k-NN, bias-variance
Statistical decision theory

Aaditya Ramdas

Dept. of Statistics and Data Science
Machine Learning Dept.
Carnegie Mellon University

Outline

1. k-NN (1/2 class)

2. Statistical decision theory (1/2 class)

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*.*

ESL Chapter 2

2.3.2 Nearest-Neighbor Methods

Nearest-neighbor methods use those observations in the training set $\mathcal{T}$ closest in input space to x to form $\hat{Y}$. Specifically, the k-nearest neighbor fit for $\hat{Y}$ is defined as follows:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i, \tag{2.8}$$

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.

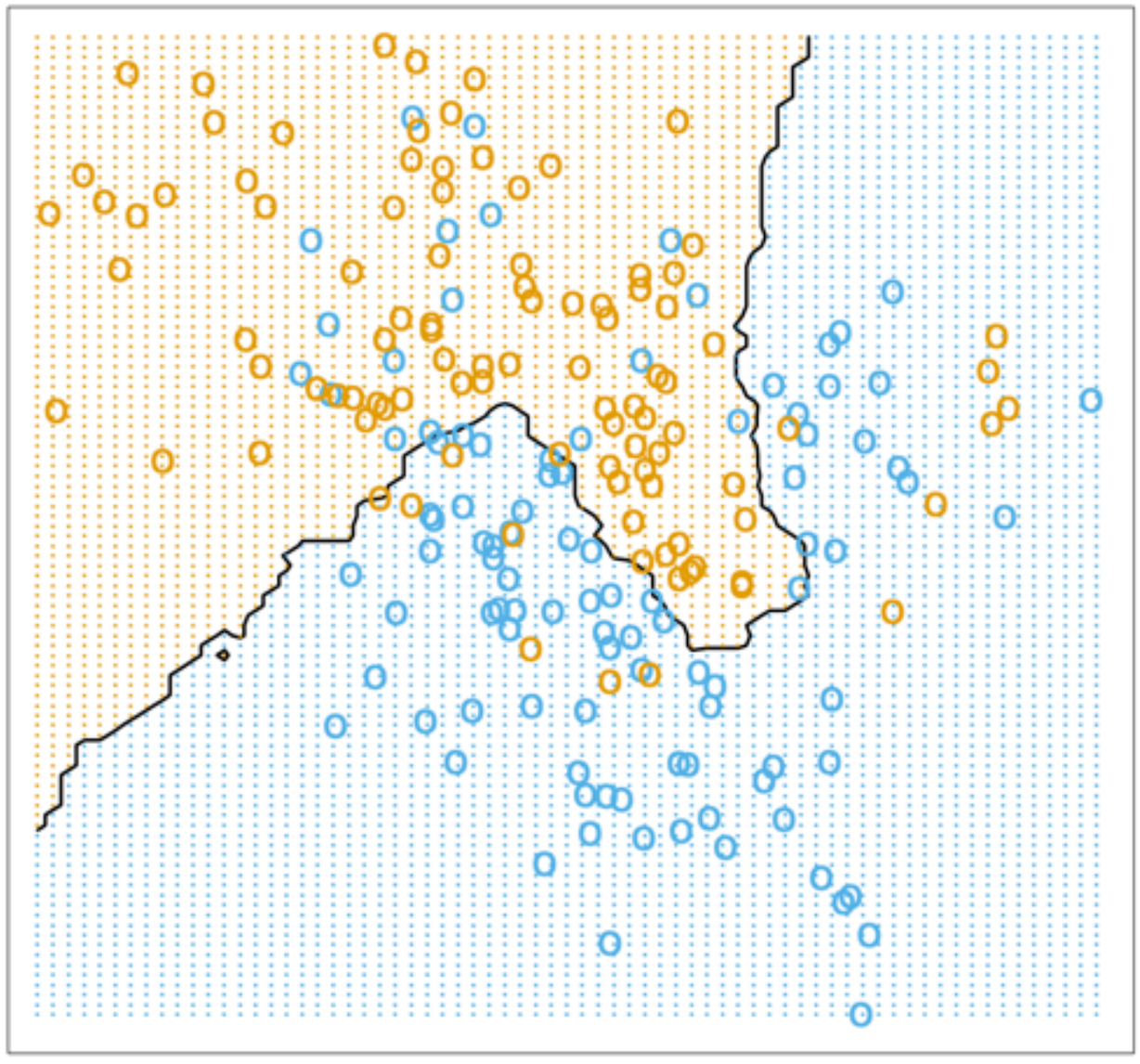

FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

