Statistical Machine Learning

Lecture 1: Introduction and Overview of Supervised Learning

W.Q.Cui Research Group

Department of Statistics and Finance
University of Science and Technology of China

2018 Spring

Books and References

- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. **The elements of statistical learning**. Vol. 1. New York: Springer series in statistics, 2001.
- V. N. Vapnik, **Statistical Learning Theory**, 1998, John Wiley & Sons.
- V. N. Vapnik, **The Nature of Statistical Learning Theory**, 2000, 2nd ed.,Springer.
- B. Schölkopf and A. J. Smola, **Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond**, 2002, MIT. Press.
- J Shawe . Shawe-Taylor and N Cristianini . Cristianini, **Kernel Methods for Pattern Analysis**, 2004, Cambridge University Press.
- P. N. Tan, M. Steinbach and V. Kumar. **Introduction to Data Mining**,2005, Addison Wesley.
- C. M. Bishop, **Pattern Recognition and Machine Learning**, 2006,Springer.

Contents

Applications of Statistical Learning

Statistical learning plays a key role in many areas of science, finance and industry. Here are some examples of learning problems:

- Medical: Predicted whether a patient, hospitalized due to a heart attack will have a second heart attack.
  Data: demographic, diet, clinical measurements
- Business/Economics: Predict the price of stock 6 months from now.
  Data: company performance, economic data
- Vision: Identify hand-written ZIP codes
  Data: Model hand-written digits
- Medical:Amount of glucose in the blood of a diabetic
  Data: Infrared absorption spectrum of blood sample
- Medical: Risk factors for prostate cancer
  Data: Clinical, demographic

Types of Data

- Two basically different types of data
  - Quantitative (numerical): e.g. stock price
  - Categorical (discrete, often binary): cancer/no cancer
- Data are predicted
  - on the basis of a set of features (e.g. diet or clinical measurements)
  - from a set of (observed) training data on these features
  - For a set of objects (e.g. people).
  - Inputs for the problems are also called predictors or independent variables
  - Outputs are also called responses or dependent variables
- The prediction model is called a learner or estimator
  - Supervised learning: learn on outcomes for observed features
  - Unsupervised learning: only the feature values available and have no measurements of the outcome

Example 1: Email Spam

- Data:
  - 4601 email messages each labeled email (+) or spam (-)
  - The relative frequencies of the 57 most commonly occurring words and punctuation marks in the message
- Prediction goal: label future messages email (+) or spam (-)
- Supervised learning problem on categorical data: classification problem

	spam	email
george	0.00	1.27
you	2.36	1.27
your	1.38	0.44
hp	0.02	0.90
free	0.52	0.07
hpl	0.01	0.43
!	0.51	0.11
our	0.51	0.18
re	0.13	0.42
edu	0.01	0.29
remove	0.28	0.01

Table: Words with largest difference between spam and email shown.

Example 1: Email Spam

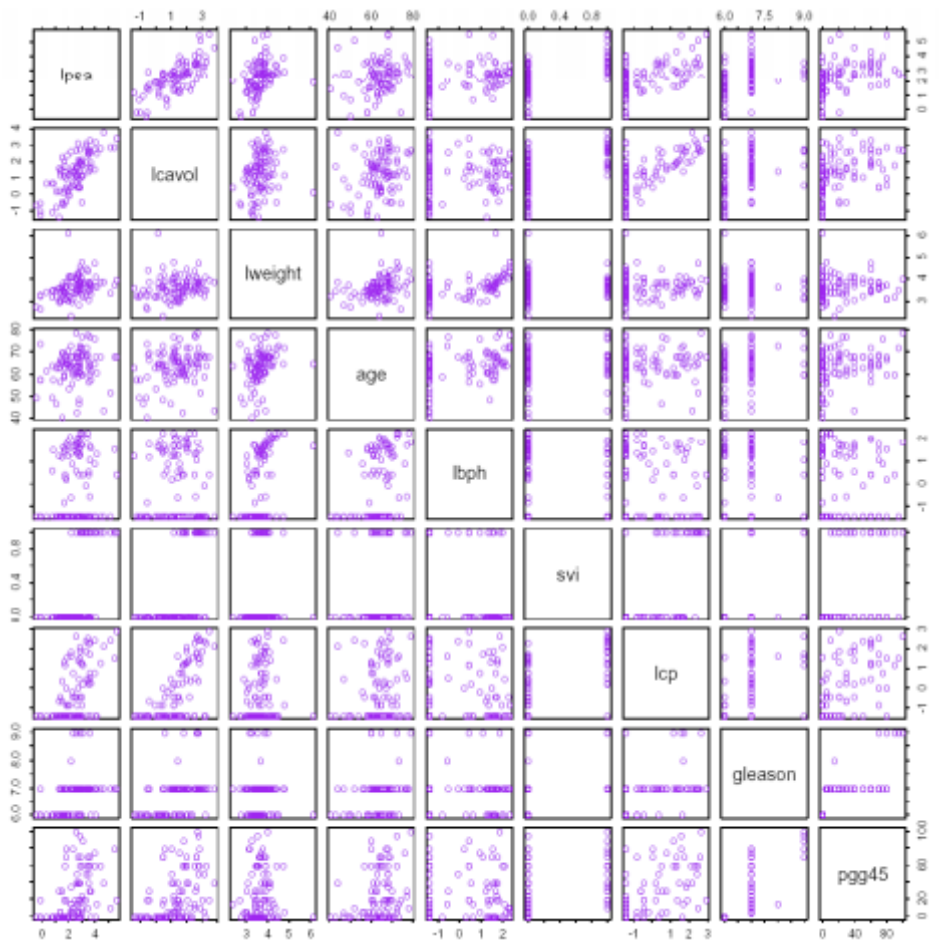
- Examples of rules for prediction:
  - If $(\%george < 0.6)$ and $(\%you > 1.5)$ then spam else email
  - If $(0.2\%you - 0.3\%george) > 0$ then spam else email
- Tolerance to errors:
  - Tolerant to letting through some spam (false positive)
  - No tolerance towards throwing out email (false negative)

	spam	email
george	0.00	1.27
you	2.36	1.27
your	1.38	0.44
hp	0.02	0.90
free	0.52	0.07
hpl	0.01	0.43
!	0.51	0.11
our	0.51	0.18
re	0.13	0.42
edu	0.01	0.29
remove	0.28	0.01

Table: Words with largest difference between spam and email shown.

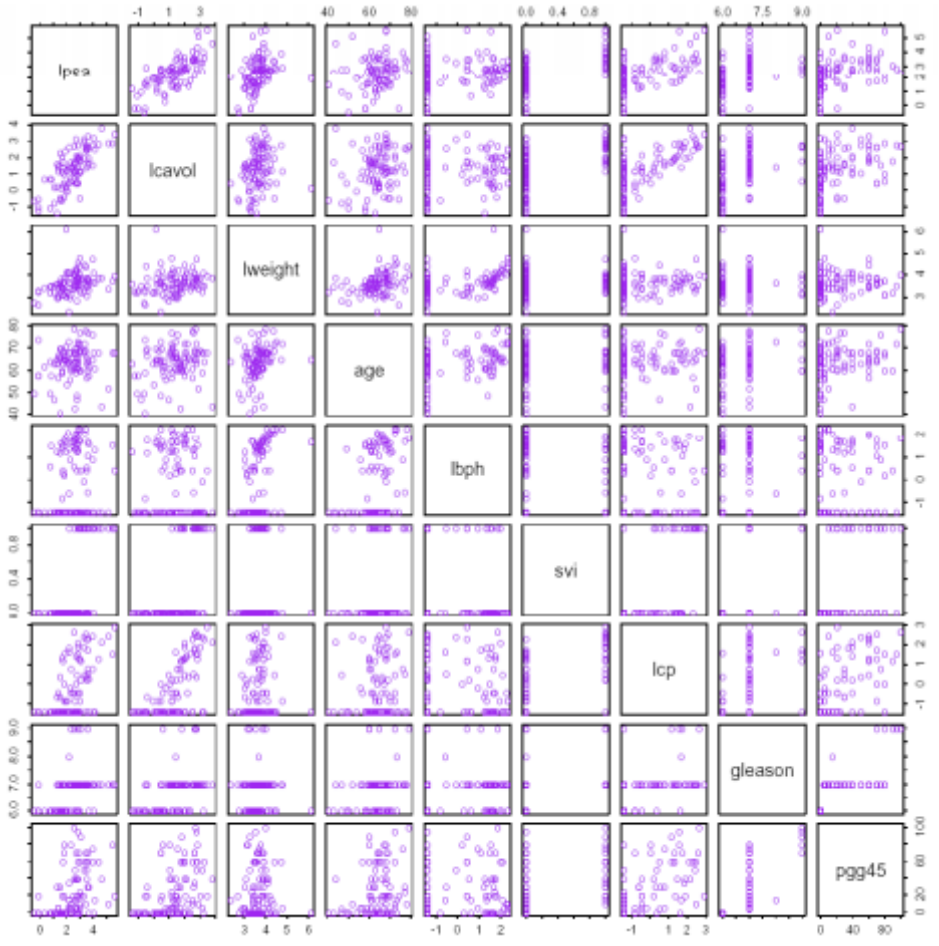
Example 2: Prostate Cancer

- Data(Stamey et al. 1989):
  lcavol log cancer volume
  lweight log prostate weight
  age
  lbph log benign hyperplasia amount
  svi seminal vesicle invasion
  lcp log capsular penetration
  gleason gleason score
  pgg45 percent gleason scores 4 or 5
- Predict:PSA (prostate specific antigen) level
- Supervised learning problem on quantitative data:
  regression problem



Example 2: Prostate Cancer

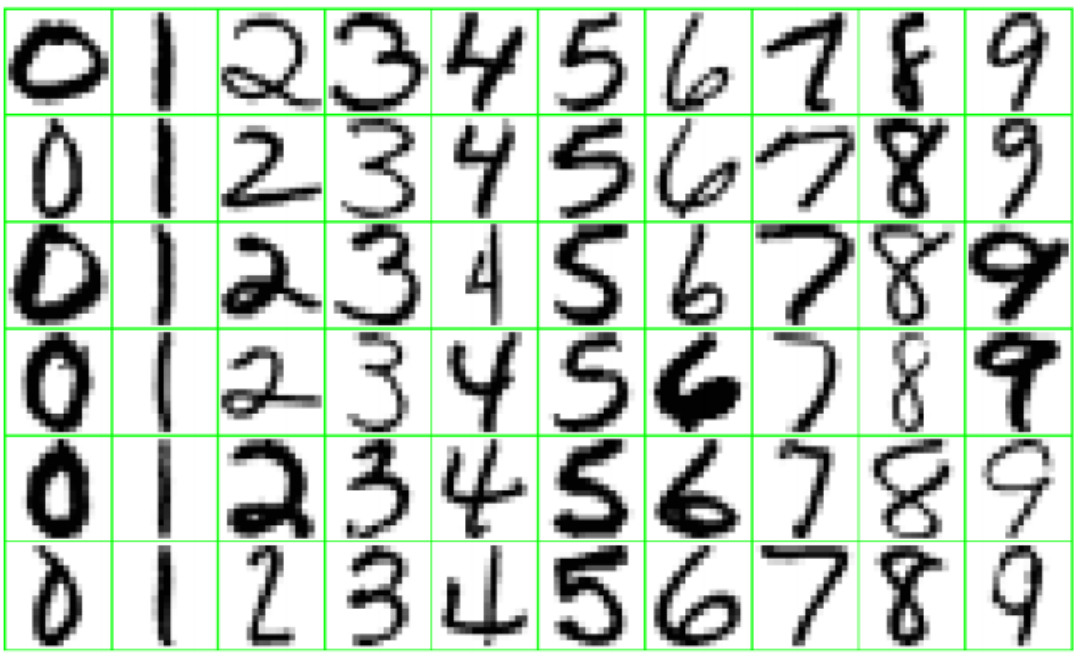
- Figure shows scatter plots of the input data, projected onto two variables, respectively.
- The first row shows the outcome of the prediction, projected onto each input variable, respectively.
- The variables svi and gleason are categorical



