Overview of Supervised Learning

Chap – 2 ; - Part - II

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

Local Methods in High Dimensions

- Two learning techniques for prediction so far: *the stable but biased linear model* and the *less stable but apparently less biased class of k-nearest-neighbor estimates*.

- It would seem that with a reasonably large set of training data, we could always approximate the theoretically optimal conditional expectation by k-nearest-neighbor averaging, since we should be able to find a fairly large neighborhood of observations close to any x and average them.

- This approach and our intuition breaks down in high dimensions, and the phenomenon is commonly referred to as the **curse of dimensionality.**

Local Methods in High Dimensions

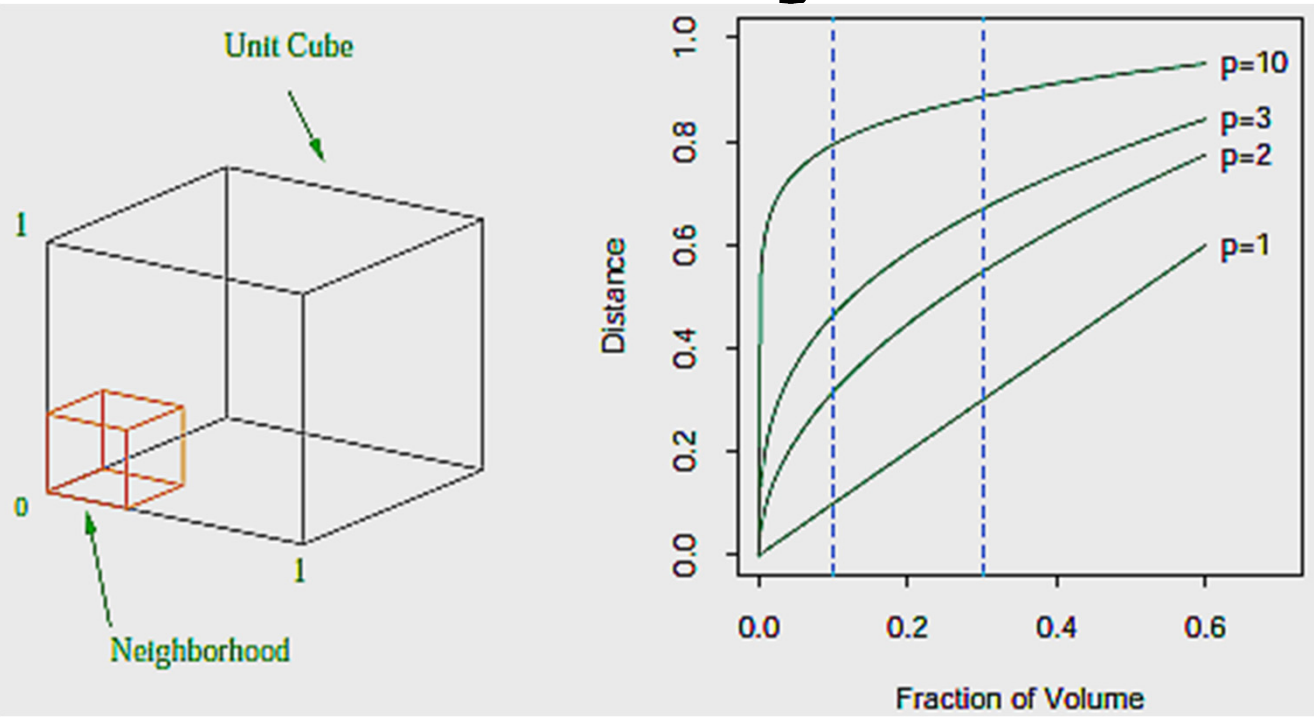


FIGURE 2.6. *The curse of dimensionality is well illustrated by a sub-cubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the sub cube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to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

