Overview of Supervised Learning

Chap – 2 ; - Part - I

T. Hastie, R.Tibshirani, J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference and Prediction", Springer Series in Statistics, 2009

Variable Types and Terminology

- Denote an input variable by the symbol X. If X is a vector, its components can be accessed by subscripts X_i . Quantitative outputs will be denoted by Y, qualitative outputs by G (for group).
- Observed values are written in lowercase; hence the i^{th} observed value of X is written as x_i (where x_i is again a scalar or vector). **Matrices** are represented by bold uppercase letters; for example, a set of N input p-vectors x_i, $i = 1, \ldots, N$ would be represented by the $N \times p$ matrix $\boldsymbol{X}$.
- In general, vectors will not be bold, except when they have N components; this convention distinguishes a p-vector of inputs x_i for the i^{th} observation from the *N-vector $\boldsymbol{x_j}$ consisting of all the observations on variable X_j* . Since all vectors are assumed to be **column vectors,** the i^{th} row of $\boldsymbol{X}$ is x_i^T, the vector transpose of x_i. i.e. #samples = N; Dimension = p;
  e.g. p sensors and N observations from each.

CONTD...

- Loosely state the learning task as follows: given the value of an input vector X, make a good prediction of the output Y, denoted by $\hat{Y}$. If Y takes values in $\mathbb{R}$ then so should $\hat{Y}$; associated with G.

- For a two-class G, one approach is to denote the binary coded target as Y , and then treat it as a quantitative output. The predictions $\hat{Y}$ will typically lie in $[0,1]$, and we can assign to $\hat{G}$ the class label according to whether $\hat{y} > 0.5$. This approach generalizes to K-level qualitative outputs as well.

- We need data to construct prediction rules, often a lot of it. We thus suppose we have available a set of measurements (x_i, y_i) or (x_i, g_i), $i = 1, \ldots, N$, known as the training data, with which to construct our prediction rule.

Two Simple Approaches to Prediction: Least Squares and Nearest Neighbors

- Two simple but powerful prediction methods: the linear model fit by least squares and the k-nearest-neighbor prediction rule. The linear model makes huge assumptions about structure and ***yields stable but possibly inaccurate predictions***. The method of k-nearest neighbors makes very mild structural assumptions: its predictions are ***often accurate but can be unstable***.

- **Linear Models and Least Squares**

- Given a vector of inputs $X^T = (X_1, X_2, \ldots, X_p)$, we predict the output Y via the model.

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

Linear Models and Least Squares

- The term $\hat{\beta}_0$ is the intercept, also known as the bias in machine learning.

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

where, X^T denotes vector or matrix transpose (X being a column vector). Here we are modeling a single output, so $\hat{Y}$ is a scalar; in general $\hat{Y}$ can be a K–vector, in which case $\hat{\beta}$ would be a $p \times K$ matrix of coefficients.

- In the $(p + 1)$-dimensional input–output space, $(X, \hat{Y})$ represents a hyperplane. If the constant is included in X, then the hyperplane includes the origin and is a subspace; if not, it is an affine set cutting the Y -axis at the point $(0,\hat{\beta}_0)$. From now on we assume that the intercept is included in $\hat{\beta}$.

- Viewed as a function over the p-dimensional input space, $f(X) = X^T\beta$ is linear, and the gradient $f_0(X) = \beta$ is a vector in input space that points in the steepest uphill direction.

- How do we fit the linear model we pick the coefficients to minimize the residual sum of squares

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

- $RSS(\beta)$ is a quadratic function of the parameters, and hence its minimum always exists, but may not be unique. The solution is easiest to characterize in matrix notation. We can write

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

- where $\boldsymbol{X}$ is an $N \times p$ matrix with each row an input vector, and $\boldsymbol{y}$ is an N-vector of the outputs in the training set. Differentiating w.r.t. β we get the normal equations (derive):

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

- If $\boldsymbol{X}^T\boldsymbol{X}$ is nonsingular, then the unique solution is given by

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

- And the fitted value at the i^{th} input x_i is $\hat{y}_i = \hat{y}(x_i) = x_i^T\hat{\beta}$
- The entire fitted surface is characterized by the p parameters $\hat{\beta}$.
- Let, (see fig. in next slide) output class variable G has the values: **BLUE** and **ORANGE**.

