Lecture 3: Measuring performance

Statistical Learning (BST 263)

Jeffrey W. Miller

Department of Biostatistics
Harvard T.H. Chan School of Public Health

Outline

K-nearest neighbors (KNN)

Measuring regression performance

Measuring classification performance

Outline

K-nearest neighbors (KNN)

Measuring regression performance

Measuring classification performance

K-nearest neighbors (KNN)

- Our first statistical learning method!
- We'll use KNN to illustrate the concepts in this lecture.

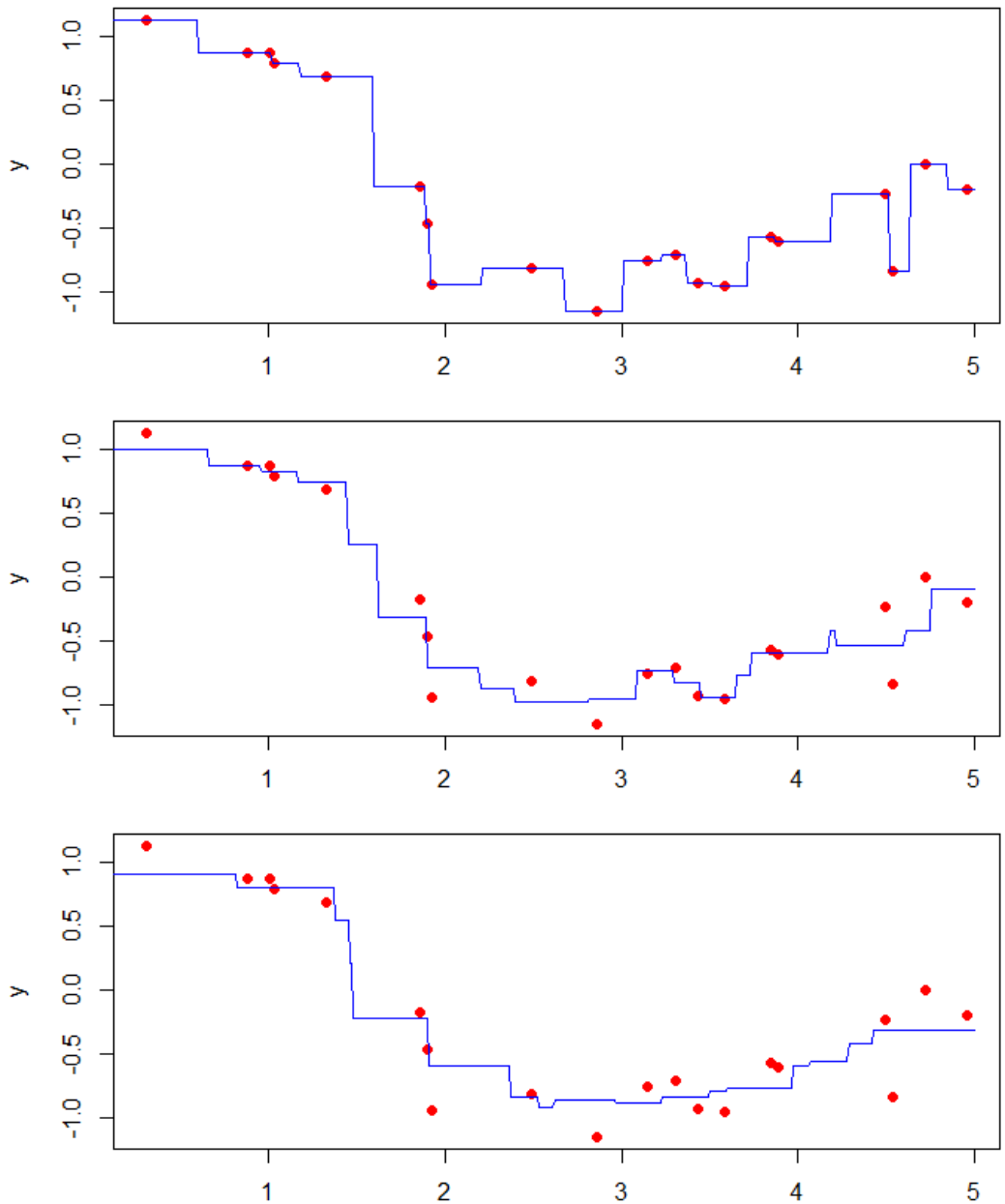
KNN regression algorithm

Input: x^*, training set $(x_1, y_1), \ldots, (x_n, y_n)$, and K.
Output: y^* (predicted outcome value at x^*)

1. Find the K training points x_i that are nearest to x^*.
2. $y^* =$ average of the training y_i values for these K points.

- Typically, $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, and Euclidean distance is used to define how near x^* is to each x_i.
- More general spaces and distances are easy to handle as well.

KNN regression using $K = 1$, $K = 2$, and $K = 4$



K-nearest neighbors (KNN) regression

(R code example)

KNN: Considerations for choosing when to use it

- Supervised learning method
- Can be used for regression or classification.
- Useful for pure prediction problems — not very useful for getting insight/understanding.
- Match to data generating process?
  - Based on assumption that nearby x's have similar y's.
  - In other words, $f(x)$ is assumed to be "smooth".
- Algorithmic method — not likelihood-based.
  - But KNN is dirt simple, so it is still easy to interpret and analyze.
- How flexible?
  - KNN is nonparametric — can fit any function.
  - Can be very flexible.
  - "Smoothness" is controlled by choice of K.
  - Smaller $K \implies$ Less smooth (more flexible)
  - Larger $K \implies$ More smooth (less flexible)

Outline

K-nearest neighbors (KNN)

Measuring regression performance

Measuring classification performance

Mean squared error

- The simplest measure of performance for regression is the mean squared error (MSE).
- The MSE on the training data set is

$$\text{training MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(x_i) - y_i)^2$$

where $\hat{f}(x_i)$ is the predicted outcome for point x_i, and the training set is $((x_1, y_1), \ldots, (x_n, y_n))$.

- The (expected) test MSE at a particular test point x_0 is

$$\text{test MSE} = \mathrm{E}\big((\hat{f}(x_0) - Y_0)^2\big)$$

where Y_0 is a random variable representing the true outcome for x_0 (e.g., $Y_0 = f(x_0) + \varepsilon$ where ε is random noise).

