k-NN, bias-variance
Statistical decision theory

Aadditya 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)

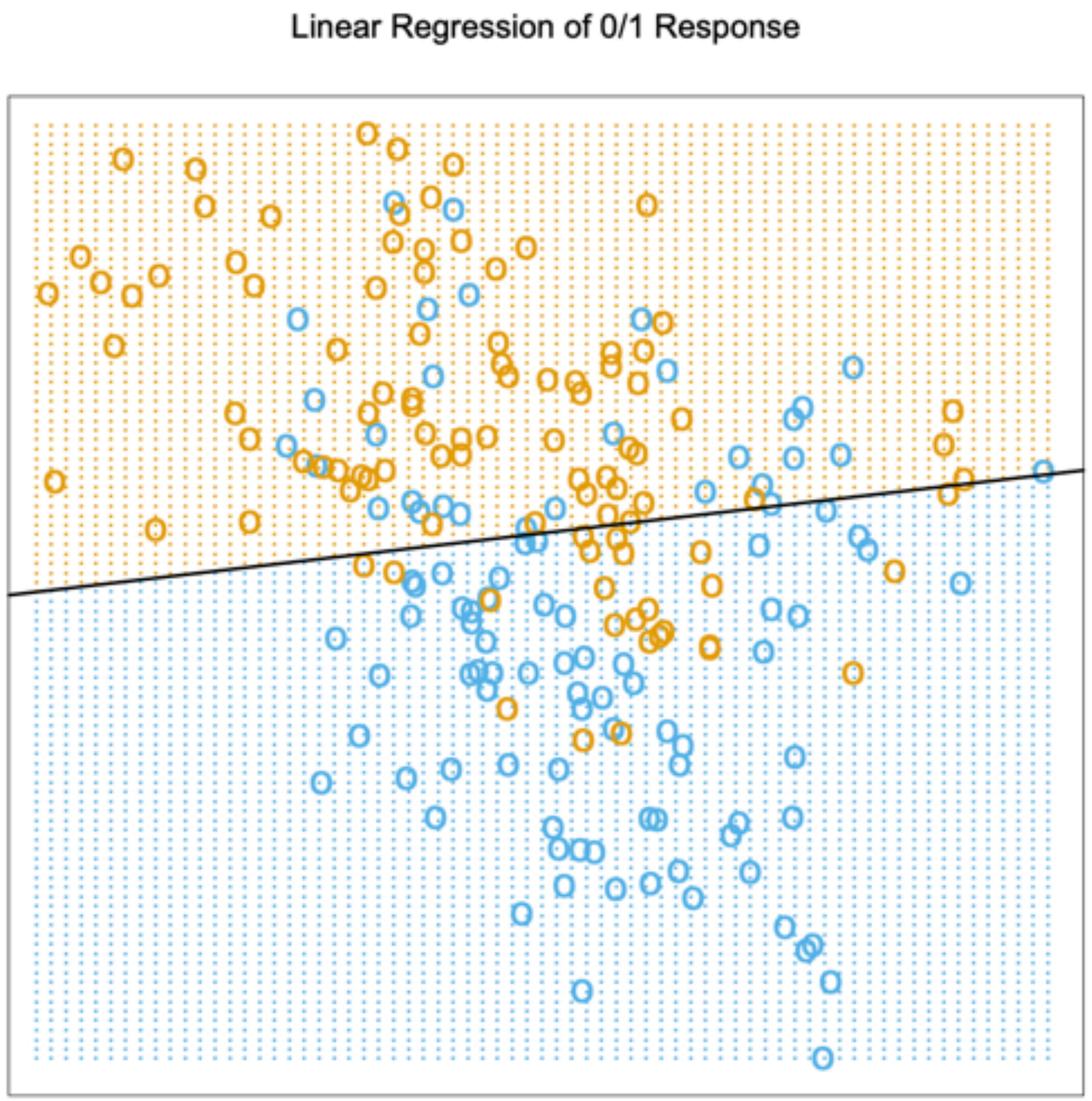
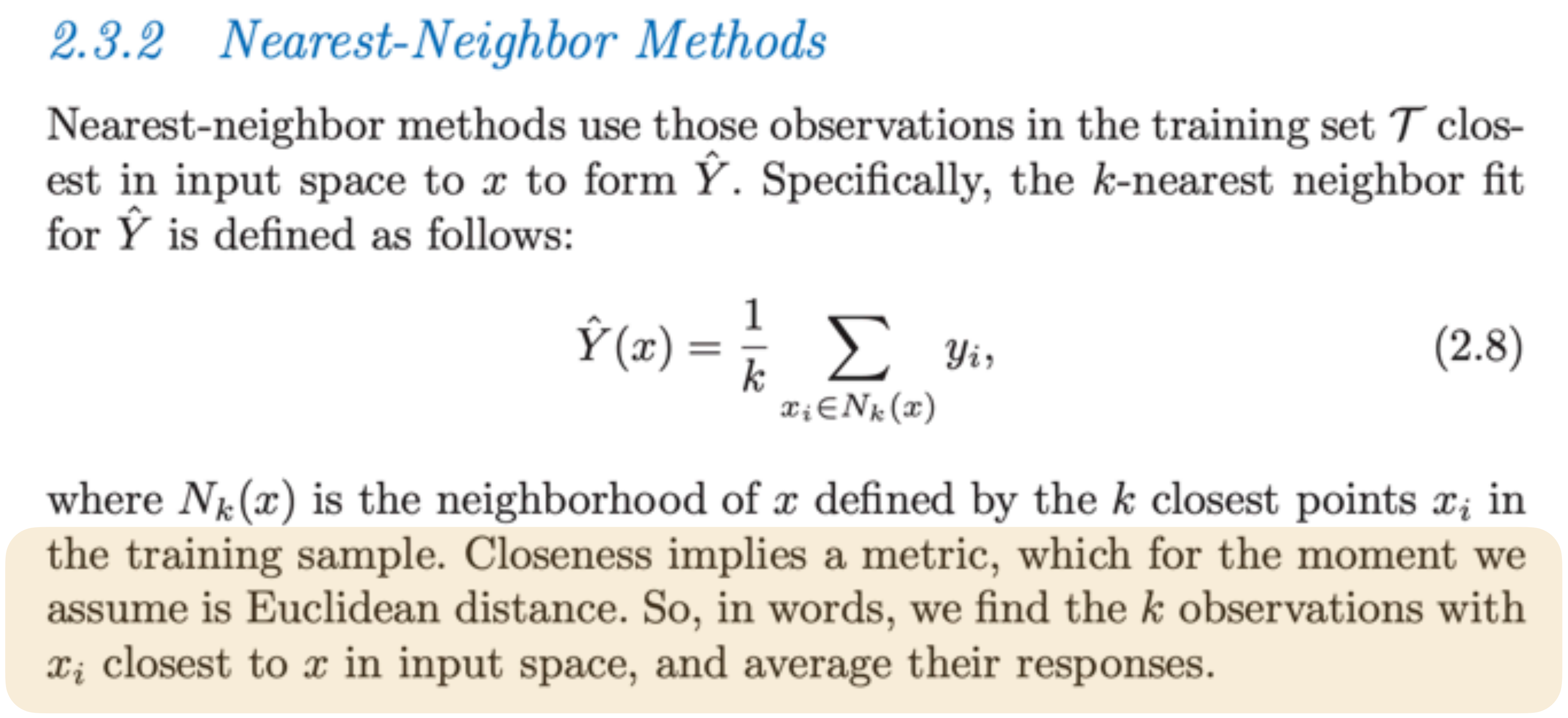


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified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

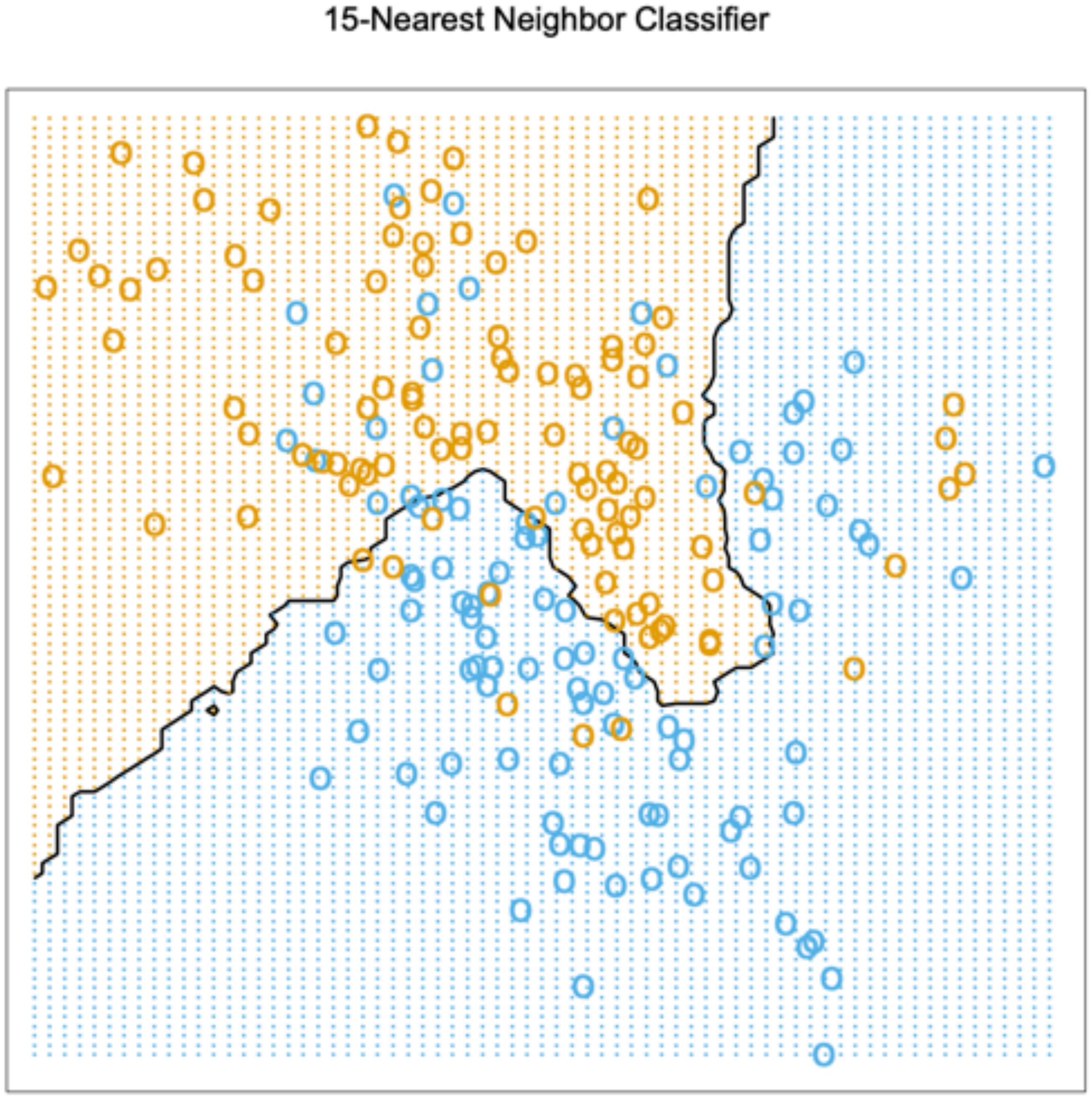


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (`BLUE` $= 0$, `ORANGE` $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