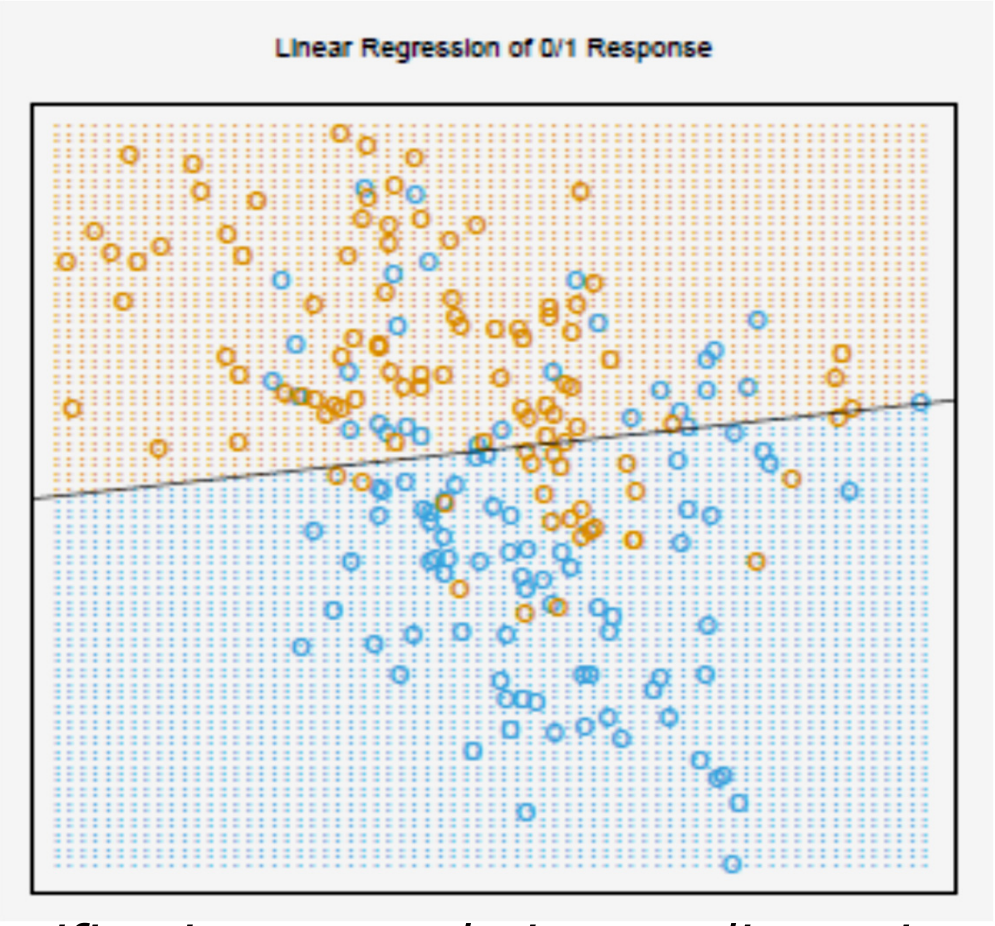

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by* $x^T \hat{\beta} = 0.5$. *The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

- Response Y coded as 0 for **BLUE** and 1 for **ORANGE**. The fitted values $\hat{Y}$ are converted to a fitted class variable $\hat{G}$ according to the rule.

$$\hat{G} = \begin{cases} \textbf{\textit{ORANGE}} & if \quad \hat{Y} \geq 0.5 \\ \textbf{\textit{BLUE}} & if \quad \hat{Y} \leq 0.5 \end{cases}$$

- The set of points in $\mathbb{R}^2$ classified as ORANGE corresponds to $\{x : x^T\hat{\beta} > 0.5\}$, indicated in Figure 2.1, and the two predicted classes are separated by the decision boundary $\{x : x^T\hat{\beta} = 0.5\}$, which is linear in this case.