Example 1: Mean squared error

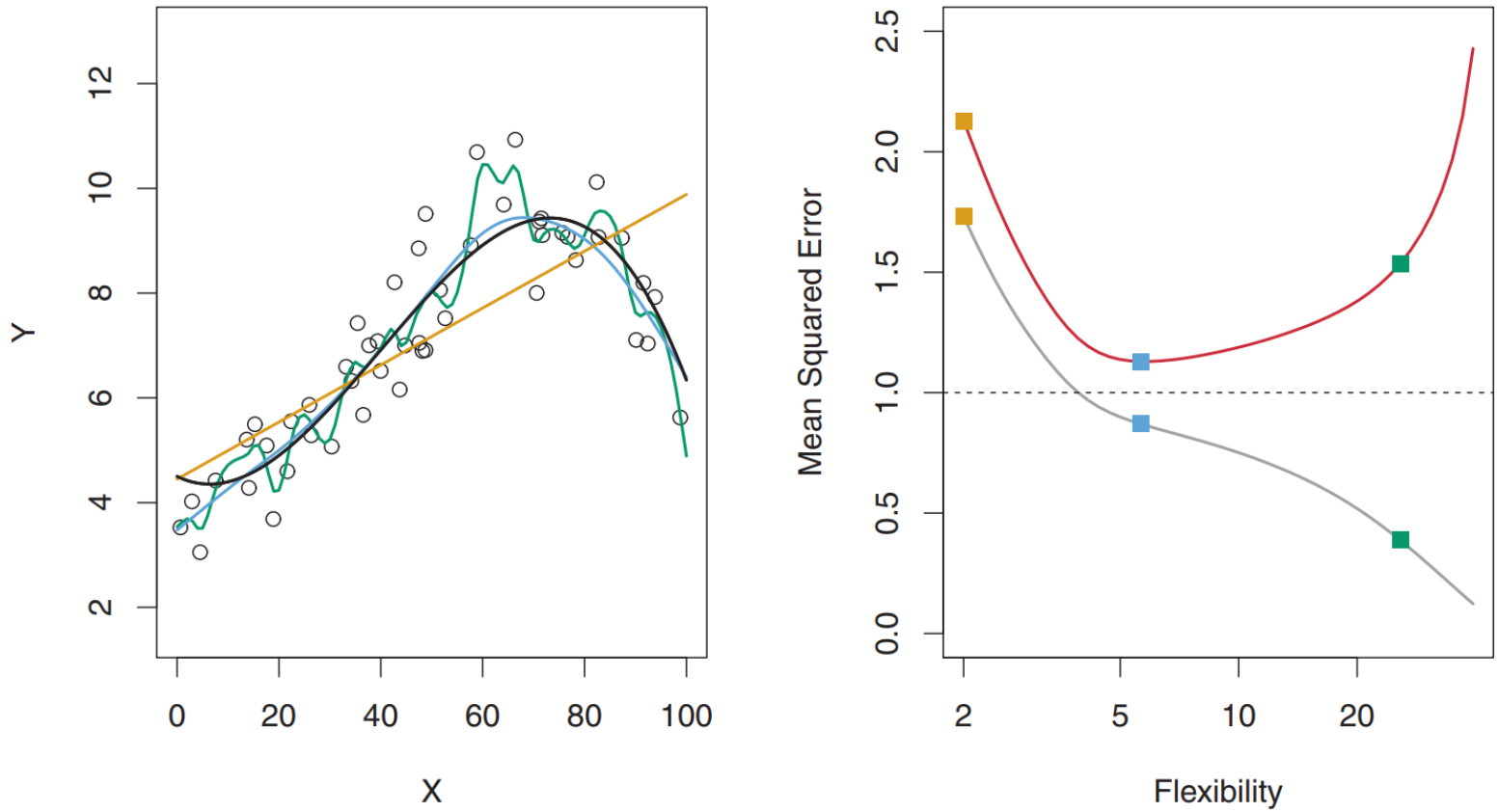


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

The flexibility knob here is the "effective degrees of freedom"; see ISL 7.5.

Example 2: Mean squared error

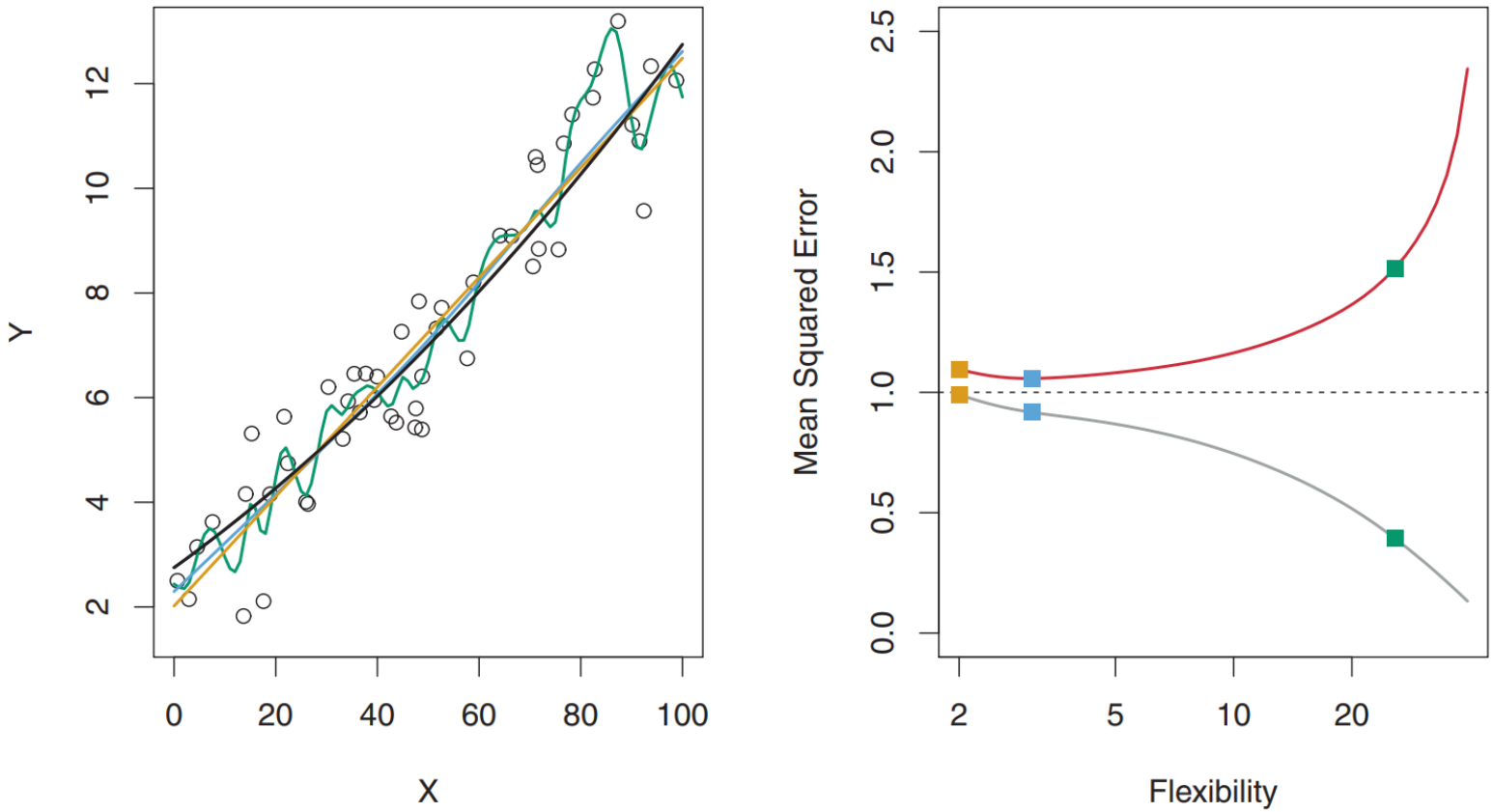


FIGURE 2.10. *Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.*

The flexibility knob here is the "effective degrees of freedom"; see ISL 7.5.