Example 3: Recognition of Handwritten Digits

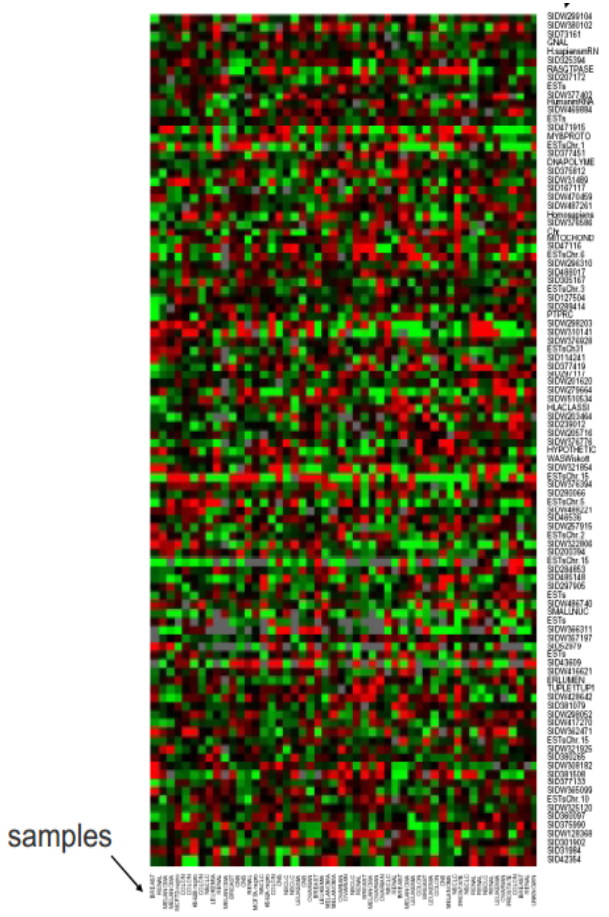
- Data:images are single digits 16×16 8-bit gray-scale, normalized for size and orientation
- Classify: newly written digits

- Non-binary classification problem
- Low tolerance to misclassification



Example 4: DNA Expression Microarrays

- Data:
  - Color intensities signifying the abundance levels of mRNA for a number of genes (6830) in several(64) different cell states(samples).
  - Red over-expressed gene
  - Green under-expressed gene
  - Black normally expressed gene(according to some predefined background)
- Predict:
  - Which genes show similar expression over the samples
  - Which samples show similar expression over the genes(unsupervised learning problem)
  - Which genes are highly over or under expressed in certain cancers(supervised learning problem)



Notation

- Inputs
  - X, X_j(j-th element of vector X)
  - p#inputs, N#observations
  - $\boldsymbol{X}$ matrix written in bold
  - Vectors written in bold $\boldsymbol{x}_i$ if they have N components and thus summarize all observations on variable X_i
  - Vectors are assumed to be column vectors
  - Discrete inputs often described by characteristic vector (dummy variables)
- Outputs
  - quantitative Y
  - qualitative G (for group)
- Observed variables in lower case
  - The i-th observed value of $\boldsymbol{X}$ is x_i and can be a scalar or a vector
- Main question of this lecture:
  - Given the value of an input vector X, make a good prediction $\hat{Y}$ of the output Y
  - Prediction should be of the same kind as the searched output (categorical vs quantitative)
  - Exception: Binary outputs can be approximated by values in $[0, 1]$, which can be interpreted as probabilities. This generalizes to k-level outputs.

Simple Approach 1: Least-Squares

- Given inputs $X = (X_1, X_2, \cdots, X_p)$
- Predict output Y via the model

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

- Include the constant variable $\mathbf{1}$ in X

$$\hat{Y} = X^T \hat{\beta}$$

- Here Y is scalar
- If Y is a K-vector then X is a $p \times K$ matrix

- In the $(p+1)$-dimension input-output space, $(X, \hat{Y})$ represents a hyperplane
- If the constant is included in X, then the hyperplane goes through the origin

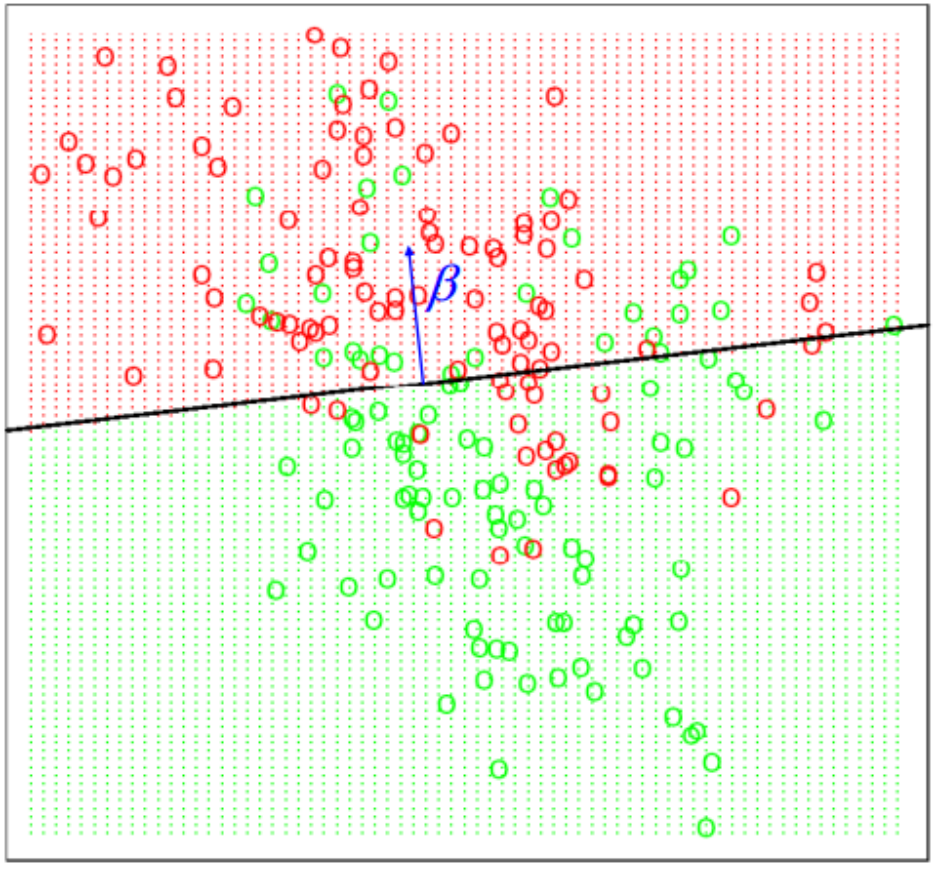
$$f(X) = X^T \beta$$

is a linear function

$$f'(X) = \beta$$

is a vector that points in the steepest uphill direction

Simple Approach 1: Least-Squares



- In the $(p+1)$-dimension input-output space, $(X, \hat{Y})$ represents a hyperplane
- If the constant is included in X, then the hyperplane goes through the origin

$$f(X) = X^T \beta$$

is a linear function

$$f'(X) = \beta$$

is a vector that points in the steepest uphill direction

Simple Approach 1: Least-Squares

- Training procedure: Method of least-square
- $N = \#$observations
- Minimize the residual sum of squares

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

- Or equivalently

$$RSS(\beta) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta)$$

- This quadratic function always has a global minimum, but it may not be unique

- Differentiating $w.r.t. \beta$ yields the normal equations

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

- If $X^T X$ is nonsingular, then the unique solution is

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

- The fitted value at input x is

$$\hat{y}(x) = x^T \hat{\beta}$$

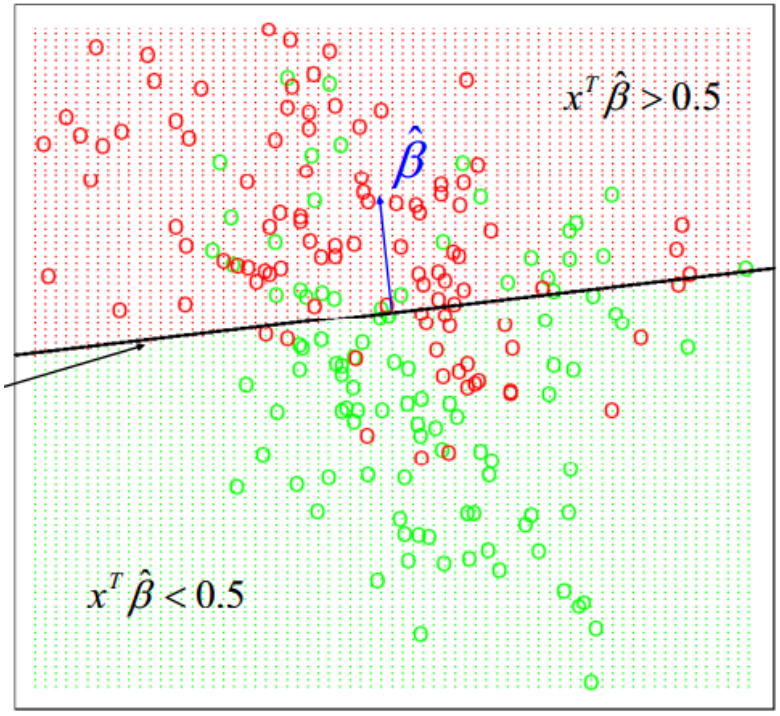
- The entire surface is characterized by $\hat{\beta}$