- 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

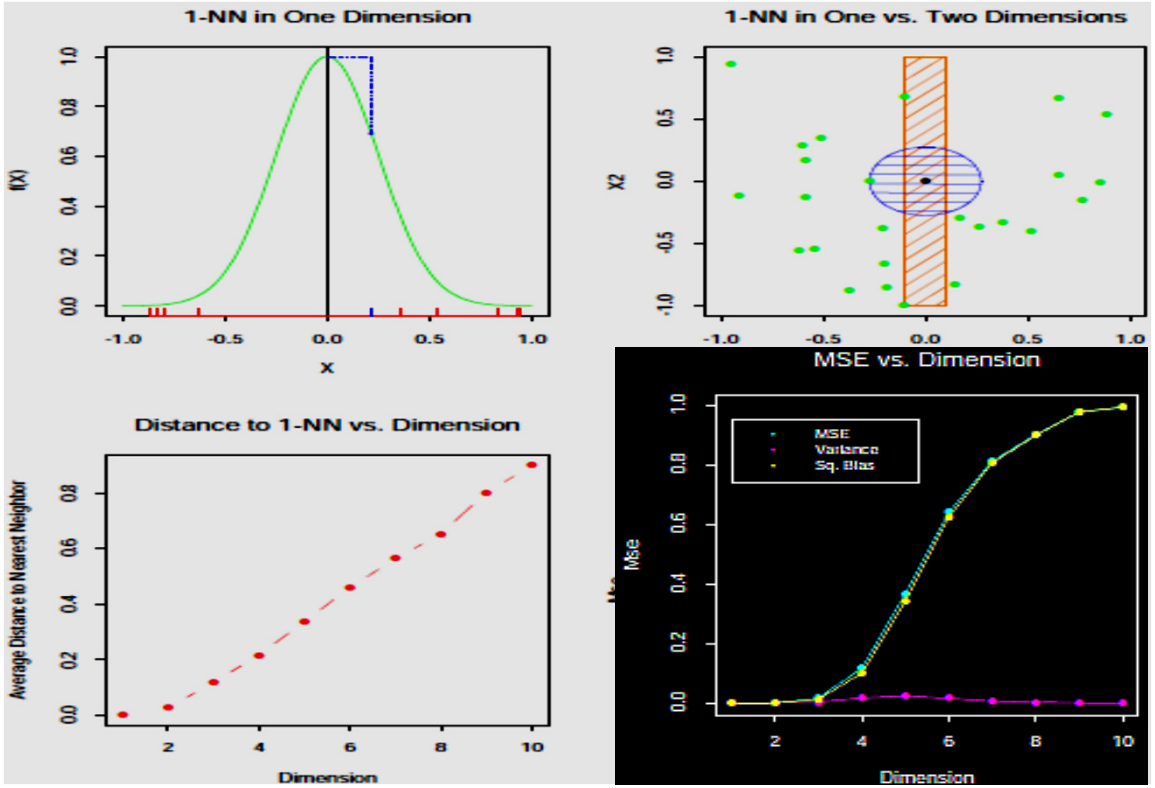
$$Y = f(X) = e^{-8||X||^2},$$

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 $\boldsymbol{\tau}$.

Compute the expected prediction error at x_0 for our procedure, averaging over all such samples of size 1000. The mean squared error (MSE) for estimating $f(0)$:

$$\begin{aligned} MSE(x_0) &= E_{\boldsymbol{\tau}}[f(x_0) - \hat{y}_0]^2 \\ &= E_{\boldsymbol{\tau}}[\hat{y}_0 - E_{\boldsymbol{\tau}}(\hat{y}_0)]^2 + [E_{\boldsymbol{\tau}}(\hat{y}_0) - f(x_0)]^2 \\ &= Var_{\boldsymbol{\tau}}(\hat{y}_0) + Bias^2(\hat{y}_0) \end{aligned} \qquad 2.25$$

Local Methods in High Dimensions



- **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* 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*.*

- **In Figure 2.7** the MSE is broken down into two components - variance and squared bias. Such a decomposition is always possible and often useful, and is known as the *bias–variance decomposition.*

- 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. 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.

- Similar phenomena occur more generally. The complexity of functions of many variables can grow exponentially with the dimension, and if we wish to be able to estimate such functions with the same accuracy as function in low dimensions, then we need the size of our training set to grow exponentially as well.

