Overview of Supervised Learning I

Statistical Data Mining I

Fall 2017

Rachael Hageman Blair

Modeling Assumptions of this class

1. Many statistical learning methods are relevant and useful in a wide-range of academic and non-academic disciplines beyond biostatistics.
2. Statistical learning should not be viewed as a series of black boxes.
3. While it is important to know what job is performed by the black box, it is not necessary to create the black box.
4. It is presumed that the student is interested in applying data-mining methods to real-world problems.

Our Goal: Practical yet rigorous.

Two Simple Approaches to Prediction

	Linear Model	K-nearest neighbors
Structural Assumptions	High	Low
Stability	Stable	Can be Unstable
Accuracy	Can be inaccurate	Accurate

Q: Why are we looking at these two simple methods?

Two Simple Approaches to Prediction

	Linear Model	K-nearest neighbors
Structural Assumptions	High	Low
Stability	Stable	Can be Unstable
Accuracy	Can be inaccurate	Accurate

Q: Why are we looking at these two simple methods?

A: Other more sophisticated methods are extensions of these!

Linear Model and Least Squares

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

Matrix-vector form: $\hat{Y} = X^T \hat{\beta}$

How do we fit a Linear Model to a set of training data?
Least Squares – find the $\hat{\beta}$ (parameters) that minimize the RSS.

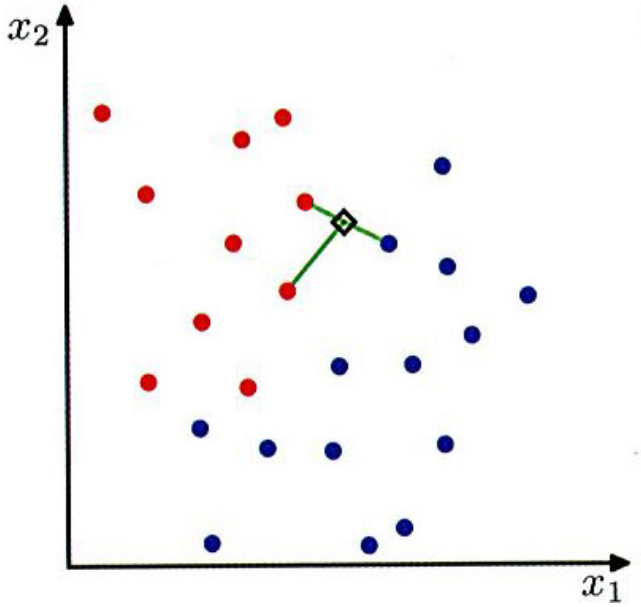
$$RSS(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$

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

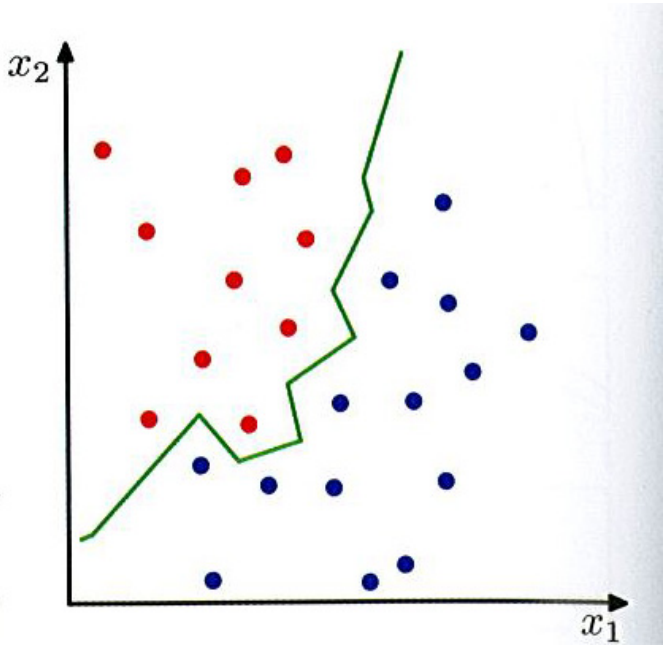
Solution: The normal equations: $\hat{\beta} = (X^T X)^{-1} X^T y$

When X is full rank, and well conditioned.

Nearest-neighbor methods

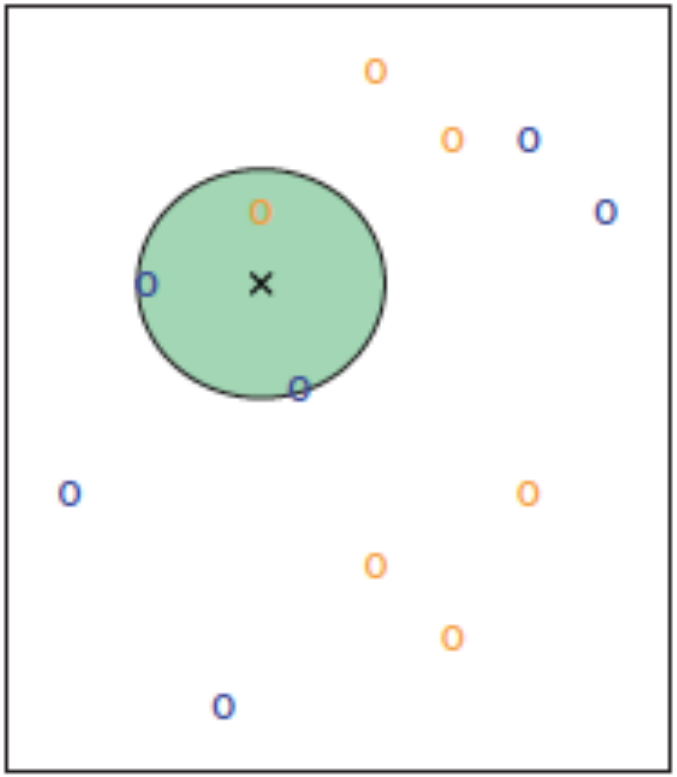


A new point arrives, it is classified according to the majority class membership of its K closest neighbors.

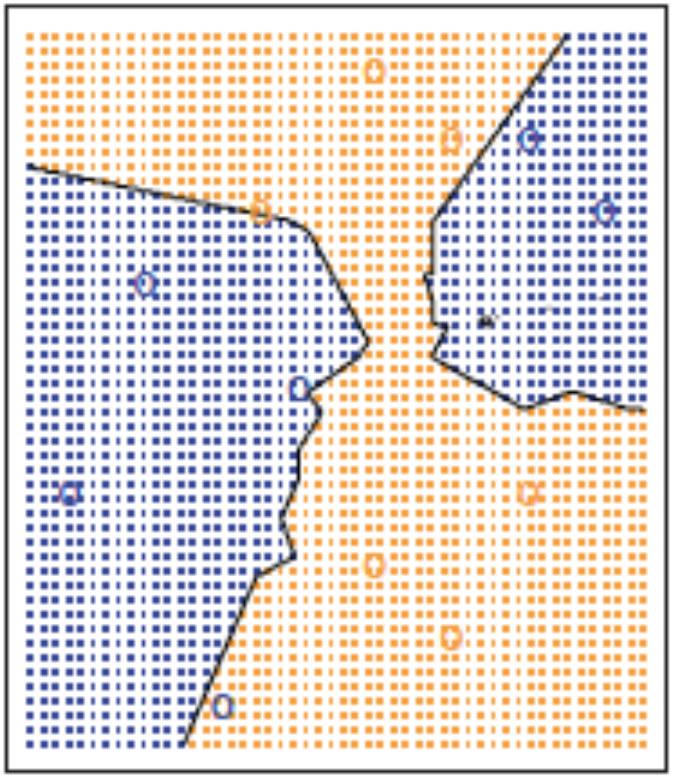


When K=1, the decision boundary is a hyper-plane that form perpendicular bisectors for pairs of points from different classes.

