Lecture 3

- K-NN vs Linear Regression
- Decision Theory for Classification
- The Bayes Classifier
- K-NN and Lin. Regression for Classific.
- The Bias-Variance Trade-Off

There is no free lunch in statistical learning: no one method dominates all other methods over all possible data sets. Usually, the more we assume, the more we can learn from data and the more accurate our predictions are, provided our assumptions are correct.

Last time we discussed two quite different regression methods:

① The k-NN method: $f(x) \approx \hat{f}(x) = \frac{1}{k} \sum_{i: x_i \in N_k(x)} y_i$

- "Flexible" (non-parametric) method: it approximates $f(x)$ by a locally constant function (class of these functions is large)
- Has a nice theoretical property: $\hat{f}(x) \to f(x) = \mathbb{E}[Y|X=x]$ as $N, k \to \infty$, $k/N \to 0$
  $\mathbb{E}[Y|X=x]$: regression function, which is the L_2-optimal prediction rule.
- But it suffers from the curse of dimensionality. (Prediction errors a large if $p \gg 1$.)

② The linear regression method: $f(x) \approx \hat{f}(x) = x^T \hat{\beta}$, $\hat{\beta} = (X^T X)^{-1} X^T y$ (based on the training data.)

- "Rigid" (parametric) method: it approximates $f(x)$ by a globally linear function (class of these functions is small)
- If the true relationship between the output Y and input X is approximately linear, $Y \approx f(X) = X^T \beta$, then the method can overcome the curse of dimensionality.
- But if the linearity assumption is wrong $\Rightarrow$ predictions are very bad.

Many popular regression methods are enhanced modifications of these two methods. There is a whole spectrum of methods between the rigid linear regression and the extremely flexible 1-nearest-neighbor method.

Here are some immediate ideas on how to enhance the k-NN and linear methods:

- The k-NN approximation can be written as follows: $\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{N} w_i y_i$, where $w_i = \begin{cases} 1, & \text{if } x_i \in N_k(x) \\ 0, & \text{if } x_i \notin N_k(x) \end{cases}$
  Instead of 0/1 weights, we can have weights w_i that decrease smoothly to zero with distance from the target input x: $\hat{f}(x) = \frac{1}{N} \sum_{i=1}^{N} w_i(\|x - x_i\|) y_i$
  This leads to the so called kernel methods. (must be chosen wisely in high dim.)
- We can extend the standard linear regression by allowing non-linear functions of each input, while maintaining additivity: $f(X) = \sum_{j=1}^{p} f_j(X_j)$. (arbitrary function)
  This leads to the generalized additive models.

The k-NN and linear regression methods can be also used for solving classification problems. They naturally appear under the framework of

Statistical Decision Theory for classification (for regression, see Lecture 1, p.4-5)

Let $X \in \mathbb{R}^p$ be a random input and $G \in \mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_K\}$ be the corresponding random qualitative output. We want to find a function (prediction rule) $f: \mathbb{R}^p \longrightarrow \mathcal{G}$ such that $\hat{G} = f(X) \approx G$ is an accurate prediction.

Remark: Re notation: It is a bit heavy, but $\mathcal{G} = \{1, \dots, K\}$ has a drawback: it assumes linear order between classes, which in general does not exist, e.g., $\mathcal{G} = \{$"stroke", "drug overdose", "epileptic seizure"$\}$

To measure the accuracy of prediction (goodness of f), we need a loss function $L: \mathcal{G} \times \mathcal{G} \to \mathbb{R}$.

If the expected loss is small, f is good; otherwise f is bad:

$$\mathbb{E}\left[L(G, \hat{G})\right] = \mathbb{E}\left[L(G, f(X))\right] = \begin{cases} \text{small} \Rightarrow f \text{ is good}, \\ \text{large} \Rightarrow f \text{ is bad}. \end{cases}$$

The most popular and often used loss function is the zero-one loss, where all misclassifications are charged a single unit:

$$L(G, \hat{G}) = I(G \neq \hat{G}) = \begin{cases} 0 & \text{if } G = \hat{G} \\ 1 & \text{if } G \neq \hat{G} \end{cases}$$

(I: indicator variable)

Remark: In regression settings, $L: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and we used the squared error loss $L(Y, \hat{Y}) = (Y - \hat{Y})^2$. In classification settings, it does not make sense.