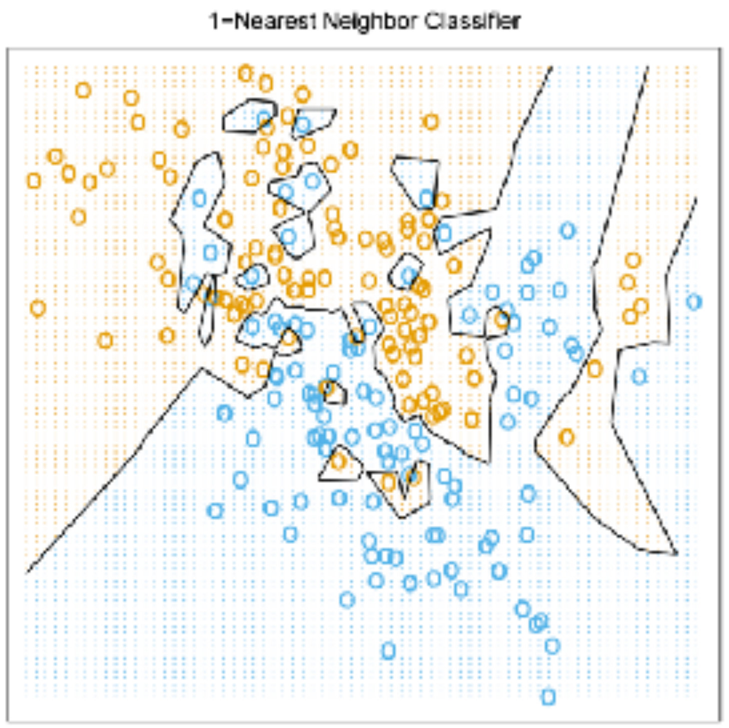


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$), *and then predicted by* 1*-nearest-neighbor classification.*

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.

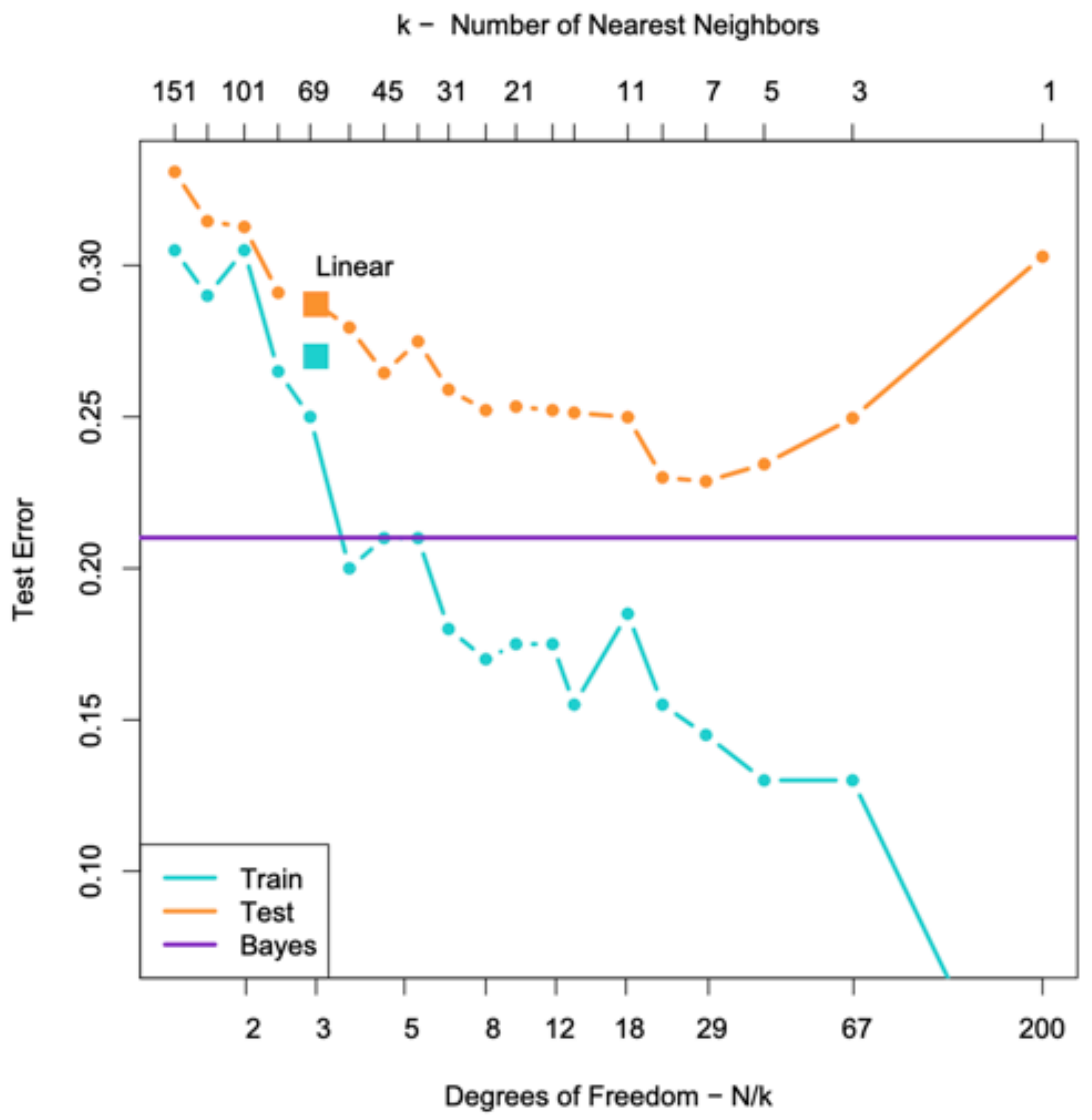
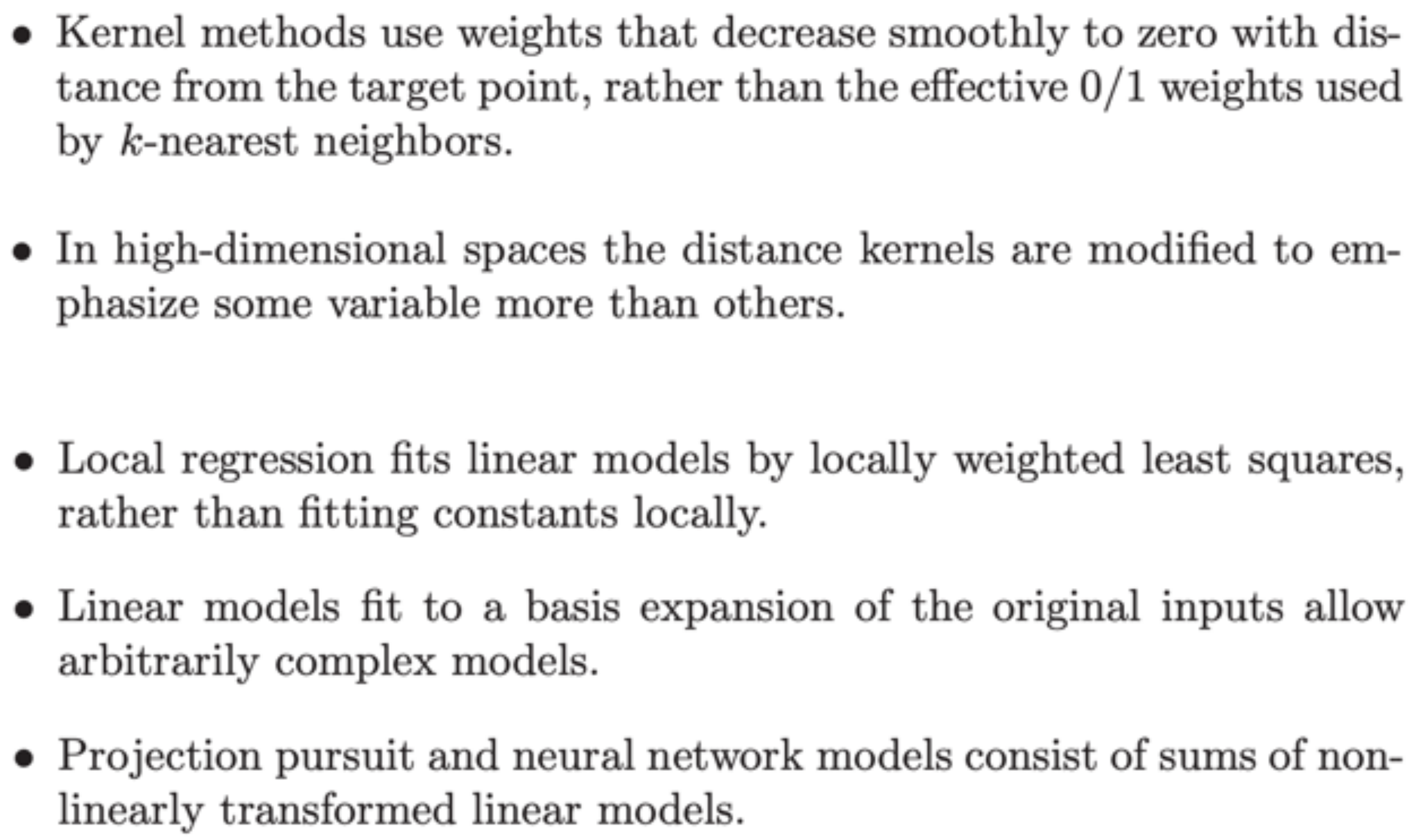