Nearest-neighbor methods

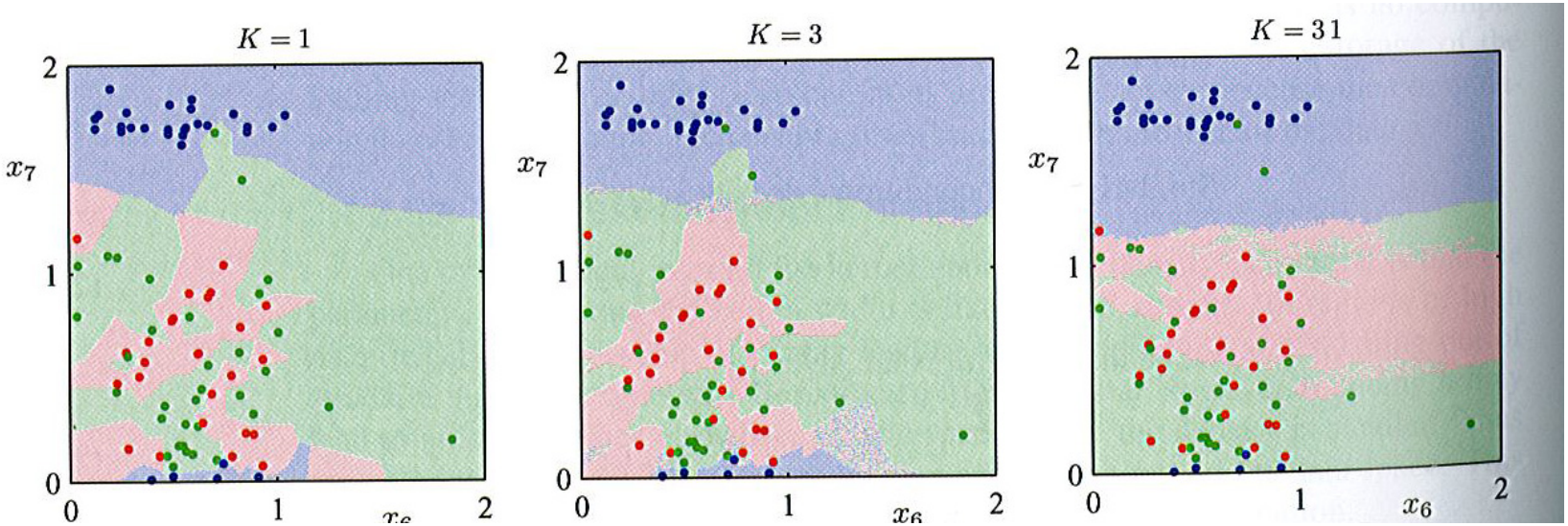


A new point arrives, it is classified according to the majority class membership of its K closest neighbors.



Simulation of a “grid” of test points.

Nearest-neighbor methods



K - pertains to the fit. The k- nearest neighbor fit for $\hat{Y}(x)$ is defined as follows:

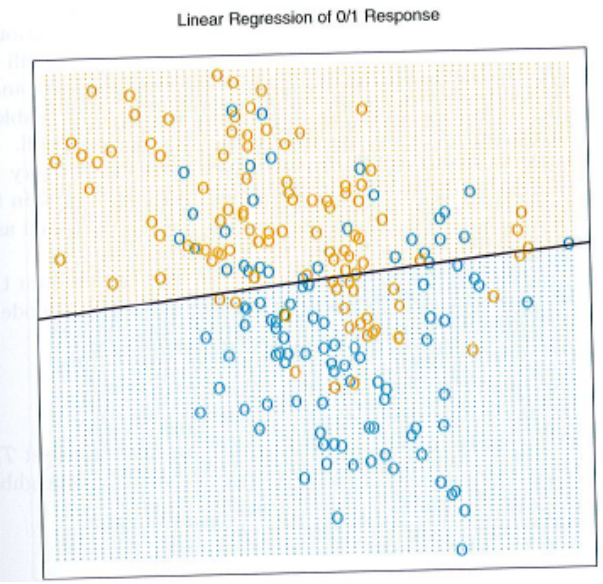
$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

Neighborhood of x defined by the k closes points x_i.

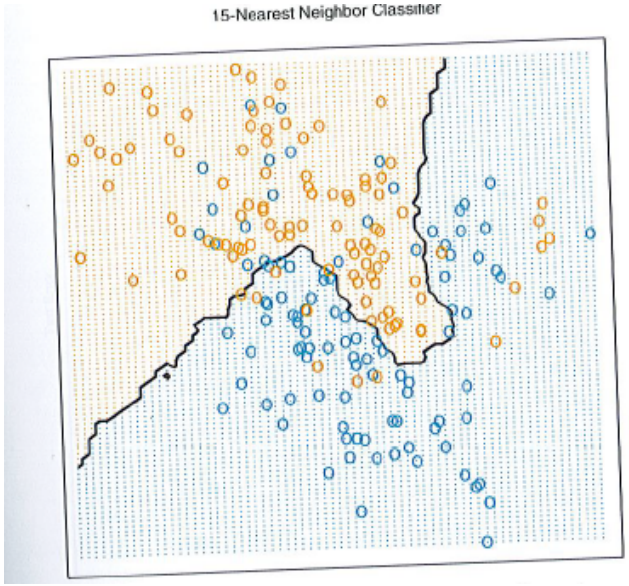
Small K - many small regions.
Larger K - fewer large regions.

Which is best…. ?