The corresponding expected loss, the expected prediction error (EPE), is then:

$$EPE(f) = \mathbb{E}\left[I(G \neq \hat{G})\right] = \mathbb{E}\left[I(G \neq f(X))\right]$$

← this expected value is w.r.t. the joint distribution of X and G.

Our goal is to find $f(x)$ that minimizes $EPE(f)$.

As in the regression case, we condition on X and use the law of total expectation:

$$EPE(f) = \mathbb{E}_X\left[\mathbb{E}_G\left[I(G \neq f(X)) \mid X\right]\right] \to \min$$

(wrt X / wrt G)

to minimize $EPE(f)$, we need to find $f(x)$ that minimizes this expression

The inner expectation:

$$\mathbb{E}\left[I(G \neq f(x)) \mid X = x\right] = \sum_{k=1}^{K} I(\mathcal{G}_k \neq f(x)) \cdot \mathbb{P}(G = \mathcal{G}_k \mid X = x)$$

$$\Rightarrow f(x) = \arg\min_{\mathcal{G}_s \in \mathcal{G}} \sum_{k=1}^{K} \underbrace{I(\mathcal{G}_k \neq \mathcal{G}_s)}_{= \begin{cases} 0, & k = s \\ 1, & k \neq s \end{cases}} \mathbb{P}(G = \mathcal{G}_k \mid X = x) = \arg\min_{\mathcal{G}_s \in \mathcal{G}} \sum_{k \neq s} \mathbb{P}(G = \mathcal{G}_k \mid X = x)$$

since $\sum_{k=1}^{K} \mathbb{P}(G = \mathcal{G}_k \mid X = x) = 1$

$$= \arg\min_{\mathcal{G}_s \in \mathcal{G}} \left(1 - \mathbb{P}(G = \mathcal{G}_s \mid X = x)\right)$$

$$= \arg\max_{\mathcal{G}_s \in \mathcal{G}} \mathbb{P}(G = \mathcal{G}_s \mid X = x)$$

So, the prediction rule that minimizes $EPE(f)$ is

$$\boxed{f(x) = \arg\max_{g \in \mathcal{G}} \mathbb{P}(G = g \mid X = x)}$$

This solution is called the Bayes classifier.

If $X = x$, then the best prediction (zero-one sense) is $\hat{G} = \mathcal{G}_k \iff \mathbb{P}(G = \mathcal{G}_k \mid X = x) = \max_{g \in \mathcal{G}} \mathbb{P}(G = g \mid X = x)$

Remark: The Bayes classifier $f(x) = \arg\max_{g \in \mathcal{G}} \mathbb{P}(G=g \mid X=x)$ is a classification analog of the regression function $f(x) = \mathbb{E}[Y \mid X=x]$

- Both are obtained by minimizing the EPE, but w.r.t. different loss functions.
- Both functions are unknown and we want to learn them from the data.
- The Bayes classifier and the regression function are, in fact, related.

Consider a 2-class problem: $G \in \mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1\}$.

Define a binary dummy-variable: $Y = 0$ if $G = \mathcal{G}_0$ and $Y = 1$ if $G = \mathcal{G}_1$.

$$\Rightarrow \mathbb{E}[Y \mid X=x] = \mathbb{P}(G=\mathcal{G}_1 \mid X=x)$$

$$\Rightarrow \arg\max_{g \in \mathcal{G}} \mathbb{P}(G=g \mid X=x) = \begin{cases} \mathcal{G}_0 & \text{if } \mathbb{E}[Y \mid X=x] < 1/2 \\ \mathcal{G}_1 & \text{if } \mathbb{E}[Y \mid X=x] > 1/2 \end{cases}$$

Remark This can be generalized to a K-class problem via

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_K \end{bmatrix} \qquad Y_i = \begin{cases} 1 & \text{if } G = \mathcal{G}_i \\ 0 & \text{if } G \neq \mathcal{G}_i \end{cases}$$

In applications, the conditional probability $\mathbb{P}(G=g \mid X=x)$ is unknown, and, therefore, we can't use the Bayes classifier directly.