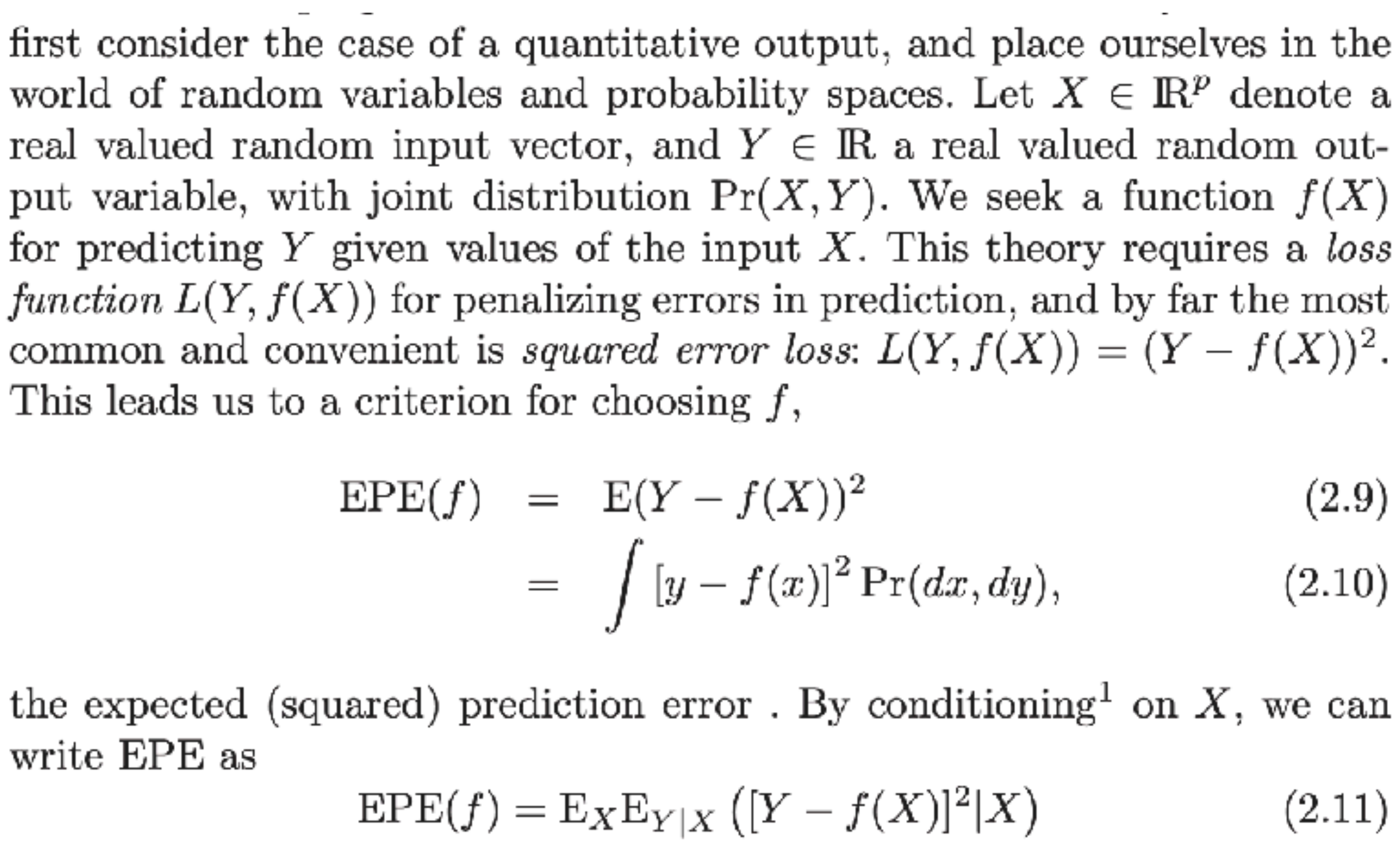
FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* 10,000*. The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear 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.

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,

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

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

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

Outline

1. High dimensional craziness (1/2 class)

2. The optimal classifier, and bias-variance tradeoffs (1/2 class)

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.

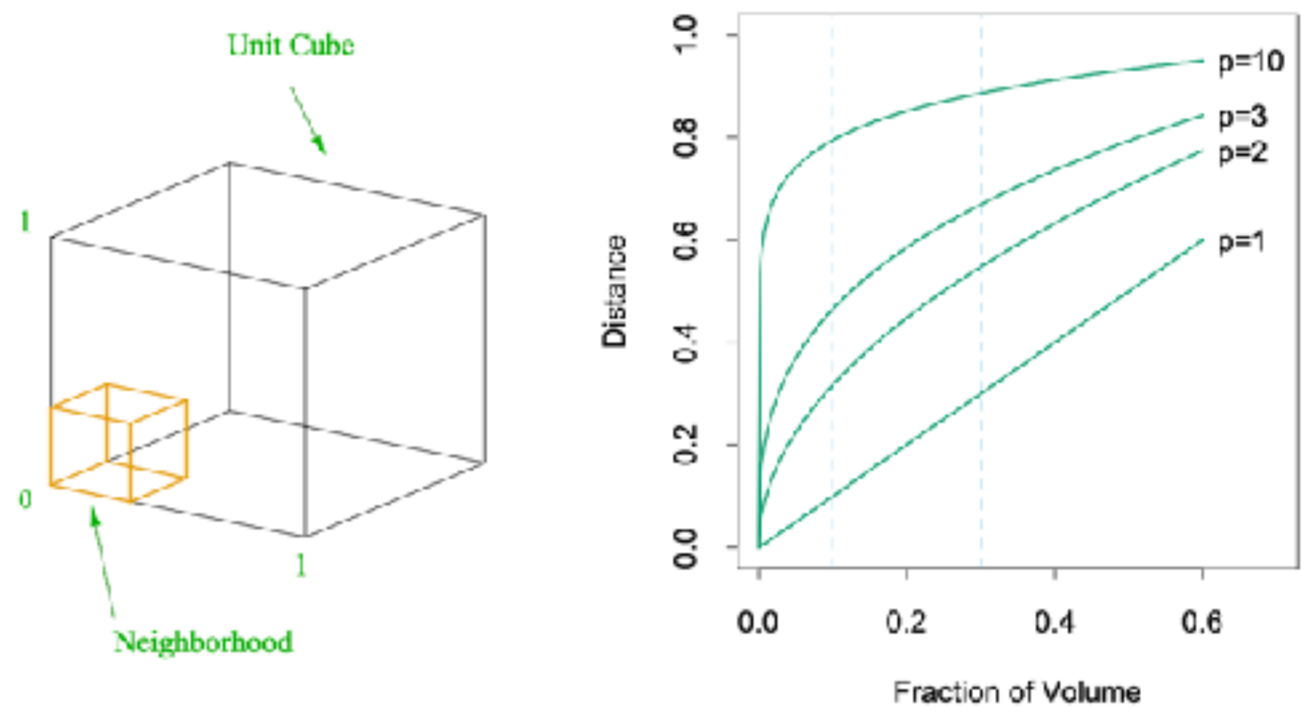


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.*

More craziness in high dimensions

Escaping Spheres

Consider a square with side length 1. At each corner of the square place a circle of radius 1/2, so that the circles cover the edges of the square. Then consider the circle centered at the center of the square that is just large enough to touch the circles at the corners of the square. In two dimensions it's clear that the inner circle is entirely contained in the square.

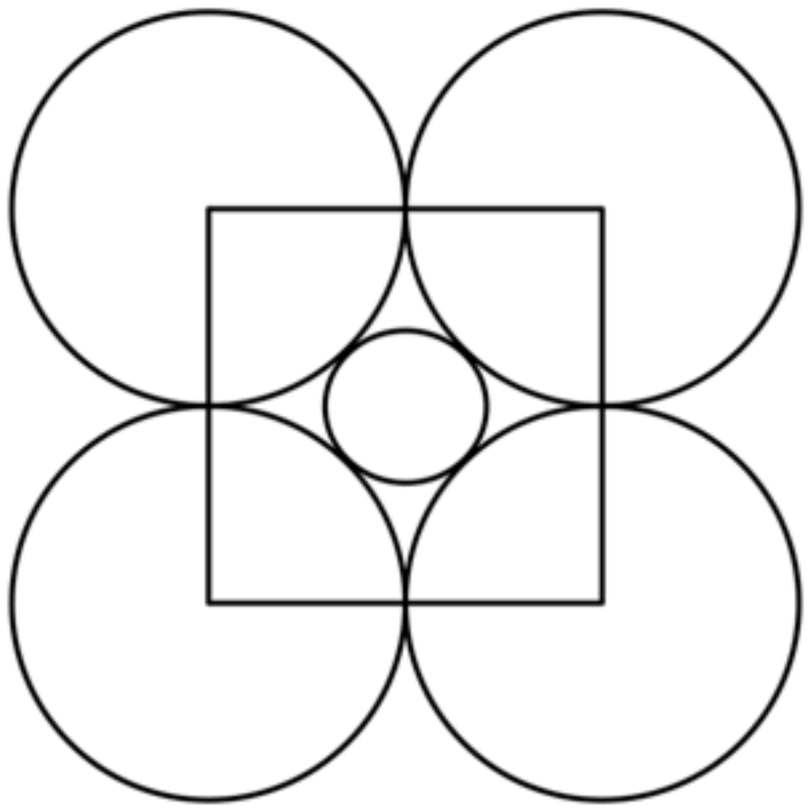


Figure 1: At each corner of the square we place a circle of radius 1/2. The inner circle is just large enough to touch the circles at the corners.

https://marckhoury.github.io/blog/counterintuitive-properties-of-high-dimensional-space/

We can do the same thing in three dimensions. At each corner of the unit cube place a sphere of radius 1/2, again covering the edges of the cube. The sphere centered at the center of the cube and tangent to spheres at the corners of the cube is shown in red in Figure 2. Again we see that, in three dimensions, the inner sphere is entirely contained in the cube.

