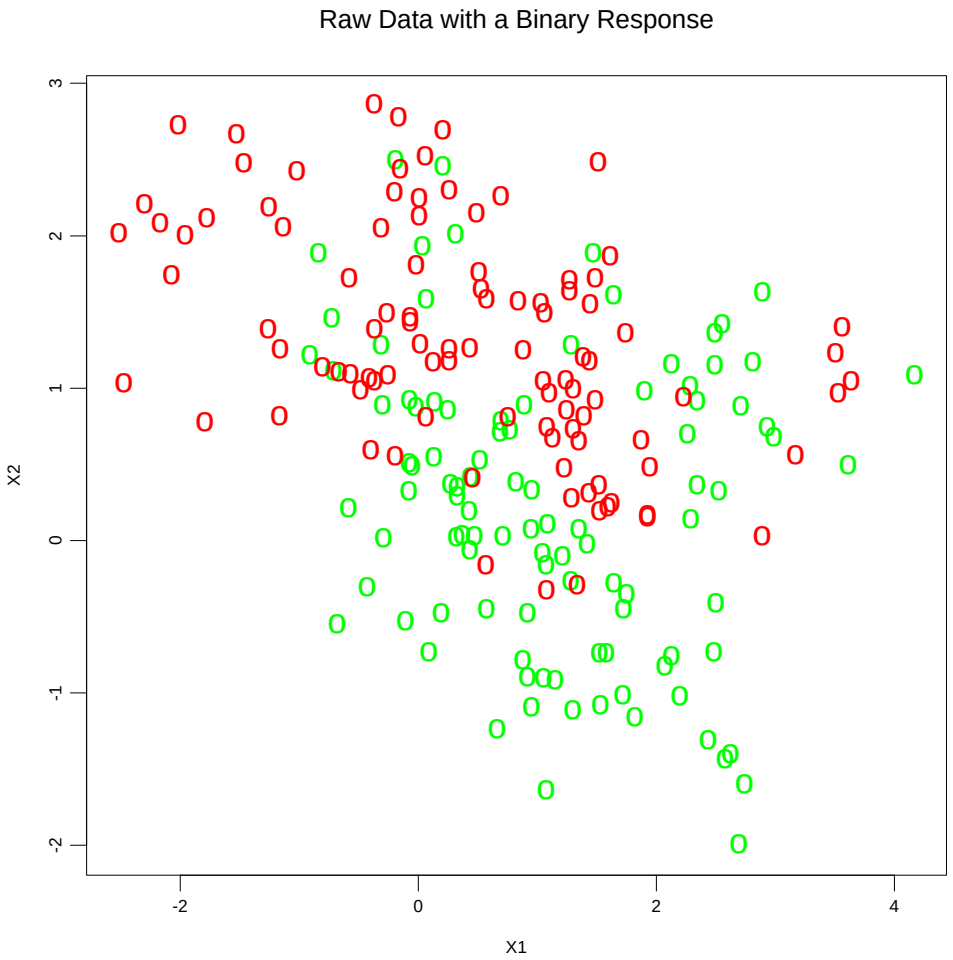
Overview of Supervised Learning

Notation

- X: inputs, feature vector, predictors, independent variables. Generally X will be a vector of p real values. Qualitative features are coded in X using, for example, dummy variables. Sample values of X generally in lower case; x_i is ith of N sample values. Bold $\mathbf{X}$ represents a matrix of feature values, with an x_i in each row.
- Y: output, response, dependent variable. Typically a scalar, can be a vector, of real values. Again y_i is a realized value.
- G: a qualitative response, taking values in a discrete set $\mathcal{G}$; e.g. $\mathcal{G} = \{\text{survived, died}\}$. We often code G via a binary indicator response vector Y.



200 points generated in $\mathbb{R}^2$ from an unknown distribution; 100 in each of two classes $\mathcal{G} = \{\texttt{GREEN}, \texttt{RED}\}$. Can we build a rule to predict the color of 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 $\mathbf{X}$ and a response vector $\mathbf{y}$,

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

(Drop β_0 and make first column of $\mathbf{X}$ equal to 1)

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

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

- The decision boundary is $\{x | x^T \hat{\beta} = 0.5\}$ is linear (Figure 2.1) (and seems to make many errors on the training data).