One strategy is to estimate $\mathbb{P}(G=g \mid X=x)$ from the data, and then classify a given observation $X=x$ to the class with highest estimated probability. One such method is

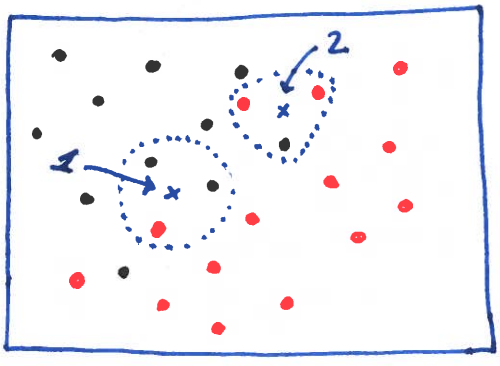
- **The k-NN Classifier** ← very much in the spirit of its regression brother.

1. Choose $k \in \{1, \dots, N\}$, where N is the sample size of training data $\{(x_1, g_1), \dots, (x_N, g_N)\}$
2. Relax the condition $X=x$ by $X \in N_k(x)$, where $N_k(x)$ is the set of the k closest to x inputs x_i in the data.
   ↳ since most likely we don't have $x_i = x$ in the data
3. Estimate the conditional probability $\mathbb{P}(G=g \mid X=x)$ by the fraction of points in $N_k(x)$ whose class is g: $\mathbb{P}(G=g \mid X=x) \approx \frac{1}{k} \sum_{x_i \in N_k(x)} I(g_i = g)$
4. Approximate the Bayes classifier:

$$f(x) \approx \hat{f}(x) = \arg\max_{g \in \mathcal{G}} \underbrace{\frac{1}{k}}_{\text{not important}} \sum_{x_i \in N_k(x)} I(g_i = g) = \arg\max_{g \in \mathcal{G}} \sum_{x_i \in N_k(x)} I(g_i = g)$$

So, the k-NN classifier is simply a majority vote in the neighborhood $N_k(x)$.

Despite the fact that the k-NN classifier is a very simple approach, it often performs very well (close to the optimal Bayes classifier) in low dimensions. In high dimensions, however, it suffers from the curse of dimensionality.



Remark: The choice of k has a drastic effect on the k-NN classifier. We will discuss later how to choose k in a systematic way.

Another strategy is to model the Bayes classifier instead of estimating it.
One method that implements this idea is the linear regression model.

- **Linear Regression for Classification**

Consider a 2-class problem : $G \in \mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1\}$.

Define a binary dummy-variable : $Y = 0$ if $G = \mathcal{G}_0$ and $Y = 1$ if $G = \mathcal{G}_1$.

Convert the training data : $\{(x_1, g_1), \dots, (x_N, g_N)\} \rightsquigarrow \{(x_1, y_1), \dots, (x_N, y_N)\}$, where

$$y_i = \begin{cases} 0 & \text{if } g_i = \mathcal{G}_0 \\ 1 & \text{if } g_i = \mathcal{G}_1 \end{cases}$$

Use linear regression for predicting output Y from input X :

$$\hat{Y} = \hat{f}(X) = X^T \hat{\beta}, \quad \text{where } \hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y,$$

$$X = \begin{bmatrix} 1 \\ X_1 \\ \vdots \\ X_p \end{bmatrix} \quad \text{and} \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} \dots & x_{1p} \\ \vdots & & \vdots \\ 1 & x_{N1} \dots & x_{Np} \end{bmatrix} = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Finally, predict class G of input X as follows :