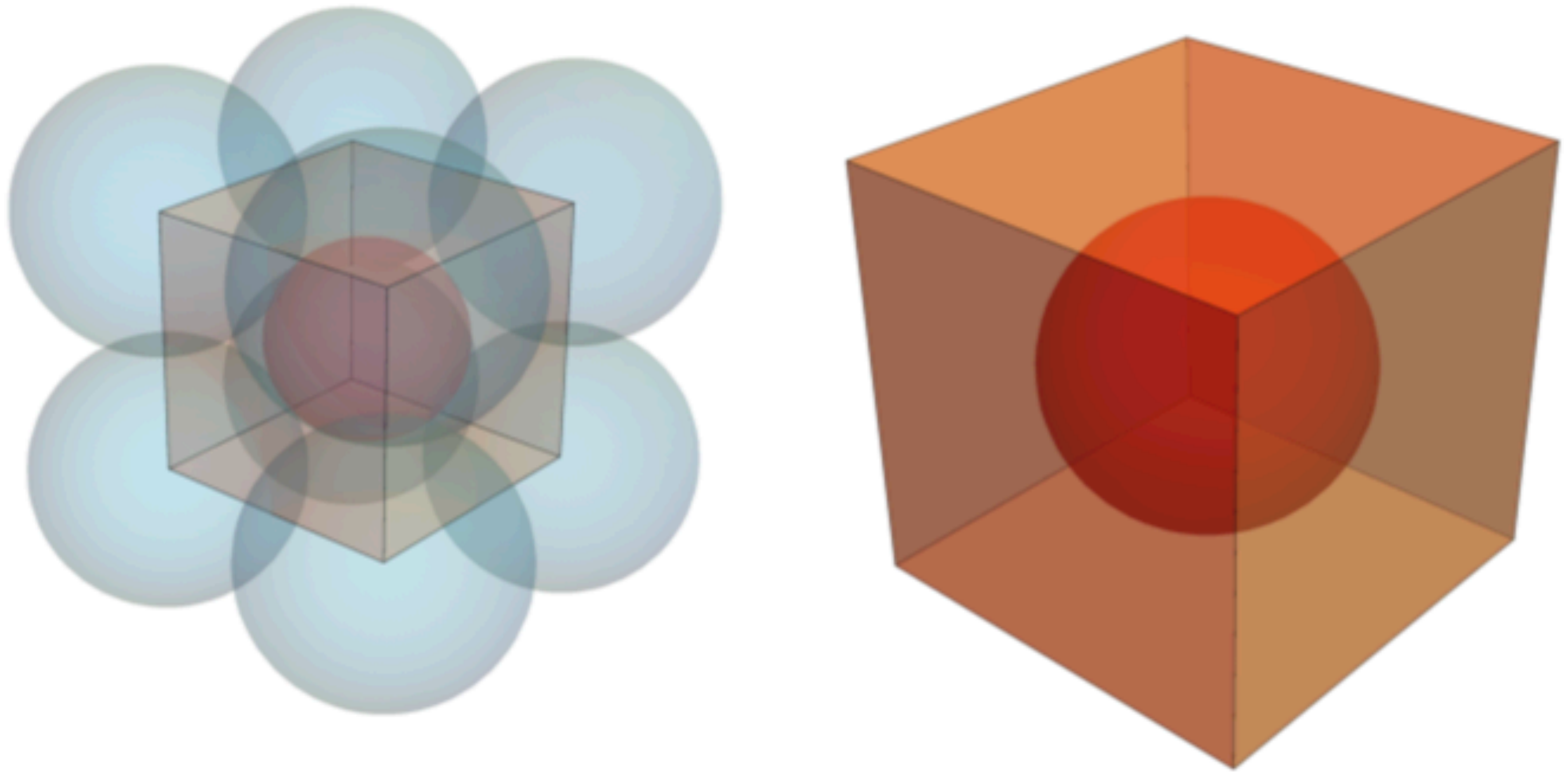


Figure 2: In three dimensions we place a sphere at the each of the eight corners of a cube.

Thus the radius of the inner sphere is \(\sqrt{d}/2 - 1/2\). Notice that the radius of the inner sphere is increasing with the dimension!

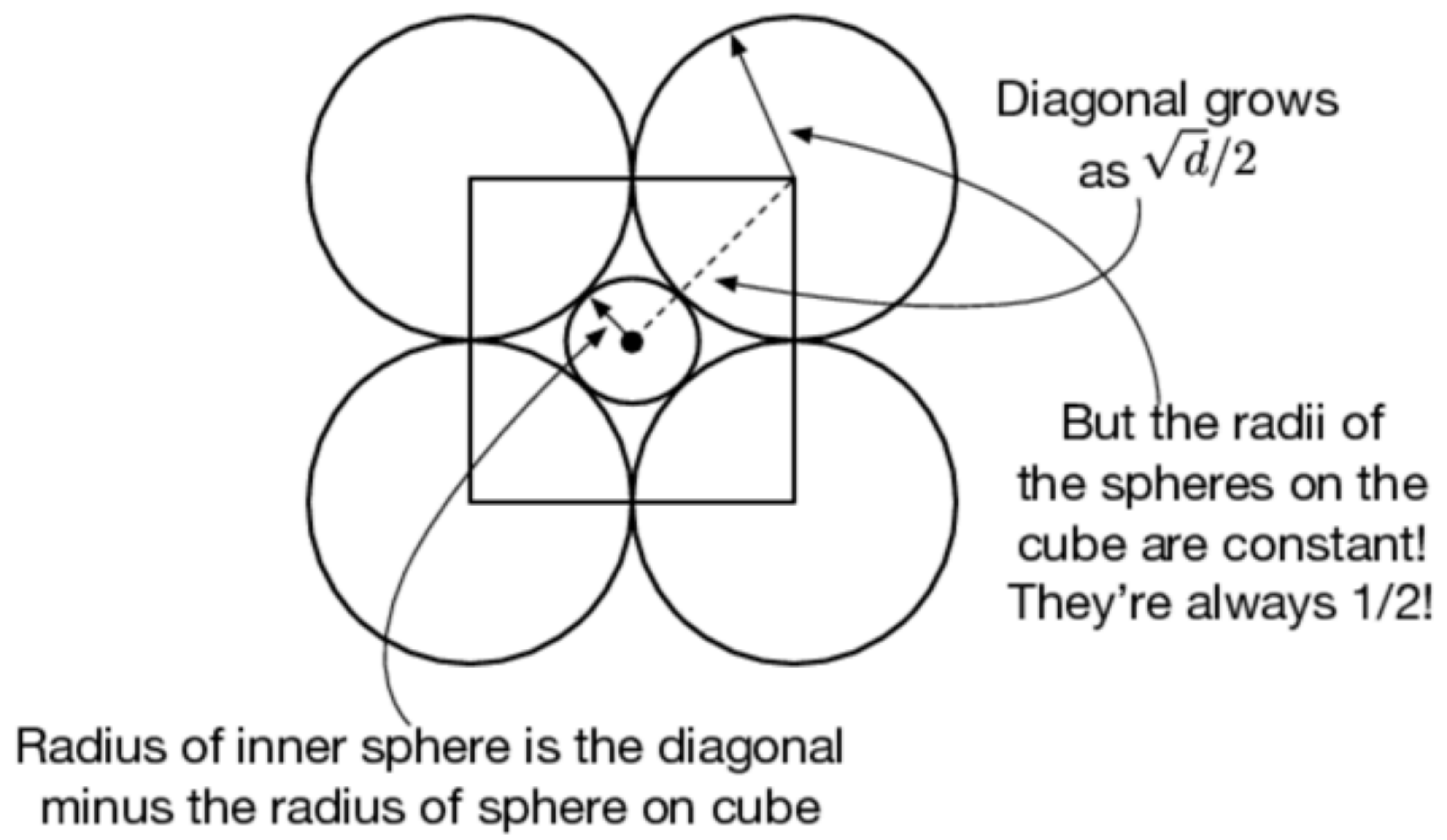

Diagonal grows as $\sqrt{d}/2$
But the radii of the spheres on the cube are constant! They're always 1/2!
Radius of inner sphere is the diagonal minus the radius of sphere on cube

Figure 3: The size of the radius of the inner sphere is growing as the dimension increases because the distance to the corner increases while the radius of the corner sphere remains constant.

In dimensions two and three, the sphere is strictly inside the cube, as we've seen in the figures above. However in four dimensions something very interesting happens. The radius of the inner sphere is exactly 1/2, which is just large enough for the inner sphere to touch the sides of the cube! In five dimensions, the radius of the inner sphere is 0.618034, and the sphere starts poking outside of the cube! By ten dimensions, the radius is 1.08114 and the sphere is poking very far outside of the cube!

High dimensional spheres have no volume

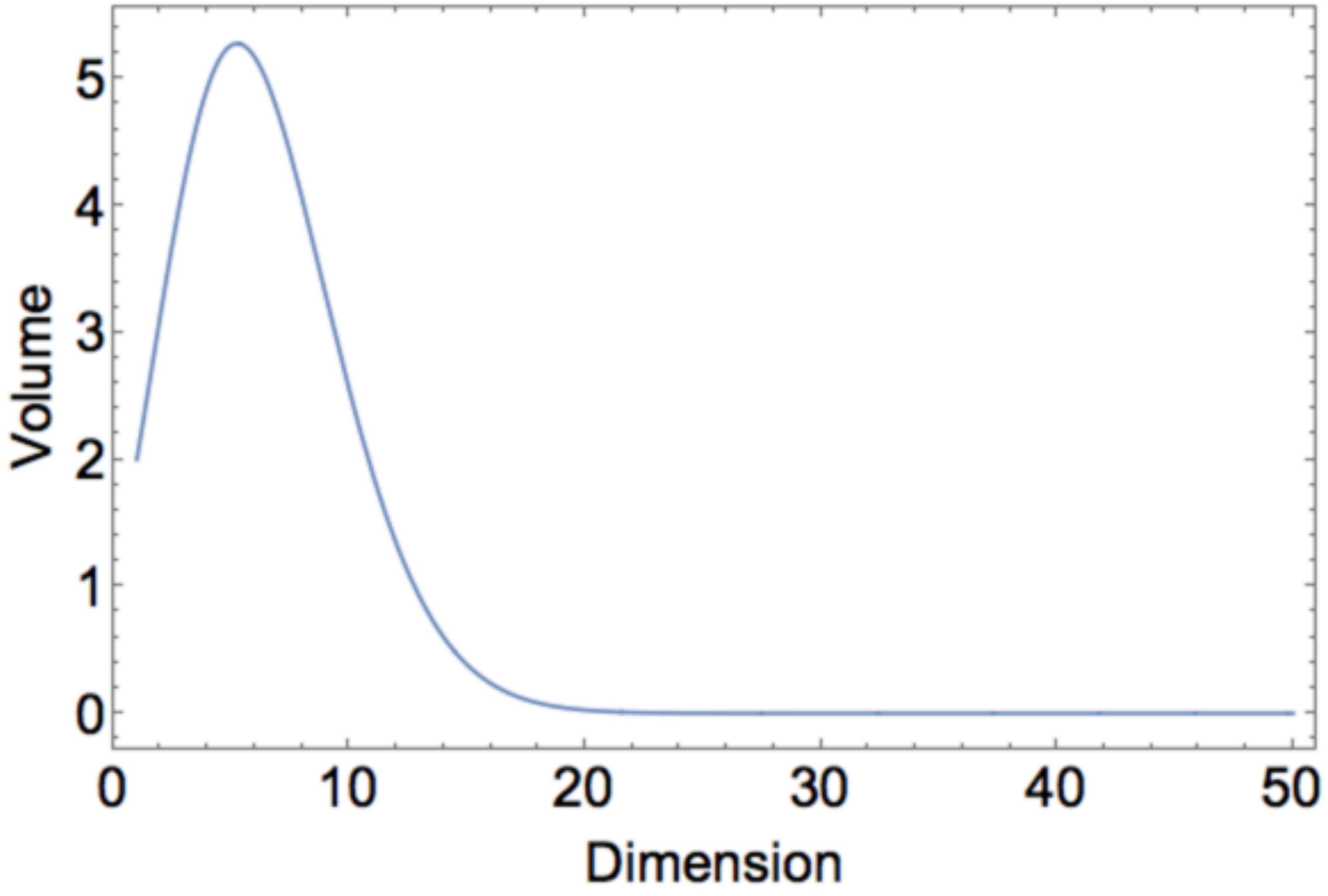


Figure 6: The volume of the unit \(d\)-sphere goes to 0 as \(d\) increases!

The volume of the unit \(d\)-sphere goes to 0 as \(d\) grows! A high dimensional unit sphere encloses almost no volume! The volume increases from dimensions one to five, but begins decreasing rapidly toward 0 after dimension six.