Example 3: Mean squared error

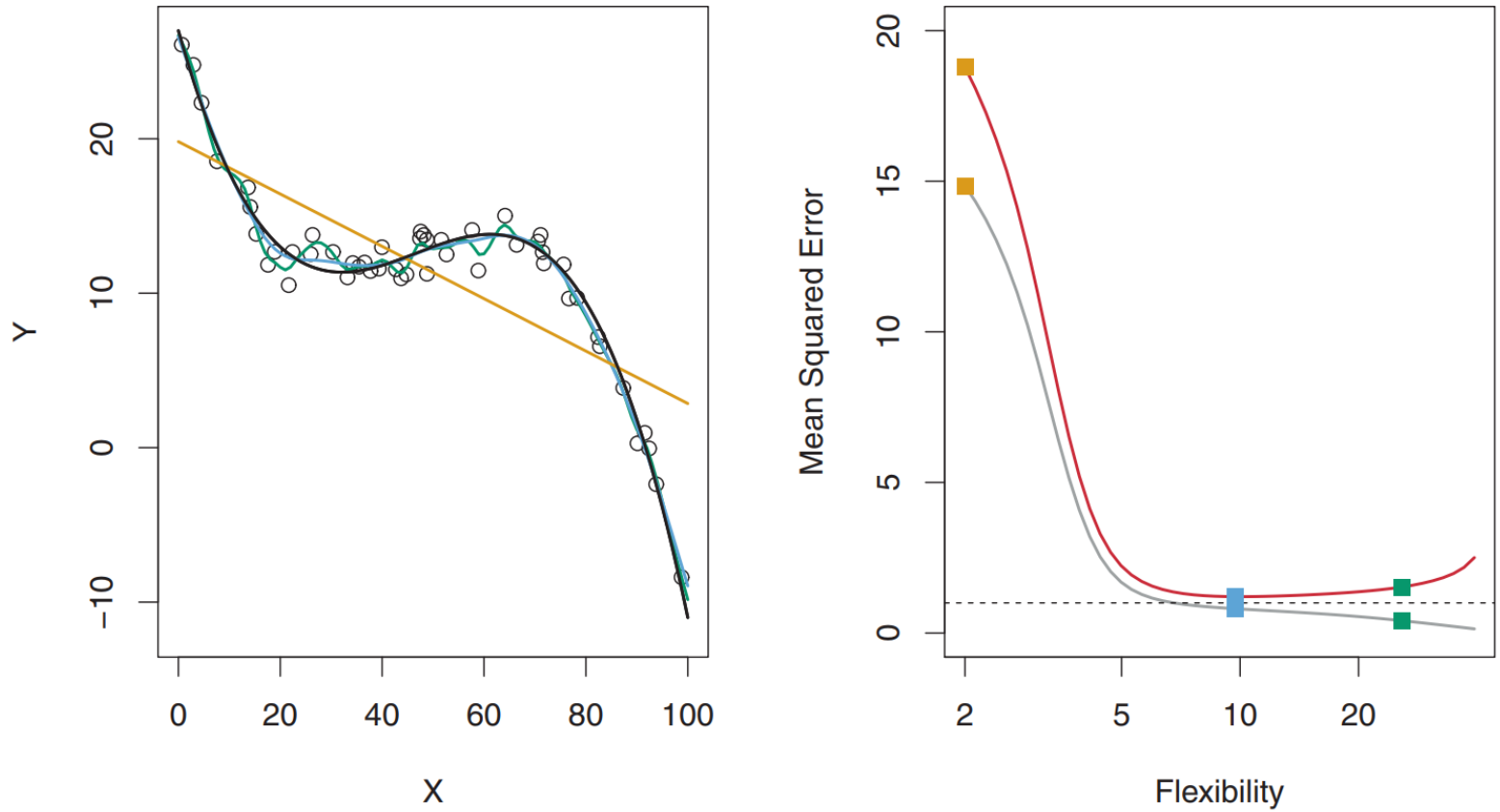


FIGURE 2.11. *Details are as in Figure 2.9, using a different f that is far from linear. In this setting, linear regression provides a very poor fit to the data.*

The flexibility knob here is the "effective degrees of freedom"; see ISL 7.5.