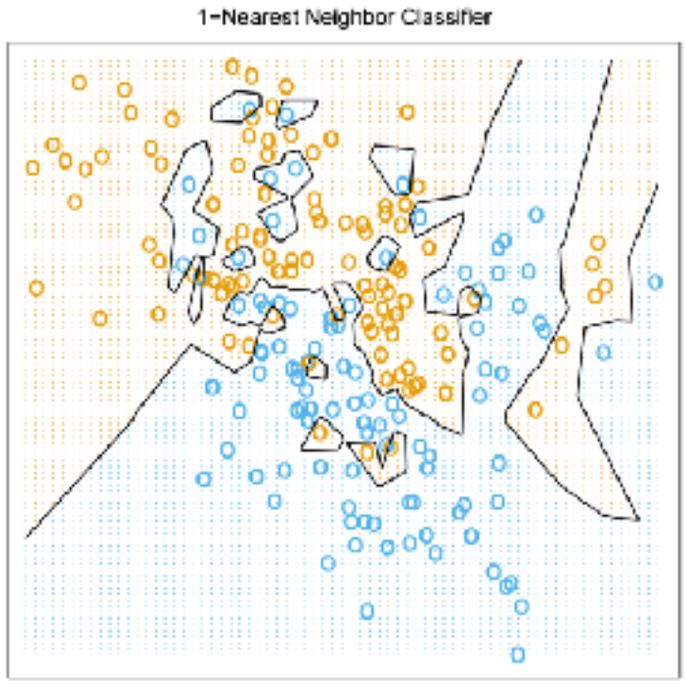

FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then predicted by 1-nearest-neighbor classification.*

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.

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. To get an idea of why, note that if the neighborhoods were nonoverlapping, there would be N/k neighborhoods and we would fit one parameter (a mean) in each neighborhood.

2.3.3 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 subregion of the decision boundary depends on a handful of input points and their particular positions, and is thus wiggly and unstable—high variance and low bias.

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.