High dimensional spheres have no volume
High dimensional cubes hide mass in corners

More Accurate Pictures

Given the rather unexpected properties of high dimensional cubes and spheres, I hope that you'll agree that the following are somewhat more accurate pictorial representations.

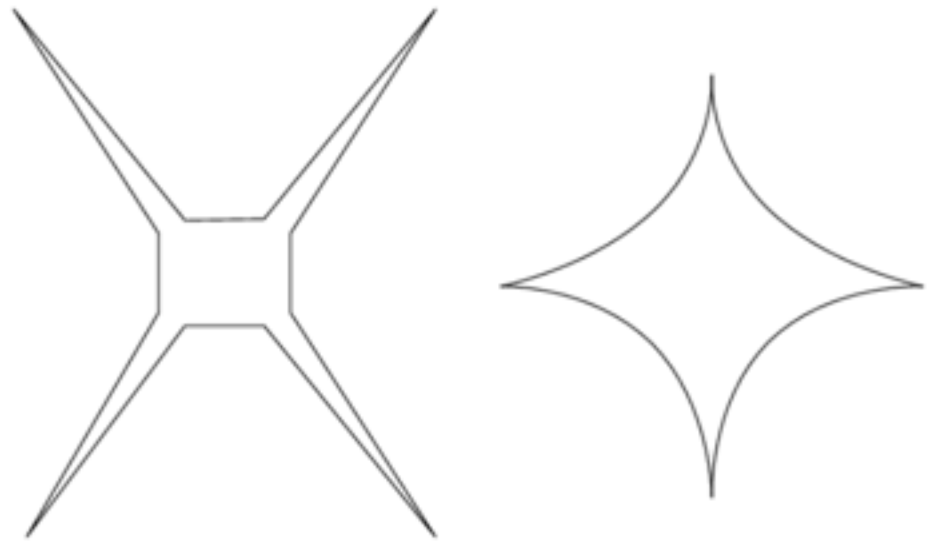


Figure 7: More accurate pictorial representations of high dimensional cubes (left) and spheres (right).

Notice that the corners of the cube are much further away from the center than are the sides. The \(d\)-sphere is drawn so that it contains almost no volume but still has radius 1. This image also suggests the next counterintuitive property of high dimensional spheres.

High dimensional spheres have fat bellies (but any equator works…)

Concentration of Measure

Suppose that you wanted to place a band around the equator of the unit sphere so that, say, 99% of the surface area of the sphere falls within that band. See Figure 8. How large do you think that band would have to be?

In two dimensions the width of the band needs to be pretty large, indeed nearly 2, to capture 99% of the perimeter of the unit circle. However as the dimension increases the width of the band needed to capture 99% of the surface area gets smaller. In very high dimensional space nearly all of the surface area of the sphere lies a very small distance away from the equator!

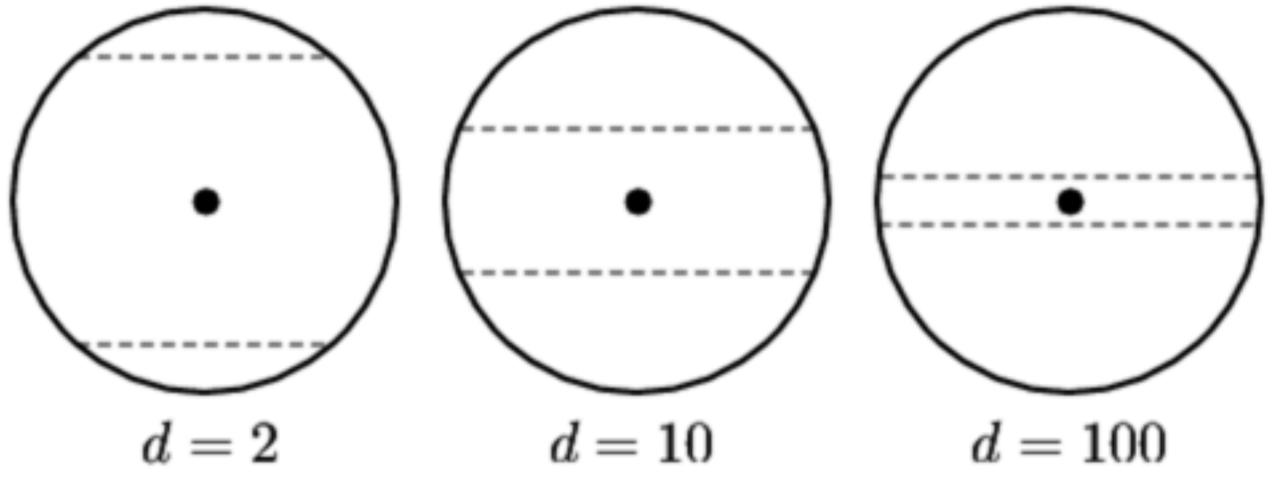


Figure 9: As the dimension increases the width of the band necessary to capture 99% of the surface area decreases rapidly. Nearly all of the surface area of a high dimensional sphere lies near the equator.

The really weird thing is that any choice of "equator" works! It must, since the sphere is, well, spherically symmetrical. We could have just as easily have chosen any of the options shown in Figure 12.

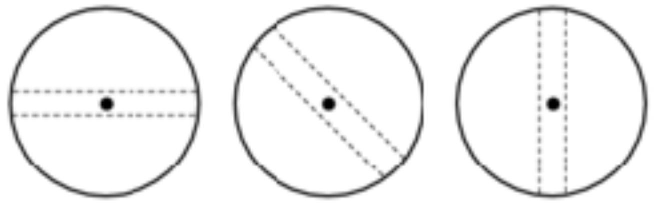


Figure 12: Any choice of equator works equally well!

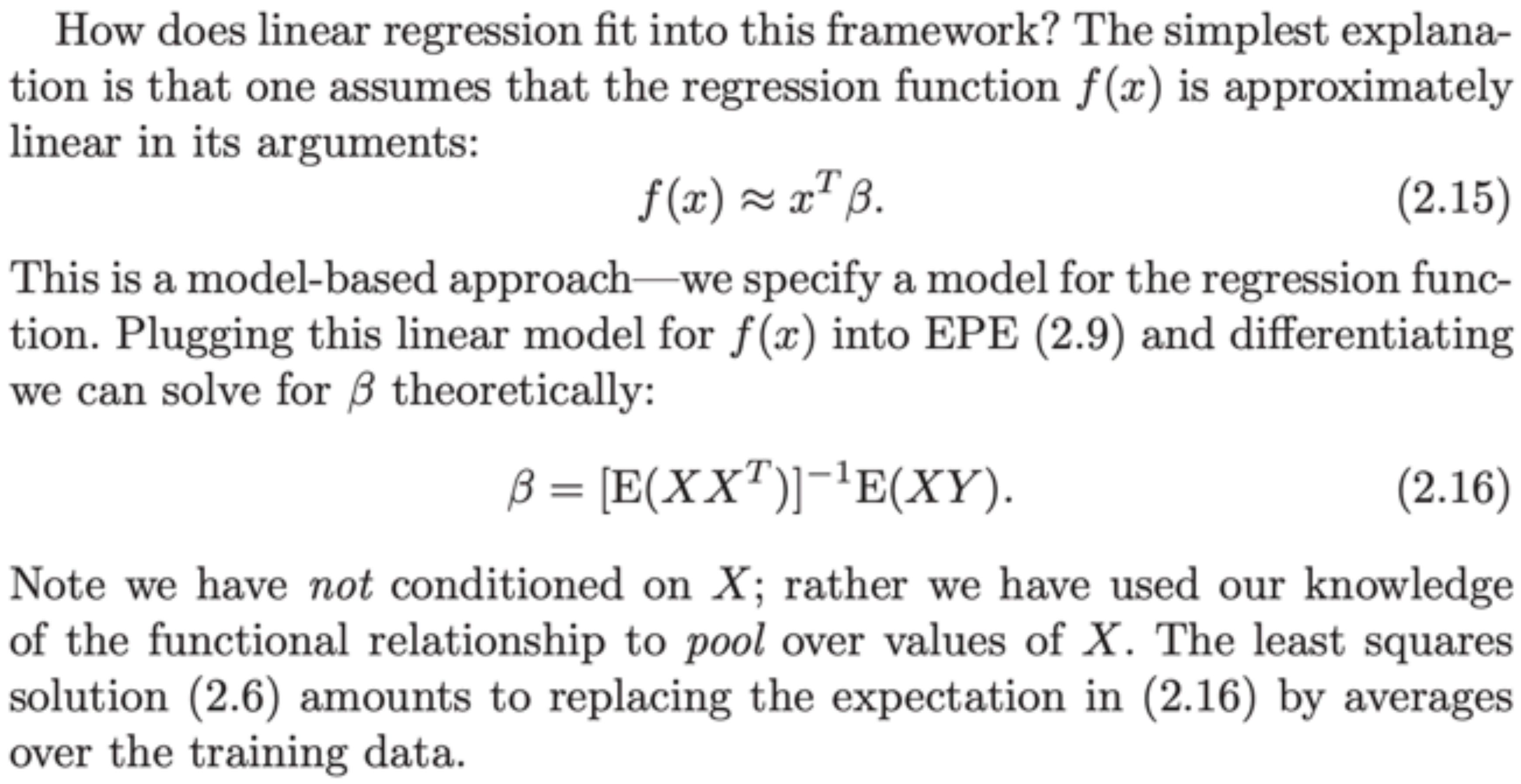
High dimensional spheres have fat bellies but they also have thick skin

All the mass is in a thin shell of the sphere.
Volume is proportional to r^d for radius r.

So if you take sphere of radius r, and an inner sphere of radius $0.99r$, then the ratio of their volumes is 0.99^d, which approaches 0 as d gets large.

A radius of $(1 - 1/d)$ will capture a constant fraction of the volume, because $(1 - 1/d)^d \to 1/e$.

Linear regression?