Bias-variance tradeoff for examples 1-3

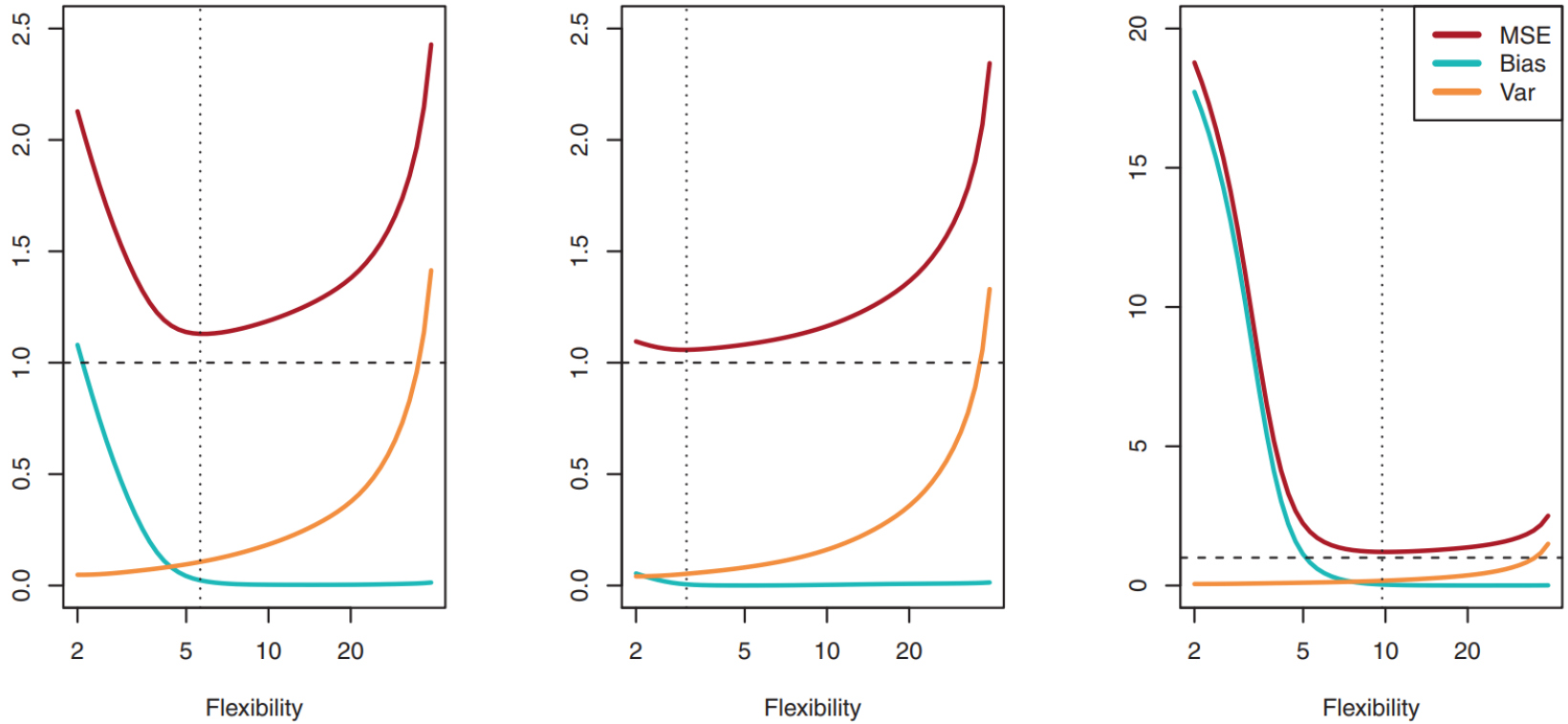


FIGURE 2.12. *Squared bias (blue curve), variance (orange curve), $Var(\epsilon)$ (dashed line), and test MSE (red curve) for the three data sets in Figures 2.9–2.11. The vertical dotted line indicates the flexibility level corresponding to the smallest test MSE.*

The flexibility knob here is the "effective degrees of freedom"; see ISL 7.5.

Bias-variance tradeoff

- A common misperception is that bias is always bad.
- In fact, allowing some bias usually improves performance!

- Why? Because the variance of the predictions can be reduced by allowing some bias.
- This is due to the bias-variance tradeoff.

Bias-variance tradeoff

The test MSE can be decomposed as

$$\text{test MSE} = \text{bias}^2 + \text{variance} + \text{noise}.$$

We will make this more precise in a little bit.

Bias-variance tradeoff

- Roughly, "variance" refers to the variability in $\hat{f}(x)$ due to the randomness in the training dataset.
- Roughly, "bias" refers to the expected difference between $\hat{f}(x)$ and the true $f(x)$.

- Less flexibility leads to:
  - more bias, since we cannot fit the data distribution as closely.
  - less variance, since there are fewer parameters to estimate.
- More flexibility leads to:
  - less bias, since we can fit the data distribution more closely.
  - more variance, since there are more parameters to estimate.

- Consequently, there is a tradeoff, and test MSE is minimized by setting the flexibility equal to some critical point.

K-nearest neighbors (KNN) regression

(R code example to illustrate bias-variance tradeoff)

Schematic of the bias-variance tradeoff

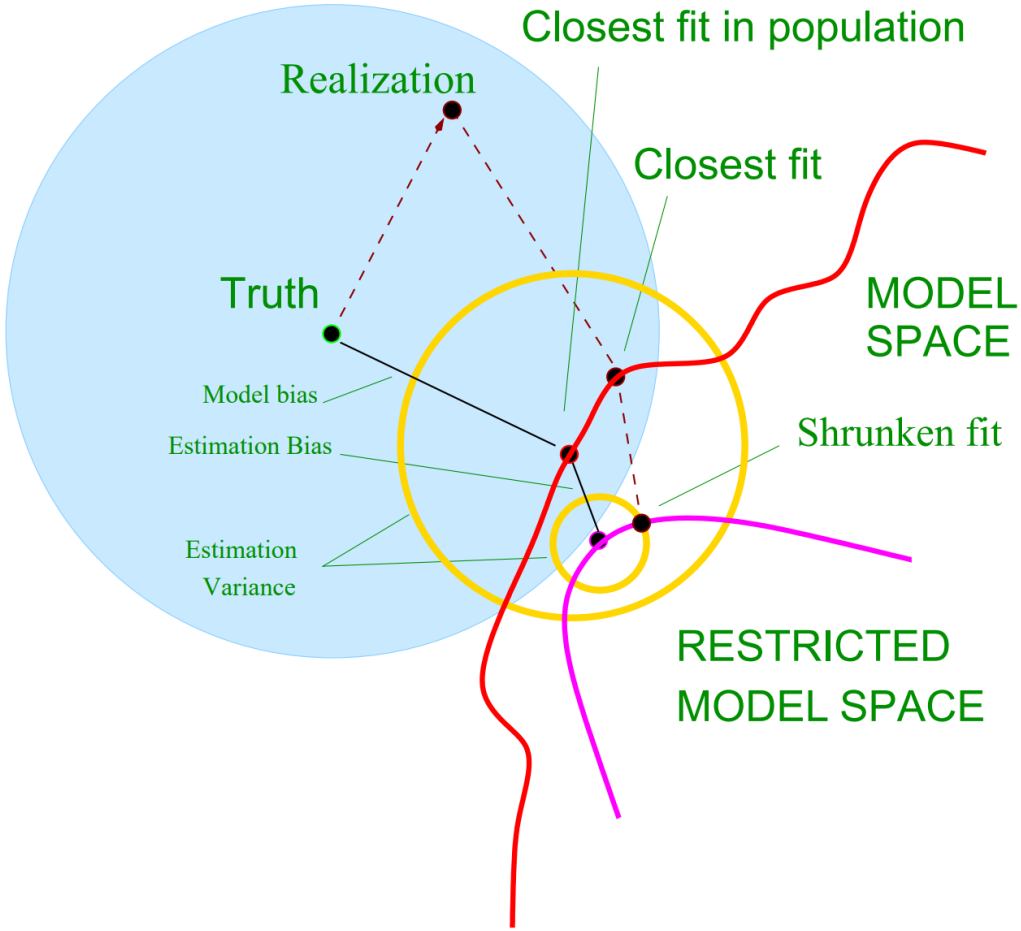


FIGURE 7.2. *Schematic of the behavior of bias and variance. The model space is the set of all possible predictions from the model, with the "closest fit" labeled with a black dot. The model bias from the truth is shown, along with the variance, indicated by the large yellow circle centered at the black dot labeled "closest fit in population." A shrunken or regularized fit is also shown, having additional estimation bias, but smaller prediction error due to its decreased variance.*

(figure from Friedman et al. (2009).)