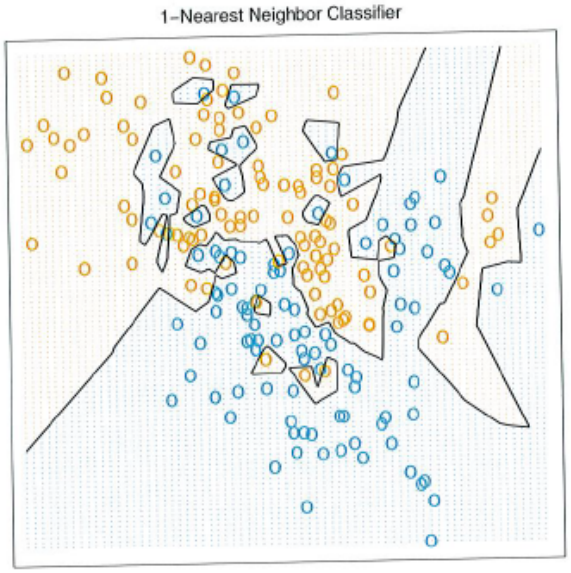
Linear Regression



Nearest Neighbor (k=15)

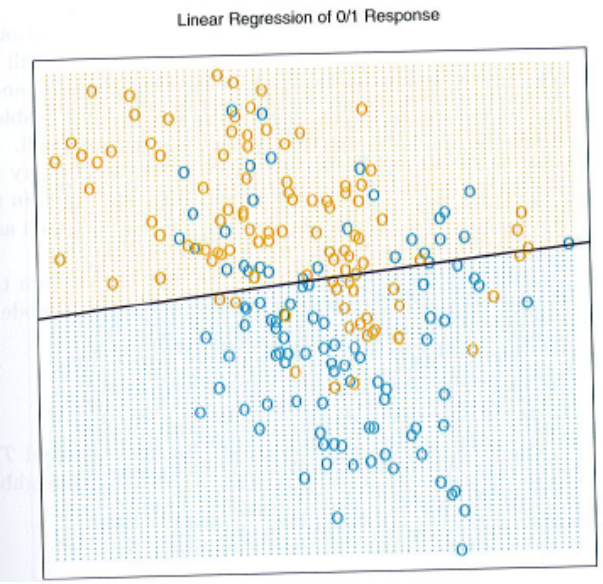


Nearest Neighbor (k=1)

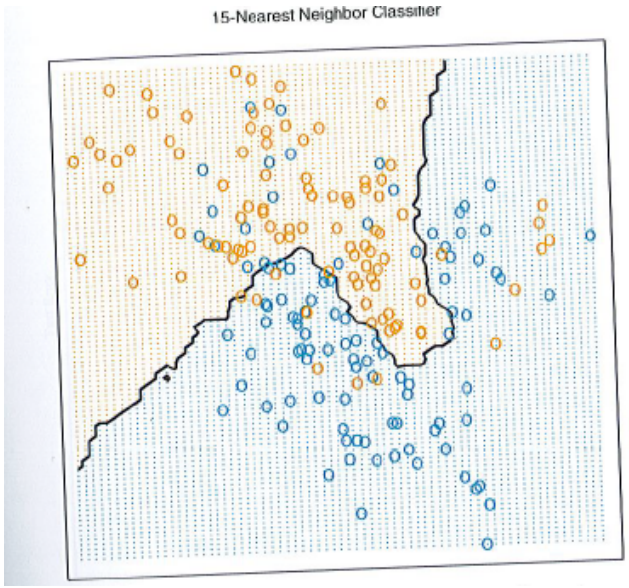


Which is best…. ?

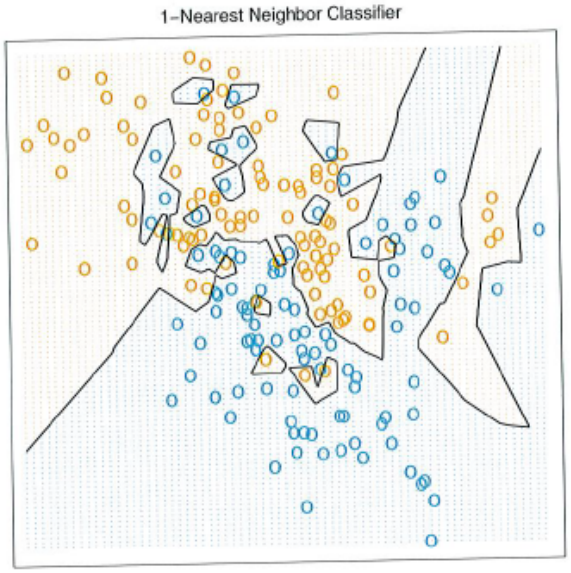
Linear Regression



Nearest Neighbor (k=15)



Nearest Neighbor (k=1)

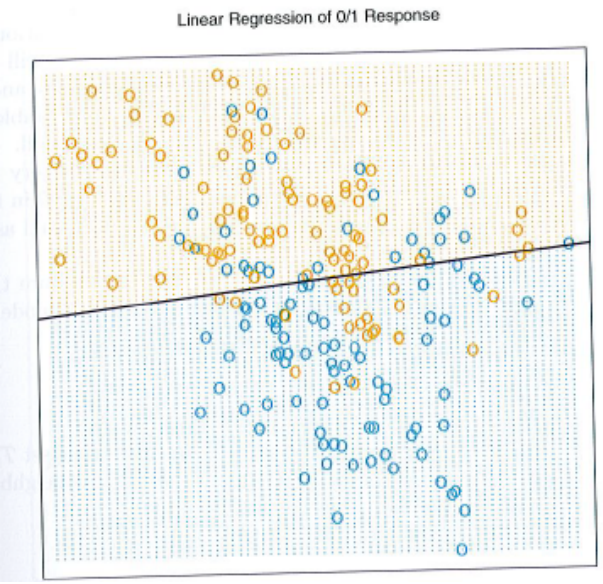


Scenario 1: The training data in each class were generated from bivariate Gaussian distributions with uncorrelated components and different means.

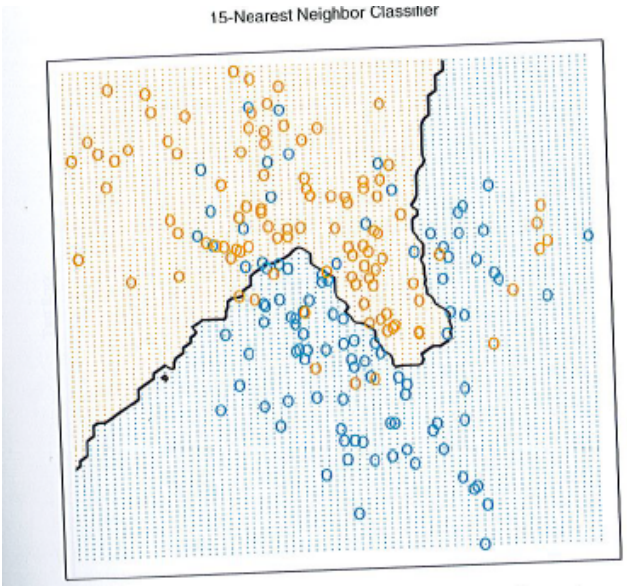
Scenario 2: The training data in each class came from a mixture of 10 low variance Gaussian distributions with individual means themselves distributed as Gaussian.

Which is best…. ?