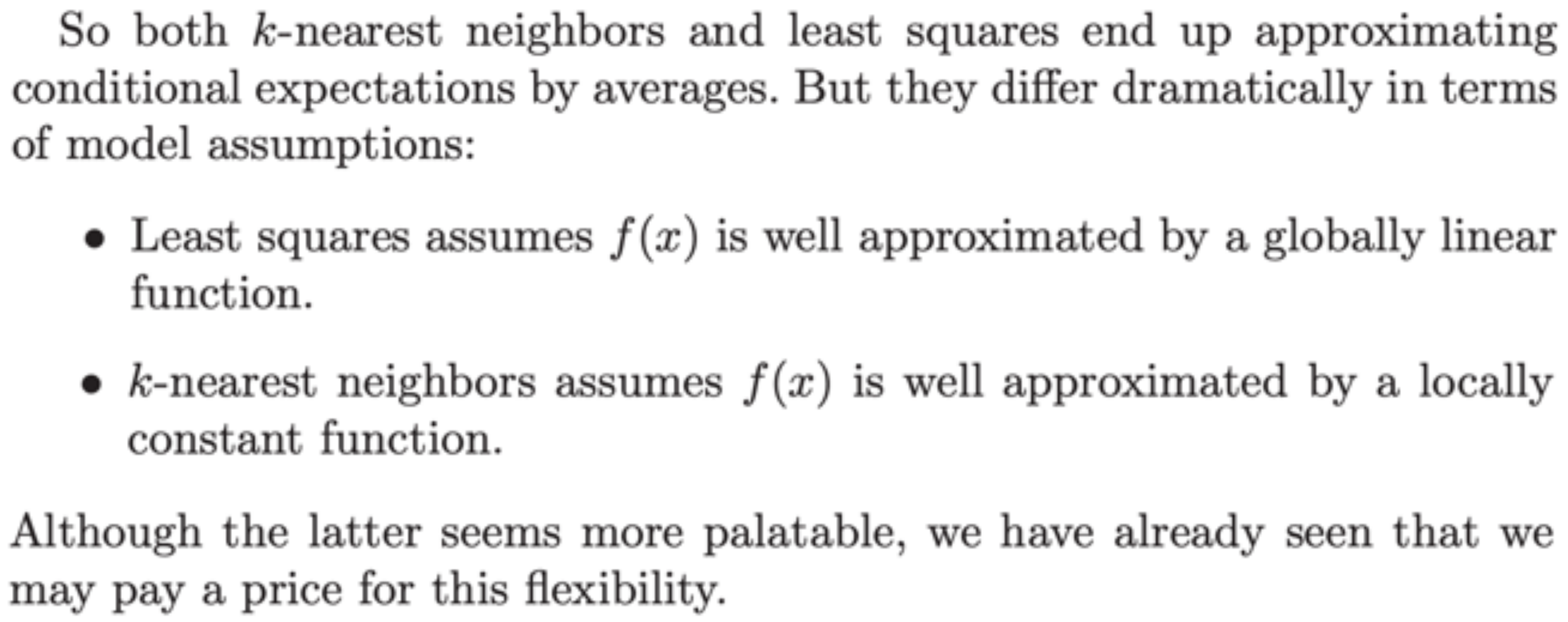
How does linear regression fit into this framework? The simplest explanation is that one assumes that the regression function $f(x)$ is approximately linear in its arguments:

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

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.

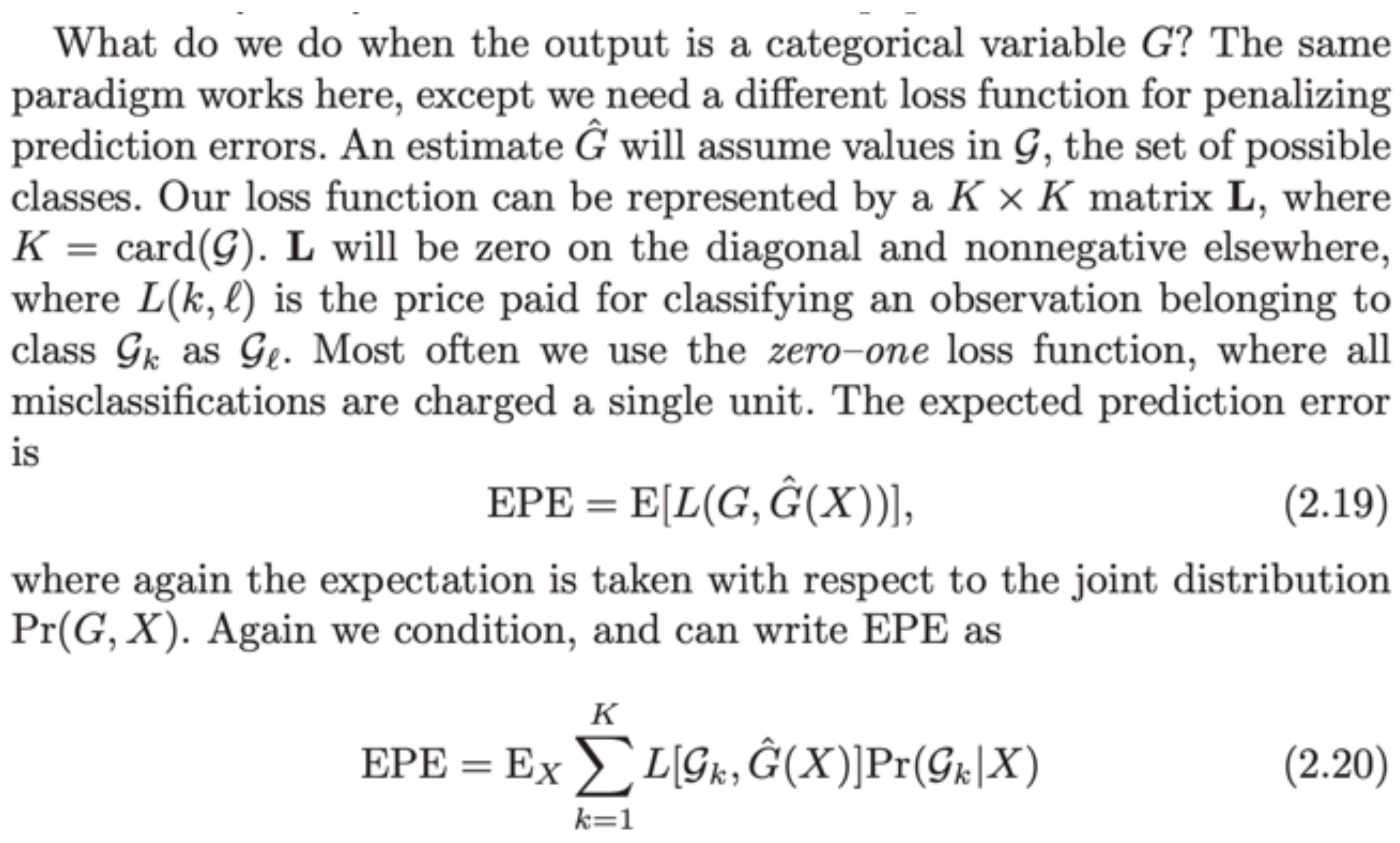

So both k-nearest neighbors and least squares end up approximating conditional expectations by averages. But they differ dramatically in terms of model assumptions:

- 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



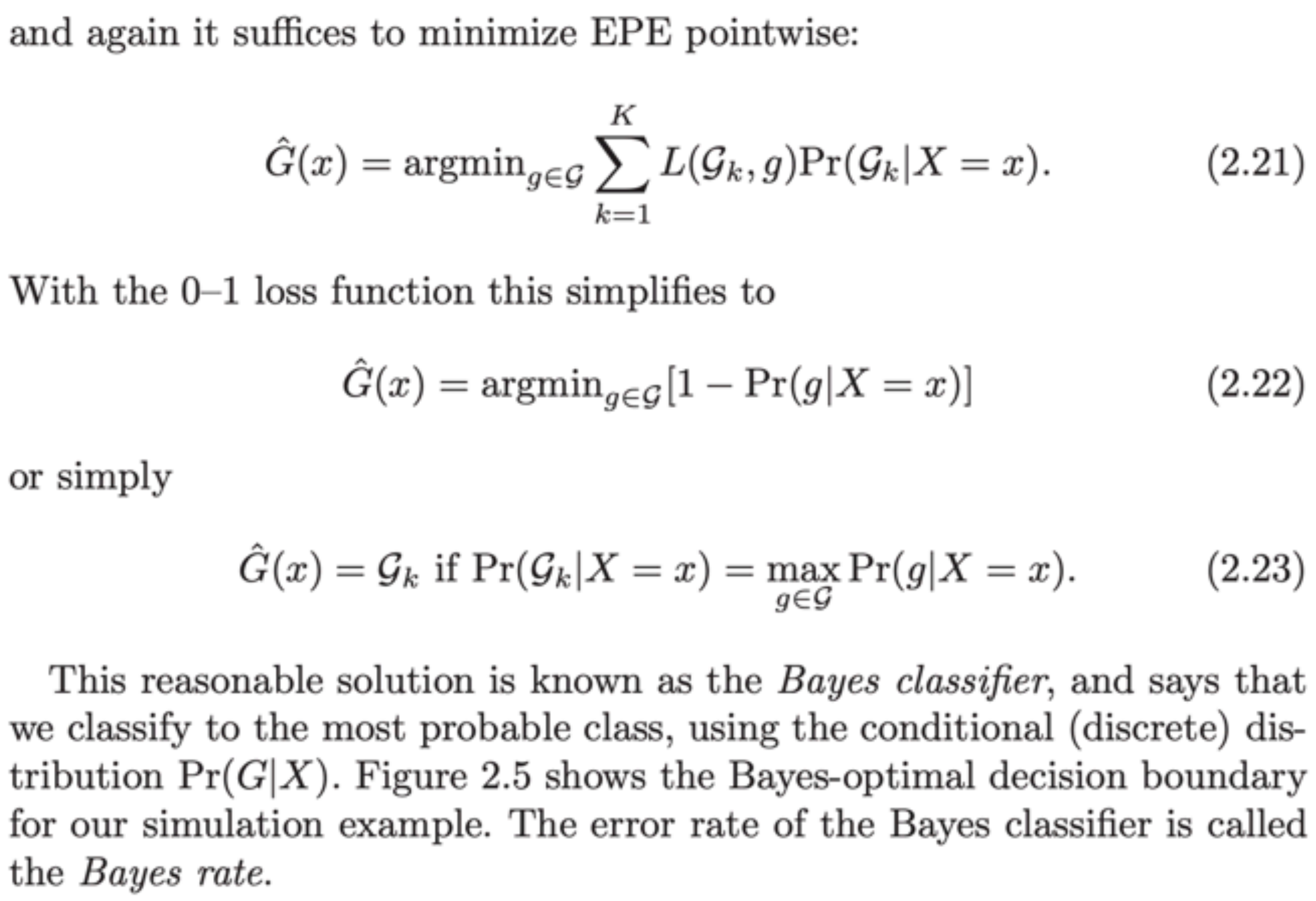
What do we do when the output is a categorical variable G? The same paradigm works here, except we need a different loss function for penalizing prediction errors. An estimate $\hat{G}$ will assume values in $\mathcal{G}$, the set of possible classes. Our loss function can be represented by a $K \times K$ 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

