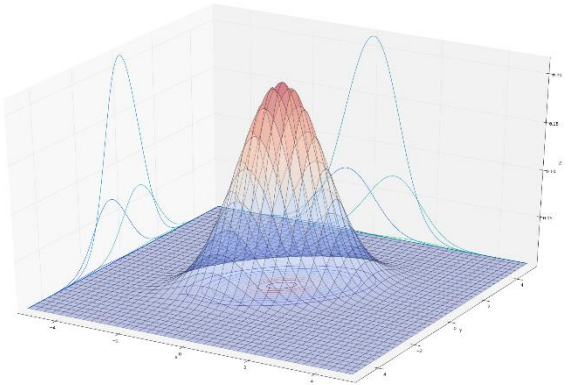


均匀分布

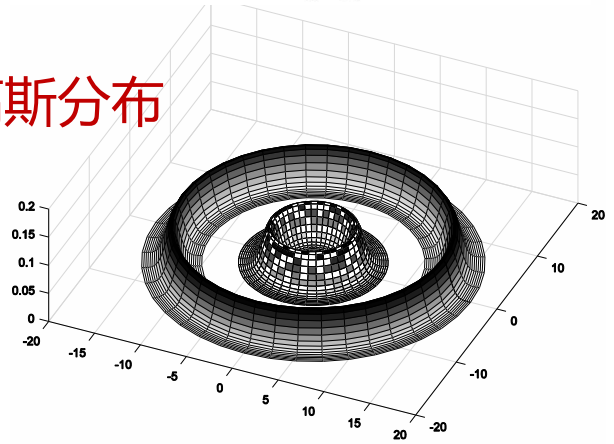


单位球内任意两点距离≈ 2

- **Problem 2:** Where is data mostly distributed for a Gaussian distribution in high dimensional spaces?



高斯分布



内圈 n=20；外圈 n=200，半径 $r=\sqrt{n-1}$

Error Decomposition

Example 1

- 1000 training examples x_i generated uniformly on [-1, 1]^p.
- $Y = f(X) = e^{-8\|X\|^2}$ (nomeasurement error)
- use the 1-nearest-neighbor rule predict y_0 at the test-point $x_0 = 0$

$$
\begin{aligned}
\mathrm{EPE}(x_0) &= E_T\left[f(x_0) - \hat{y}_0\right]^2 \\
&= E_T\left[f(x_0) - E(\hat{y}_0) + E(\hat{y}_0) - \hat{y}_0\right]^2 \\
&= \boxed{E_T\left[\hat{y}_0 - E_T(\hat{y}_0)\right]^2} + \boxed{E_T\left[E(\hat{y}_0) - f(x_0)\right]^2} \\
&\quad + 2E\left\{\left[\hat{y}_0 - E_T(\hat{y}_0)\right]\left[E(\hat{y}_0) - f(x_0)\right]\right\} \\
&= \boxed{\mathrm{Var}_T(\hat{y}_0)} + \boxed{\mathrm{Bias}^2(\hat{y}_0)}
\end{aligned}
$$