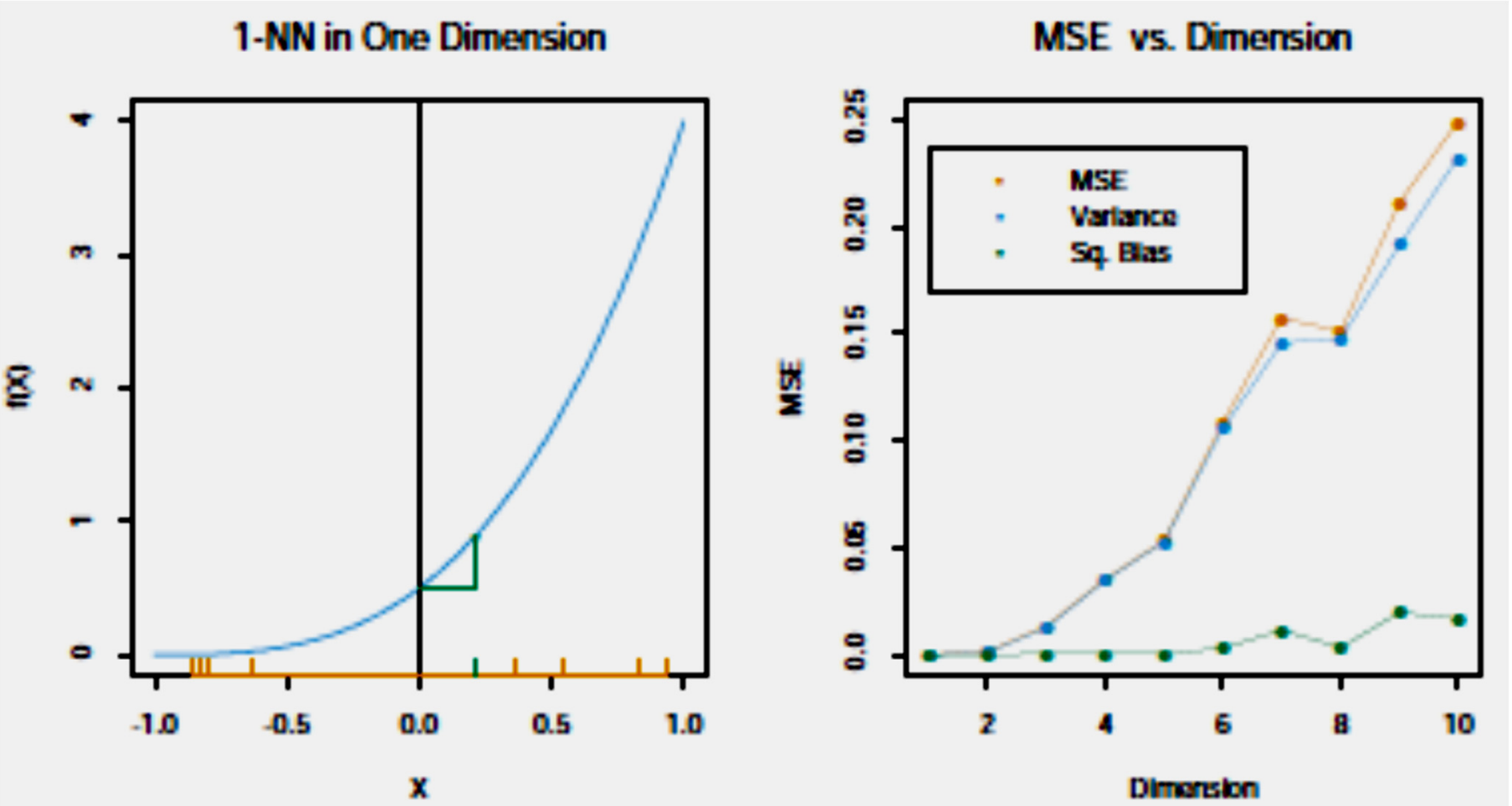


FIGURE 2.8. *A simulation example with the same setup as in Figure 2.7. Here the function is constant in all but one dimension:* $F(X) = \frac{1}{2}(X_1 + 1)^3$. *The variance dominates.*

- If the function always involves only a few dimensions as in **Figure** $\mathbf{2.8}$, then the variance can dominate instead.

- Suppose, on the other hand, that we know that the relationship between Y and X is linear,

$$Y = X^T\beta + \varepsilon,$$

where $\varepsilon \sim N(0, \sigma^2)$ and we fit the model by least squares to the training data. For an arbitrary test point x_0, we have $\hat{y} = x^T{}_0\hat{\beta}$, which can be written as

$$\hat{y} = x^T{}_0\beta + \sum_{i=1}^{N} \ell_i(x_0)\varepsilon_i,$$

where $\ell_i(x_0)$ is the i^{th} element of $X(X^TX)^{-1}x_0$.

$$Y = X^T\beta + \varepsilon,$$

where $\ell_i(x_0)$ is the i^{th} element of $X(X^TX)^{-1}x_0$

$$\hat{y} = x^T{}_0\beta + \sum_{i=1}^{N} \ell_i(x_0)\varepsilon_i,$$

$$\begin{aligned}
\mathrm{EPE}(x_0) &= \mathrm{E}_{y_0|x_0}\mathrm{E}_{\mathcal{T}}(y_0 - \hat{y}_0)^2 \\
&= \mathrm{Var}(y_0|x_0) + \mathrm{E}_{\mathcal{T}}[\hat{y}_0 - \mathrm{E}_{\mathcal{T}}\hat{y}_0]^2 + [\mathrm{E}_{\mathcal{T}}\hat{y}_0 - x_0^T\beta]^2 \\
&= \mathrm{Var}(y_0|x_0) + \mathrm{Var}_{\mathcal{T}}(\hat{y}_0) + \mathrm{Bias}^2(\hat{y}_0) \\
&= \sigma^2 + \mathrm{E}_{\mathcal{T}}x_0^T(\mathbf{X}^T\mathbf{X})^{-1}x_0\sigma^2 + 0^2.
\end{aligned}$$

If N is large and τ were selected at random, and assuming E(X) = 0,

then X^TX -> NCov(X) and

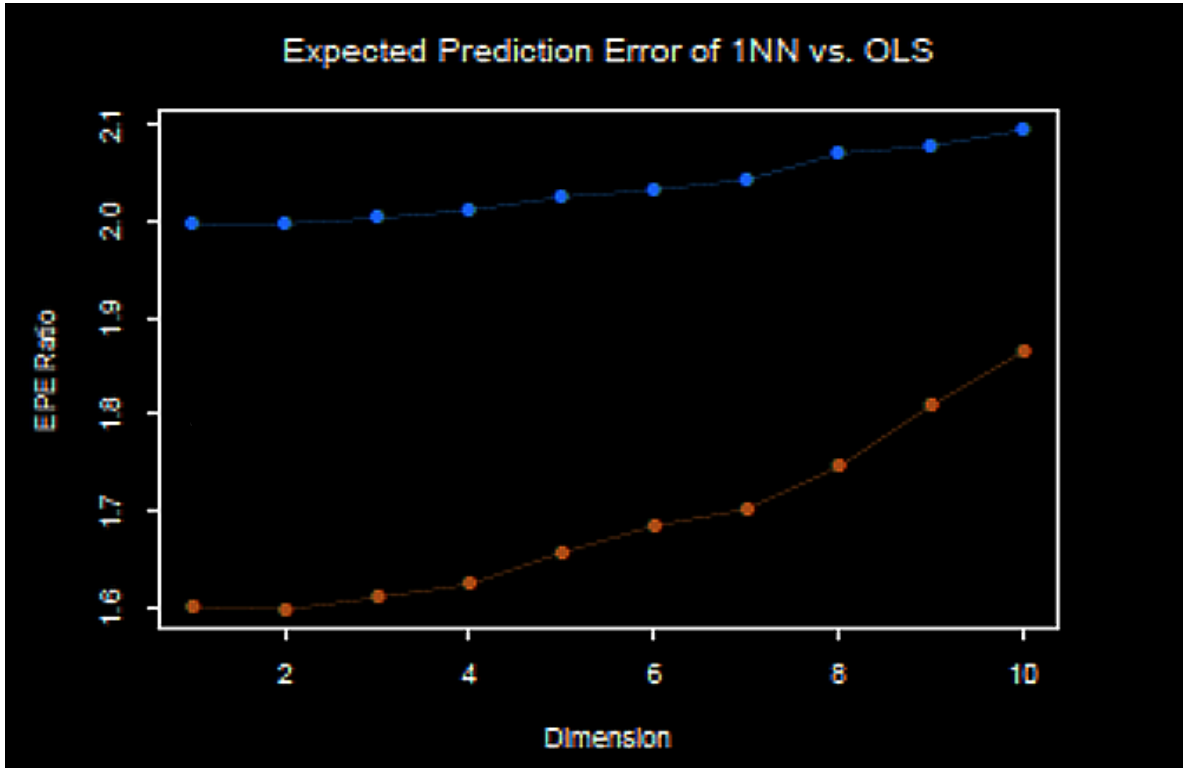
$$E_{x_0} EPE(x_0) \sim E_{x_0} x_0^T Cov(X)^{-1} x_0 \sigma^2/N + \sigma^2$$