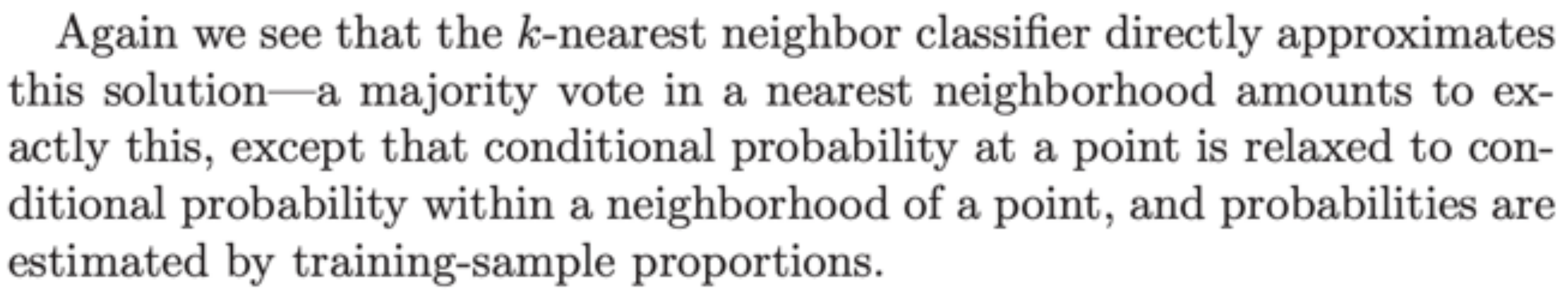
$$\hat{G}(x) = \text{argmin}_{g\in\mathcal{G}}[1 - \Pr(g|X = x)] \tag{2.22}$$

or simply

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

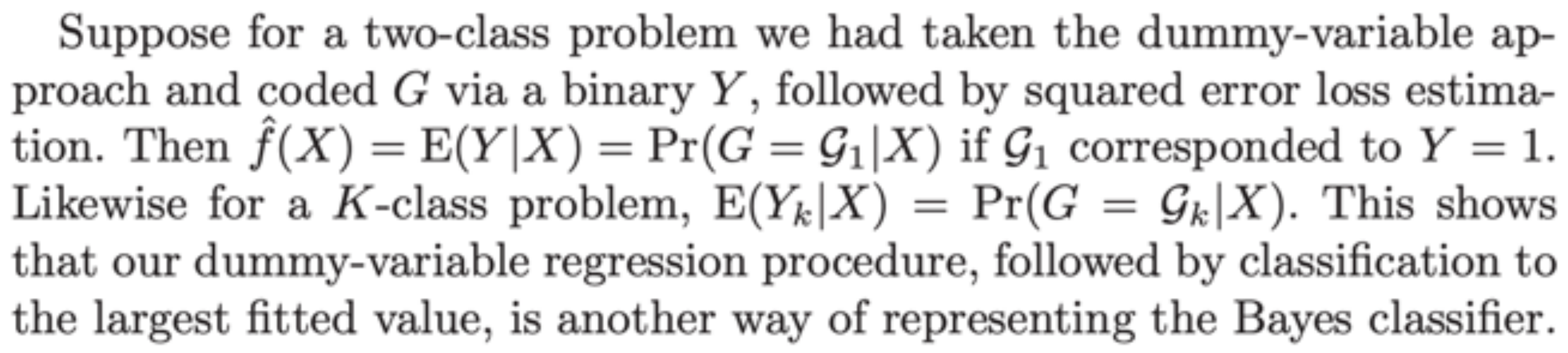
This reasonable solution is known as the *Bayes classifier*, and says that we classify to the most probable class, using the conditional (discrete) distribution $\Pr(G|X)$. Figure 2.5 shows the Bayes-optimal decision boundary for our simulation example. The error rate of the Bayes classifier is called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

kNN



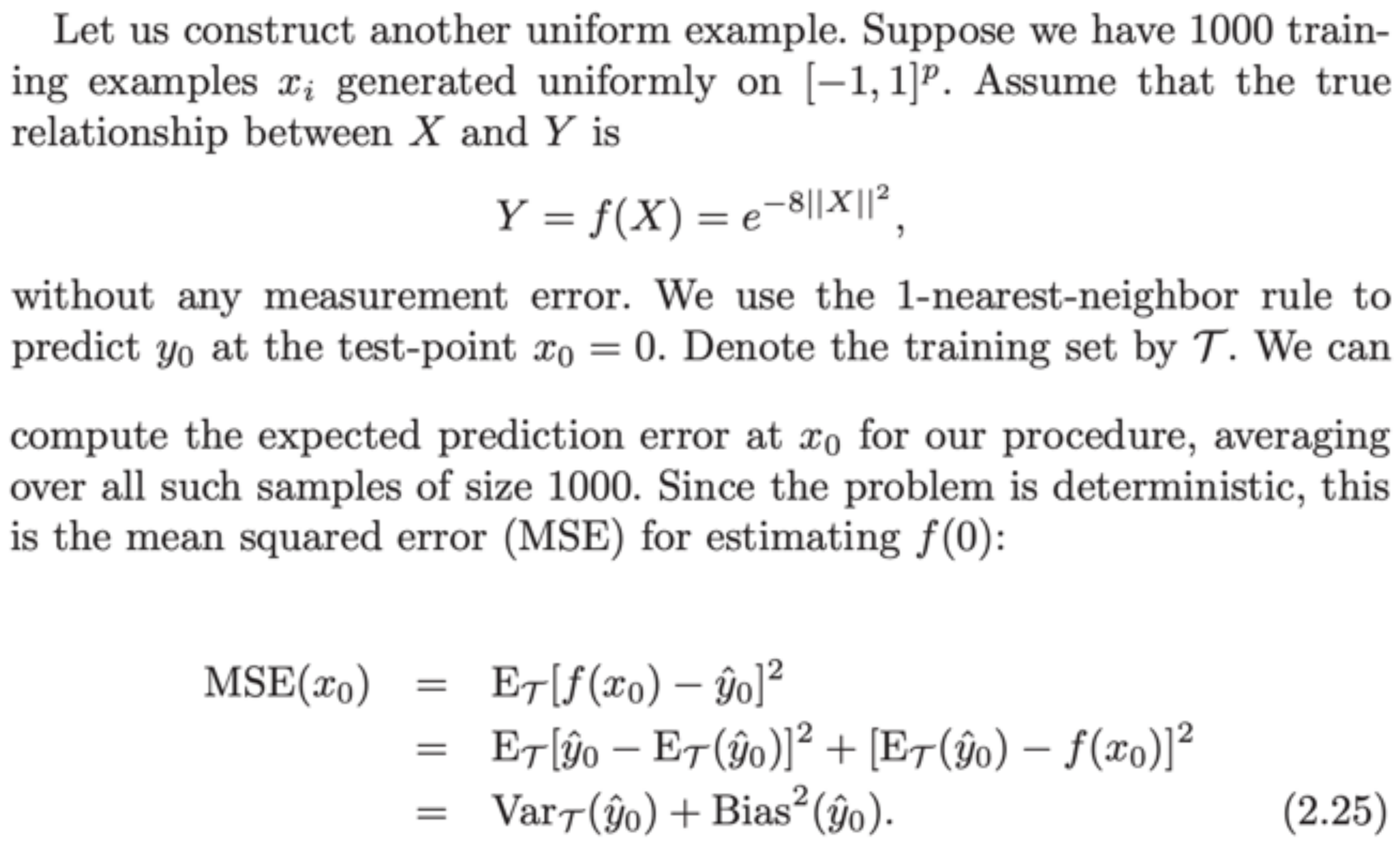
Again we see that the k-nearest neighbor classifier directly approximates this solution—a majority vote in a nearest neighborhood amounts to exactly this, except that conditional probability at a point is relaxed to conditional probability within a neighborhood of a point, and probabilities are estimated 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.

Bias-variance decomposition: regression with zero noise



Let us construct another uniform example. Suppose we have 1000 training examples x_i generated uniformly on $[-1,1]^p$. Assume that the true relationship between X and Y is

$$Y = f(X) = e^{-8||X||^2},$$

without any measurement error. We use the 1-nearest-neighbor rule to predict y_0 at the test-point $x_0 = 0$. Denote the training set by $\mathcal{T}$. We can

compute the expected prediction error at x_0 for our procedure, averaging over all such samples of size 1000. Since the problem is deterministic, this is the mean squared error (MSE) for estimating $f(0)$:

$$\begin{aligned} \text{MSE}(x_0) &= \text{E}_{\mathcal{T}}[f(x_0) - \hat{y}_0]^2 \\ &= \text{E}_{\mathcal{T}}[\hat{y}_0 - \text{E}_{\mathcal{T}}(\hat{y}_0)]^2 + [\text{E}_{\mathcal{T}}(\hat{y}_0) - f(x_0)]^2 \\ &= \text{Var}_{\mathcal{T}}(\hat{y}_0) + \text{Bias}^2(\hat{y}_0). \end{aligned} \qquad (2.25)$$