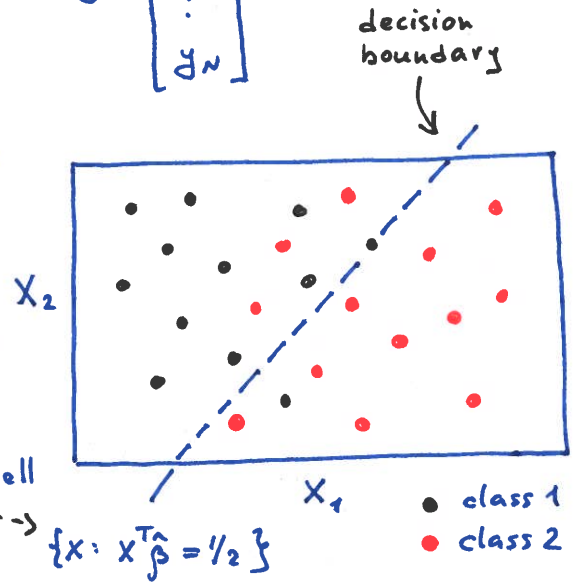
$$\hat{G} = \begin{cases} \mathcal{G}_0 & \text{if } \hat{Y} < 1/2 \\ \mathcal{G}_1 & \text{if } \hat{Y} > 1/2 \end{cases}$$

The sets of inputs in $\mathbb{R}^p$ classified as $\mathcal{G}_0$ and $\mathcal{G}_1$ correspond to

$\{X : X^T \hat{\beta} < \frac{1}{2}\} \leftarrow \mathcal{G}_0$

$\{X : X^T \hat{\beta} > \frac{1}{2}\} \leftarrow \mathcal{G}_1$

This approach will work well if $\exists$ a hyperplane ----> $\{X : X^T \hat{\beta} = 1/2\}$ that approximately divides the two classes.



Remark : This can be generalized to the K-class problem using dummy-variable

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_K \end{bmatrix} \quad \text{where } Y_i = \begin{cases} 0 & \text{if } G \neq \mathcal{G}_i \\ 1 & \text{if } G = \mathcal{G}_i \end{cases}$$

Use multiple linear regression (↓ will discuss later) for predicting Y from X : $\hat{Y}^T = X^T \hat{B}$ ($\hat{B}$: matrix of estimated regression parameters) $\Rightarrow$ predict class G by $\hat{G} = \mathcal{G}_{\arg\max_k \hat{Y}_k}$ (does not work well)

We will discuss this in more detail when we will be discussing classification problems.

As mentioned at the beginning, no one method dominates all others over all possible data sets. On a particular data set, however, one specific method may performe better than others. Selecting the best method from a collection of methods for a given training data set is called model selection, and it is one of the most difficult tasks in statistical learning.

Example :
- Should we use a more flexible k-NN method or a less flexible linear regression?
- If k-NN, then what value of k should we use ?
- If linear regression, then how many parameters $\beta_0 \dots \beta_p$ should we use ? (May be some inputs X_i are not relevant for predicting Y and we can set $\beta_i = 0$)

We will dicuss model selection later. Here let's discuss two competing statistical forces that make model selection a challenging task.

The Bias-Variance Trade-Off

Let's consider the regression setting with the squared error loss function.

Remark: For other reasonable loss functions and for classification, the story is similar.

Suppose that: $Y = f(X) + \varepsilon$, where $\mathbb{E}[\varepsilon] = 0$, $\mathbb{V}[\varepsilon] = \sigma^2$ (not necessarily small).

Then $\mathbb{E}[Y \mid X = x] = \mathbb{E}[f(x) + \varepsilon \mid X = x] = f(x)$ ← the regression function that we want to learn.

Let $\mathcal{T} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ be the training data,

and $\hat{f}(X) \approx f(X)$ be an approximation of $f(X)$ obtained from $\mathcal{T}$ by some method (different regression methods will lead to different $\hat{f}(x)$).

So, our prediction for output Y from input X is $\hat{Y} = \hat{f}(X)$.

A reasonable model $\hat{f}$ should perform well on the training data $\mathcal{T}$.

This performance is quantified by the **training error**:

$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i)) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{f}(x_i))^2$$

Q: Can we use $\overline{\text{err}}$ for choosing between different models $\hat{f}$?

A: No. We are not really interested how well $\hat{f}$ works on $\mathcal{T}$.

Instead, we are interested in the accuracy $Y \approx \hat{f}(X)$, where X is new previously unseen input.

Example: Suppose, we want to use the k-NN method and need to select k.