Simple Approach 1: Least-Squares

- Data on two inputs X_1 and X_2
- Output variable has values GREEN(coded 0) and RED(coded 1)
- 100 points per class
- Regression line is defined by

$$x^T \hat{\beta} = 0.5$$

- Easy but many misclassifications if the problem is not linear

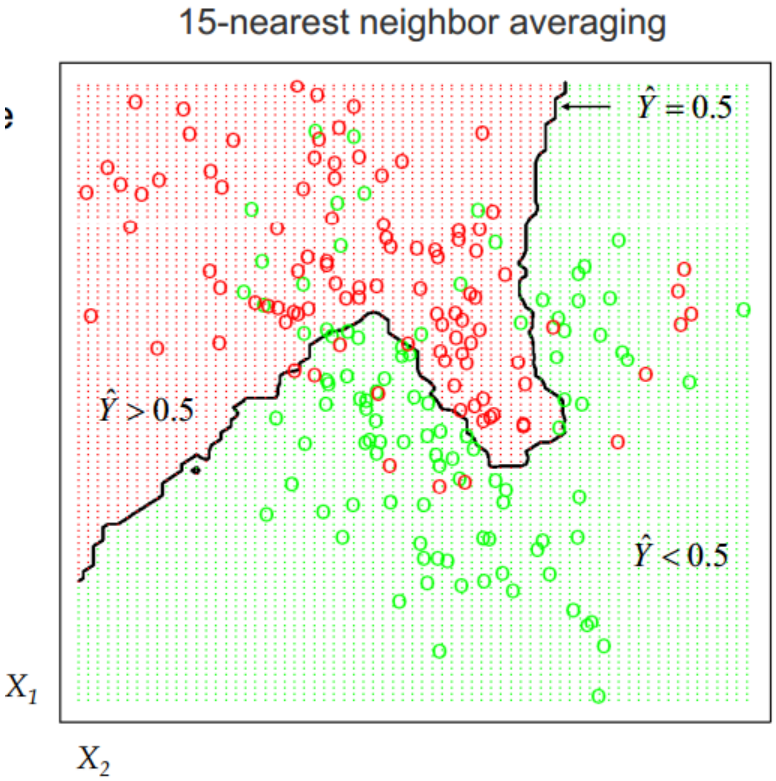


Simple Approach 2: Nearest Neighbors

- Uses those observations in the training set closest to the given input.

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

- $N_k(x)$ is the set of the k closest points to x is the training sample
- Average the outcome of the k closest training sample points
- Fewer misclassifications

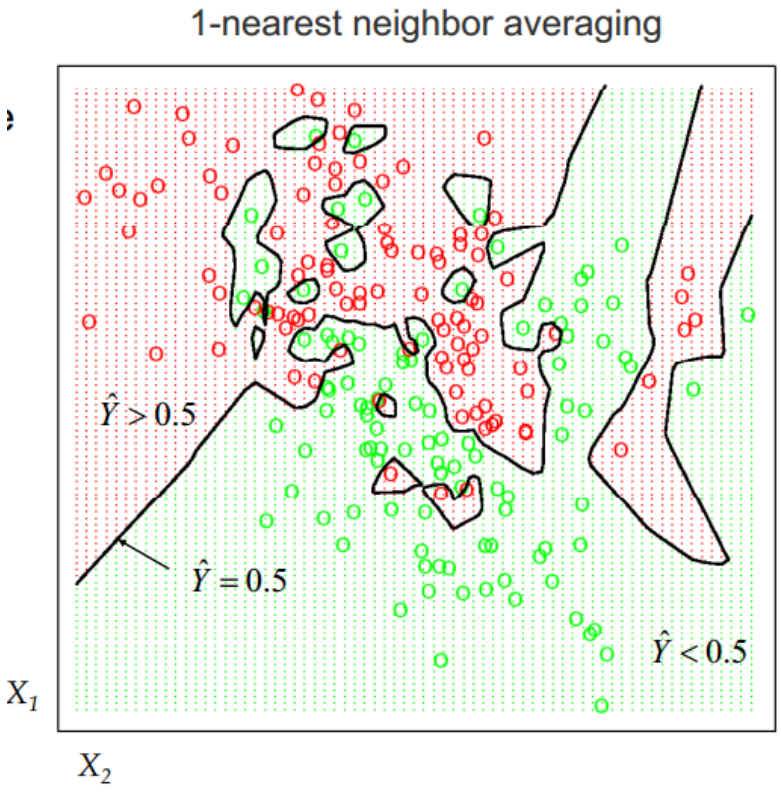


Simple Approach 2: Nearest Neighbors

- Uses those observations in the training set closest to the given input.

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

- $N_k(x)$ is the set of the k closest points to x is the training sample
- Average the outcome of the k closest training sample points
- No misclassifications: Overtraining

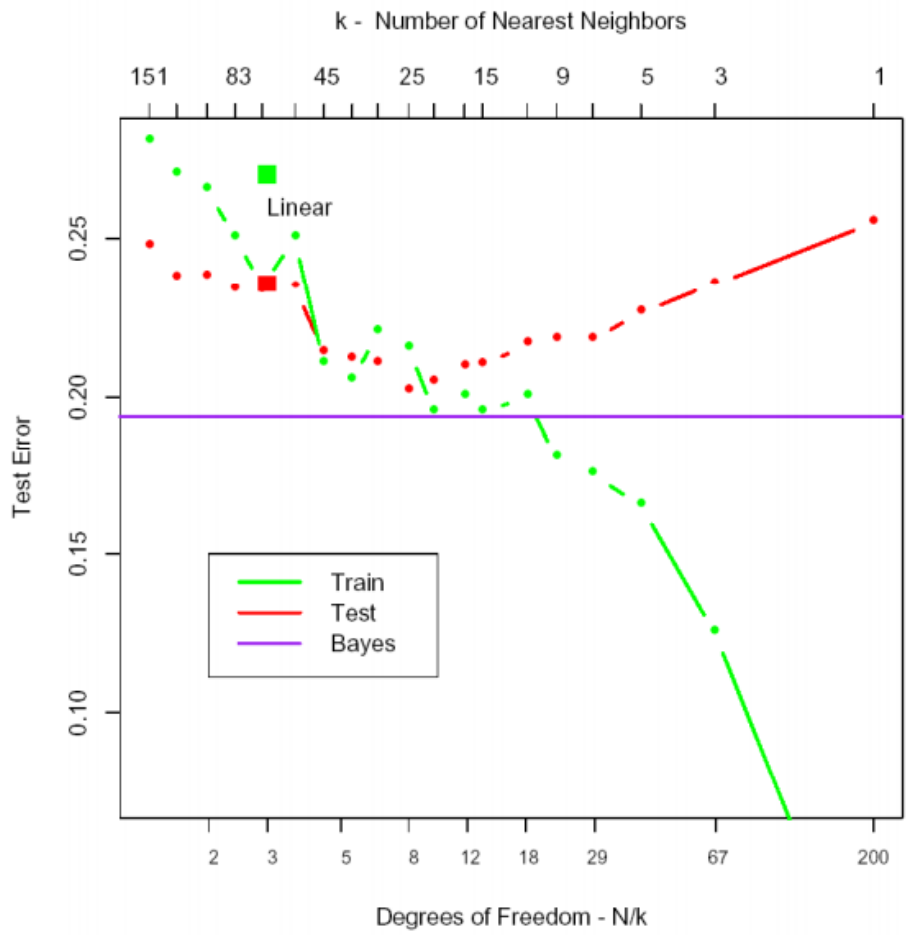


Comparison of the Two Approaches

- Least squares
- p parameters
  $p = \#\text{features}$
- Low variance (robust)
- High bias (rests on strong assumptions)
- Good for Scenario 1:
  Training data in each class generated from a two-dimensional Gaussian, the two Gaussians are independent and have different means
- K-nearest neighbors
- Apparently one parameter k. In fact N/k parameters.
  $N = \#\text{observations}$
- High variance (not robust)
- Low bias (rests only on weak assumptions)
- Good for Scenario 2:
  Training data in each class from a mixture of 10 low-variance Gaussians, with means again distributed as Gaussian

Simple Approach 2: Nearest Neighbors

- step 1:
  Generate 10 means m_k from the bivariate Gaussian distribution $N((1,0)^T, \boldsymbol{I})$ and label this class GREEN
- step 2:
  Similarly, generate 10 means from the bivariate Gaussian distribution $N((0,1)^T, \boldsymbol{I})$ and label this class RED
- step 3:
  For each class, generate 100 observations as follows:
  - For each observation, pick an m_k at random with probability $1/10$
  - Generate a point according to $N(m_k, \boldsymbol{I}/5)$



Result of 10 000 classifications

Variants of These Simple Methods

- Kernel methods: use weights that decrease smoothly to zero with distance from the target point, rather than the $0/1$ cutoff used in nearest-neighbor methods
- In high-dimensional spaces, some variables are emphasized more than others
- Local regression fits linear models (by least squares) locally rather than fitting constants locally
- Linear models fit to a basis expansion of the original inputs allow arbitrarily complex models
- Projection pursuit and neural network models are sums of nonlinearly transformed linear models

Statistical Decision Theory

- Random input vector: $X \in R^p$
- Random output variable: $Y \in R$
- Joint distribution: $Pr(X, Y)$
- We are looking for a function $f(x)$ for predicting Y given the values of the input X
- The loss function $L(Y, f(X))$ shall penalize errors
- Squared error loss: $L(Y, f(X)) = (Y - f(X))^2$

- Expected prediction error (EPE):

$$EPE(f) = E(Y - f(X))^2 = \int (y - f(x))^2 Pr(dx, dy)$$

- Since $Pr(X, Y) = Pr(Y|X)Pr(X)$
- EPE can also be written as $EPE(f) = E_X E_{Y|X}([Y - f(X)]^2|X)$
- Thus it suffices to minimize EPE pointwise $f(x) = argmin_c E_{Y|X}([Y - c]^2|X = x)$

Regression function: $f(x) = E(Y|X = x)$

Statistical Decision Theory

- Nearest neighbor methods try to directly implement this recipe

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

- Several approximations
  - Since no duplicate observations, expectation over a neighborhood
  - Expectation approximated by averaging over observations
- With increasing k and number of observations the average gets (provably) more stable
- But often we do not have large samples
- By making assumptions(linearity) we can reduce the number of required observations greatly
- With increasing dimension the neighborhood grows exponentially. Thus the rate of convergence to the true estimator (with increasing k) decreases

Regression function: $f(x) = E(Y|X = x)$

Statistical Decision Theory

- Linear regression
- Assumes that the regression function is approximately linear

$$f(x) \approx x^T \beta$$

- This is a model-based approach
- After plugging this expression into EPE and differentiating w.r.t. β, we can solve for β

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

$$\beta = [E(XX^T)]^{-1}E(XY)$$

- Again, linear regression replaces the theoretical expectation by averaging over the observed data

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

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

- Summary:
  - Least squares assumes that $f(x)$ is well approximated by a globally linear function
  - Nearest neighbors assumes that $f(x)$ is well approximated by a locally constant function