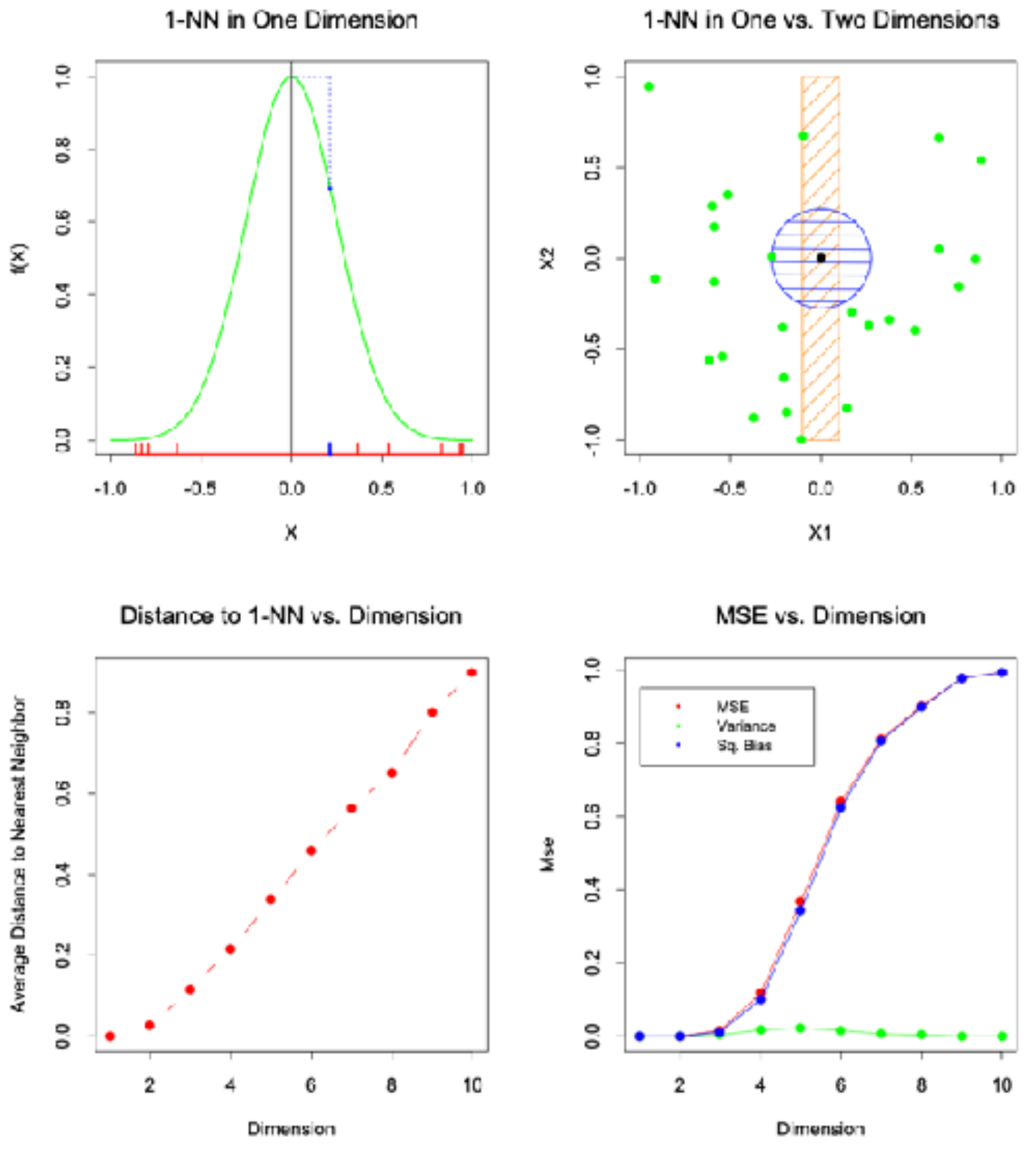


FIGURE 2.7. *A simulation example, demonstrating the curse of dimensionality and its effect on MSE, bias and variance. The input features are uniformly distributed in $[-1,1]^p$ for $p = 1,\ldots,10$ The top left panel shows the target function (no noise) in $\mathbb{R}$: $f(X) = e^{-8||X||^2}$, and demonstrates the error that 1-nearest neighbor makes in estimating $f(0)$. The training point is indicated by the blue tick mark. The top right panel illustrates why the radius of the 1-nearest neighborhood increases with dimension p. The lower left panel shows the average radius of the 1-nearest neighborhoods. The lower-right panel shows the MSE, squared bias and variance curves as a function of dimension p.*

Bias-variance decomposition: regression with zero noise

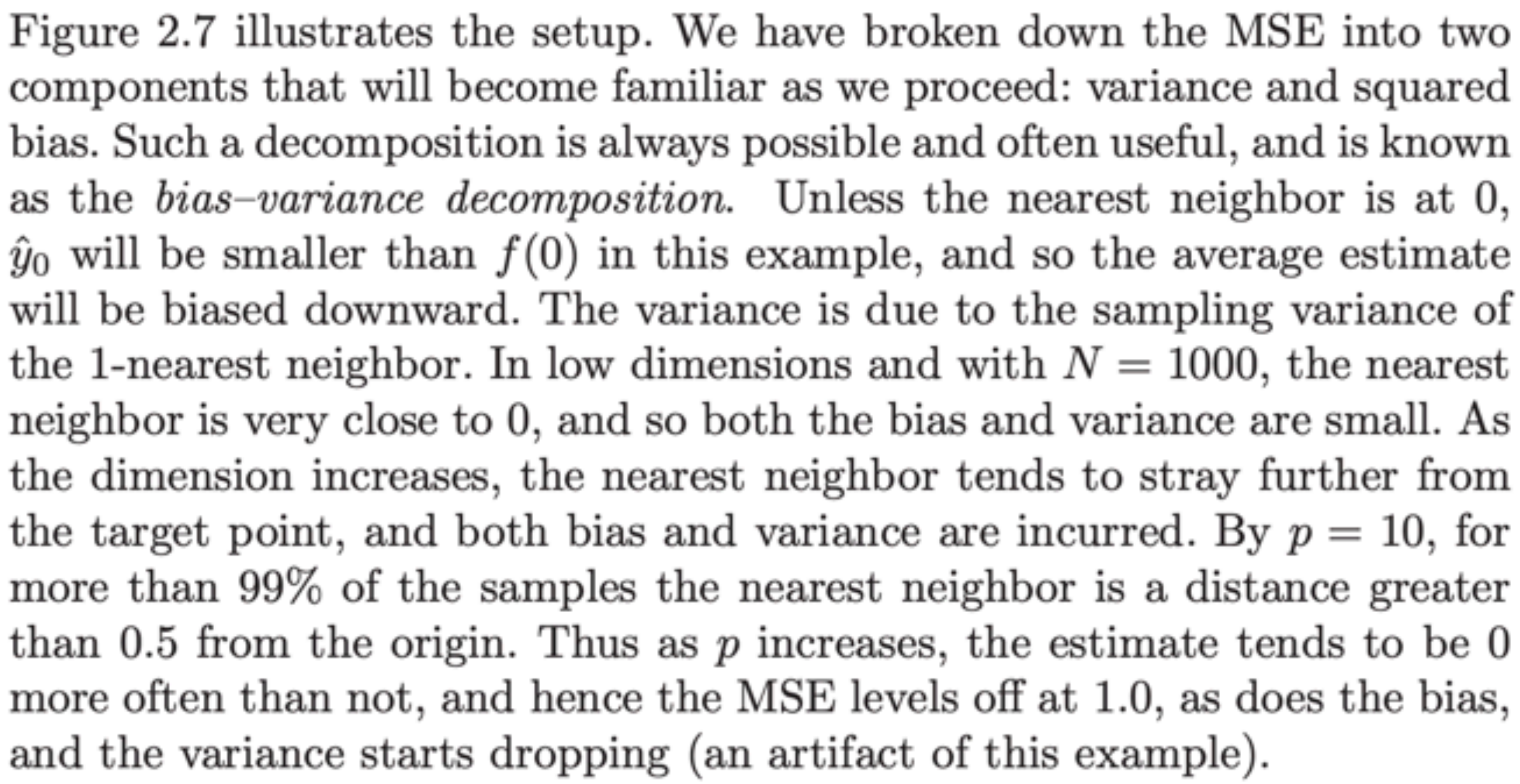


Figure 2.7 illustrates the setup. We have broken down the MSE into two components that will become familiar as we proceed: variance and squared bias. Such a decomposition is always possible and often useful, and is known as the *bias–variance decomposition*. Unless the nearest neighbor is at 0, $\hat{y}_0$ will be smaller than $f(0)$ in this example, and so the average estimate will be biased downward. The variance is due to the sampling variance of the 1-nearest neighbor. In low dimensions and with $N = 1000$, the nearest neighbor is very close to 0, and so both the bias and variance are small. As the dimension increases, the nearest neighbor tends to stray further from the target point, and both bias and variance are incurred. By $p = 10$, for more than 99% of the samples the nearest neighbor is a distance greater than 0.5 from the origin. Thus as p increases, the estimate tends to be 0 more often than not, and hence the MSE levels off at 1.0, as does the bias, and the variance starts dropping (an artifact of this example).

Bias-variance decomposition: regression with zero noise

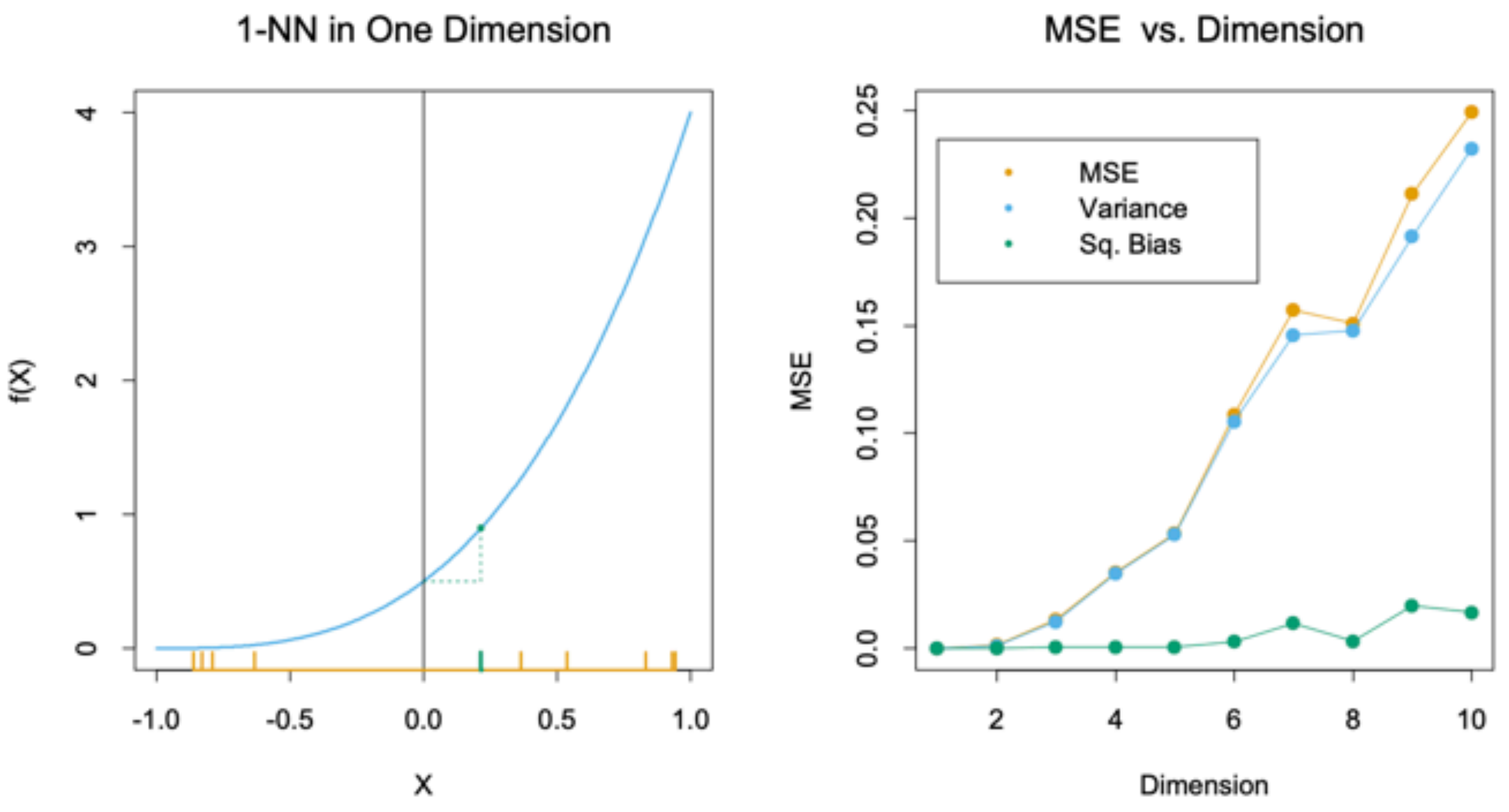


FIGURE 2.8. *A simulation example with the same setup as in Figure 2.7. Here the function is constant in all but one dimension:* $f(X) = \frac{1}{2}(X_1 + 1)^3$. *The variance dominates.*

Next class

K - nearest neighbors
Bias-variance tradeoffs