Precise statement of the bias-variance tradeoff

- Suppose the training data set is $\mathcal{D} = ((x_1, Y_1), \ldots, (x_n, Y_n))$.
  (The x_i's are fixed, whereas the Y_i's are random variables.)
- Suppose $\hat{f}_{\mathcal{D}}(x)$ is the prediction function generated by some algorithm using $\mathcal{D}$.
- Suppose x_0 is a fixed test point, and we want to predict the true unobserved Y_0.
- We then predict $\hat{Y}_0 = \hat{f}_{\mathcal{D}}(x_0)$.
  ($\hat{Y}_0$ is a random variable, since the Y_i's are random variables.)

Bias-variance tradeoff

If $Y_0 = f(x_0) + \varepsilon$, where $\varepsilon \perp\!\!\!\perp \mathcal{D}$ and $\mathrm{E}(\varepsilon) = 0$, then

$$\mathrm{E}\big((\hat{Y}_0 - Y_0)^2\big) = \big(\mathrm{E}(\hat{Y}_0) - f(x_0)\big)^2 + \mathrm{Var}(\hat{Y}_0) + \mathrm{Var}(\varepsilon).$$

In other words, test MSE = bias^2 + variance + noise.

Outline

K-nearest neighbors (KNN)

Measuring regression performance

Measuring classification performance

KNN classifier with $K = 3$ (on data with 2-dim x's)

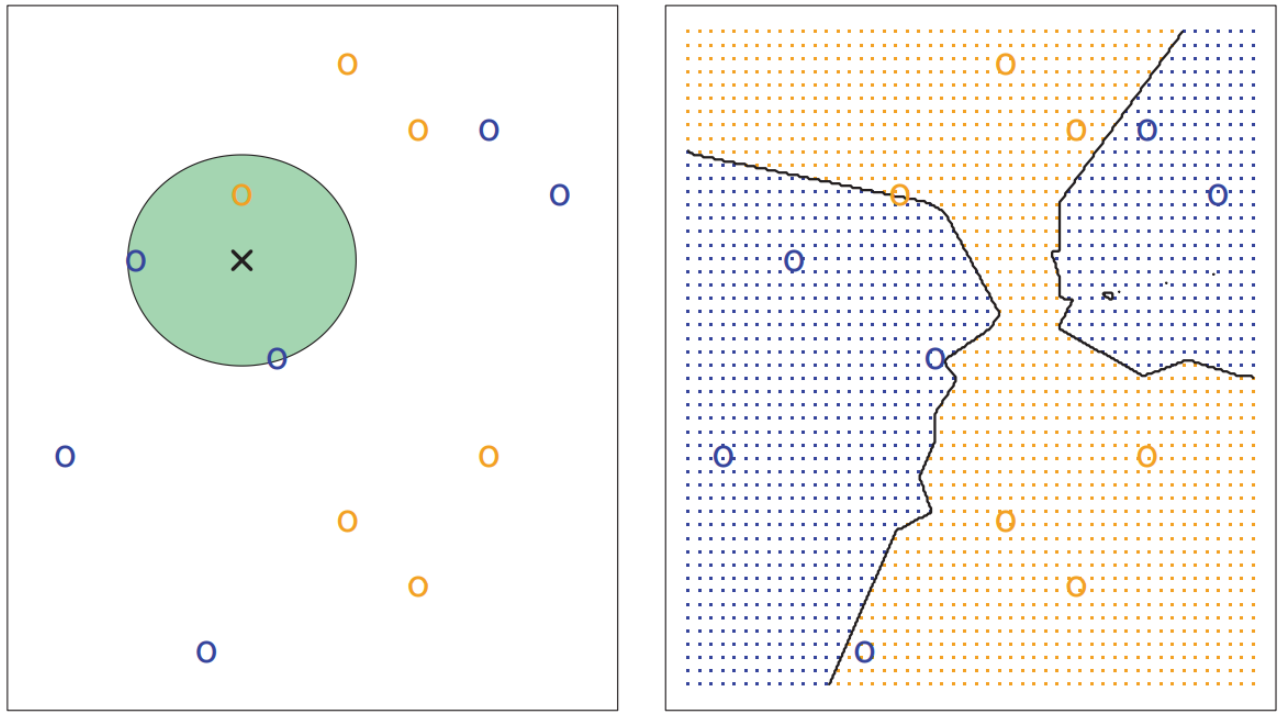


FIGURE 2.14. *The KNN approach, using $K = 3$, is illustrated in a simple situation with six blue observations and six orange observations.* Left: *a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.* Right: *The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.*