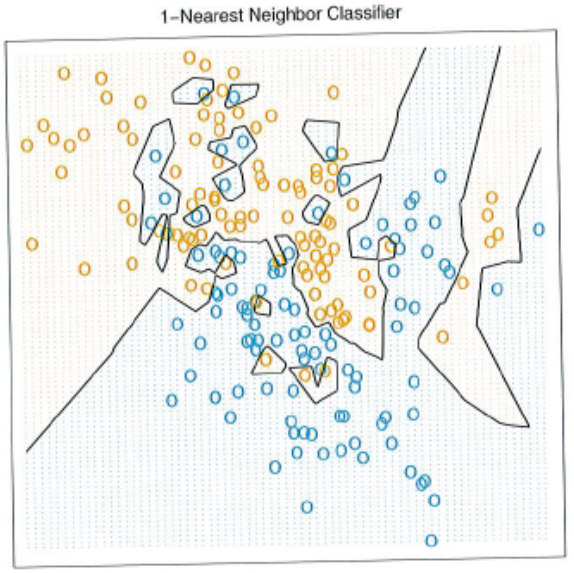
Linear Regression



Nearest Neighbor (k=15)



Nearest Neighbor (k=1)



LINEAR REGRESSION

Scenario 1: The training data in each class were generated from bivariate Gaussian distributions with uncorrelated components and different means.

Scenario 2: The training data in each class came from a mixture of 10 low variance Gaussian distributions with individual means themselves distributed as Gaussian.

K-Nearest Neighbors

Statistical Decision Theory

The Setup: Let $X \in \mathbf{R}^p$ denote a real valued random input vector and $Y \in \mathbf{R}$ be a real valued random output variable, with a joint distribution $P(X,Y)$.

The Objective: We want a function, for predicting the output for given values of the input. We can use **squared error loss** to penalize errors in prediction: $L(Y, f(x)) = (Y - f(x))^2$.

Statistical Decision Theory

Let $X_1, X_2, \ldots, X_p$ denote a set of predictors that contain relevant Information for the risk of disease Y.

Natural to use $X_1, X_2, \ldots, X_p$ to predict Y.

Rephrase as functional approximation:

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

"black box"

Statistical Decision Theory

How do we choose f(x)?

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

Best predictor when dealing with squared error loss.

Nearest neighbors tries to do this using training data:

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

Neighborhood containing the k points In Training Data that are closest to x.

Least squares also averages over the training data:

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

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

Local Methods in High Dimensions

Curse of Dimensionality… where the wheels come off!

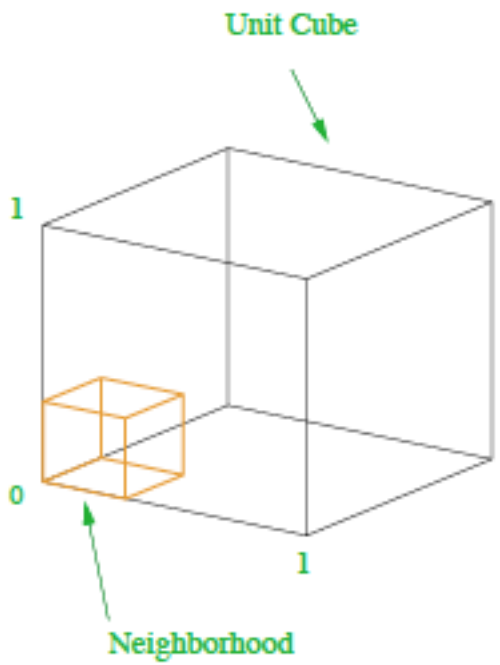
A contrived example:
Consider a p-dimensional hypercube on the range [0,1].

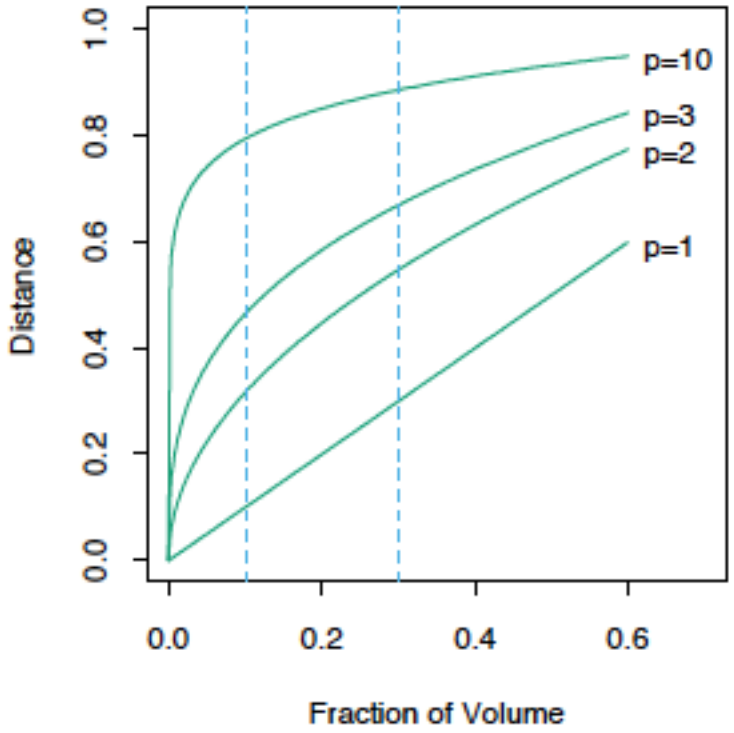
The expected edge length for an "r" fraction of observations (unit volume): $e_p(r) = r^{1/p}$.