Error Decomposition

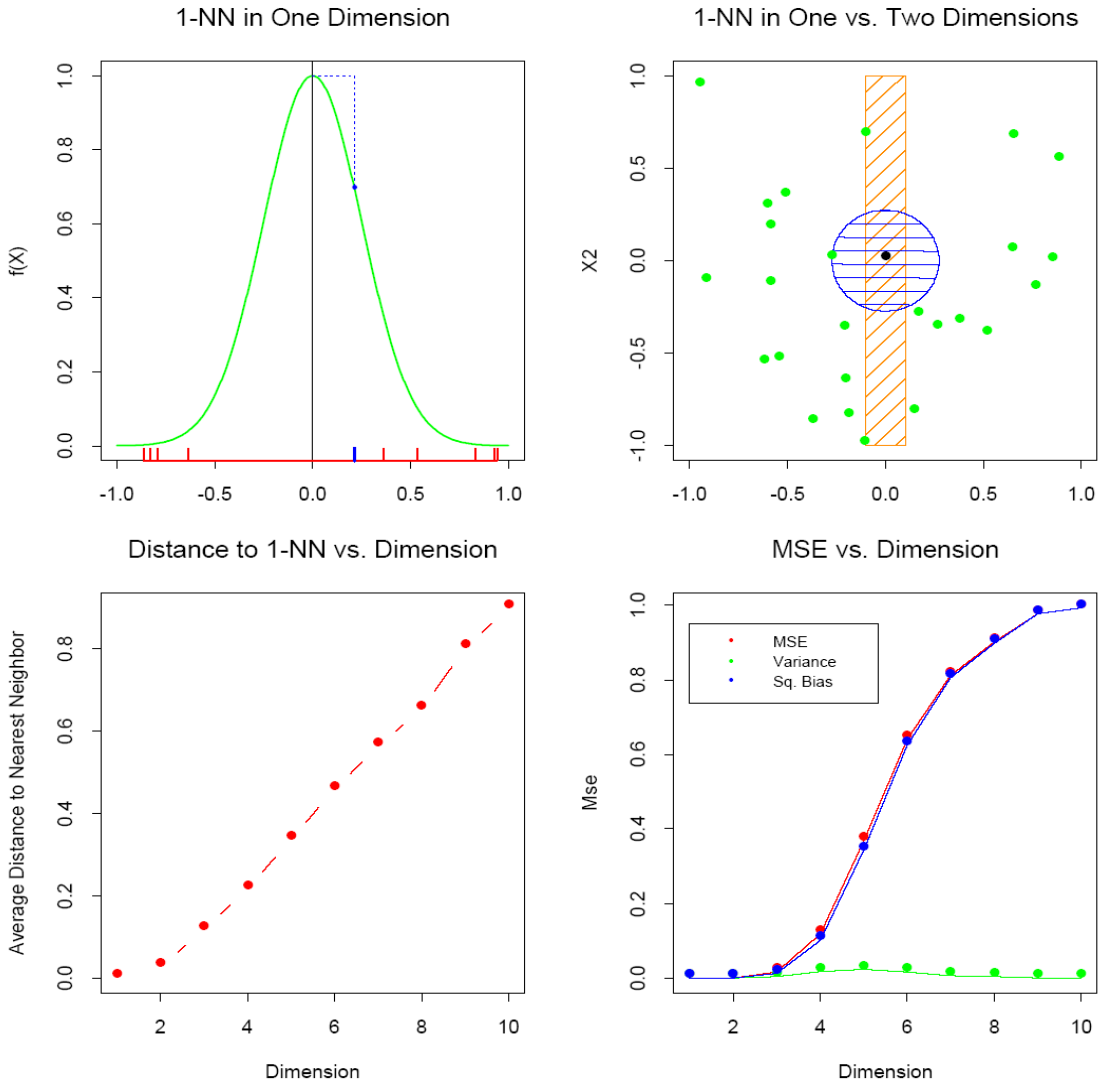


Figure 2.7: A simulation example, demonstrating the curse of dimensionality and its effect on MSE, bias and variance. The input features are uniformly distributed in $[-1,1]^p$, for $p = 1, ..., 10$.

Linear Model

- Linear Model $$Y = X^T \beta + \varepsilon$$
- Linear Regression $$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$
- Test error $$\hat{y} = x_0^T \hat{\beta} = x_0^T \beta + \sum_{i=1}^{N} l_i(x_0) \varepsilon_i$$

$l_i(x_0)$ -- the i-th component of $X(X^T X)^{-1} x_0$

Linear Model

- Linear Model $Y = X^T \beta + \varepsilon$
- Linear Regression Solution $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
- The expected prediction error

$$\begin{aligned}
\mathrm{EPE}(x_0) &= \mathrm{E}_{y_0|x_0}\mathrm{E}_T\left[y_0 - \hat{y}_0\right]^2 \\
&= \mathrm{Var}(y_0 \mid x_0) + \mathrm{E}_T\left[\hat{y}_0 - \mathrm{E}_T\hat{y}_0\right]^2 + \left[\mathrm{E}_T\hat{y}_0 - x_0^T\beta\right]^2 \\
&= \mathrm{Var}(y_0 \mid x_0) + \mathrm{Var}_T\left[\hat{y}_0\right]^2 + Bias^2(\hat{y}_0) \\
&= \sigma_\varepsilon^2 + \mathrm{E}_T\left[x_0^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}x_0\right]\sigma_\varepsilon^2 + 0
\end{aligned}$$

Linear Model

- Linear Model $Y = X^T \beta + \varepsilon$

- Linear Regression Solution $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

If N is large and T were selected at random, E[X]=0, then we have

$$\mathbf{X}^T\mathbf{X} \to NCov(X)$$

$$\begin{aligned}
\mathrm{E}_{x_0}\mathrm{EPE}(x_0) &= \sigma_\varepsilon^2 + \mathrm{E}_{x_0}\mathrm{E}_T\left[x_0^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}x_0\right]\sigma_\varepsilon^2 \\
&= \sigma_\varepsilon^2 + \mathrm{E}_{x_0}\left[x_0^T Cov(X)^{-1}x_0\right]\sigma_\varepsilon^2 / N \\
&= \sigma_\varepsilon^2 + trace\left[Cov(X)^{-1}Cov(x_0)\right]\sigma_\varepsilon^2 / N \\
&= \sigma_\varepsilon^2 + \sigma_\varepsilon^2(p/N)
\end{aligned}$$