To minimize $\overline{\text{err}}$, regardless of the training data $\mathcal{T}$, we should always use $k = 1$ (which does not feel right especially in the context of the limiting result on p. 6: $\hat{f} \to \mathbb{E}[Y \mid X = x]$ $k, N \to \infty$ $\frac{k}{N} \to 0$).

In this case, $\hat{f}(x_i) = y_i$ (since $N_1(x_i) = x_i$), and $\overline{\text{err}} = 0$.

So, instead of $\overline{\text{err}}$, we are interested in how well $\hat{f}$ can "generalize" from the training data to new unseen data.

Let X be a new fixed input. The quantity of interest is the expected prediction error (EPE) at X, aka **generalization or test error**

$$\text{Err}(X) = \mathbb{E}[L(Y, \hat{f}(X))] = \mathbb{E}[(Y - \hat{f}(X))^2]$$

w.r.t. randomness in Y and $\mathcal{T}$

This test error can always be decomposed into the sum of three fundamental quantities:

$$\text{Err}(X) = \mathbb{E}[(\varepsilon + f(x) - \hat{f}(x))^2] = \mathbb{E}[\varepsilon^2] + \mathbb{E}[(f(x) - \hat{f}(x))^2] + 2\,\mathbb{E}[\underbrace{\varepsilon}_{\text{independent}} \cdot (f(x) - \hat{f}(x))]$$

$$= 2\,\underbrace{\mathbb{E}[\varepsilon]}_{0} \cdot \mathbb{E}[f(x) - \hat{f}(x)] = 0$$

$$= \sigma^2 + \mathbb{E}[(f(x) - \mathbb{E}[\hat{f}(x)] + \mathbb{E}[\hat{f}(x)] - \hat{f}(x))^2] \quad \text{(wrt randomness in } \mathcal{T}\text{)}$$

$$= \sigma^2 + \mathbb{E}[\underbrace{(f(x) - \mathbb{E}[\hat{f}(x)])^2}_{\text{const}}] + \underbrace{\mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]}_{\mathbb{V}[\hat{f}(x)]}$$

$$+ 2\,\mathbb{E}[\underbrace{(f(x) - \mathbb{E}[\hat{f}(x)])}_{\text{const}} (\mathbb{E}[\hat{f}(x)] - \hat{f}(x))]$$

$$\cdot\, \mathbb{E}[(\mathbb{E}[\hat{f}(x)] - \hat{f}(x))] = 0$$

So $$\mathrm{Err}(x) = \underbrace{\sigma^2}_{\text{irreducible error}} + \underbrace{\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)^2}_{\text{squared bias of } \hat{f}(x)} + \underbrace{\mathbb{V}[\hat{f}(x)]}_{\text{variance of } \hat{f}(x)}$$ ← bias-variance trade-off.

All three terms are non-negative and we want them to be as small as possible.

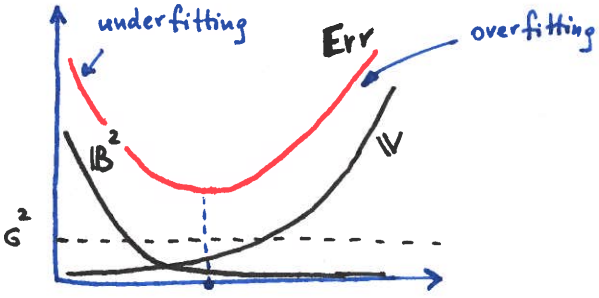
- σ^2 : irreducible error, we can't control it, even if we know the true value $f(x)$.
- $\mathbb{B}[\hat{f}(x)]^2$: squared bias of $\hat{f}(x)$, quantifies the error made by approximating the true value $f(x)$ by a model $\hat{f}(x)$. ($\mathbb{E}$ is w.r.t to training data $\mathcal{T}$)
- $\mathbb{V}[\hat{f}(x)]$: variance of $\hat{f}(x)$, quantifies the variability of $\hat{f}(x)$ w.r.t. $\mathcal{T}$. ideally, $\hat{f}(x)$ should not change a lot between different training sets.