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 (Chapter 4).

For Scenario 2, it is far too rigid.

K-Nearest Neighbors

A natural way to classify a new point is to have a look at its neighbors, and take a vote:

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

where $N_k(x)$ is a neighborhood of x that contains exactly k neighbors (k-nearest neighborhood).

If there is a clear dominance of one of the classes in the neighborhood of an observation x, then it is likely that the observation itself would belong to that class, too. Thus the classification rule is the majority voting among the members of $N_k(x)$. As before,

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

Figure 2.2 shows the result of 15-nearest neighbor classification. Fewer training data are misclassified, and the decision boundary adapts to the local densities of the classes.

Figure 2.3 shows the result of 1-nearest neighbor classification. None of the training data are misclassified.

Linear regression uses 3 parameters to describe its fit. Does K-nearest neighbors use 1 (the value of k here)?

More realistically, k-nearest neighbors uses N/k effective number of parameters

Many modern procedures are variants of linear regression and K-nearest neighbors:

- Kernel smoothers
- Local linear regression
- Linear basis expansions
- Projection pursuit and neural networks

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 $1/5$.

See page 17 for more details, or the book website for the actual data.

Figure 2.4 shows the results of classifying 10,000 test observations generated from this distribution.

The Bayes Error is the best performance possible (Figure 2.5).

Statistical decision theory

Case 1: Quantitative output Y

- Let $X \in R^p$ denote a real valued random input vector
- We have a Loss function $L(Y, f(X))$ for penalizing errors in prediction.
- Most common and convenient is squared error loss: $L(Y, f(X) = (Y - f(X))^2$.
- This leads us to a criterion for choosing f,

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

  the Expected (squared) Prediction Error,

-

$$EPE(f) \quad = \quad E_x[E_{y|x}(Y - E(Y|x) + E(Y|x) - f(x))]^2$$

$$
\begin{aligned}
&= E_x(E_{y|x}(Y - E(Y|x))^2 + E_x(E(Y|x) - f(x)))^2 \\
&= \text{Bayes error} + MSE \qquad (1)
\end{aligned}
$$

- Minimizing $EPE(f)$ leads to a solution $f(x) = E(Y|X = x)$, the conditional expectation, also known as the regression function.

Case 2: Qualitative output G:

- Suppose our prediction rule is $\hat{G}(X)$, and G and $\hat{G}(X)$ take values in $\mathcal{G}$, with card$(\mathcal{G}) = K$.
- We have a different loss function for penalizing prediction errors. $L(k, \ell)$ is the price paid for classifying an observation belonging to class $\mathcal{G}_k$ as $\mathcal{G}_\ell$.
- Most often we use the 0-1 loss function where all misclassifications are charged a single unit.
- The expected prediction error is

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

- Solution is

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

With the 0-1 loss function this simplifies to

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

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 did we construct the Bayes classifier for our simulation example?

- K-nn tries to implement conditional expectations directly, by
  - Approximating expectations by sample averages
  - Relaxing the notion of conditioning at a point, to conditioning in a region about a point.
- As $N, k \to \infty$, such that $k/N \to 0$, the K-nearest neighbor estimate $\hat{f}(x) \to E(Y|X = x)$ — it is consistent.
- Linear regression assumes a (linear) structural form for $f(x) = x^T\beta$, and minimizes sample version of EPE directly.
- As sample size grows, our estimate of linear coefficients $\hat{\beta}$ converges to the optimal $\beta_{opt} = E(XX^T)^{-1}E(XY)$.
- Model is limited by the linearity assumption

Question: Why not always use k-nearest neighbors?

Curse of dimensionality

K-nearest neighbors can fail in high dimensions, because it becomes difficult to gather K 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.

See Figure 2.6.