- Reason for error: Source of data (not discussed) or inappropriate model ??

❑ DATA SOURCE:

- *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.

- A mixture of Gaussians is best described in terms of the generative model. In the case of one Gaussian per class, a linear decision boundary is the best one can do, and that our estimate is almost optimal. The region of overlap is inevitable, and future data to be predicted will be plagued by this overlap as well.

- In the case of mixtures of tightly clustered Gaussians the story is different. A linear decision boundary is unlikely to be optimal, and in fact is not. The optimal decision boundary is nonlinear and disjoint, and as such will be much more difficult to obtain.

Nearest-Neighbor Methods

- Specifically, the k-nearest neighbor fit for $\hat{Y}$ is defined as follows:

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

- Where $N_k(x)$ is the neighborhood of x defined by the k closest points x_i in the training sample. Closeness implies a metric, which for the moment we assume is Euclidean distance.

- So, in words, we find the k observations with x_i closest to x in input space, and average their responses.

15-Nearest Neighbor Classifier

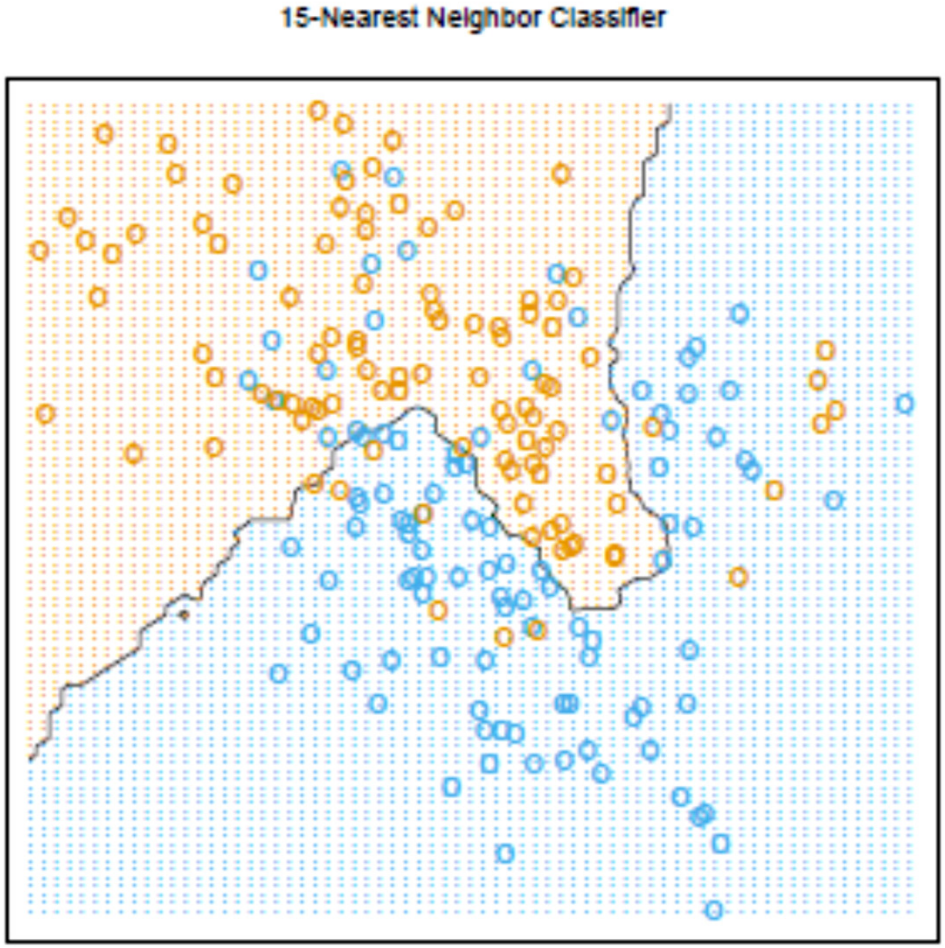

FIGURE 2.2. The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors

1−Nearest Neighbor Classifier

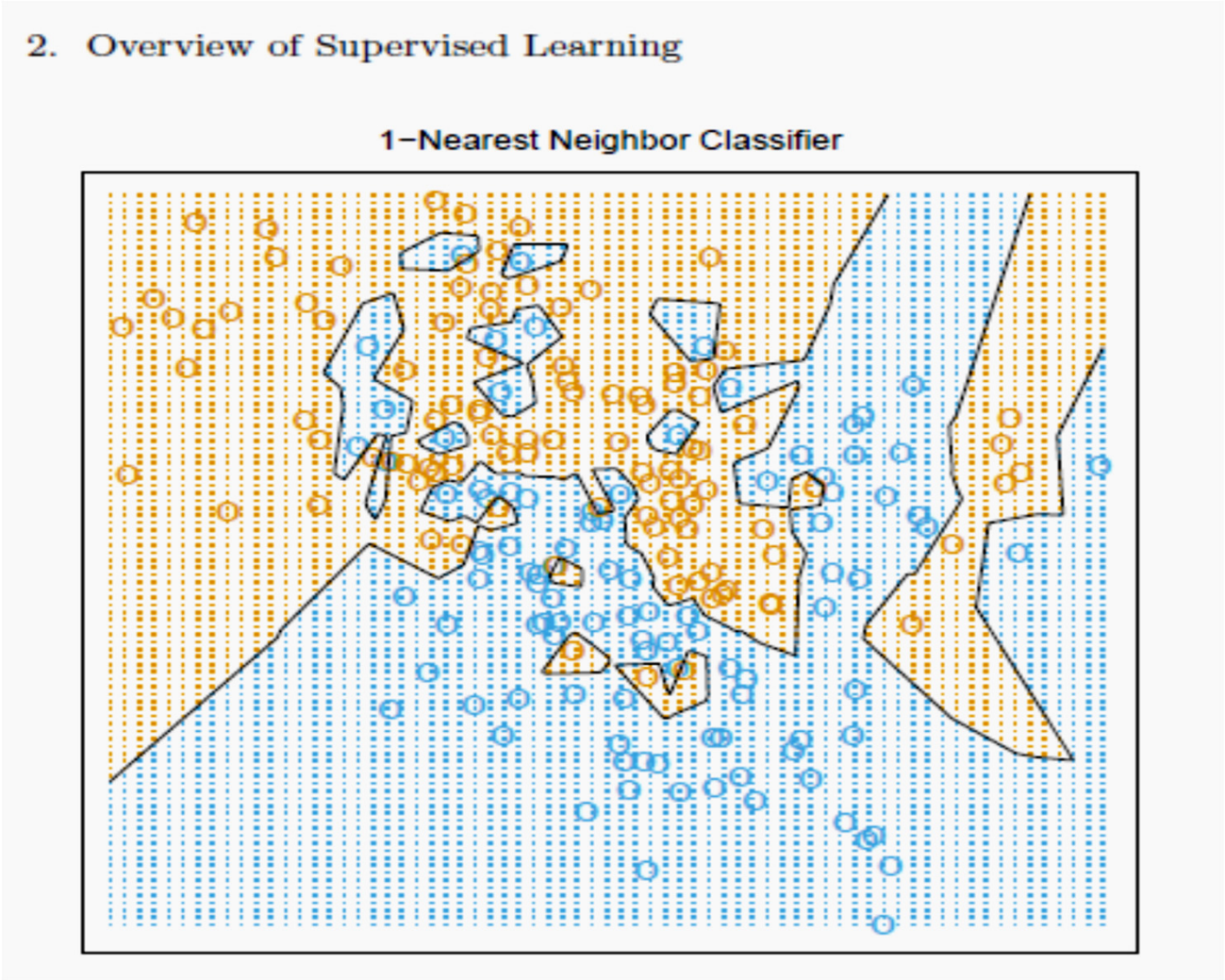

FIGURE 2.3. *predicted by 1-nearest-neighbor classification.*

Nearest Neighbor Classifier

- In Figure 2.2 we see that far fewer training observations are misclassified than in Figure 2.1.
- This should not give us too much comfort, though, since in Figure 2.3 none of the training data are misclassified.