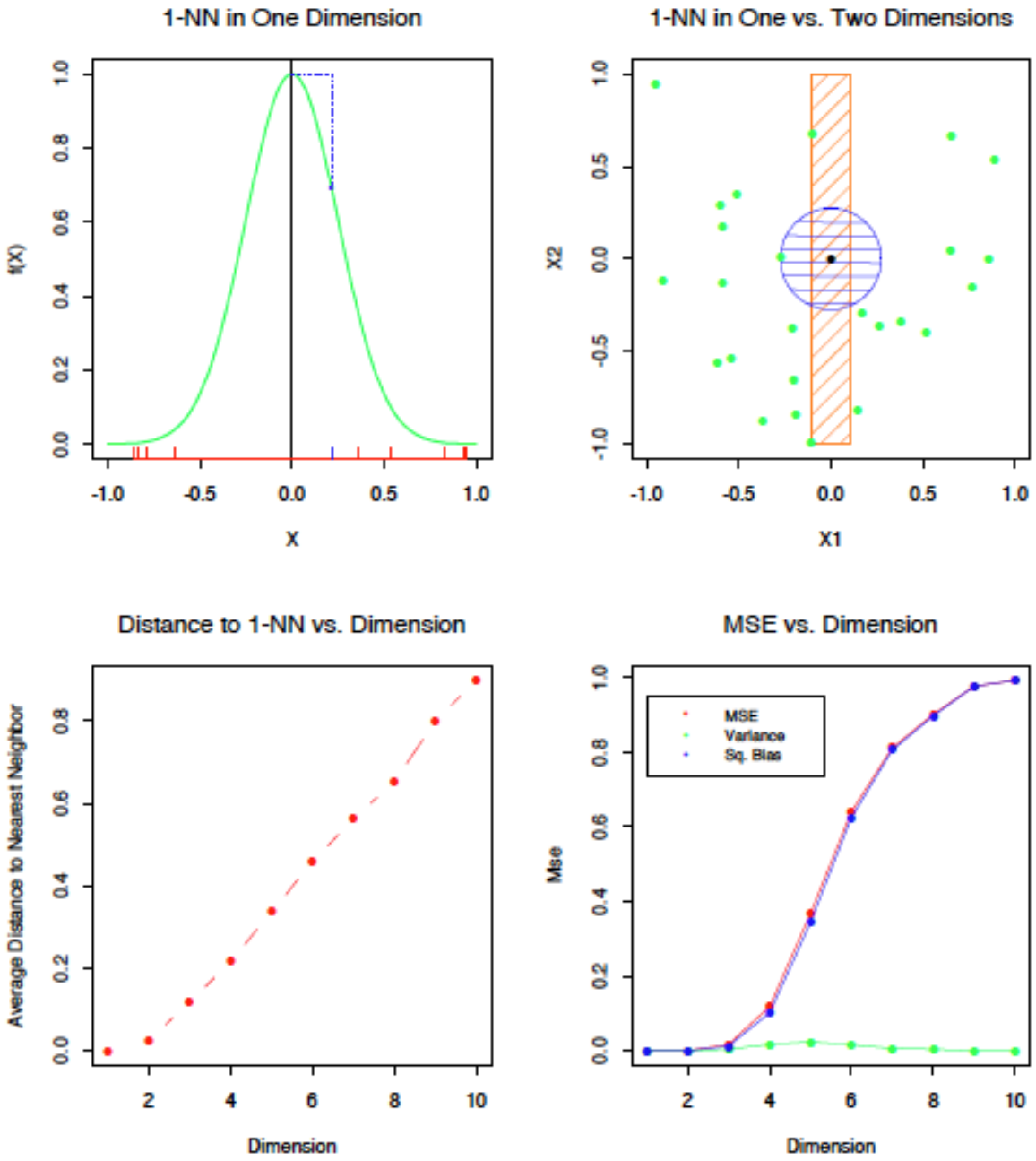
In 10 dimensions ….
If you want to capture 1% of the data, $e_p(.01) = .01^{1/10} = .63$.
If you want to capture 10% of the data, $e_p(.1) = .1^{1/10} = .80$.

Suppose 1000 data points generated in a p-dimensional hypercube on the range [0,1].

Local Methods in High Dimensions






1-NN in One Dimension
f(X)
X
1-NN in One vs. Two Dimensions
X2
X1
Distance to 1-NN vs. Dimension
Average Distance to Nearest Neighbor
Dimension
MSE vs. Dimension
MSE
Variance
Sq. Bias
Mse
Dimension

Local Methods in High Dimensions

- The complexity of functions grows of many variables grows exponentially with the number of dimensions.

- In order to estimate with accuracy with local models, we need massive coverage (lots of training samples).

The relationship between X and Y

Back to our goal ….
We want to estimate $\hat{f}(x)$

Our example was contrived to illustrated a point.

The reality: We don't know $f(x)$.

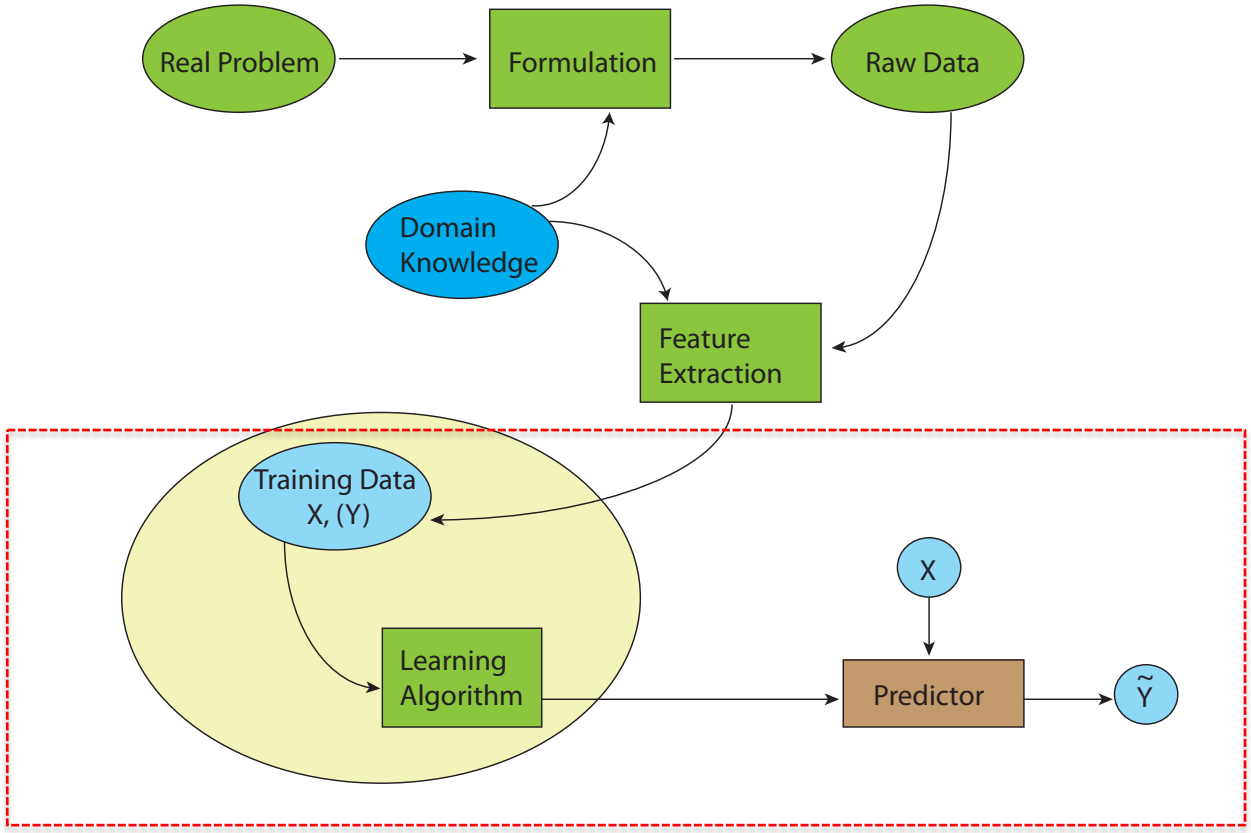
- (X,Y) may not even have a deterministic relationship.
- Unmeasured variables may contribute to Y (e.g, measurement error, technical effects, latent variables).

Our goal

Find a good approximation $f(X)$ of $\hat{f}(X)$ that accurately describes the relationship between input and output variables.