2d example: KNN classifier with $K = 10$

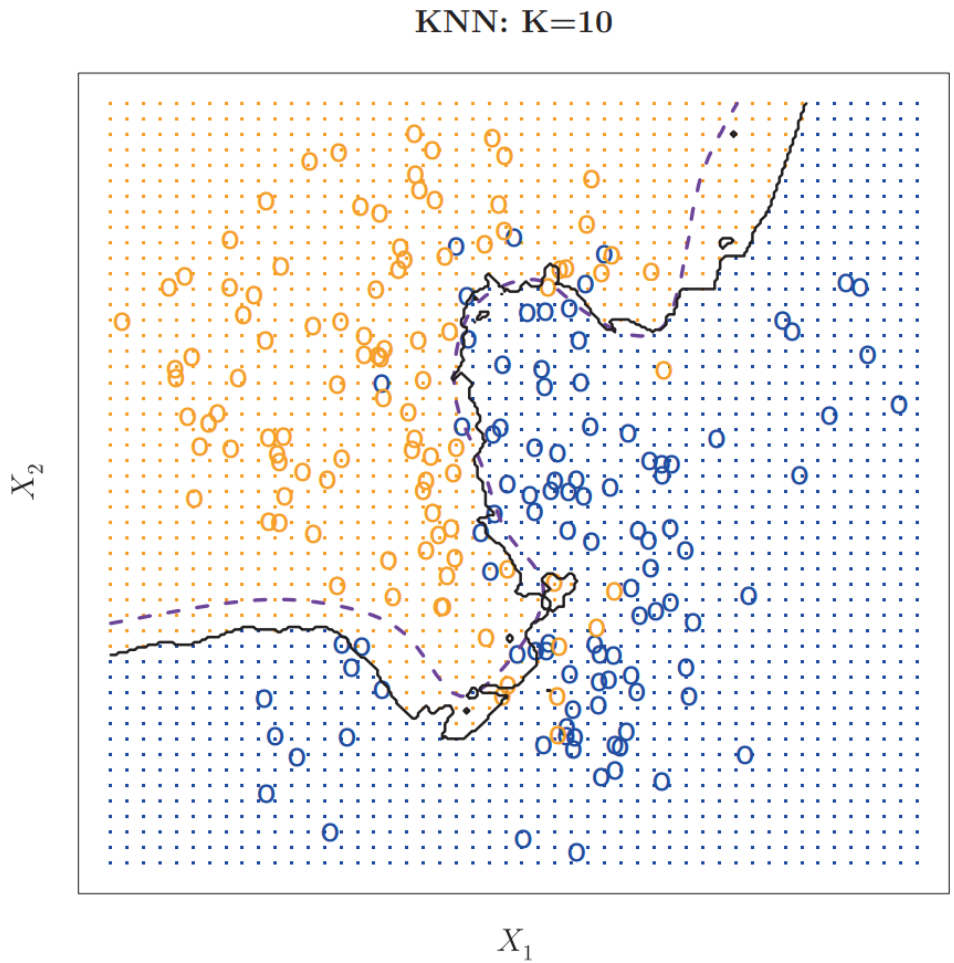


FIGURE 2.15. *The black curve indicates the KNN decision boundary on the data from Figure 2.13, using $K = 10$. The Bayes decision boundary is shown as a purple dashed line. The KNN and Bayes decision boundaries are very similar.*

2d example: KNN classifier with $K = 1$ and $K = 100$

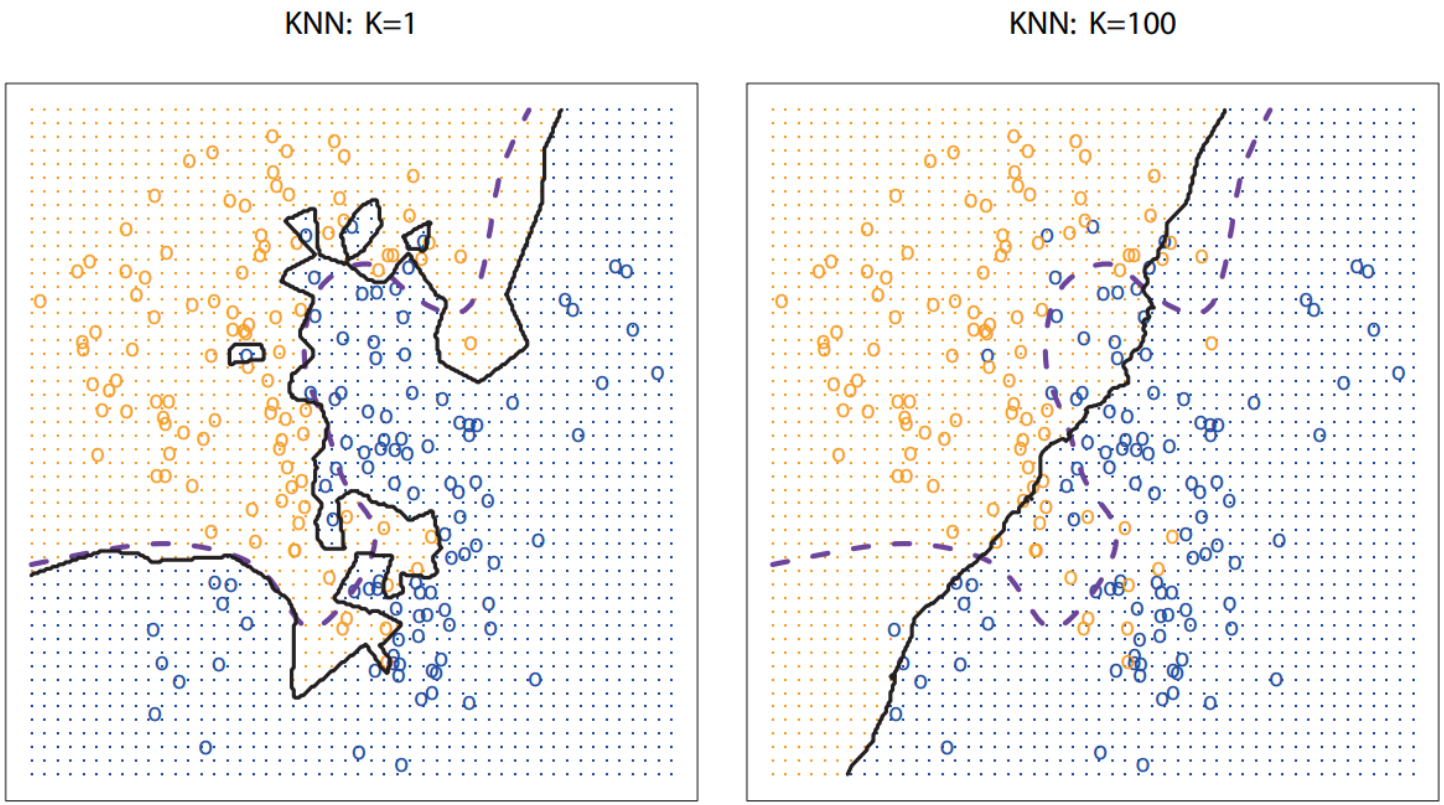


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*

K-nearest neighbors (KNN) classifier algorithm

- In classification, y_i is categorical, e.g., $y_i \in \{1, \ldots, C\}$.
- Usually $x_i \in \mathbb{R}^d$, but other spaces are common as well.

KNN classifier algorithm – class prediction version

Input: x^*, training set $(x_1, y_1), \ldots, (x_n, y_n)$, and K.
Output: y^* (predicted class at x^*)

1. Find the K training points x_i that are nearest to x^*.
2. $y^* =$ most frequently occurring class y_i over these K points.

- In other words, take a majority vote of the classes of the K nearest points.
- Ties can be broken arbitrarily, e.g., randomly if you wish.

KNN classifier – probability version

KNN classifier algorithm – probability version

Input: x^*, training set $(x_1, y_1), \ldots, (x_n, y_n)$, and K.
Output: $\hat{p}_y$ = estimated probability of class y at x^*, for each y.