- A little thought suggests that for k-nearest-neighbor fits, the error on the training data should be approximately an increasing function of k, and will always be 0 for $k = 1$.
- An independent test set would give us a more satisfactory means for comparing the different methods.

Nearest Neighbor Classifier

- It appears that k-nearest-neighbor fits have a single parameter, the number of neighbors k, compared to the p parameters in least-squares fits.
- Although this is the case, we will see that **the effective number of parameters of k-nearest neighbors is N/k** and is generally bigger than p, and decreases with increasing k.

- It would seem that k-nearest-neighbor methods would be more appropriate for the mixture Scenario 2 described earlier, while for Gaussian data the decision boundaries of k-nearest neighbors would be unnecessarily noisy.

From Least Squares to Nearest Neighbors

- The **linear decision boundary from least squares is very smooth, and apparently stable to fit**. It does appear to rely heavily on the assumption that a linear decision boundary is appropriate. In language we will develop shortly, it has **low variance and potentially high bias.**

- On the other hand, the k-nearest-neighbor procedures do not appear to rely on any stringent assumptions about the underlying data, and can adapt to any situation.

- However, any particular sub region of the **decision boundary (for K-NN)** depends on a handful of input points and their particular positions, and is thus **wiggly and unstable—high variance and low bias.**

  - ❖ Local regression fits linear models by locally weighted least squares, 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 consist of sums of nonlinearly transformed linear models.