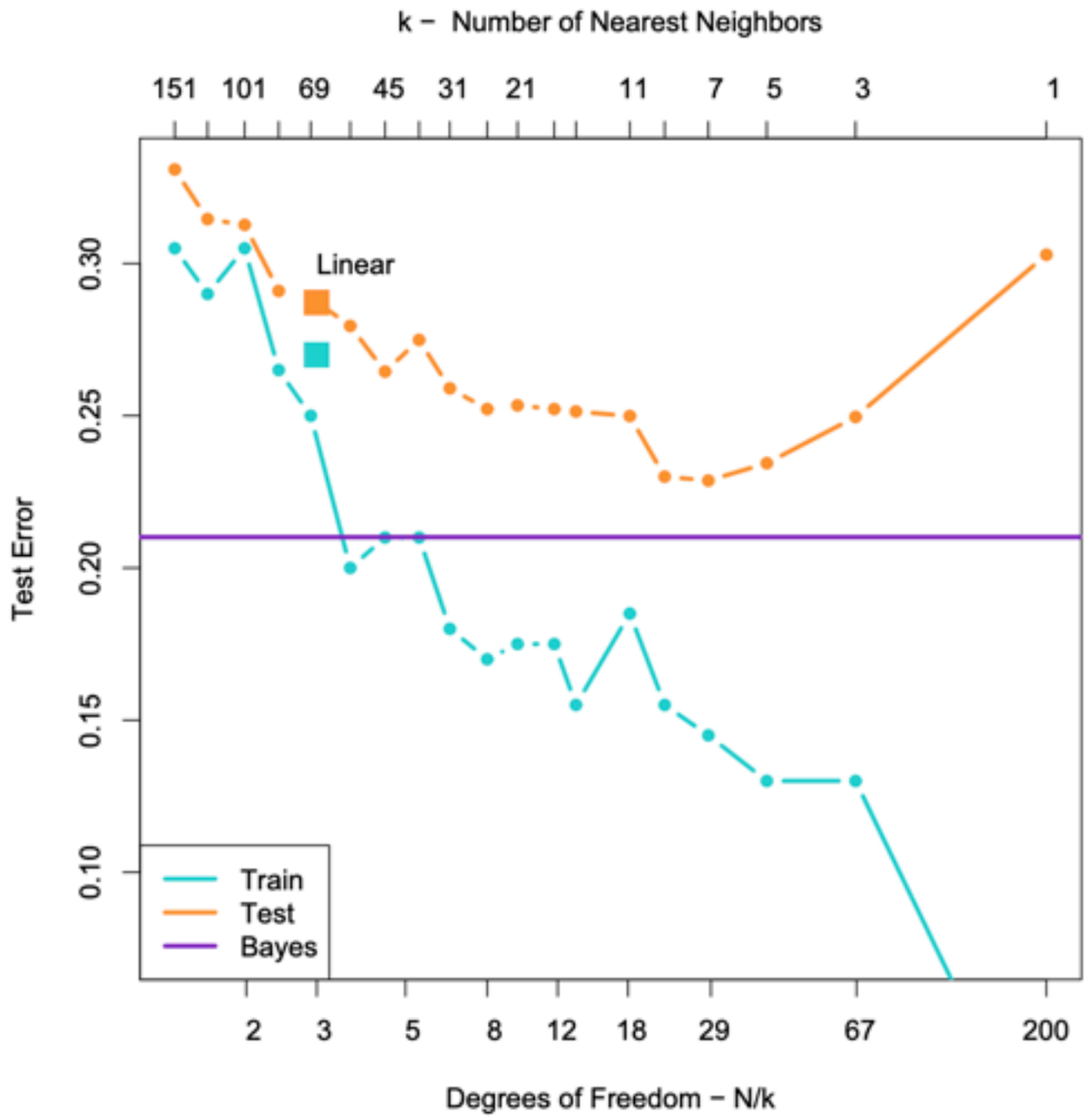

FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* 10,000. *The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear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

Lots of extensions/variations

- Kernel methods use weights that decrease smoothly to zero with distance from the target point, rather than the effective 0/1 weights used by k-nearest neighbors.

- In high-dimensional spaces the distance kernels are modified to emphasize some variable more than others.

- 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.

Outline

1. k-NN (1/2 class)

2. Statistical decision theory (1/2 class)

First, regression

first consider the case of a quantitative output, and place ourselves in the world of random variables and probability spaces. Let $X \in \mathbb{R}^p$ denote a real valued random input vector, and $Y \in \mathbb{R}$ 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*: $L(Y, f(X)) = (Y - f(X))^2$. This leads us to a criterion for choosing f,

$$\begin{aligned} \mathrm{EPE}(f) &= \mathrm{E}(Y - f(X))^2 && (2.9) \\ &= \int [y - f(x)]^2 \Pr(dx, dy), && (2.10) \end{aligned}$$

the expected (squared) prediction error . By conditioning[1] on X, we can write EPE as

$$\mathrm{EPE}(f) = \mathrm{E}_X \mathrm{E}_{Y|X} \left([Y - f(X)]^2 | X \right) \qquad (2.11)$$

The optimal regressor! "Regression function"

$$\mathrm{EPE}(f) = \mathrm{E}_X \mathrm{E}_{Y|X}\left([Y - f(X)]^2 | X\right) \tag{2.11}$$

and we see that it suffices to minimize EPE pointwise:

$$f(x) = \mathrm{argmin}_c \mathrm{E}_{Y|X}\left([Y - c]^2 | X = x\right). \tag{2.12}$$

The solution is

$$f(x) = \mathrm{E}(Y|X = x), \tag{2.13}$$

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.

kNN

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_is with input $x_i = x$. Since there is typically at most one observation at any point x, we settle for

$$\hat{f}(x) = \mathrm{Ave}(y_i | x_i \in N_k(x)), \tag{2.14}$$

where "Ave" denotes average, and $N_k(x)$ is the neighborhood containing the k points in $\mathcal{T}$ 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.