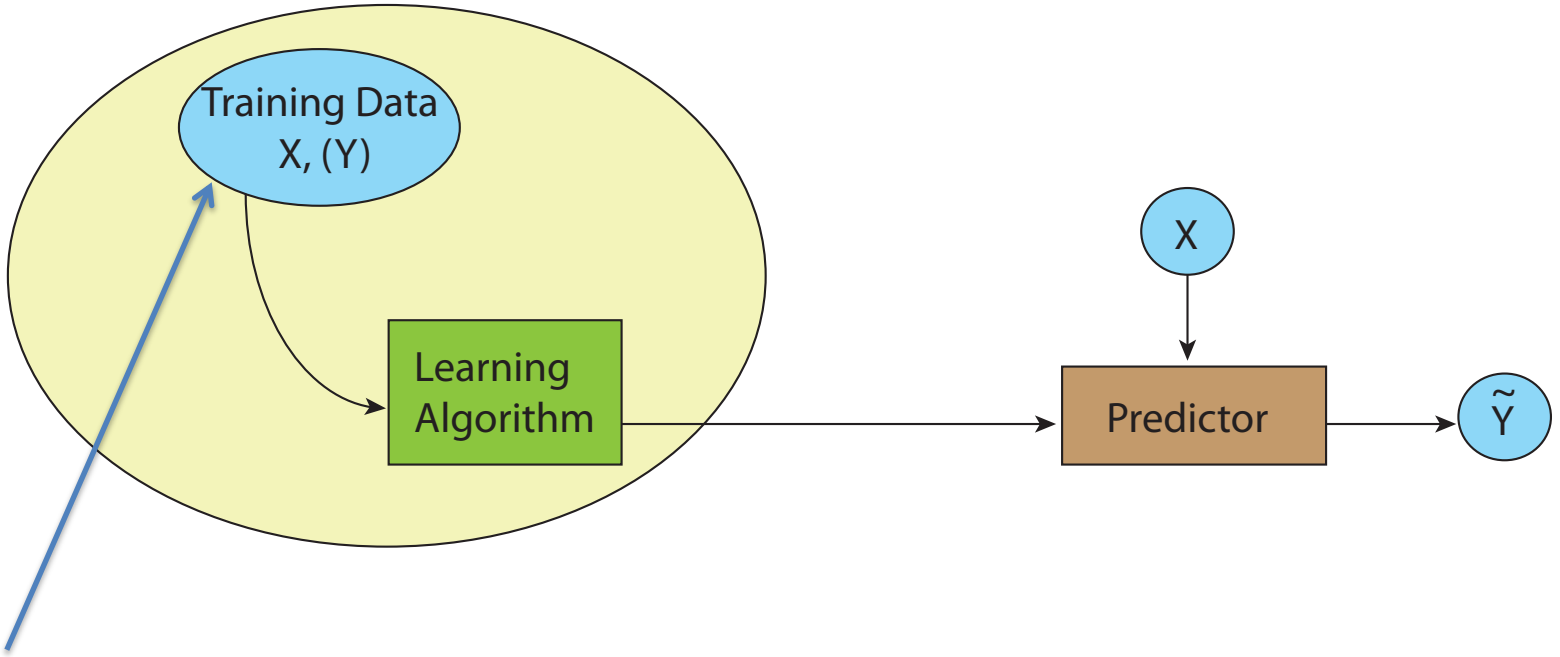
Functional Approximation

Supervised Learning with the additive error model $Y = f(x) + \varepsilon$



Functional Approximation

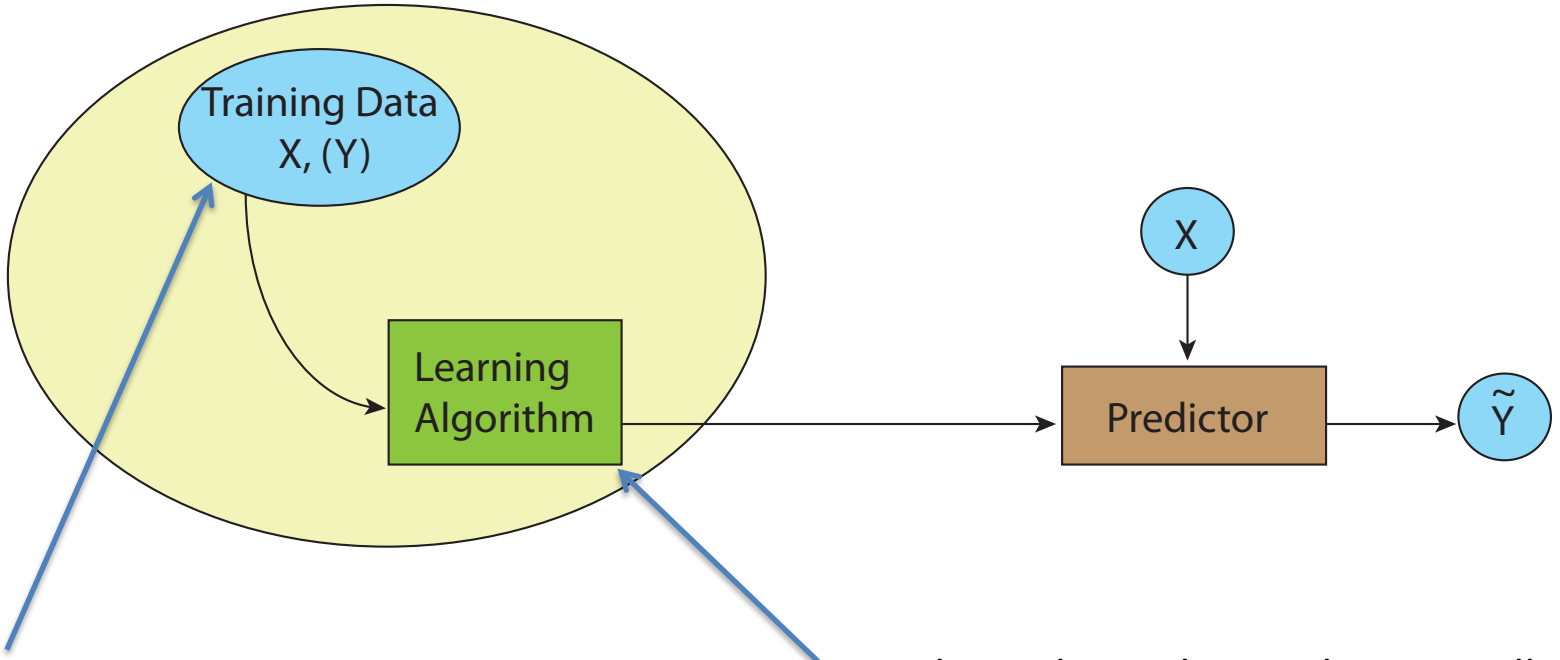
Supervised Learning with the additive error model $Y = f(x) + \varepsilon$



f(x) may or may not be deterministic, we only observe X and Y. Assume we have no prior knowledge