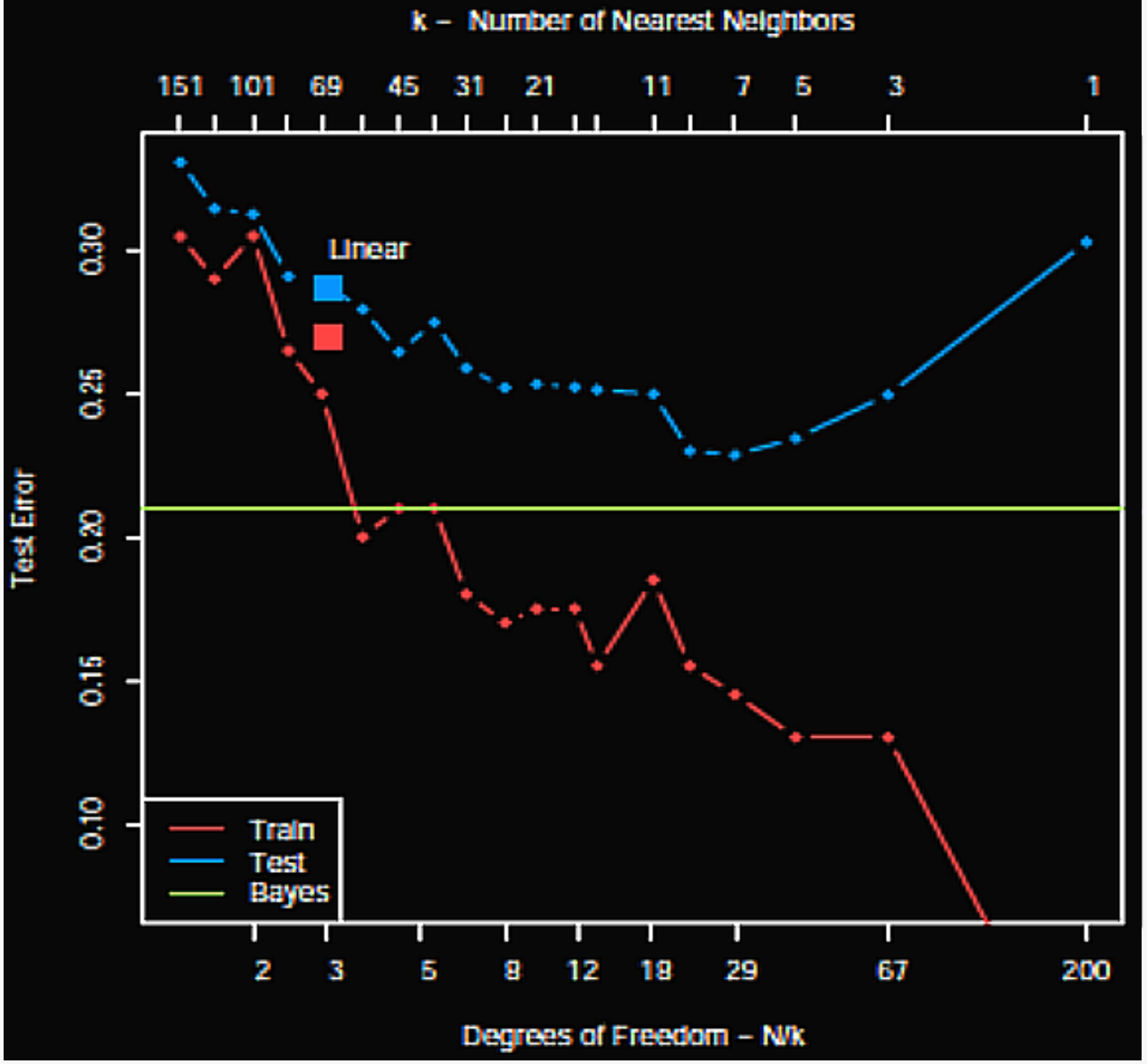

FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* 10,000. *The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.*

Statistical Decision Theory (SDT)

- SDT provides a framework for developing models such as those discussed informally so far. First consider the case of a quantitative output, and place ourselves in the world of random variables and probability spaces.
- *Let $X \in R^p$ denote a real valued random input vector, and $Y \in R$ a real valued random output variable*, with joint distribution $Pr(X, Y)$. We seek a function $f(X)$ for predicting Y given values of the input X. This theory requires a loss function $L(Y, f(X))$ for penalizing errors in prediction, and by far the most common and convenient is squared error loss:

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

This leads us to a criterion for choosing f,

$$EPE(f) = E\big(Y - f(X)\big)^2 \qquad (2.9)$$

$$= \int [y - f(x)]^2 \, \Pr(dx, dy),$$