Regression function: $f(x) = E(Y|X = x)$

Statistical Decision Theory

- Additional methods in this book are often model-based but more flexible than the linear model
- Additive models

$$f(X) = \sum_{j=1}^{p} f_j(X_j)$$

- Each f_j is arbitrary

- What happens if we use another loss function?
  $L_1(Y, f(X)) = |Y - f(X)|$
- In this case

$$\hat{f}(x) = median(Y|X = x)$$

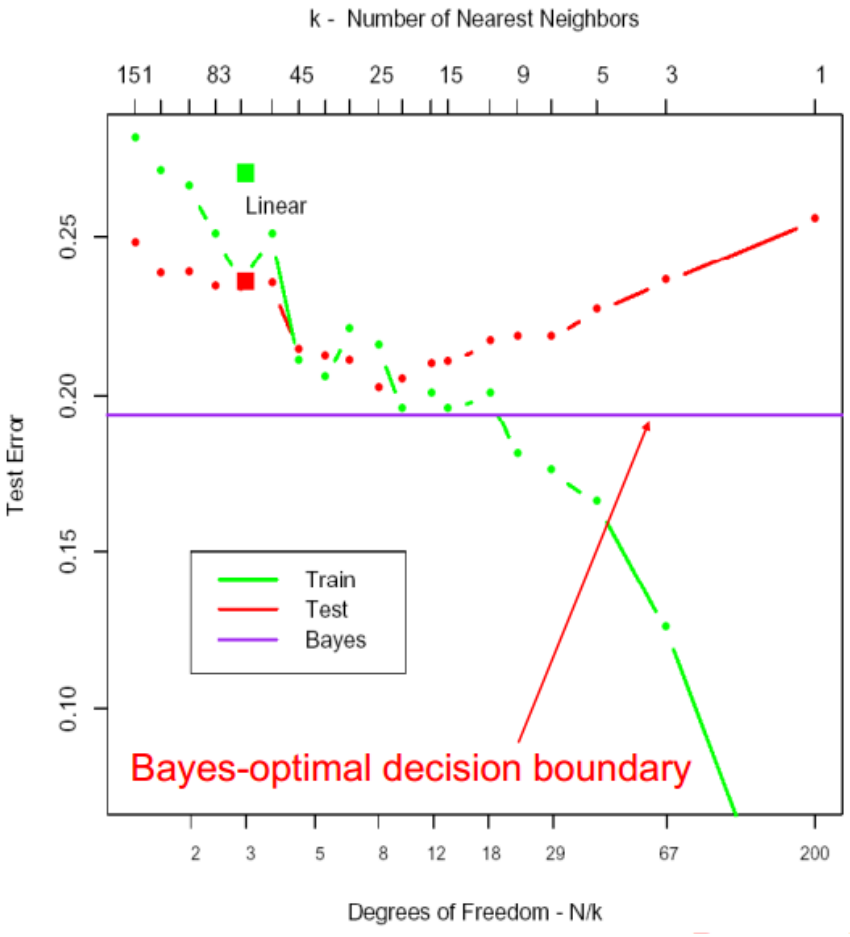
- More robust than the conditional mean
- L_1 criterion not differentiable
- Squared error most popular

Statistical Decision Theory

- Procedure for categorical output variable G with values from $\mathcal{G}$
- Loss function is a $K \times K$ matrix $\boldsymbol{L}$ where $K = card(\mathcal{G})$
- $\boldsymbol{L}$ is zero on the diagonal
- $\boldsymbol{L}(k,l)$ is the price paid for misclassifying a element from class $\mathcal{G}_k$ as belonging to class $\mathcal{G}_l$
- Frequently $0-1$ loss function used: $\boldsymbol{L}(k,l) = 1 - \delta_{kl}$

- Expected prediction error (EPE) $EPE = E[L(G, \hat{G}(X))]$
- Expectation taken w.r.t. the joint distribution $Pr(G,X)$
- Conditioning yields $$EPE = E_X \sum_{k=1}^{K} L[\mathcal{G}_k, \hat{G}(X)] Pr(\mathcal{G}_k|X)$$
- Again, pointwise minimization suffices $\hat{G}(X) =$ $$\arg\min_{g \in \mathcal{G}} \sum_{k=1}^{K} L(\mathcal{G}_k, g) Pr(\mathcal{G}_k|X = x)$$
- Or simply(Bayes Classifier) $\hat{G}(X) = \mathcal{G}_k$ if $Pr(\mathcal{G}_k|X = x) = \max_{g \in \mathcal{G}} Pr(g|X = x)$

Statistical Decision Theory



- Expected prediction error (EPE)
  $EPE = E[L(G, \hat{G}(X))]$
- Expectation taken w.r.t. the joint distribution $Pr(G, X)$
- Conditioning yields
  $$EPE = E_X \sum_{k=1}^{K} L[\mathcal{G}_k, \hat{G}(X)] Pr(\mathcal{G}_k|X)$$
- Again, pointwise minimization suffices
  $\hat{G}(X) =$
  $$\arg\min_{g \in \mathcal{G}} \sum_{k=1}^{K} L(\mathcal{G}_k, g) Pr(\mathcal{G}_k|X = x)$$
- Or simply(Bayes Classifier) $\hat{G}(X) = \mathcal{G}_k$ if $Pr(\mathcal{G}_k|X = x) = \max_{g \in \mathcal{G}} Pr(g|X = x)$

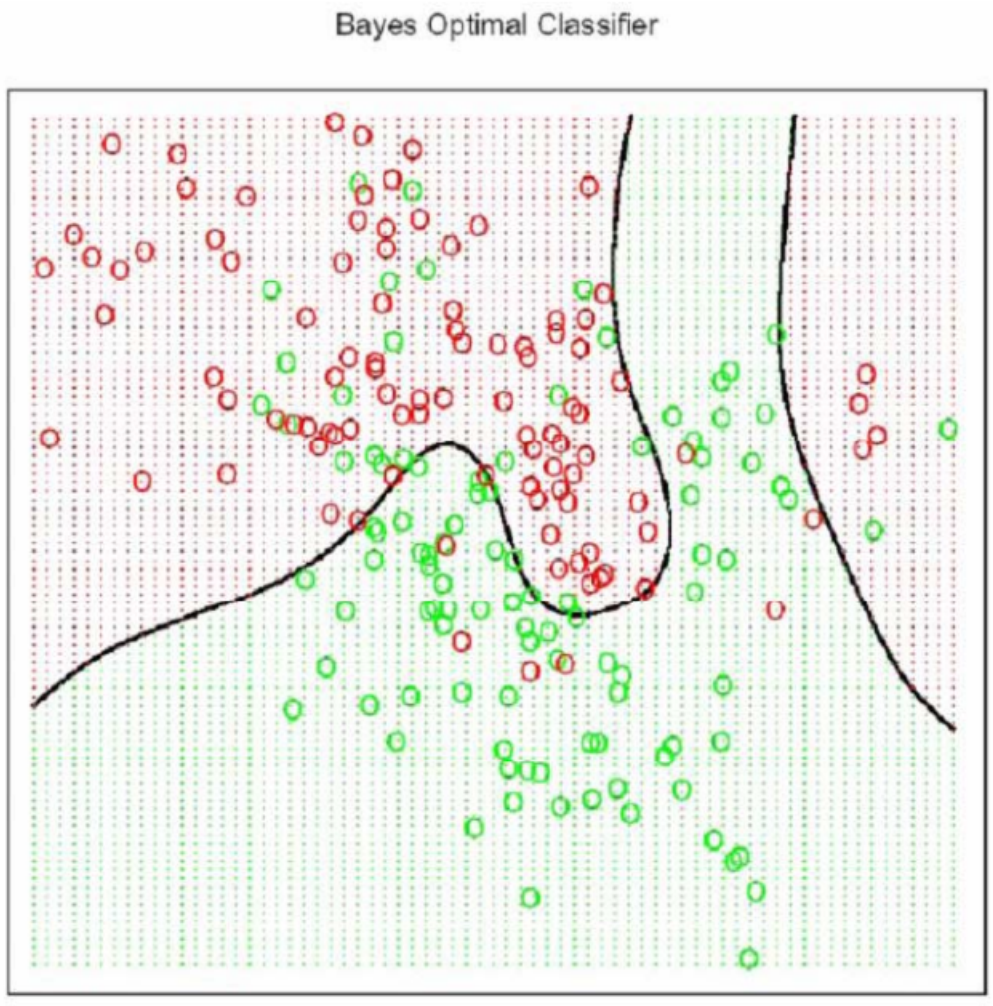
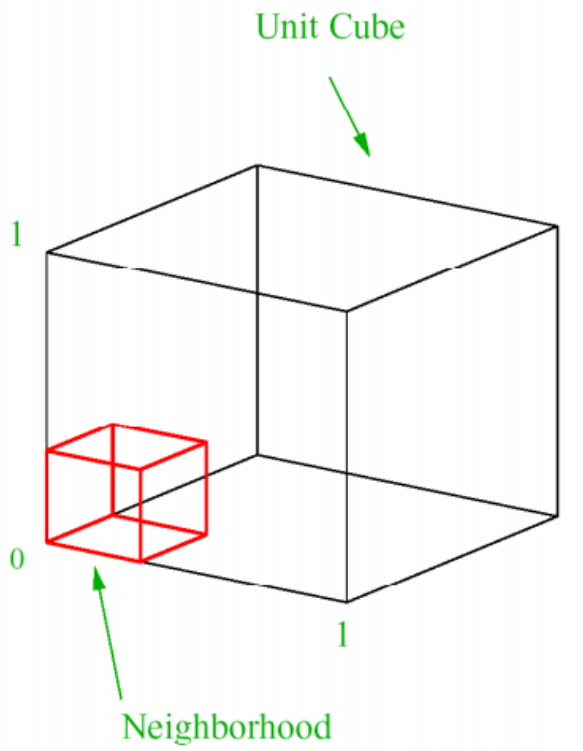


Figure: The optimal Bayes decision boundary for the simulation example

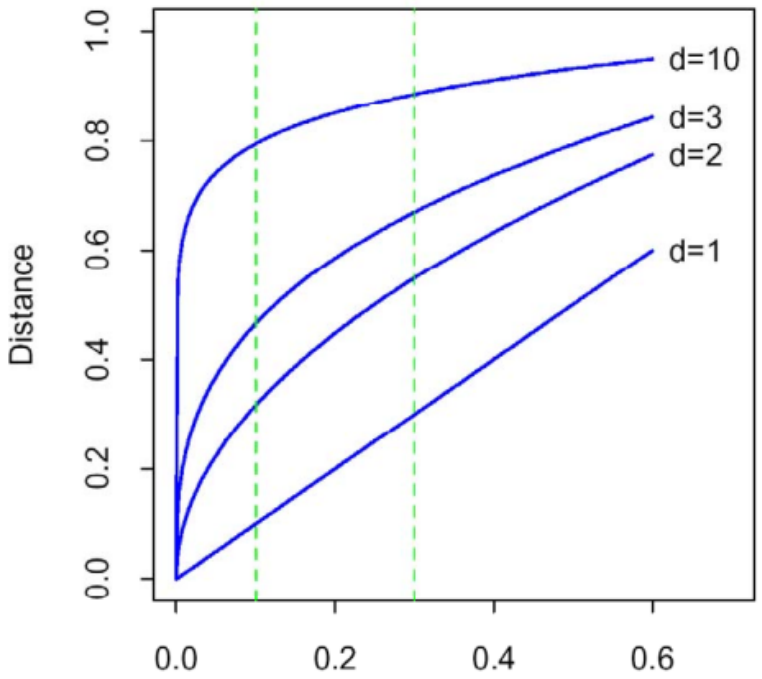
Local Methods in High Dimensions

- Curse of Dimensionality: Local neighborhoods become increasingly global, as the number of dimension increases
- Example: Points uniformly distributed in a p-dimensional unit hypercube.
- Hypercubical neighborhood in p dimensions that captures a fraction r of the data
  - Has edge length $e_p(r) = r^{1/p}$
  - $e_{10}(0.01) = 0.63$
  - $e_{10}(0.1) = 0.82$