$$= trace[Cov(X)^{-1} Cov(x_0)]_{\sigma^2}/N + \sigma^2$$

$$= \sigma^2(p/N) + \sigma^2. \qquad 2.28$$

- we see that the expected EPE increases linearly as a function of p,with slope σ^2/N. If N is large and/or σ^2 is small, this growth in variance is negligible (0 in the deterministic case). By imposing some heavy restrictions on the class of models being fitted, we have avoided the curse of dimensionality.

Expected Prediction Error of 1NN vs. OLS



- ***FIGURE 2.9***. *The curves show the expected prediction error* $(at\ x_0 = 0)$ *for* $1-$*nearest neighbor relative to least squares for the model* $Y = f(X) + \varepsilon$ *. For the* ***blue curve,*** $f(x) = x_1$, *while for the* ***orange curve*** $f(x) = \frac{1}{2}(x_1 + 1)^3$.

- **Figure 2.9** compares 1 −nearest neighbor vs. Least squares in two situations, both of which have the form $Y = f(X) + \boldsymbol{\varepsilon}$, X is uniform as before, and $\boldsymbol{\varepsilon} \sim N(0,1)$. The sample size is $N = 500$.

- For the blue curve, $f(x)$ is linear in the first coordinate, for the orange curve, cubic as in Figure 2.8. Shown is the relative EPE of 1-nearest neighbor to least squares, which appears to start at around 2 for the linear case. Least squares is unbiased in this case, and as discussed above the EPE is slightly above $\sigma^2 = 1$.

- The EPE for 1-nearest neighbor is always above 2, since the variance of $\hat{f}(x_0)$ in this case is at least σ^2, and the ratio increases with dimension as the nearest neighbor strays from the target point. For the cubic case, least squares is biased, which moderates the ratio. Clearly we could manufacture examples where the bias of least squares would dominate the variance, and the 1-nearest neighbor would come out the winner.

At $\boldsymbol{x}_0 = 0$; For LSQ:

$E_{x_0} EPE(x_0)^2 = \sigma^2(p/N) + \sigma^2$; p = 1; N = 500; $\text{EPE}_{\text{LSQ}} = \sigma^2 (1 + p/N) \sim \sigma^2$

For NN: **EPE = MSE = Var + Bias² = 2 σ^2 - WHY ??**

Statistical Models, Supervised Learning and Function Approximation

- Squared error loss lead us to the regression function $f(x) = E(Y|X = x)$ for a quantitative response. The class of nearest-neighbor methods can be viewed as direct estimates of this conditional expectation, but we have seen that they can fail in at least two ways:

  - ❖ If the dimension of the input space is high, the nearest neighbors need not be close to the target point, and can result in large errors;

  - ❖ If special structure is known to exist, this can be used to reduce both the bias and the variance of the estimates.

A Statistical Model for the Joint Distribution $Pr(X, Y)$

- Suppose in fact that our data arose from a statistical model.

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

where the random error ε has $E(\varepsilon) = 0$ and is independent of X. Note that for this model, $f(x) = E(Y|X = x)$, and in fact the conditional distribution $\Pr(Y|X)$ depends on X only through the conditional mean $f(x)$.

- The additive error model is a useful approximation to the truth. For most systems the input–output pairs (X, Y) will not have a deterministic relationship $Y = f(X)$. Generally there will be other unmeasured variables that also contribute to Y, including measurement error. The additive model assumes that we can capture all these departures from a deterministic relationship via the error ε.

Supervised Learning

- Assume a training set of observations $T = (xi, yi), i = 1, \ldots, N$. The observed input values to the system x_i are also fed into an artificial system, known as a learning algorithm (usually a computer program), which also produces outputs $\hat{f}(xi)$ in response to the inputs. The learning algorithm has the property that it can modify its input/output relationship $\hat{f}$ in response to differences $y_i - \hat{f}(x_i)$ between the original and generated outputs.