the expected (squared) prediction error.

By conditioning[1] on X, we can write EPE as

$$EPE(f) = E_X E_{(Y|X)}([Y - f(X)]^2|X) \quad \text{.....(2.11)}$$

- And we see that it suffices to minimize EPE pointwise:

$$f(x) = argmin_c E_{Y|X}([Y - c]^2|X = x)$$

- The solution is

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

the conditional expectation, also known as the **regression function**. Thus the best prediction of Y at any point $X = x$ is the conditional mean, when best is measured by average squared error.

$$\begin{aligned}
&\mathbb{E}[(Y - \hat{f}(\vec{X}))^2] \\
&\quad = \mathbb{E}[(Y - \mathbb{E}[Y|\vec{X}] + \mathbb{E}[Y|\vec{X}] - \hat{f}(\vec{X}))^2] \\
&\quad = \mathbb{E}[(Y - \mathbb{E}[Y|\vec{X}])^2] + \mathbb{E}[(\mathbb{E}[Y|\vec{X}] - \hat{f}(\vec{X}))^2] \\
&\qquad + 2\mathbb{E}[(Y - \mathbb{E}[Y|\vec{X}])(\mathbb{E}[Y|\vec{X}] - \hat{f}(\vec{X})].
\end{aligned}$$

[1]Conditioning h)Pr(X)(
where $\Pr(Y|X) =$ rdingly.