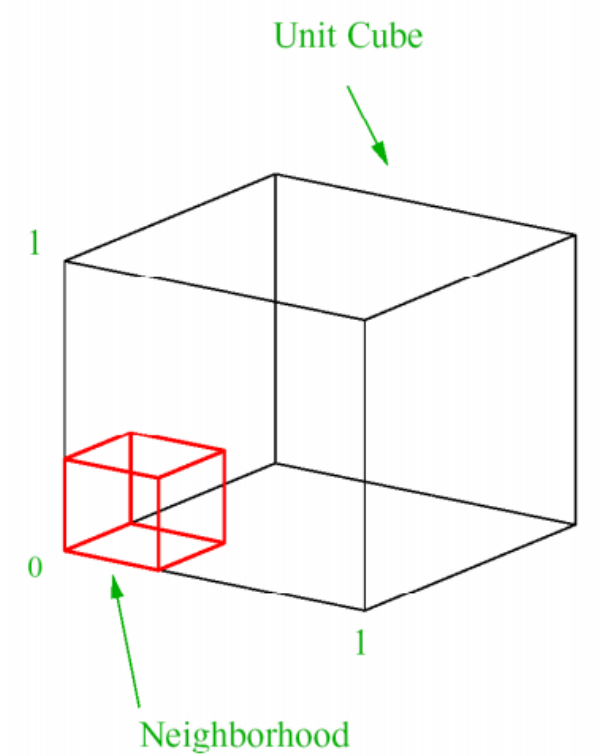


- To cover 1% of the data we must cover 63% of the range of an input variable

Local Methods in High Dimensions



The figure 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



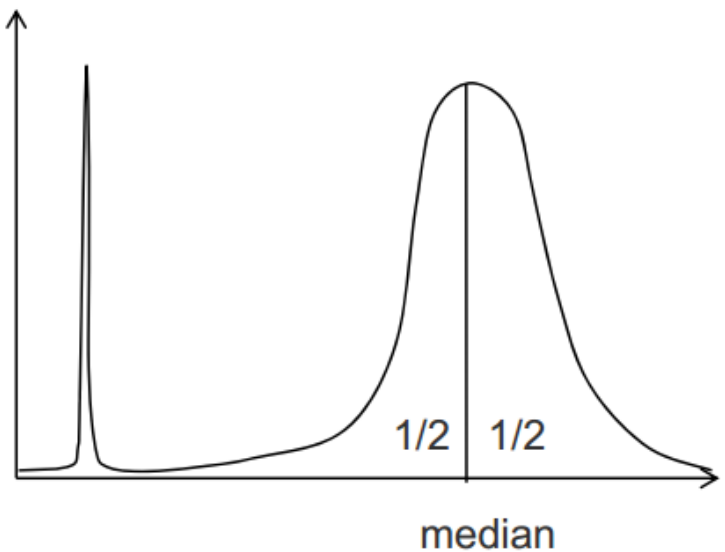
- To cover 1% of the data we must cover 63% of the range of

Local Methods in High Dimensions

- In high dimensions, all sample points are close to the edge of the sample
- N data points uniformly distributed in a p-dimensional unit ball centered at the origin
- Median distance from the closest point to the origin

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

- $d(10, 500) = 0.52$
  $d(20, 500) = 0.72$
- More than half the way to the boundary
- Sampling density is proportional to $N^{1/p}$
- If $N_1 = 100$ is a dense sample for one input then $N_{10} = 100^{10}$ is an equally dense sample for 10 inputs.

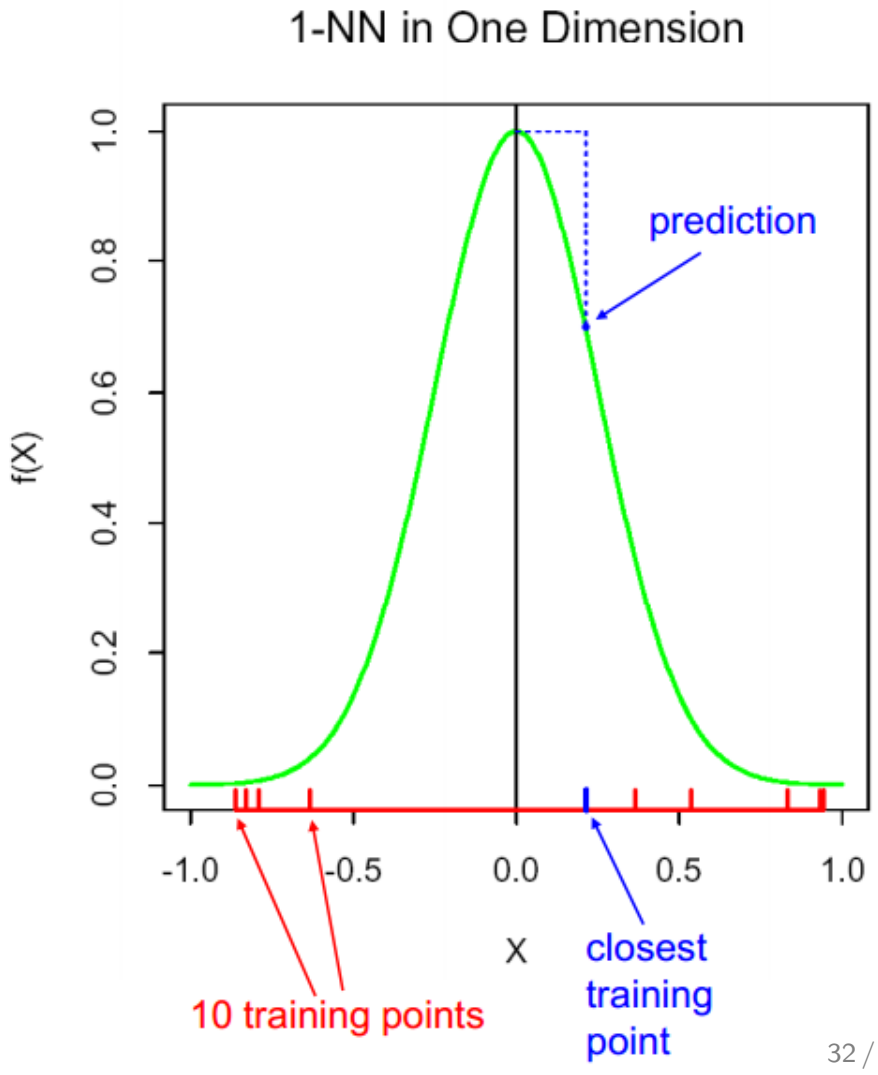


Local Methods in High Dimensions

- Another example
- $\mathcal{T}$ set of training points x_i generated uniformly in $[-1, 1]^p$(red)
- Functional relationship between X and Y(green)

$$Y = f(X) = e^{-8\|X\|^2}$$

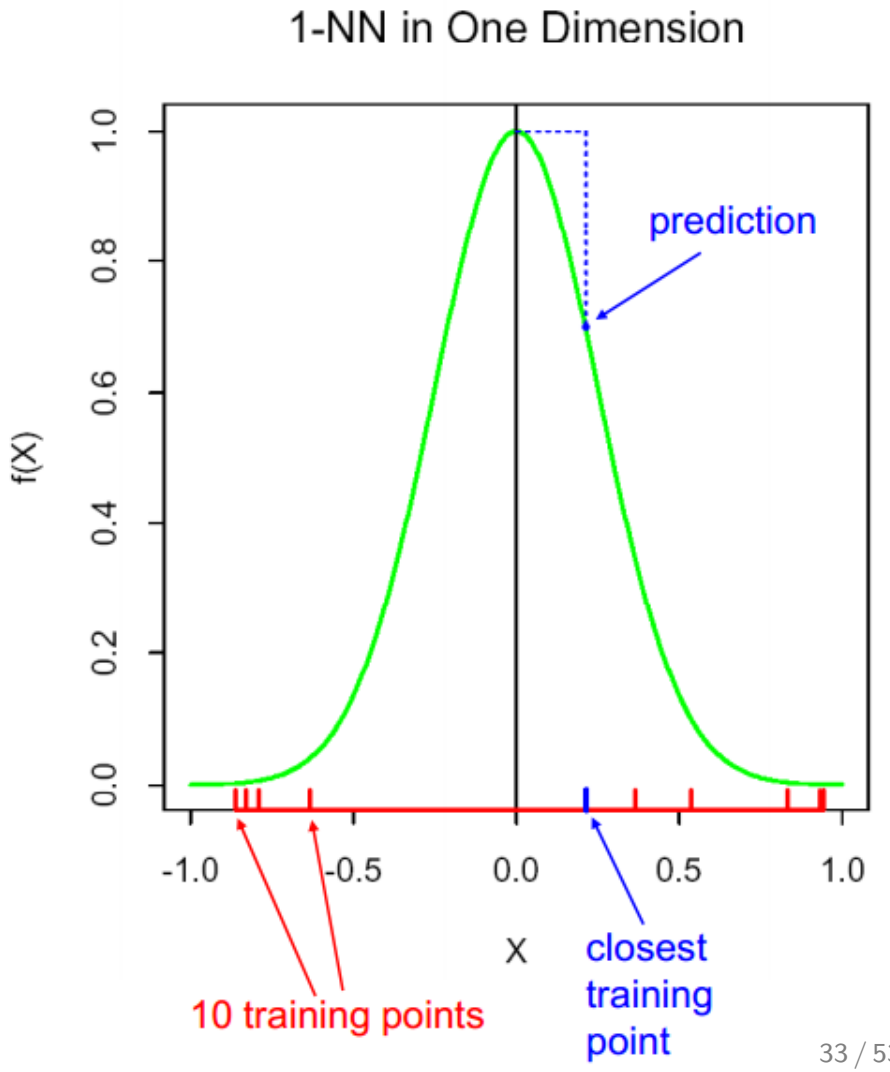
- No measurement error
- Error of a 1-nearest neighbor classifier in estimating $f(0)$(blue)



Local Methods in High Dimensions

- Another example
- Problem deterministic:
  Prediction error is the mean-squared error for estimating $f(0)$