- This process is known as learning by example. Upon completion of the learning process the hope is that the artificial and real outputs will be close enough to be useful for all sets of inputs likely to be encountered in practice.

Function Approximation

- The approach taken in applied mathematics and statistics has been from the perspective of function approximation and estimation. Here the data pairs $\{x_i, y_i\}$ are viewed as points in a $(p + 1)$-dimensional Euclidean space. The function $f(x)$ has domain equal to the p-dimensional input subspace, and is related to the data via a model $y_i = f(x_i) + \varepsilon_i$. Although somewhat less glamorous than the learning paradigm, *treating supervised learning as a problem in function approximation* encourages the geometrical concepts of Euclidean spaces and mathematical concepts of probabilistic inference to be applied to the problem.

- Many of the approximations we will encounter have associated a set of parameters θ that can be modified to suit the data at hand. For example, the linear model $f(x) = x^T\beta$ has $\theta = \beta$. Another class of useful approximates can be expressed as *linear basis expansions*.

$$f\theta(x) = \sum_{k=1}^{K} h_k(x)\theta_k,$$

- where the h_k are a suitable set of functions or transformations of the input vector x. Traditional examples are polynomial and trigonometric expansions, where for example h_k might be x_1^2, $x_1 x_2^2$, $\cos(x_1)$ and so on. Encounter nonlinear expansions, such as the sigmoid transformation common to neural network models,

$$h_k(x) = \frac{1}{1+\exp(-x^T \beta_k)}$$

- We can use least squares to estimate the parameters θ in f_θ as we did for the linear model, by minimizing the residual sum-of squares

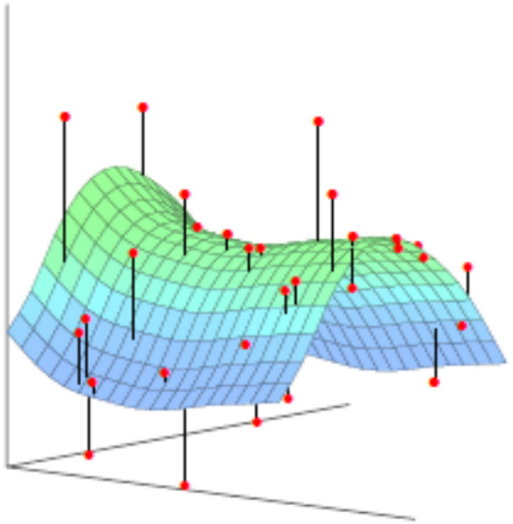
$$RSS(\theta) = \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2$$

- As a function of θ This seems a reasonable criterion for an additive error model. In terms of function approximation, we imagine our parameterized function as a surface in $p + 1$ space, and what we observe are noisy realizations from it. This is easy to visualize when $p = 2$ and the vertical coordinate is the output y, as in Figure 2.10. The noise is in the output coordinate, so we find the set of parameters such that the fitted surface gets as close to the observed points as possible, where close is measured by the sum of squared vertical errors in $RSS(\theta)$.

- For the linear model we get a simple closed form solution to the minimization problem. This is also true for the basis function methods, if the basis functions themselves do not have any hidden parameters. Otherwise the solution requires either iterative methods or numerical optimization.

- While least squares is generally very convenient, it is not the only criterion used and in some cases would not make much sense.

Statistical Models, Supervised Learning and Function Approximation



- **FIGURE 2.10**. *Least squares fitting of a function of two inputs. The parameters of $f_\theta(x)$ are chosen so as to minimize the sum-of-squared vertical errors.*

Basis Functions and Dictionary Methods

- Radial basis functions are symmetric p-dimensional kernels located at particular centroids,
- $$f_\theta(x) = K_{\lambda_m}(\mu_m, x)\theta_m;$$
- For example, the Gaussian kernel $K_\lambda(\mu, x) = e^{-\frac{||x-\mu||^2}{2\lambda}}$ is popular.
- Radial basis functions have centroids μ_m and scales λ_m that have to be determined. The spline basis functions have knots.
- A single-layer feed-forward neural network model with linear output weights can be thought of as an adaptive basis function method. The model has the form

$$f_\theta(x) = \sum_{m=1}^{M} \beta_m \sigma(\alpha_m^T x + bm),$$

Model Selection and the Bias–Variance Tradeoff

- All the models described above and many others discussed in later chapters have a smoothing or complexity parameter that has to be determined:
  - ❖ the multiplier of the penalty term;
  - ❖ the width of the kernel;
  - ❖ or the number of basis functions.

- The data arise from a model $Y = f(X) + \varepsilon$, with $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$. The expected prediction error at x_0, also known as test or generalization error, can be decomposed:

$$EPE_k(x_0) = E[(Y - \hat{f}_k(x_0))^2 | X = x_0]$$
$$= \sigma^2 + [Bias^2(\hat{f}_k(x_0)) + Var_{\boldsymbol{\tau}}(\hat{f}_k(x_0))] \qquad 2.46$$
$$= \sigma^2 + \left[f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})\right]^2 + \frac{\sigma 2}{k}. \qquad 2.47$$

- The first term $\boldsymbol{\sigma^2}$ is the *irreducible Error*.