- The **nearest-neighbor** methods attempt to directly implement this recipe using the training data. At each point x, we might ask for the average of all those $y_i s$ with input $x_i = x$. Since there is typically at most one observation at any point x, we settle for.

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

where "Ave" denotes average, and $N_k(x)$ is the neighborhood containing the k points in T closest to x.

Two approximations are happening here:

- ❖ expectation is approximated by averaging over sample data;
- ❖ conditioning at a point is relaxed to conditioning on some region "close" to the target point.

Issues with NN:

We often do not have very large samples?

If the linear or some more structured model is appropriate, then we can usually get a more stable estimate than k-nearest neighbors, although such knowledge has to be learned from the data as well.

As the dimension p gets large, so does the metric size of the k-nearest neighborhood. So settling for nearest neighborhood as a surrogate for conditioning will fail.

Under mild regularity conditions on the joint probability distribution Pr(X, Y), one can show that as N, k ->∞ such that k/N -> 0,

$$\hat{f}(x) \rightarrow E(Y \mid X = x).$$

The convergence above still holds, but the rate of convergence decreases as the dimension increases.

- How does linear regression fit into this framework? The simplest explanation is that one assumes that the regression function $f(x)$ is approximately linear in its arguments:

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

Statistical Decision Theory…. Cont'd

- Model-based approach—we specify a model for the regression function. Plugging this linear model for $f(x)$ into EPE (2.9) and differentiating we can solve for β theoretically (derive it):

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

Compare this with: $\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbf{y}$ <- Matrix notation

*<*Vector vs matrix vs scalar notations; Dimensions of both expressions ??*
*(p*p)*(p*1) = (p*1); (p*p)*(p*N)*(N*1) = (p*1); *>*

- So both k-nearest neighbors and least squares end up approximating conditional expectations by averages. But they differ dramatically in terms of model assumptions:

  - ❖ Least squares assumes $f(x)$ is well approximated by a globally linear function.

  - ❖ k-nearest neighbors assumes $f(x)$ is well approximated by a locally constant function.

- Are we happy with the criterion (2.11)? What happens if we replace the L_2 loss function with the L_1: $E|Y - f(X)|$?

- The solution in this case is the conditional median,

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

which is a different measure of location, and its estimates are more robust than those for the conditional mean. L_1 criteria has discontinuities – so rarely used in practice; more on this later.

What do we do when the output is a categorical variable G?

Our loss function can be represented by a K×K matrix **L**, where K = card(G). **L** will be zero on the diagonal and nonnegative elsewhere, where L(k, l) is the price paid for classifying an observation belonging to class G_k as G_l. Most often we use the zero–one loss function, where all misclassifications are charged a single unit.

- The expected prediction error is

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

where again the expectation is taken with respect to the joint distribution $\Pr(G, X)$. Again we condition, and can write EPE as.

$$EPE = E_X \sum_{k=1}^{K} L[\mathcal{G}_k, \hat{G}(X)] \Pr(\mathcal{G}_k | X)$$

Minimize EPE pointwise, and simplifying, gets (check derivation):

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

With 0-1 loss, this simplifies to:

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

- This reasonable solution is known as the ***Bayes classifier***, and says that we classify to the most probable class, using the conditional (discrete) distribution $\Pr(G|X)$.
- Figure 2.5 shows the Bayes-optimal decision boundary for our simulation example. The error rate of the Bayes classifier is called the ***Bayes rate.***