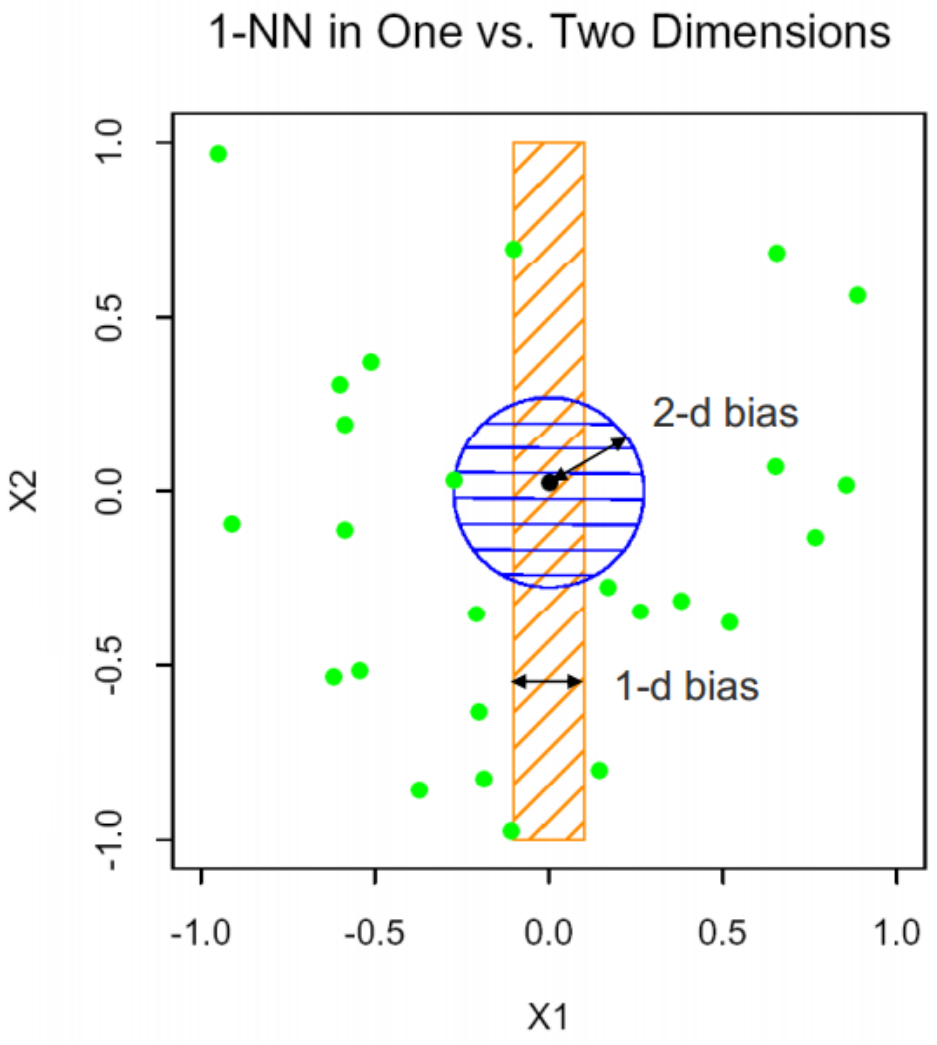
$$MSE(x_0) = E_T[f(x_0) - \hat{y}_0]^2$$
$$= Var_T(\hat{y}_0) + Bias_T^2(\hat{y}_0)$$



Local Methods in High Dimensions

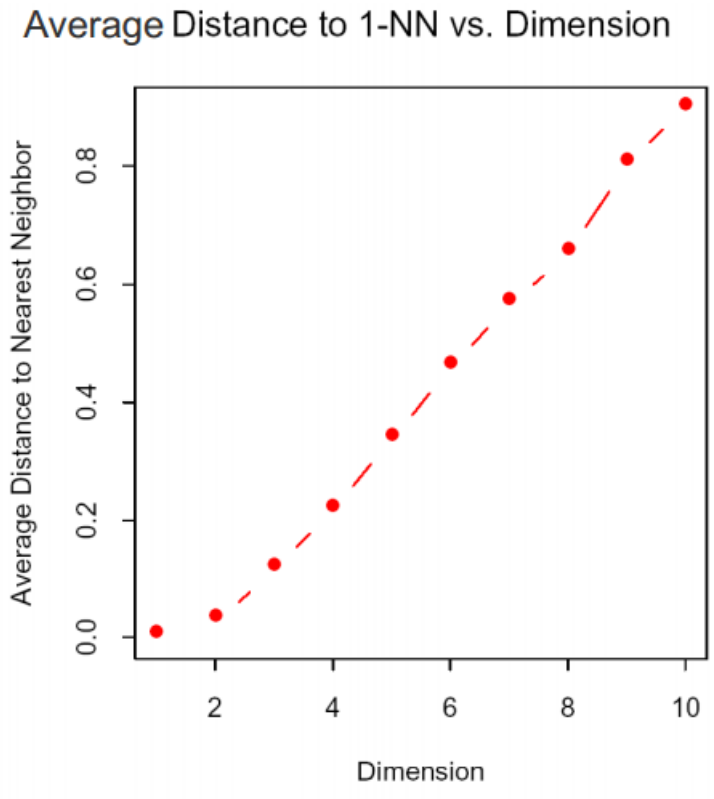
- Another example
- 1-d vs 2-d
- Bias increases

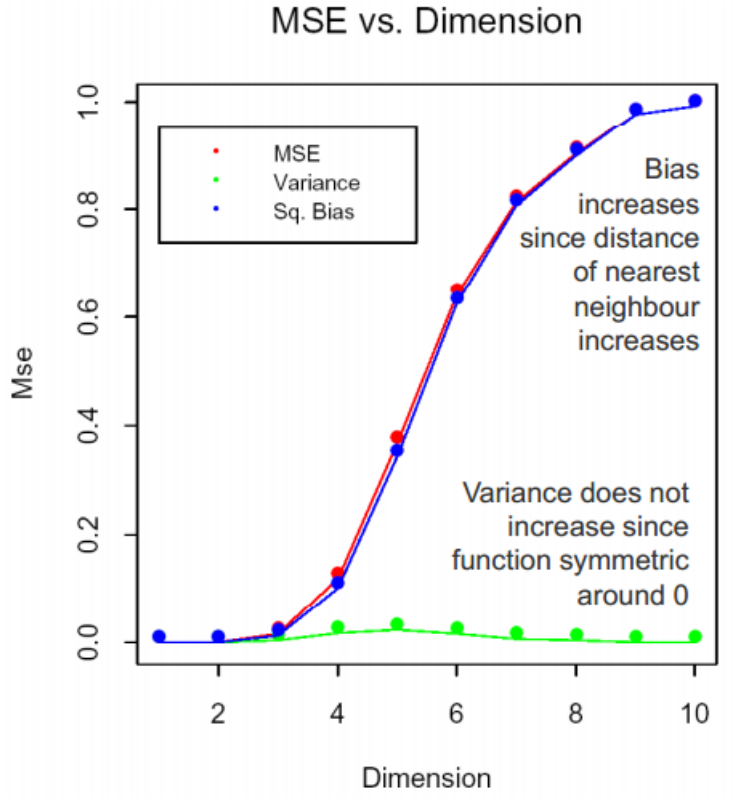
$$MSE(x_0) = E_T[f(x_0) - \hat{y}_0]^2$$
$$= Var_T(\hat{y}_0) + Bias_T^2(\hat{y}_0)$$



Local Methods in High Dimensions

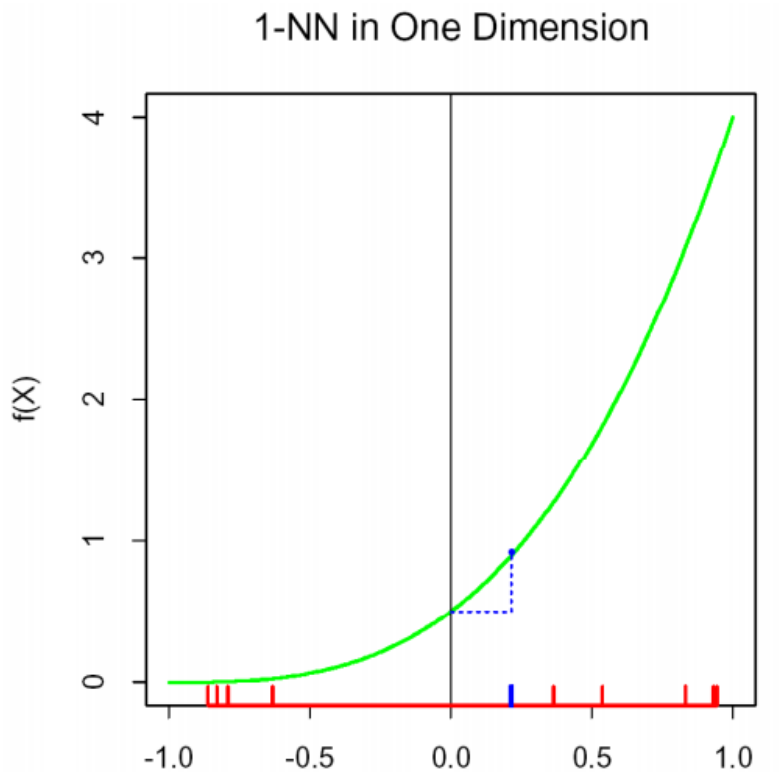
- The case on $N = 1000$ training points

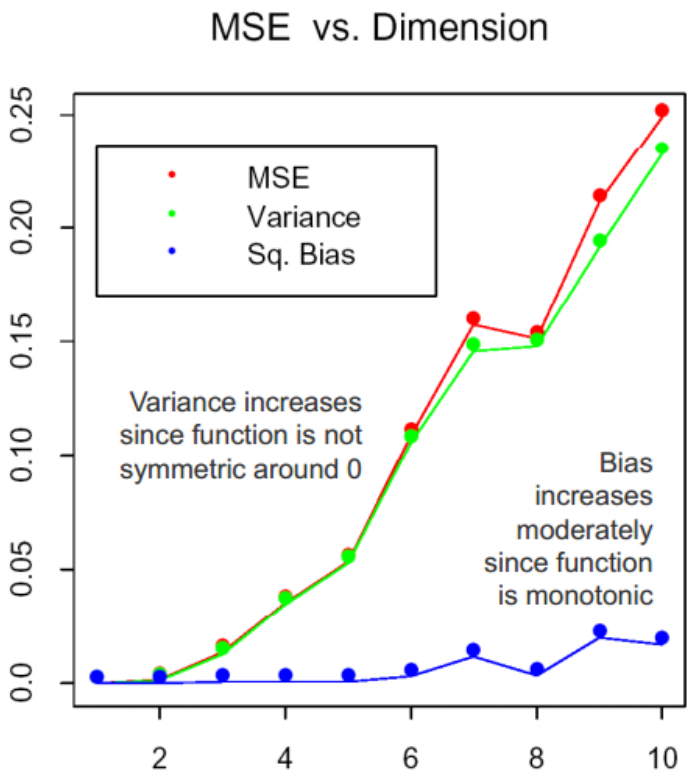




Local Methods in High Dimensions

- Yet another example $Y = f(X) = \frac{1}{2}(X_1 + 1)^3$





Local Methods in High Dimensions

- Assume now a linear relationship with measurement error
  $Y = X^T\beta + \varepsilon\ ,\ \varepsilon \sim N(0, \sigma^2)$
- We fit the model with least squares, for arbitrary test point x_0
  $\hat{y}_0 = x_0^T\hat{\beta} = x_0^T\beta + \sum_{i=1}^{N} l_i(x_0)\varepsilon_i$
  $l_i(x_0)$ is the i-th element of $X(X^TX)^{-1}x_0$

$$EPE(x_0) = E_{y_0|x_0}E_T[y_0 - \hat{y}_0]^2$$
$$= \sigma^2 E_T[x_0^T(X^TX)^{-1}x_0]\sigma^2$$