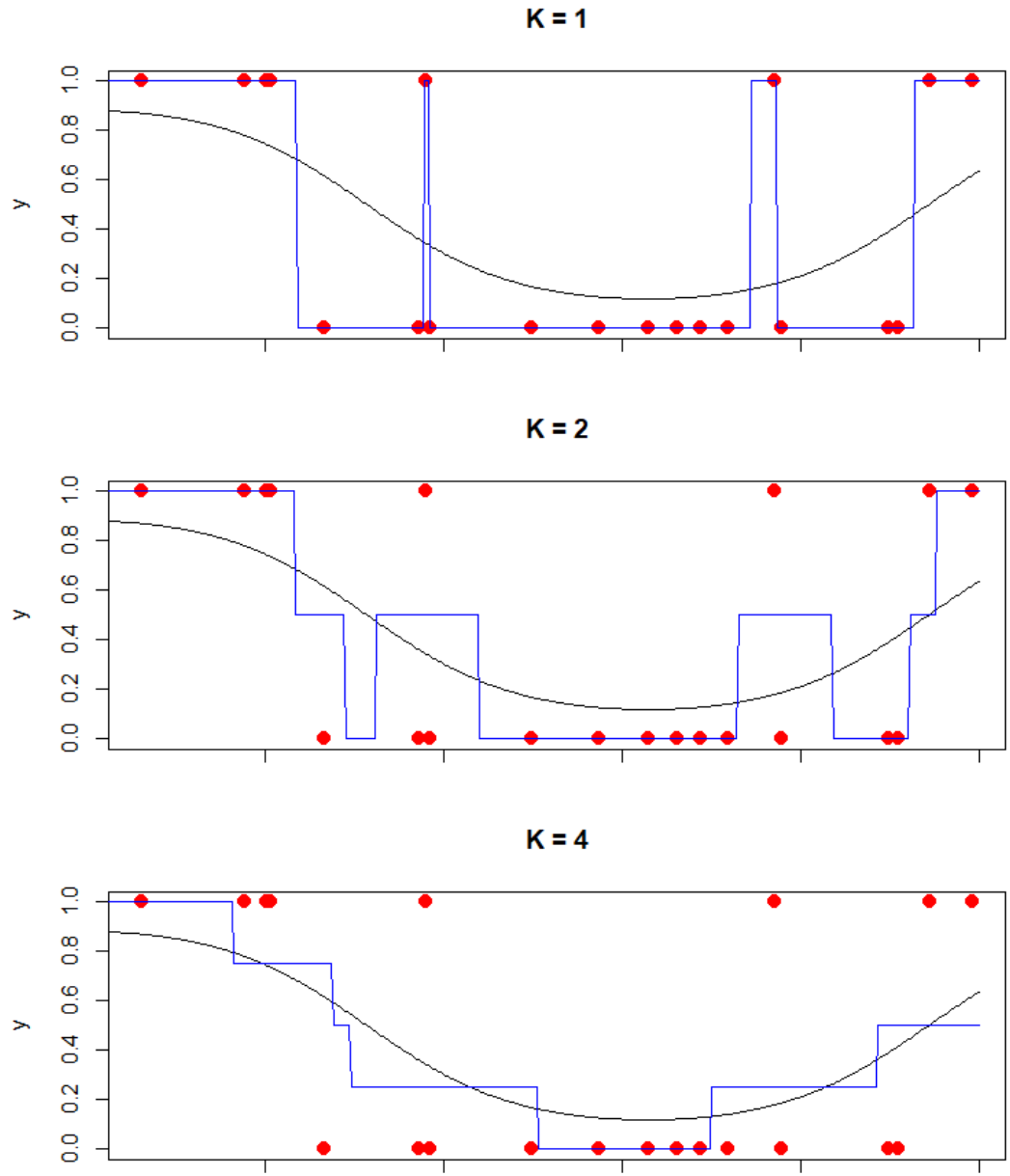
1. Find the K training points x_i that are nearest to x^*.
2. $\hat{p}_y$ = proportion of these K points that have $y_i = y$.

- In other words, for each class y, compute what fraction of the K nearest points have class y.
- The class prediction version can be recovered by setting

$$y^* = \underset{y}{\operatorname{argmax}}\ \hat{p}_y.$$

How would you explain, in words, what is this formula doing?
Notation: $\operatorname{argmax}_x g(x)$ is the x at which $g(x)$ is maximized.

KNN classifier on univariate x's and binary y's



KNN classifier on univariate x's and binary y's

(R code example)

Error rate (a.k.a. misclassification rate)

- Classification is supervised learning with categorical y's.
- Simplest measure of classification performance is error rate.
- *Error rate* = fraction of points that are classified incorrectly.

- The *training error rate* is

$$\text{train error} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(\hat{y}_i \neq y_i)$$

  where $\hat{y}_i$ is the predicted class for point x_i, and the training set is $((x_1, y_1), \ldots, (x_n, y_n))$.

- Notation: $\mathrm{I}(\cdot)$ denotes the *indicator function*: $\mathrm{I}(A) = 1$ if A is true, and $\mathrm{I}(A) = 0$ otherwise.

Error rate (a.k.a. misclassification rate)

- The (expected) test error rate is

$$\text{test error} = \mathrm{E}(\mathrm{I}(\hat{Y}_0 \neq Y_0)) = \mathbb{P}(\hat{Y}_0 \neq Y_0)$$

  where (X_0, Y_0) is a random data point distributed according to the true data generating process, and $\hat{Y}_0 = \hat{f}_{\mathcal{D}}(X_0)$ where $\hat{f}_{\mathcal{D}}$ is constructed from the training data $\mathcal{D}$.

- Randomness of $\hat{Y}_0$ can come from three sources:
  1. randomness of the test point X_0,
  2. randomness of the training x's in $\mathcal{D}$, and/or
  3. randomness of the training y's in $\mathcal{D}$.
- Sometimes we consider test error given one or more of these three sources. Need to be careful to clarify.
- For example, in the bias-variance discussion, we conditioned on $X_0 = x_0$ and on the training x's (but not the training y's).

Example: Error rate of KNN classifier

(R code example to illustrate error rate)

The Bayes optimal classifier

- The *Bayes optimal classifier* (or *Bayes classifier*) is defined as the classifier that has the smallest test error rate:

$$f_{\text{optimal}} = \underset{f}{\operatorname{argmin}}\ \mathbb{P}(f(X_0) \neq Y_0).$$

- Notation: $\operatorname{argmin}_x g(x)$ is the x at which $g(x)$ is minimized.

  (Technically, it is the set of all such minimizing values x.)

- The Bayes optimal classifier is a theoretical construct, not a practical classification method.

- It's what we would ideally use if we knew the distribution of (X_0, Y_0), i.e., if we knew the true data generating process.

The Bayes optimal classifier

- What can we say about the Bayes optimal classifier?

- First, if $f(x_0)$ minimizes $\mathbb{P}(f(x_0) \neq Y_0 \mid X_0 = x_0)$ for each x_0, then f minimizes the test error rate $\mathbb{P}(f(X_0) \neq Y_0)$.

  (This is probably not obvious. . . Can you see why this is true?)

- This is equivalent to $f(x_0)$ being the class y with the highest probability given x_0, i.e., the highest $\mathbb{P}(Y_0 = y \mid X_0 = x_0)$.

  (Can you see why this is true?)

- Therefore,

$$f_{\text{optimal}}(x_0) = \underset{y}{\operatorname{argmax}}\ \mathbb{P}(Y_0 = y \mid X_0 = x_0).$$

  (How would you state this result in words?)