Caveat?

For large training sample size N, the points in the neighborhood are likely to be close to x, and as k gets large the average will get more stable. In fact, under mild regularity conditions on the joint probability distribution $\Pr(X, Y)$, one can show that as $N, k \to \infty$ such that $k/N \to 0$, $\hat{f}(x) \to \mathrm{E}(Y|X = x)$. In light of this, why look further, since it seems we have a universal approximator? 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. There are other problems though, sometimes disastrous. In Section 2.5 we see that 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 us miserably. The convergence above still holds, but the *rate* of convergence decreases as the dimension increases.

Linear regression?

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:

$$f(x) \approx x^T \beta. \tag{2.15}$$

This is a 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:

$$\beta = [\mathrm{E}(XX^T)]^{-1}\mathrm{E}(XY). \tag{2.16}$$

Note we have *not* conditioned on X; rather we have used our knowledge of the functional relationship to *pool* over values of X. The least squares solution (2.6) amounts to replacing the expectation in (2.16) by averages over the training data.

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.

Although the latter seems more palatable, we have already seen that we may pay a price for this flexibility.

Comment: additive models, L1 regression

Now, classification

What do we do when the output is a categorical variable G? The same paradigm works here, except we need a different loss function for penalizing prediction errors. An estimate $\hat{G}$ will assume values in $\mathcal{G}$, the set of possible classes. Our loss function can be represented by a $K \times K$ matrix $\mathbf{L}$, where $K = \text{card}(\mathcal{G})$. $\mathbf{L}$ will be zero on the diagonal and nonnegative elsewhere, where $L(k, \ell)$ is the price paid for classifying an observation belonging to class $\mathcal{G}_k$ as $\mathcal{G}_\ell$. Most often we use the *zero–one* loss function, where all misclassifications are charged a single unit. The expected prediction error is

$$\text{EPE} = \text{E}[L(G, \hat{G}(X))], \tag{2.19}$$

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

$$\text{EPE} = \text{E}_X \sum_{k=1}^{K} L[\mathcal{G}_k, \hat{G}(X)]\Pr(\mathcal{G}_k|X) \tag{2.20}$$

The optimal classifier! "Bayes classifier"

and again it suffices to minimize EPE pointwise:

$$\hat{G}(x) = \text{argmin}_{g \in \mathcal{G}} \sum_{k=1}^{K} L(\mathcal{G}_k, g) \Pr(\mathcal{G}_k | X = x). \tag{2.21}$$

With the 0–1 loss function this simplifies to

$$\hat{G}(x) = \text{argmin}_{g \in \mathcal{G}} [1 - \Pr(g | X = x)] \tag{2.22}$$

or simply

$$\hat{G}(x) = \mathcal{G}_k \text{ if } \Pr(\mathcal{G}_k | X = x) = \max_{g \in \mathcal{G}} \Pr(g | X = x). \tag{2.23}$$

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*.

kNN

Again we see 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.

Classification via regression

Suppose for a two-class problem we had taken the dummy-variable approach and coded G via a binary Y, followed by squared error loss estimation. Then $\hat{f}(X) = \mathrm{E}(Y|X) = \Pr(G = \mathcal{G}_1|X)$ if $\mathcal{G}_1$ corresponded to $Y = 1$. Likewise for a K-class problem, $\mathrm{E}(Y_k|X) = \Pr(G = \mathcal{G}_k|X)$. This shows that our dummy-variable regression procedure, followed by classification to the largest fitted value, is another way of representing the Bayes classifier.

Next class

K - nearest neighbors
Bias-variance tradeoffs