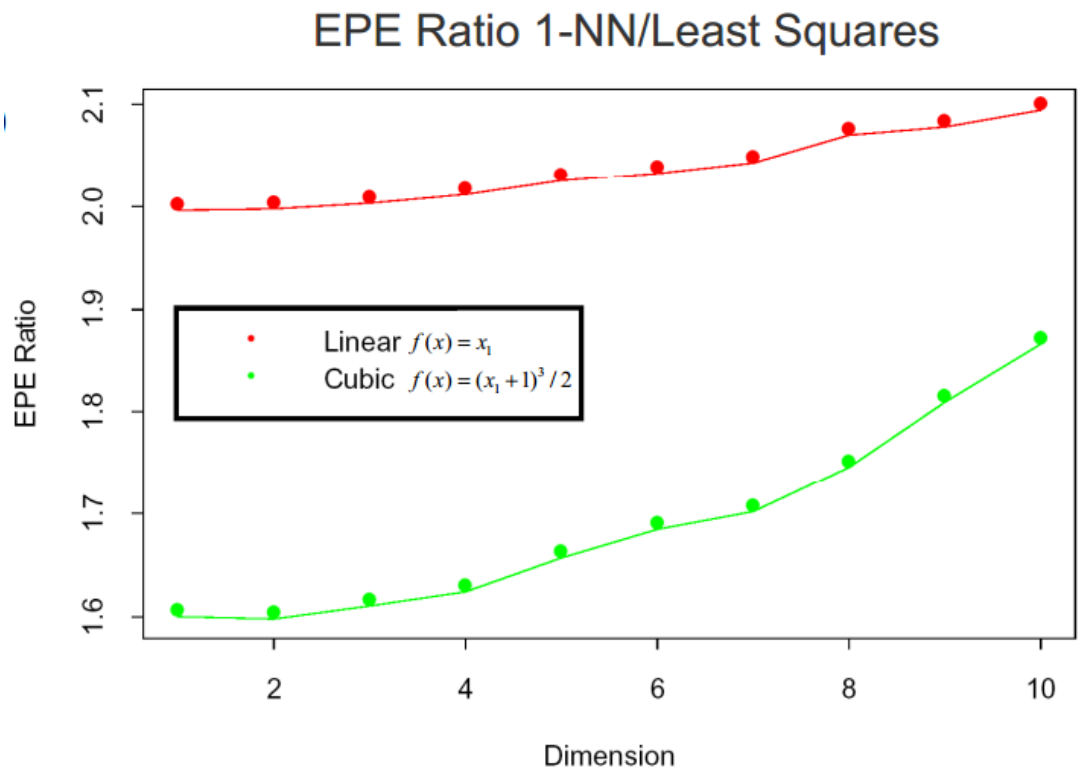
- Additional variance σ^2 since output nondeterministic
- Variance depends on x_0
- If N is large we get

$$E_{x_0}EPE(x_0) \to \sigma^2(p/N) + \sigma^2$$

- Variance negligible for large N or small σ
- No bias
- Curse of dimensionality controlled

Local Methods in High Dimensions

- More generally
  $Y = f(X) + \varepsilon\ ,\ \varepsilon \sim N(0,1)$
- Sample size: $N = 500$
- Linear case
  - EPE(Least Squares) is slightly above 1 no bias
  - EPE(1-NN) always above 2, grows slowly as nearest training point strays from target
- Cubic case
  - EPE(LeastSquares) is biased, thus ratio smaller



Statistical Models

- NN methods are the direct implementation of

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

- But can fail in two ways
  - With high dimensions NN need not be close to the target point
  - If special structure exists in the problem, this can be used to reduce variance and bias

Additive Error Model

- Assume additive error model

$$Y = f(X) + \varepsilon$$
$$E(\varepsilon) = 0$$
$$\varepsilon \text{ independent of } X$$

- Then $Pr(Y|X)$ depends only on the conditional mean of $f(x)$
- This model is a good approximation in many cases
- In many cases, $f(x)$ is deterministic and error enters through uncertainty in the input. This can often be mapped on uncertainty in the output with deterministic input.

Supervised Learning

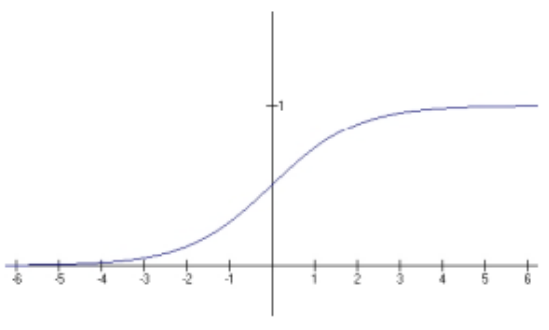
- Supervised learning
- The learning algorithm modifies its input/output relationship in dependence on the observed error

$$y_i - \hat{f}(x_i)$$

- This can be a continuous process

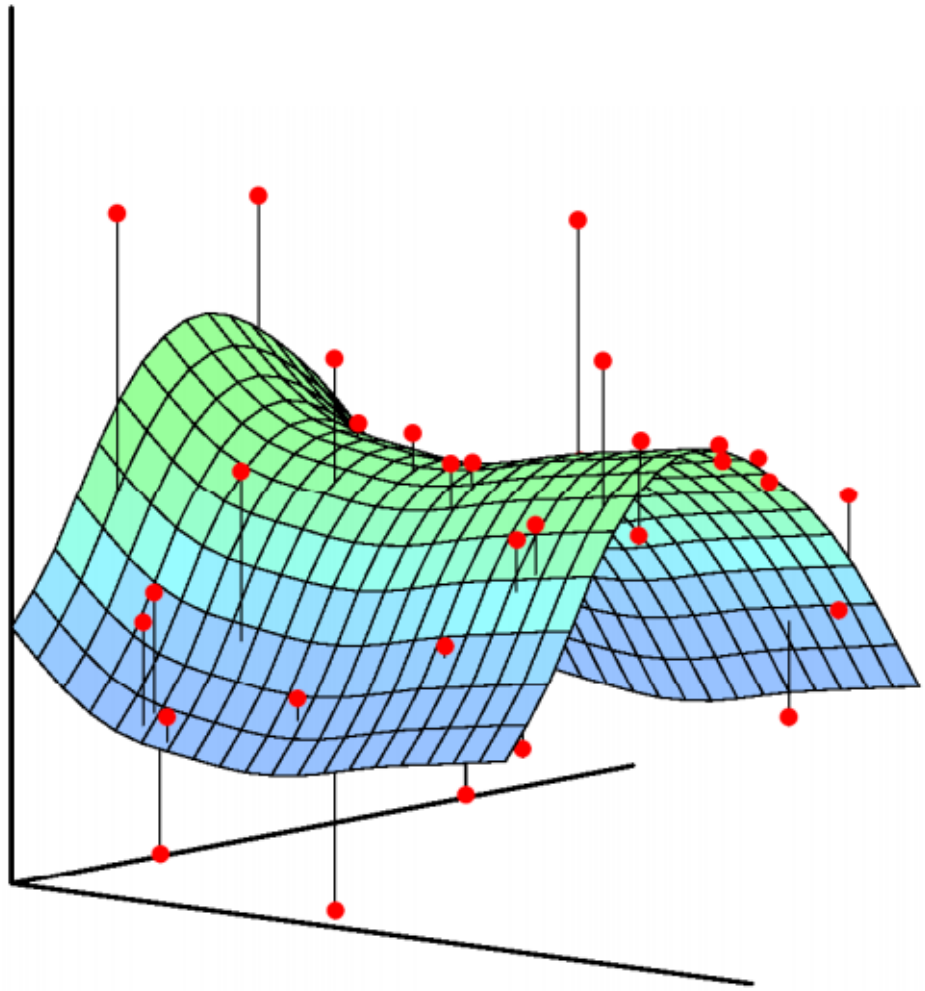
Function Approximation

- Data: pairs (x_i, y_i) that are points in $(p+1)$-space
  $F : R^p \to R,\ y_i = f(x_i) + \varepsilon_i$
- More general input spaces are possible
- Want a good approximation of $f(x)$ in some region of input space, given the training set $\mathcal{T}$
- Many models have certain parameters θ
- e.g. for the linear model
  $f(x) = x^T\beta$ and $\theta = \beta$
- Approximating f_θ by minimizing the residual sum of squares
  $RSS(\theta) = \sum_{i=1}^{N}(y_i - f_\theta(x_i))^2$
- Linear basis expansions have the more general form
  $f_\theta(x) = \sum_{k=1}^{K} h_k(x)\theta_k$
- Examples
  - Polynomial expansions:
    $h_k(x) = x_1x_2^2$
  - Trigonometric expansions:
    $h_k(x) = cos(x_1)$
  - Sigmoid expansion:
    $h_(x) = \dfrac{1}{1+\exp(-x^T\beta_k)}$



Function Approximation

- Approximating f_θ by minimizing the residual sum of squares
  $RSS(\theta) = \sum_{i=1}^{N}(y_i - f_\theta(x_i))^2$
- Intuition
  - f surface in $(p+1)$-space
  - Observe noisy realizations
  - Want fitted surface as close to the observed points as possible
  - Distance measured by RSS
- Methods
  - Closed form: if basis function have no hidden parameters
  - Iterative: otherwise