Classes of Restricted Estimators

Some of the classes of restricted methods that we cover are

- Roughness Penalty and Bayesian Methods

$$\mathrm{PRSS}(f, \lambda) = \mathrm{RSS}(f) + \lambda J(f)$$

- Kernel Methods and Local Regression

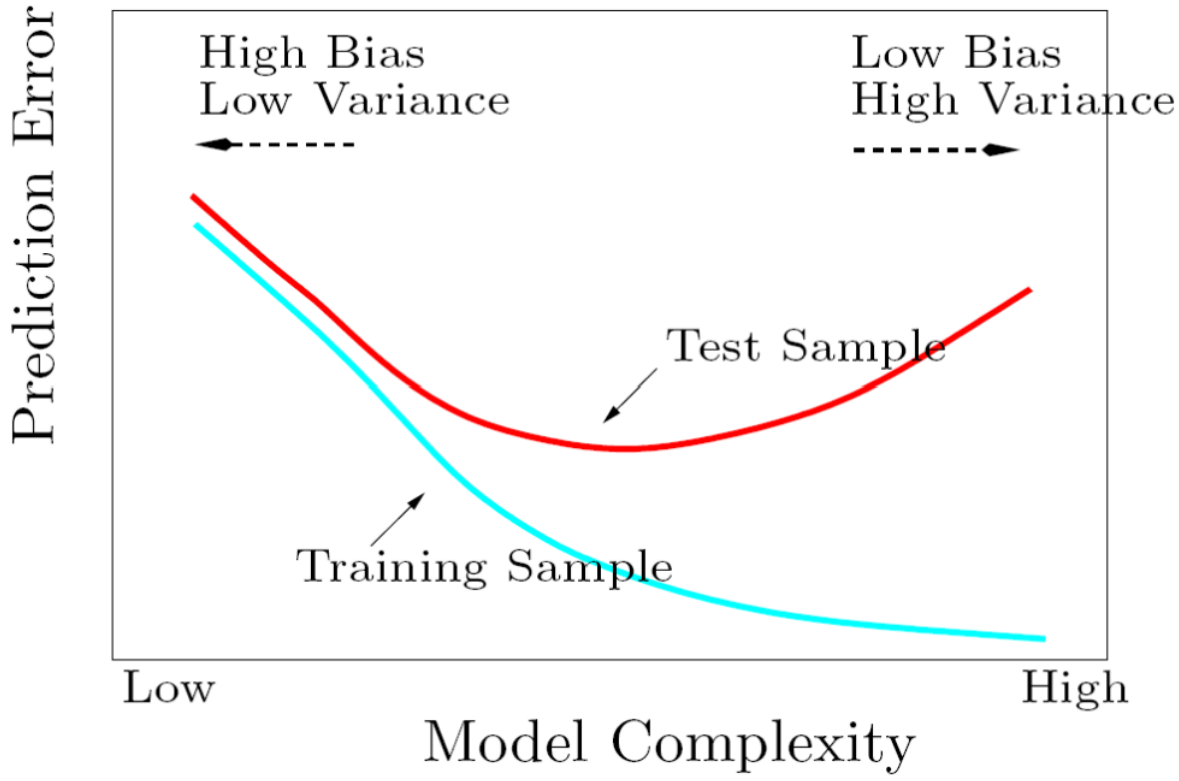
$$\mathrm{RSS}\left(f_\theta, x_0\right) = \sum_{i=1}^{N} K_\lambda\left(x_0, x_i\right)\left(y_i - f_\theta\left(x_i\right)\right)^2$$

- Basis functions and dictionary methods

$$f_\theta(x) = \sum_{m=1}^{M} \theta_m h_m(x)$$

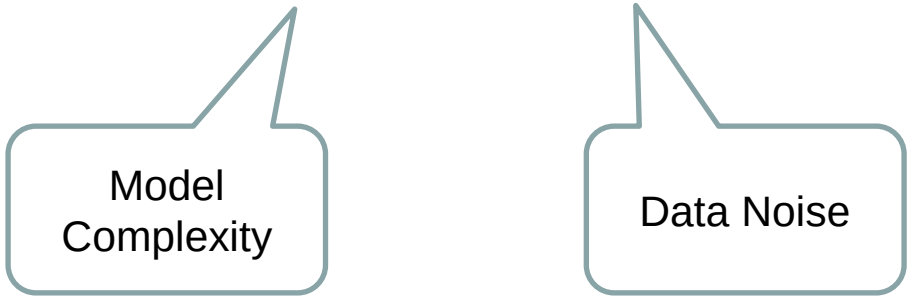
Model Selection & the Bias-Variance Tradeoff

- Test and training error as a function of model complexity.



Key Points

- The objective of statistical learning is to identify the model with **best generalization performance** or **with minimum training error**?
- In what conditions, linear regression is the best as a classifier?
- KNN is one of implementations of the optimal decision function, but why we do not use it as a classifier in high dimensional space?
- Generalization error = Model Bias^2 + Variance



THE END

Ex.

Problem 1

- For uniform distribution, the median distance from the origin to the closest data point is given by the expression

$$d(p,N)=\left(1-\left(\frac{1}{2}\right)^{1/N}\right)^{1/p}$$

- For $\mathrm{N} = 500$, $\mathrm{p} = 10$, $\mathrm{d(p,N)} \approx 0.52$, more than halfway to the boundary.

Problem 2

- $D=\{x_i\}_{i=1}^{N}$, $x_i \in \mathrm{R}^p$ is i-th sample. If the data follow Gaussian distribution, then the maximum likelihood estimators for the mean and covariance are given by

$$\hat{\mu}=\frac{1}{N}\sum_{i=1}^{N}x_i, \qquad \hat{\Sigma}=\frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{\mu})(x_i-\hat{\mu})^T$$