The Bayes optimal classifier

- In practice, we don't know the true distribution of (X_0, Y_0).
- But we do have samples from it, namely, the training set $((x_1, y_1), \ldots, (x_n, y_n))$.
- Thus, a probabilistic model-based approach to classification is:
  1. estimate the true distribution of (X_0, Y_0) from the training set,
  2. use the Bayes optimal classifier for the estimated distribution.

Example: Bayes optimal classifier

(R code example to illustrate)

2d example: Bayes optimal classifier

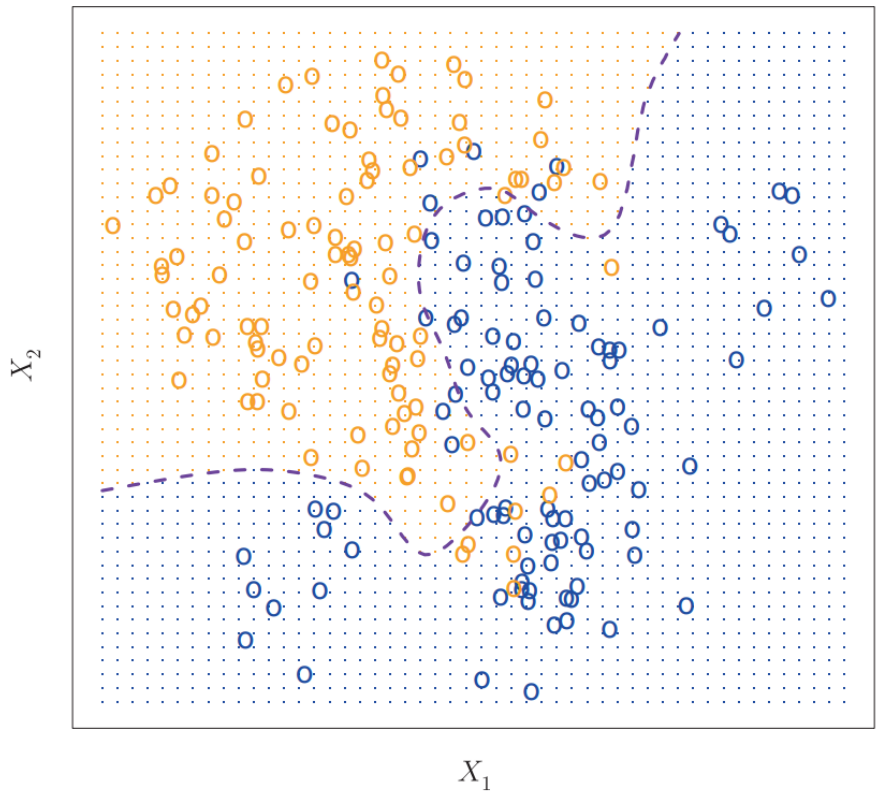


FIGURE 2.13. *A simulated data set consisting of* 100 *observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

Example: Bias-variance tradeoff

(R code example to illustrate)

Bias-variance tradeoff for classification vs regression

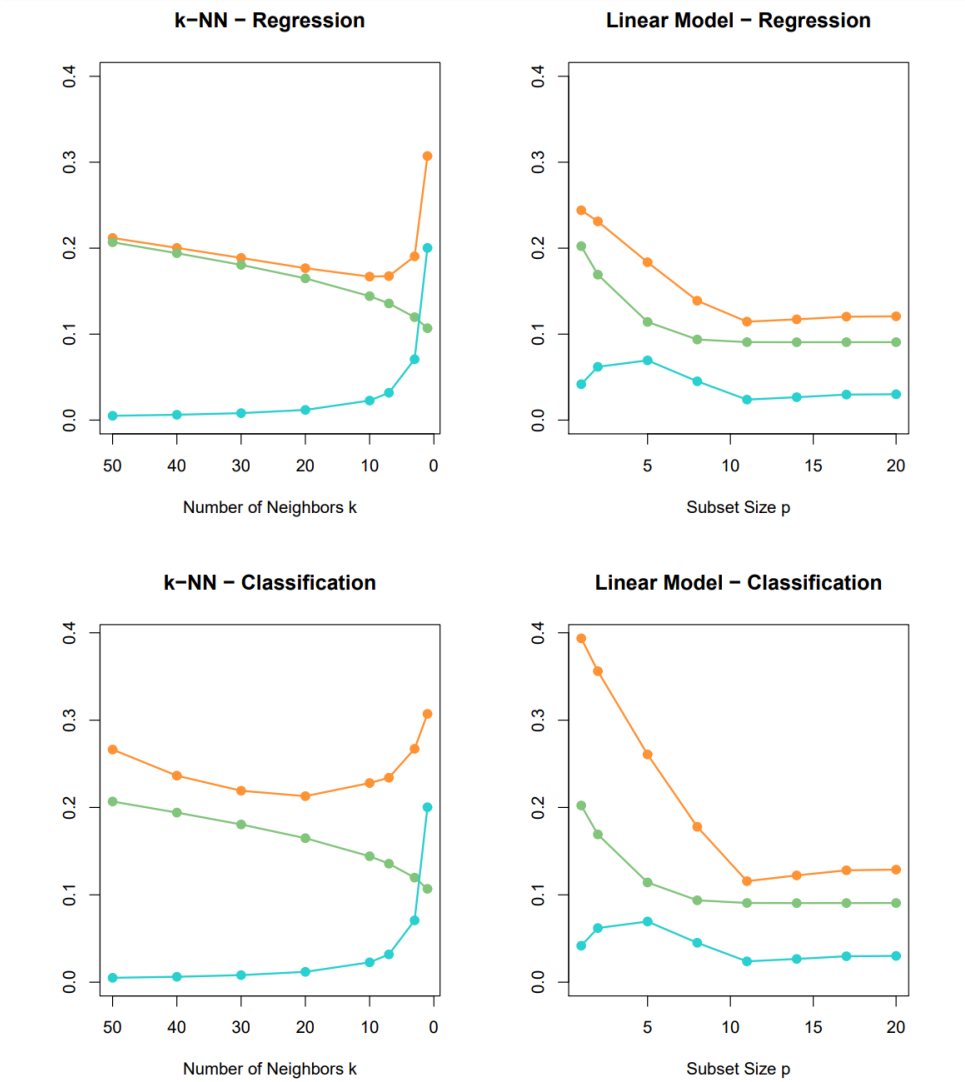


FIGURE 7.3. *Expected prediction error (orange), squared bias (green) and variance (blue) for a simulated example. The top row is regression with squared error loss; the bottom row is classification with 0–1 loss. The models are k-nearest neighbors (left) and best subset regression of size p (right). The variance and bias curves are the same in regression and classification, but the prediction error curve is different.*

References

J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Springer Series in Statistics, New York, 2009.