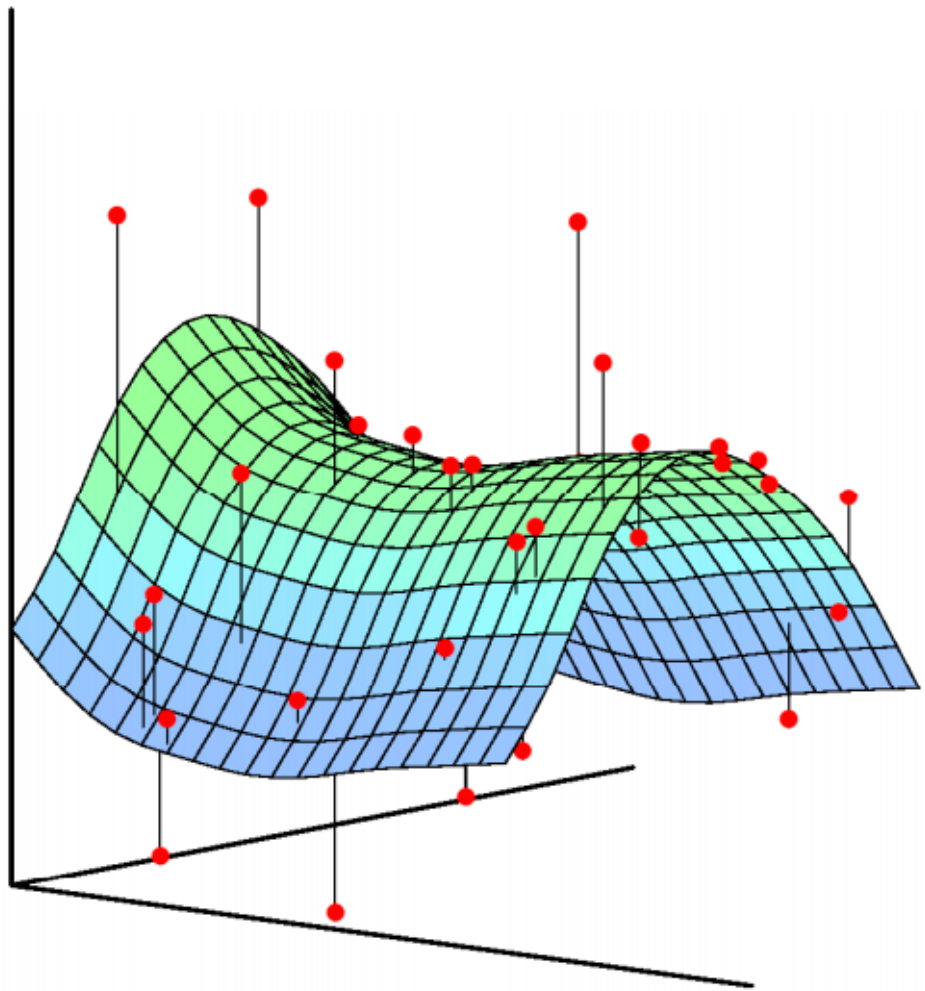


Function Approximation

- Approximating f_θ by maximizing the likelihood
- Assume an independently drawn random sample $y_i, i = 1, \cdots, N$ from a probability density $Pr_\theta(y)$. The log-probability of observing the sample is

$$L(\theta) = \sum_{i=1}^{N} \log Pr_\theta(y_i)$$

- Set θ to maximize $L(\theta)$



Function Approximation

- Approximating f_θ by maximizing the likelihood
- Assume an independently drawn random sample $y_i, i = 1, \cdots, N$ from a probability density $Pr_\theta(y)$. The log-probability of observing the sample is

$$L(\theta) = \sum_{i=1}^{N} \log Pr_\theta(y_i)$$

- Set θ to maximize $L(\theta)$

- Least squares with the additive error model $Y = f_\theta(X) + \varepsilon, \varepsilon \sim N(0, \sigma^2)$
- is equivalent to maximum likelihood with the likelihood function $Pr(Y|X, \theta) \sim N(f_\theta(X), \sigma^2)$
- This is, because in this case the log-likelihood function is

$$L(\theta) = -\frac{N}{2}\log(2\pi) - N\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f_\theta(x_i))^2$$

proportional to RSS → $-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - f_\theta(x_i))^2$

Function Approximation

- Approximating the regression function $Pr(G|X)$ by maximizing the likelihood for a qualitative output G
- Conditional probability of each class given X

$$Pr(G = \mathcal{G}_k|X = x) = p_{k,\theta}(x) \ , \ k = 1, \cdots, K$$

- Then the log-likelihood, also called the cross-entropy, is

$$L(\theta) = \sum_{i=1}^{N} \log p_{g_i,\theta}(x_i)$$

Structured Regression Models

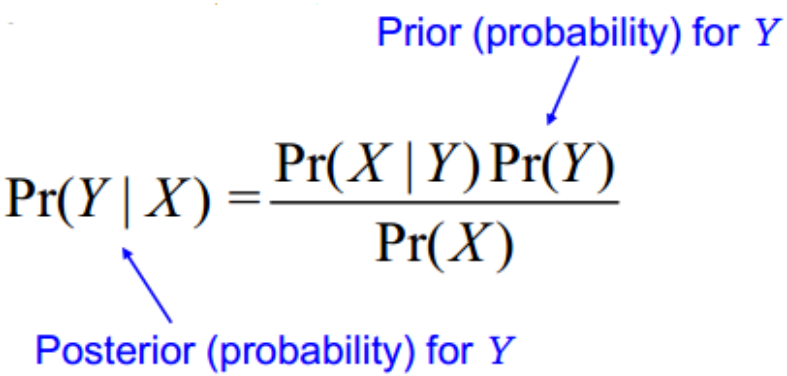
- Problem with regression:
- Minimizing $RSS(\theta) = \sum_{i=1}^{N}(y_i - f_\theta(x_i))^2$ has infinitely many (interpolating) solutions
- If we have repeated outcomes at each point, we can use them to decrease the variance by better estimating the average
- Otherwise restrict the set of functions to "smooth" functions
  - Choice of set is model choice
  - Major topic of this course
- Restricting function spaces:
- Choose function space of low complexity
  * Close to constant, linear or low-order polynomial in small neighborhoods
  * VC dimension is a relevant complexity measure in this context
  * Estimator does averaging or local polynomial fitting
  * The larger the neighborhood, the stronger the constraint
  * Metric used is important, either explicitly or implicitly defined
  * All such methods run into problems with high dimensions, therefore need metrics that allow neighborhoods to be large in at least some dimensions

Structured Regression Models

- Bayesian Methods
- Formula for joint probabilities

$$Pr(X,Y) = Pr(Y|X)Pr(X) = Pr(X|Y)Pr(Y)$$

- Bayes Formula



- RSS is penalized with a roughness penalty

$$PRSS(f;\lambda) = RSS(f) + \lambda J(f)$$

- $J(f)$ is large for ragged functions
  - E.g. cubic smoothing spline is the solution for the least squares problem

$$PRSS(f;\lambda) = \sum_{i=1}^{N}(y_i - f(x_i))^2 + \lambda \int [f''(x)]^2 dx$$

  - Large second derivative is penalized

Structured Regression Models

- Introducing penalty functions is a type of regularization
  - It works against overfitting
  - It implements beliefs about unseen parts of the problem
- In Bayesian terms
  - Penalty J is the log-prior (probability distribution)
  - PRSS is the log-posterior (probability distribution)

- RSS is penalized with a roughness penalty

$$PRSS(f;\lambda) = RSS(f) + \lambda J(f)$$

- $J(f)$ is large for ragged functions
  * E.g. cubic smoothing spline is the solution for the least squares problem

$$PRSS(f;\lambda) = \sum_{i=1}^{N}(y_i - f(x_i))^2 + \lambda \int [f''(x)]^2 dx$$

  * Large second derivative is penalized

Kernel Methods and Local Regression

- Kernel functions
  - model the local neighborhoods in NN methods
  - define the function space used for approximation
- Gaussian kernel

$$K_\lambda(x_0, x) = \frac{1}{\lambda} \exp\left[-\frac{\|x - x_0\|^2}{2\lambda}\right]$$

  * assigns weights to points that die exponentially with the square of the distance from the point x_0
  * λ controls the variance

- Simplest Kernel estimate: Nadaraya-Watson weighted average

$$\hat{f}(x_0) = \frac{\sum_{i=1}^N K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^N K_\lambda(x_0, x_i)}$$

- General local regression estimate of $f(x_0)$ is $f_{\hat{\theta}}(x_0)$ where $\hat{\theta}$ minimizes

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

Kernel Methods and Local Regression

- f_θ is a simple function such as a low-order polynomial
  - $f_\theta = \theta_0$ Nadaraya-Watson estimate
  - $f_\theta(x) = \theta_0 + \theta_1 x$ local linear regression model
- NN methods can be regarded as kernel methods with a special metric

$K_k(x, x_0) =$
$I\left(\|x - x_0\| \leq \|x_{(k)} - x_0\|\right)$

$x_{(k)}$: training sample ranked k in distance from x_0

I: indicator function $I(b) = \delta_{b,true}$

- Simplest Kernel estimate:Nadaraya-Watson weighted average

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)}$$

- General local regression estimate of $f(x_0)$ is $f_{\hat{\theta}}(x_0)$ where $\hat{\theta}$ minimizes

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

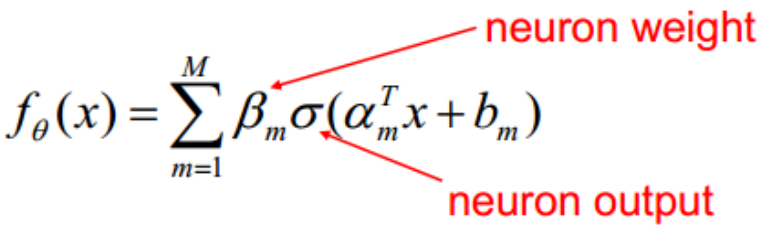
Basis Functions and Dictionary Methods

- Include linear and polynomial expansions and more
- General form

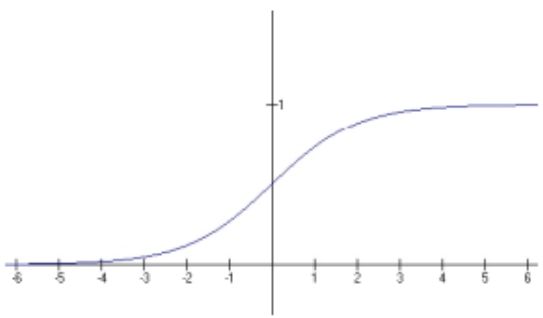
$$f_\theta(x) = \sum_{m=1}^{M} \theta_m h_m(x)$$

- Linear in θ
- Examples
- Splines
  - Parameters are the points of attachment of the polynomials(knots)

- Radial Basis Functions
  $f_\theta(x) = \sum_{m=1}^{M} K_{\lambda_m}(\mu_m, x)\theta_m$
  - Parameters are
    - Centroids μ_m
    - Scales λ_m
- Neural Networks



$$\sigma(x) = 1/(1+e^{-x})$$



Model Selection

- Smoothing and complexity parameters
  - Coefficient of the penalty term
  - Width of the kernel
  - Number of basis functions
- The setting of the parameters implements a tradeoff between bias and variance
- Example: k-NN methods

$$Y = f(X) + \varepsilon$$
$$E(\varepsilon) = 0$$
$$Var(\varepsilon) = \sigma^2$$

- Assume that the values of the x_i are fixed in advance

- Generalization error

$$EPE_k(x_0) = E\left[Y - \hat{f}_k(x_0)|X = x_0\right]$$
$$= \sigma^2 + \left[Bias^2(f_k(x_0)) + Var_{\mathcal{T}}(f_k(x_0))\right]$$
$$= \sigma^2 + \left[f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})\right]^2 + \frac{\sigma^2}{k}$$

irreducible error — mean-square error