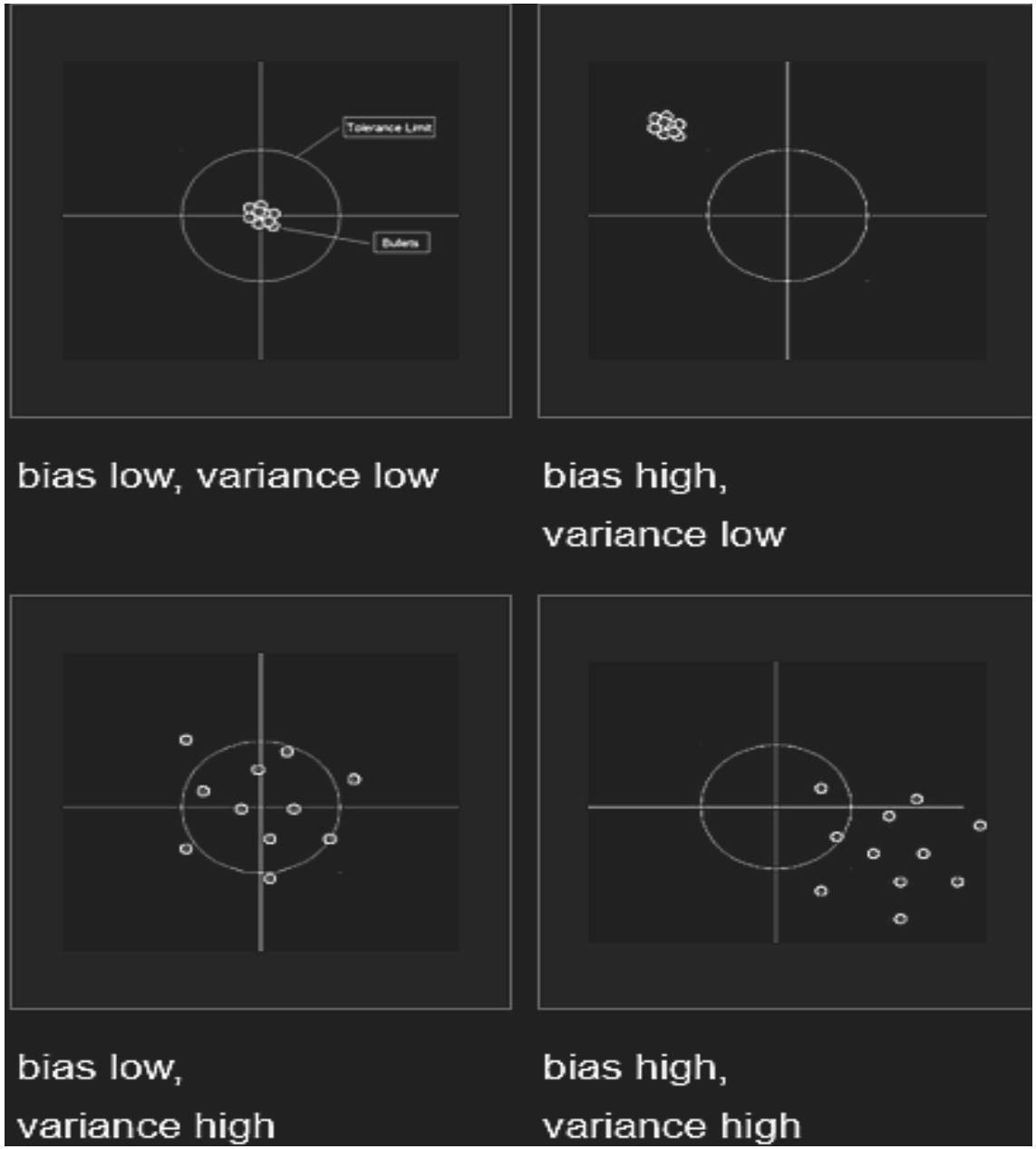
- The second and third terms are under our control, and make up the **mean squared error (MSE)** of $\hat{f}_k(x_0)$ in estimating $f(x_0)$, which is broken down into a bias component and a variance component.

- The bias term is the squared difference between the true mean $f(x_0)$ and the expected value of the estimate $\left[E_\tau(\widehat{f_k}(x_0)) - f(x_0)\right]^2$ —where the expectation averages the randomness in the training data. This term will most likely increase with k, if the true function is reasonably smooth. For small k the few closest neighbors will have values $f(x_{(\ell)})$ close to $f(x_0)$, so their average should close to $f(x_0)$. As k grows, the neighbors are further away, and then anything can happen.

- The variance term decreases as the inverse of k. So as k varies, there is a bias–variance tradeoff.

- More generally, as the model complexity of our procedure is increased, the variance tends to increase and the squared bias tends to decrease. The opposite behavior occurs as the model complexity is decreased. For k-nearest neighbors, the model complexity is controlled by k.

- Typically we would like to choose our model complexity to trade bias off with variance in such a way as to minimize the test error. An obvious estimate of test error is the training error $\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$. Unfortunately training error is not a good estimate of test error, as it does not properly account for model complexity.


Tolerance Limit
Bullets
bias low, variance low
bias high,
variance low
bias low,
variance high
bias high,
variance high

Accuracy is a description of bias; a sample will appear accurate (i.e. have low bias), but may result in underfitting.