Functional Approximation

Supervised Learning with the additive error model $Y = f(x) + \varepsilon$

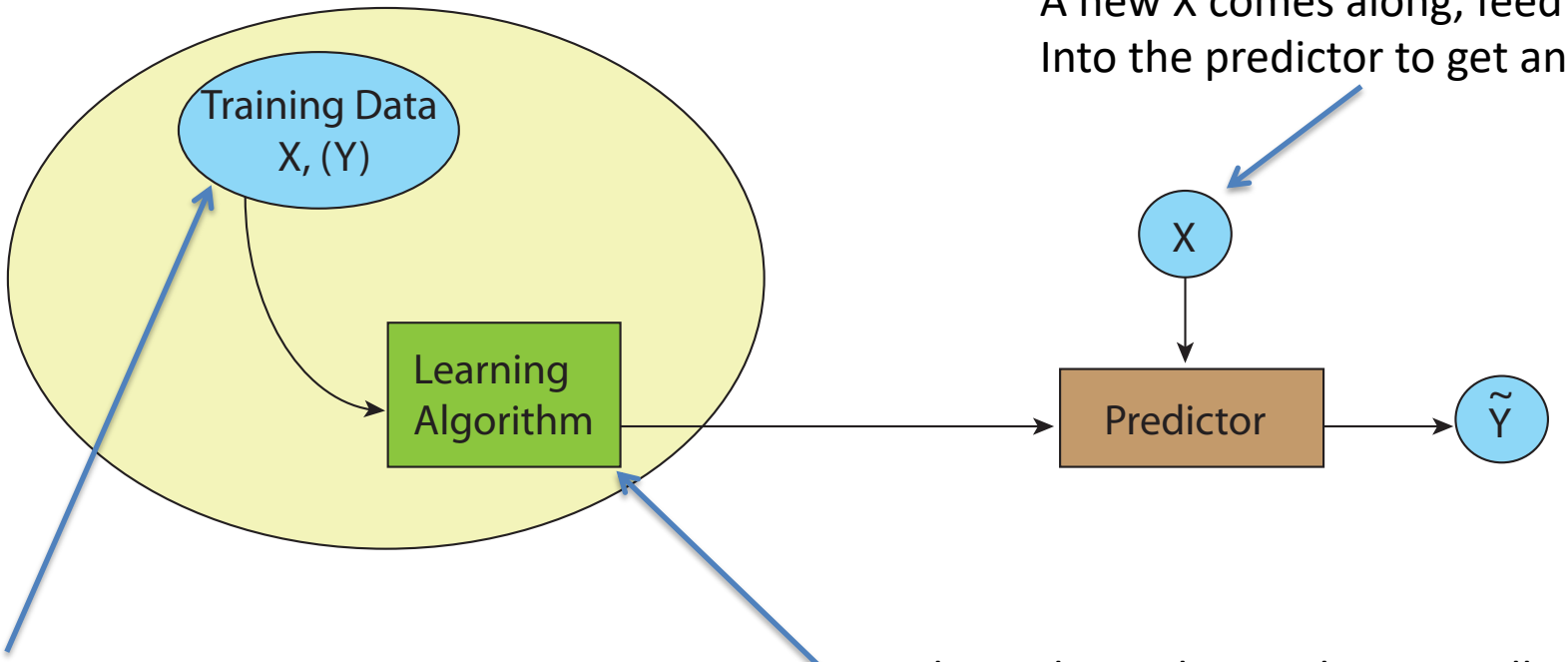


f(x) may or may not be deterministic, we only observe X and Y. Assume we have no prior knowledge

Feed X and Y to the machine, it will tell you the $\hat{f}(x)$ that minimizes the residual sum of squares. We "learn" the optimal predictor.

Functional Approximation

Supervised Learning with the additive error model $Y = f(x) + \varepsilon$



f(x) may or may not be deterministic, we only observe X and Y. Assume we have no prior knowledge

Feed X and Y to the machine, it will tell you the $\hat{f}(x)$ that minimizes the residual sum of squares. We "learn" the optimal predictor.

Functional Approximation

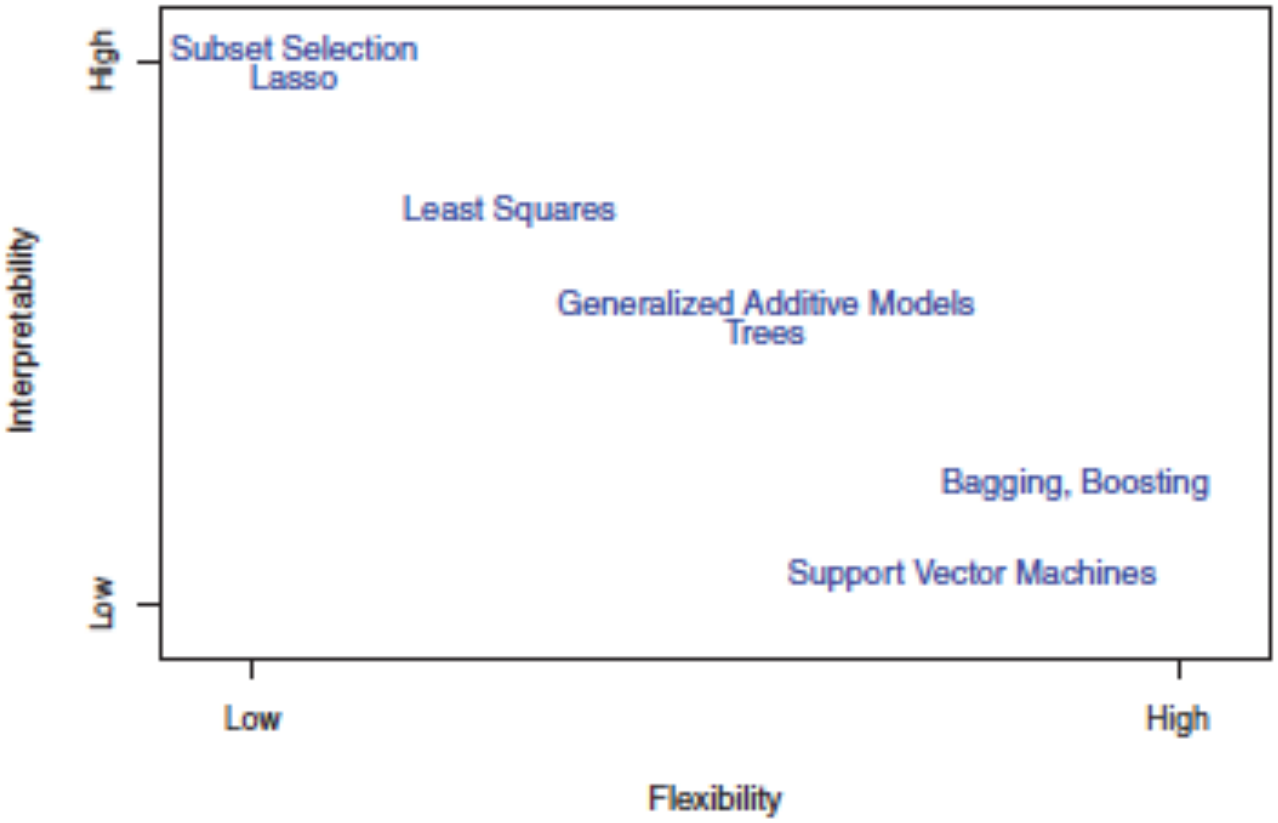
There are two reasons to estimate f(x):

(1) Prediction – minimize “reducible error”.