- 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 $\mathbb{R}^{10}$

Example 1

- 1000 training examples x_i generated uniformly on $[-1, 1]^p$.
- $Y = f(X) = e^{-8||X||^2}$ (no measurement error).
- use the 1-nearest-neighbor rule to predict y_0 at the test-point $x_0 = 0$.

$$\begin{aligned} \text{MSE}(x_0) &= \text{E}_{\mathcal{T}}[f(x_0) - \hat{y}_0]^2 \\ &= \text{E}_{\mathcal{T}}[\hat{y}_0 - \text{E}_{\mathcal{T}}(\hat{y}_0)]^2 + [\text{E}_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\ &= \text{Var}_{\mathcal{T}}(\hat{y}_0) + \text{Bias}^2(\hat{y}_0). \end{aligned}$$

Figure 2.7 shows what happens as p increases.

Figure 2.8 is the same, except here $f(X) = \frac{1}{2}(X_1 + 1)^3$. Here we have

- $\text{E}\hat{f}(x_0) = \text{E}f(X_{(1)}) \approx f(\text{E}X_{(1)}) = f(x_0)$ for all dimensions
- $\text{Var}\hat{f}(x_0) = \text{Var}f(X_{(1)}) \uparrow$ with dimension

Example 2

If the linear model is correct, or almost correct, K-nearest neighbors will do much worse than linear regression.

In cases like this (and of course, assuming we know this is the case), simple linear regression methods are not affected by the dimension.

Figure 2.9 illustrates two simple cases.