Precision is a description of variance - may also result in an overreliance on the training data (overfitting).

$$\mathrm{E}_{D,\varepsilon}\left[(y-\hat{f}(x;D))^2\right] = \left(\mathrm{Bias}_D\left[\hat{f}(x;D)\right]\right)^2 + \mathrm{Var}_D\left[\hat{f}(x;D)\right] + \sigma^2$$

where

$$\mathrm{Bias}_D\left[\hat{f}(x;D)\right] = \mathrm{E}_D\left[\hat{f}(x;D) - f(x)\right] = \mathrm{E}_D\left[\hat{f}(x;D)\right] - \mathrm{E}\left[y(x)\right]$$

and

$$\mathrm{Var}_D\left[\hat{f}(x;D)\right] = \mathrm{E}_D[\left(\mathrm{E}_D[\hat{f}(x;D)] - \hat{f}(x;D)\right)^2].$$

The expectation ranges over different choices of the training set $D = \{(x_1, y_1)\ldots, (x_n, y_n)\}$

$$\operatorname{Var}[X] = \operatorname{E}[X^2] - \operatorname{E}[X]^2.$$

Rearranging, we get:

$$\operatorname{E}[X^2] = \operatorname{Var}[X] + \operatorname{E}[X]^2.$$

Since f is deterministic, i.e. independent of D,

$$\operatorname{E}[f] = f.$$

Thus, given $y = f + \varepsilon$ and $\operatorname{E}[\varepsilon] = 0$ (because ε is noise), implies $\operatorname{E}[y] = \operatorname{E}[f + \varepsilon] = \operatorname{E}[f] = f.$

Also, since $\operatorname{Var}[\varepsilon] = \sigma^2$,

$$\operatorname{Var}[y] = \operatorname{E}[(y - \operatorname{E}[y])^2] = \operatorname{E}[(y - f)^2] = \operatorname{E}[(f + \varepsilon - f)^2] = \operatorname{E}[\varepsilon^2] = \operatorname{Var}[\varepsilon] + \operatorname{E}[\varepsilon]^2 = \sigma^2 + 0^2 = \sigma^2.$$

Thus, since ε and $\hat{f}$ are independent, we can write

$$\begin{aligned}
MSE = \operatorname{E}\left[(y - \hat{f})^2\right] &= \operatorname{E}\left[(f + \varepsilon - \hat{f})^2\right] \\
&= \operatorname{E}\left[(f + \varepsilon - \hat{f} + \operatorname{E}[\hat{f}] - \operatorname{E}[\hat{f}])^2\right] \\
&= \operatorname{E}\left[(f - \operatorname{E}[\hat{f}])^2\right] + \operatorname{E}[\varepsilon^2] + \operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})^2\right] + 2\operatorname{E}\left[(f - \operatorname{E}[\hat{f}])\varepsilon\right] + 2\operatorname{E}\left[\varepsilon(\operatorname{E}[\hat{f}] - \hat{f})\right] + 2\operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})(f - \operatorname{E}[\hat{f}])\right] \\
&= (f - \operatorname{E}[\hat{f}])^2 + \operatorname{E}[\varepsilon^2] + \operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})^2\right] + 2(f - \operatorname{E}[\hat{f}])\operatorname{E}[\varepsilon] + 2\operatorname{E}[\varepsilon]\operatorname{E}\left[\operatorname{E}[\hat{f}] - \hat{f}\right] + 2\operatorname{E}\left[\operatorname{E}[\hat{f}] - \hat{f}\right](f - \operatorname{E}[\hat{f}]) \\
&= (f - \operatorname{E}[\hat{f}])^2 + \operatorname{E}[\varepsilon^2] + \operatorname{E}\left[(\operatorname{E}[\hat{f}] - \hat{f})^2\right] \\
&= (f - \operatorname{E}[\hat{f}])^2 + \operatorname{Var}[\varepsilon] + \operatorname{Var}\left[\hat{f}\right] \\
&= \operatorname{Bias}[\hat{f}]^2 + \operatorname{Var}[\varepsilon] + \operatorname{Var}\left[\hat{f}\right] \\
&= \operatorname{Bias}[\hat{f}]^2 + \sigma^2 + \operatorname{Var}\left[\hat{f}\right].
\end{aligned}$$

Approaches

Dimensionality reduction and feature selection can decrease variance by simplifying models. Similarly, a larger training set tends to decrease variance. Adding features (predictors) tends to decrease bias, at the expense of introducing additional variance. Learning algorithms typically have some tunable parameters that control bias and variance; for example,

- linear and Generalized linear models can be regularized to decrease their variance at the cost of increasing their bias.
- In artificial neural networks, the variance increases and the bias decreases as the number of hidden units increase, although this classical assumption has been the subject of recent debate. Like in GLMs, regularization is typically applied.
- In *k*-nearest neighbor models, a high value of k leads to high bias and low variance
- In instance-based learning, regularization can be achieved varying the mixture of prototypes and exemplars.
- In decision trees, the depth of the tree determines the variance. Decision trees are commonly pruned to control variance.

One way of resolving the trade-off is to use mixture models and ensemble learning. For example, boosting combines many "weak" (high bias) models in an ensemble that has lower bias than the individual models, while bagging combines "strong" learners in a way that reduces their variance.