Bayes Optimal Classifier

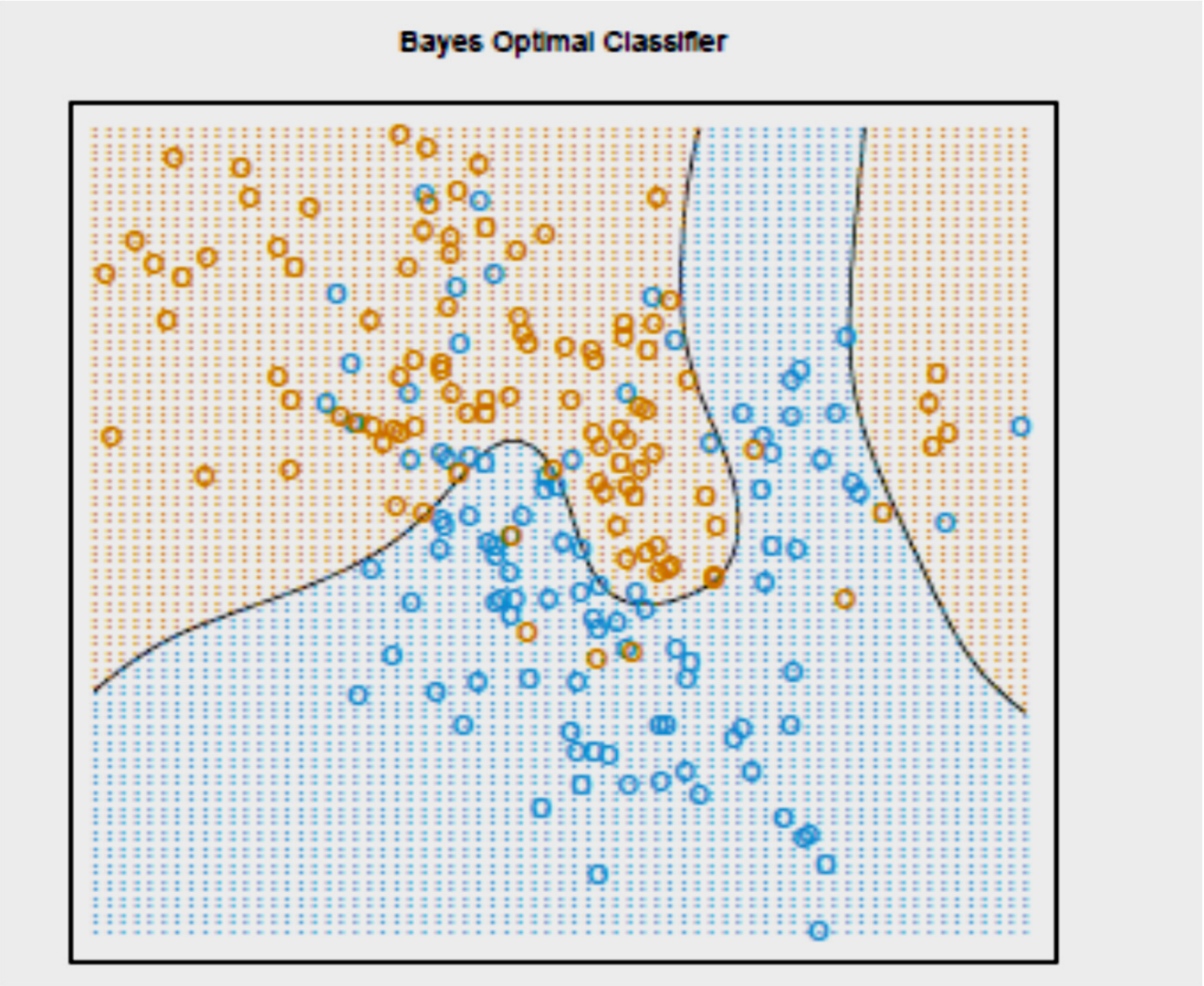

FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly.*

- That the k-nearest neighbor classifier directly approximates this solution—a majority vote in a nearest neighborhood amounts to exactly this, except that conditional probability at a point is relaxed to *conditional probability within a neighborhood of a point*, and probabilities are estimated by training-sample proportions.

- A regression procedure, followed by classification to the largest fitted value, is another way of representing the Bayes classifier (see analytics in book)