Statistical Models

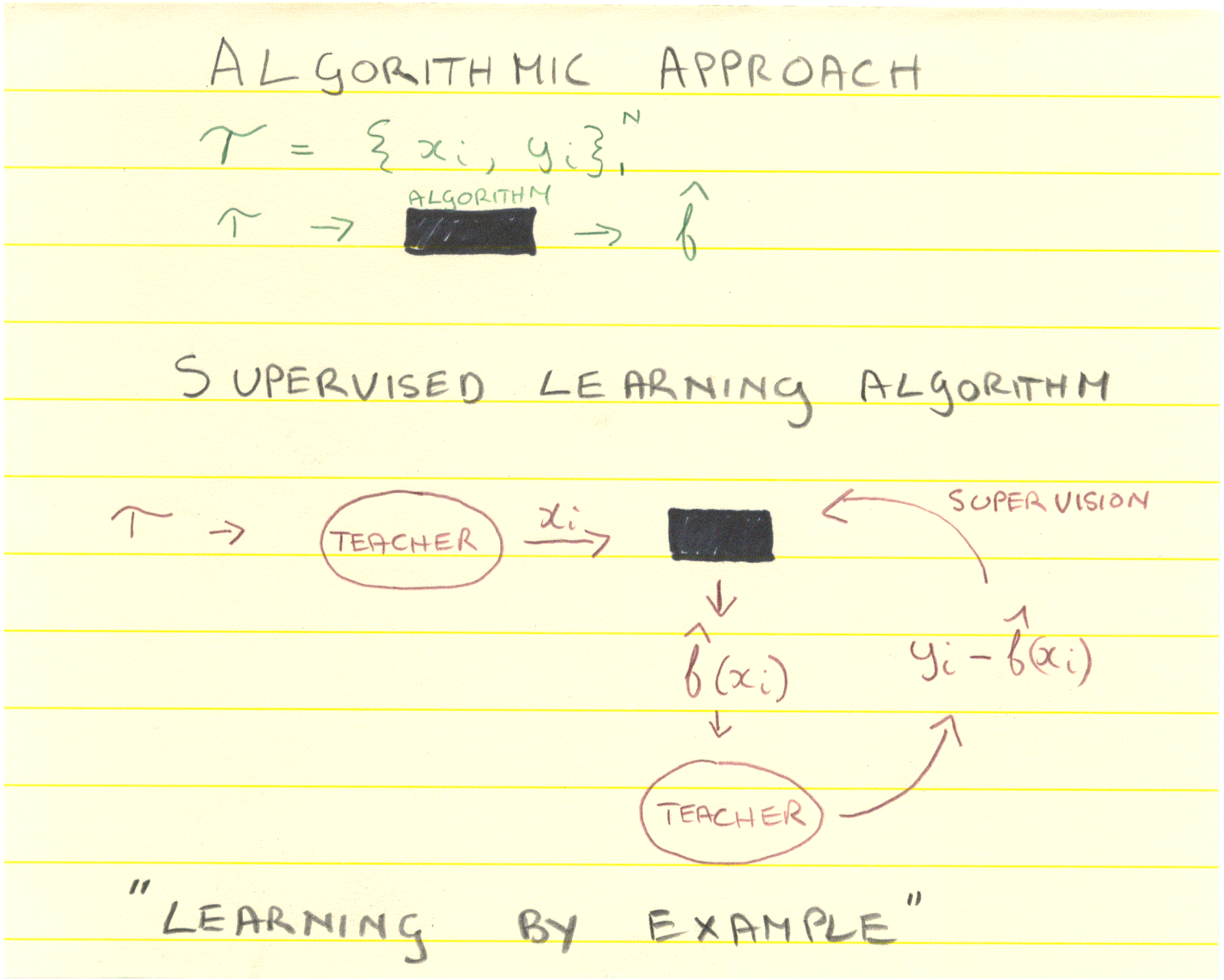
$$Y = f(X) + \varepsilon$$

with $\mathrm{E}(\varepsilon) = 0$ and X and ε independent.

- $\mathrm{E}(Y|X) = f(X)$
- $\Pr(Y|X)$ depends on X only through $f(X)$.
- Useful approximation to the truth — all unmeasured variables captured by ε
- N realizations $y_i = f(x_i) + \varepsilon_i,\ i = 1, \ldots, N$
- Assume ε_i and ε_j are independent.

More generally can have, for example, $\mathrm{Var}(Y|X) = \sigma^2(X)$.

For qualitative outcomes $\{\Pr(G = \mathcal{G}_k|X)\}_1^K = p(X)$ which we model directly.


ALGORITHMIC APPROACH
$\mathcal{T} = \{x_i, y_i\}_1^N$
ALGORITHM
$\hat{f}$
SUPERVISED LEARNING ALGORITHM
TEACHER
x_i
SUPERVISION
$\hat{f}(x_i)$
$y_i - \hat{f}(x_i)$
TEACHER
"LEARNING BY EXAMPLE"

Function Approximation

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

Assumes

- $x_i,\ y_i$ are points in, say $\mathbb{R}^{p+1}$.
- A (parametric or non-parametric) form for $f(X)$
- A loss function for measuring the quality of the approximation.

Figure 2.10 illustrates the situation.

More generally, Maximum Likelihood Estimation provides a natural basis for estimation. We will see examples such as logistic regression via the binomial likelihood.

Structured Regression Models

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

- Any function passing through (x_i, y_i) has $RSS = 0$
- Need to restrict the class
- Usually restrictions impose local behavior — see equivalent kernels in chapters 5 and 6
- Any method that attempts to approximate locally varying functions is "cursed"
- Alternatively, any method that "overcomes" the curse, assumes an implicit metric that does not allow neighborhoods to be simultaneously small in all directions.

Classes of Restricted Estimators

Some of the classes of restricted methods that we cover are

- Roughness Penalty and Bayesian Methods (chap 5)

$$\text{Penalized RSS}(f, \lambda) = \text{RSS}(f) + \lambda J(f)$$

- Basis functions and dictionary methods (chap 5)

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

- Kernel Methods and Local Regression (chap 6)

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

Model Selection and the Bias-Variance Tradeoff

Many of the flexible methods have a complexity parameter:

- the multiplier of the penalty term
- the width of the kernel
- the number of basis functions

Cannot use RSS to determine this parameter — why?

Can use Prediction error on unseen test cases to guide us

E.g. $Y = f(X) + \varepsilon$, K-nn (and assume the sample x_i are fixed):

$$\begin{aligned} \mathrm{E}[(Y - \hat{f}_k(x_0))^2 | X = x_0] &= \sigma^2 + \mathrm{Bias}^2(\hat{f}_k(x_0)) + \mathrm{Var}_{\mathcal{T}}(\hat{f}_k(x_0)) \\ &= \sigma^2 + [f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})]^2 + \frac{\sigma^2}{k} \end{aligned}$$

Selecting k is a bias-variance tradeoff — see Figure 2.11.