Model validation methods such as cross-validation (statistics) can be used to tune models so as to optimize the trade-off.

k-nearest neighbors

In the case of k-nearest neighbors regression, when the expectation is taken over the possible labeling of a fixed training set, a closed-form expression exists that relates the bias–variance decomposition to the parameter k:

$$\mathbb{E}[(y - \hat{f}(x))^2 \mid X = x] = \left(f(x) - \frac{1}{k} \sum_{i=1}^{k} f(N_i(x)) \right)^2 + \frac{\sigma^2}{k} + \sigma^2$$

where $N_1(x), \ldots, N_k(x)$ are the k nearest neighbors of x in the training set. The bias (first term) is a monotone rising function of k, while the variance (second term) drops off as k is increased. In fact, under "reasonable assumptions" the bias of the first-nearest neighbor (1-NN) estimator vanishes entirely as the size of the training set approaches infinity.

Same as Eqn. 2.46 (Hastie)

Watch the fun:

https://mlu-explain.github.io/bias-variance/

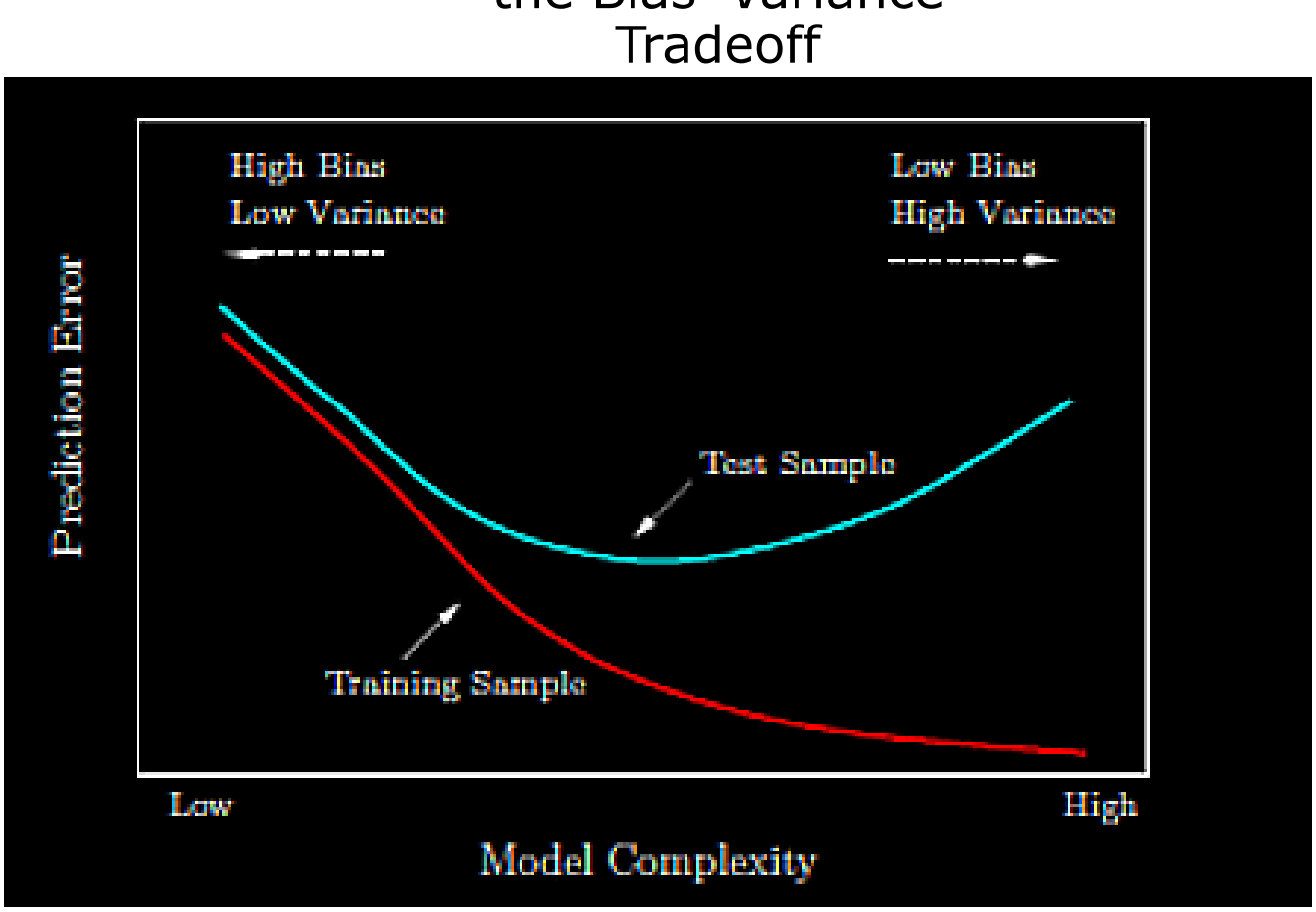


FIGURE 2.11. *Test and training error as a function of model complexity.*

- **Figure 2.11** shows the typical behavior of the test and training error, as model complexity is varied.

- The training error tends to decrease whenever we increase the model complexity, that is, whenever we fit the data harder. However with too much fitting, the model adapts itself too closely to the training data, and will not generalize well (i.e., have large test error). In that case the predictions $\hat{f}(x_0)$ will have large variance, as reflected in the last term of expression (2.46). In contrast, if the model is not complex enough, it will under-fit and may have large bias, again resulting in poor generalization.