(2) Inference

$\hat{f}(x)$

Functional Approximation



Functional Approximation

Goal: Obtain a 'useful approximation' to $f(x)$ for all $x \in \Re^p$ in a Euclidean space, given the training data T.

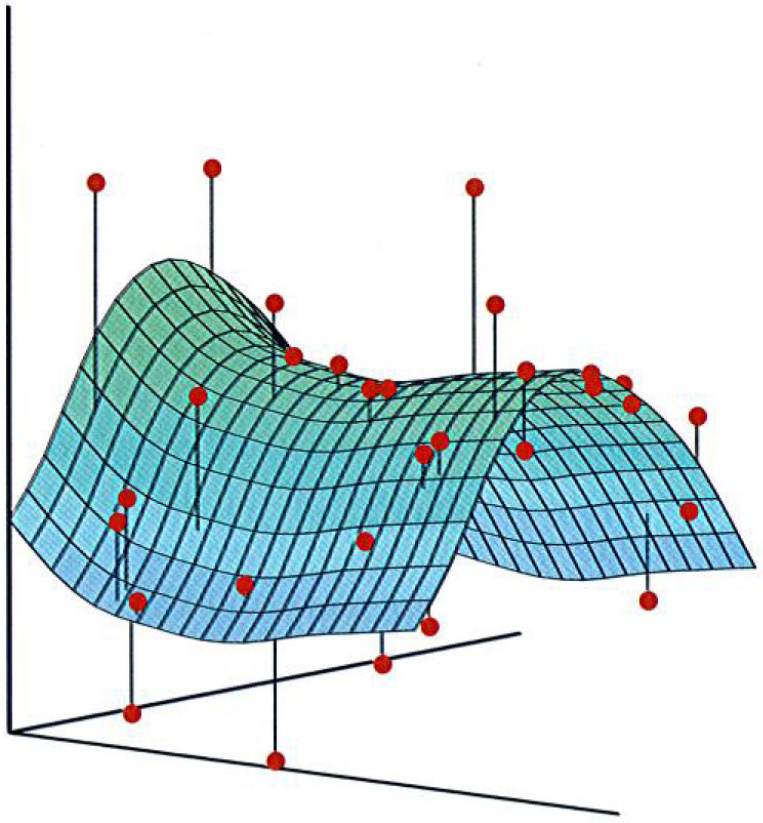
Parameters: Approximations depend on parameters θ, e.g.,

- Linear Model: $f(x) = x^T \beta$, the parameters $\theta = \beta$.
- Linear Basis Expansions: $f_\theta(x) = \sum_{k=1}^{K} h_k(x)\theta_k$,
  h_k where can be polynomial, trigonometric, sigmodial, etc.

Functional Approximation

We can take a least squares approach and find the parameters that minimize the RSS:

$$RSS(\theta) = \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2.$$



Functional Approximation

A **more general approach** is maximum likelihood.
Suppose we have a random sample $y_i,\ i = 1 \ldots N$ from a density $\Pr_\theta(y)$ indexed by some parameters θ.

The log-probability of the observed sample is:

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

Structured Regression Models

Consider the RSS:

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

Difficulty of the problem:

- There are infinitely many solutions.
- May be a poor predictor of non-test data.
- There may be multiple observation pairs per x_i .

We need to place constraints on the solution space!

Structured Regression Models

Solution Constraints - How can we do it?
A couple of ways…..

- Encoded via parametric representation of the function we are trying to approximate.
- Build into the learning method, implicitly or explicitly. Often called a “complexity restriction”.

- Constraints are often imposed in local neighborhoods.
- Strong constraints – large neighborhood size. The more sensitive the solution is to the adopted constraint.

Structured Regression Models

Solution Constraints - How can we do it?
A couple of ways…..

- Encoded via parametric representation of the function we are trying to approximate.
- Build into the learning method, implicitly or explicitly. Often called a "complexity restriction".

- Constraints are often imposed in local neighborhoods.
- Strong constraints – large neighborhood size. The more sensitive the solution is to the adopted constraint.

IMPORTANT NOTE: Before we have ambiguity arising from infinitely many solutions**.** Constraint restrict the class of feasible functions. However, the ambiguity remains.
Why? Because there are infinitely many constraints.

Roughness Penalty and Bayesian Methods

Penalized RSS:

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

User-selected functional
Lots of possibilities.

Example - cubic smoothing spline

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

Penalizes large second derivatives.

Prior – we expect a smooth solution. Tuning parameter lambda conveys our belief in the smoothness.

Kernel Methods and Local Regression

Specifying the local neighborhood by a **Kernel function:**
$K_\lambda(x_0, x)$ which assigns weights to points x in a region around x_0.

Example – Gaussian Kernel
Weights die exponentially with their squared Euclidean Distance from the center point.

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

Incorporates variance of the Gaussian density and controls neighborhood size.

Residual Sum of Squares:

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

Example – Nearest Neighbor (data-centric)

Model Selection and Bias-Variance tradeoff

These methods all employ a **smoothing** or **complexity parameter**:

- The multiplier on the penalty term.
- The width of the kernel.
- The number of basis functions.

How to choose complexity parameter? $\hat{\lambda}$

We can't use RSS, because we would always choose the parameters that gives zeros residual. Would be a bad predictor for future data.

Model Selection and Bias-Variance tradeoff

Consider a given estimate $\hat{f}$ and a set of predictors X, which yields the prediction $\hat{Y} = f(X)$.

$$E(Y - \hat{Y})^2 = E\left[f(X) + \varepsilon - \hat{f}(X)\right]^2$$

$$= \left[f(X) - \hat{f}(X)\right]^2 + Var(\varepsilon)$$

Reducible

Irreducible

Model Selection and Bias-Variance tradeoff

Back to nearest neighbors: The expected prediction error (EPE) at x_0:

$$EPE_k(x_0) = E\left[(Y - \hat{f}_k(x_0))^2 \mid X = x_0\right]$$

$$= \sigma^2 + \left[Bias^2(\hat{f}_k(x_0)) + Var_T(\hat{f}_k(x_0))\right]$$

$$= \sigma^2 + \left[f(x_0) - \frac{1}{k}\sum_{l=1}^{k} f(x_l)\right]^2 + \frac{\sigma^2}{k}.$$

MSE

Irreducible error – beyond out control.

Squared difference between the true mean and the estimated. Likely to Increase with k.

The variance of an average. As k Increases this decreases.

Model Selection and Bias-Variance tradeoff