} can control to some degree

Generally: the more flexible the model $\hat{f}$ is, the lower the bias $\mathbb{B}[\hat{f}(x)]^2$ but the higher the variance $\mathbb{V}[\hat{f}(x)]$ } that is why "trade-off"

Remark: Intuitively, model flexibility (aka model complexity) refers to the ability to fit many different functional forms. For instance, quadratic polynomials is a more flexible model than linear functions. Often (but not always), more flexible models have more parameters. The concept of flexibility / complexity is formalized by the notion of the degrees of freedom ↷ aka effective number of parameters.

Schematically, the picture looks as follows:



We want to choose the model flexibility to trade bias off with variance to minimize the test error.

The training error $\overline{\mathrm{err}}$ is a bad estimate of the test error: it does not account for model flexibility. The training error $\overline{\mathrm{err}}$ consistently ↓ as flexibility ↑. Typically, if model is flexible enough, then $\overline{\mathrm{err}} = 0$.

Remark: A U-shape of the test error Err is a fundamental property of statistical learning that holds regardless of the statistical method being used.

Remark: In applications, f is unknown, and it is not possible to compute Err, $\mathbb{B}$, $\mathbb{V}$. But it is useful to keep the general picture in mind.

There are 2 bad cases:

- **Overfitting**: $\hat{f}$ has small $\overline{\mathrm{err}}$, but large Err (small bias, but large variance) The model adapts itself too closely to $\mathcal{T}$ (finds patters specific to $\mathcal{T}$, but not to f), and will not generalize well (have large test error)
- **Underfitting**: $\hat{f}$ has large Err, because variance is small, but bias is large. The model is not flexible enough to approximate f well.

Remark: Regardless of whether Overfitting / Underfitting occurs, we almost always expect $\overline{\mathrm{err}} < \mathrm{Err}$, since most learning methods seek to minimize $\overline{\mathrm{err}}$ (directly or indirectly)

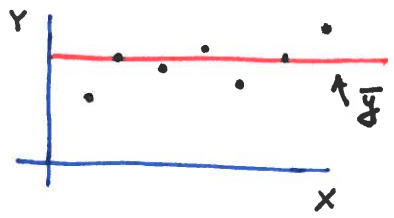
Example: To illustrate the bias-variance trade-off, let's consider the k-NN regression method:

$$f(x) \approx \hat{f}_k(x) = \frac{1}{k} \sum_{i : x_i \in N_k(x)} y_i \quad , \quad k \in \{1, 2, \dots, N\}$$

- If $\underline{k = N} \Rightarrow \hat{f}_N(x) = \frac{1}{N} \sum_{i=1}^{N} y_N$ ← for a given $\mathcal{T}$, it is a constant.

In this case, we approximate $f(x)$ by a constant function (↓ that depends on $\mathcal{T}$).

effectively one parameter



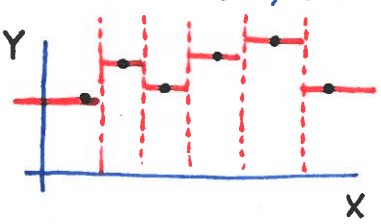
Whatever input X is, our prediction for the output Y is $\hat{Y} = \bar{y}$

Very rigid (not flexible) method.

- If $\underline{k = 1} \Rightarrow \hat{f}_1(x) = y_{i(x)}$, where $i(x) = \arg\min_{i=1,\dots,N} dist(x, x_i)$

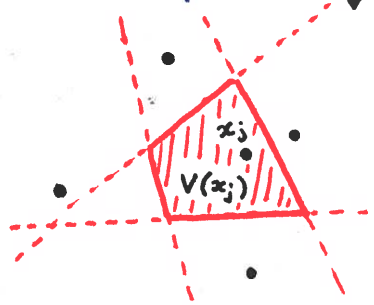
In this case, we approximate $f(x)$ by a locally constant function on a Voronoi tessellation of the $\mathcal{T}$.

effectively N parameters



This is the most flexible k-NN method.

If $p = 2$ ↓



$V(x_j) = \{x : x_j \text{ is the closest to } x \text{ among all } x_1, \dots x_N\}$

$\forall x \in V(x_j) \quad \hat{f}_1(x) = y_j$

Remark: As discussed above, $\overline{err}(\hat{f}_1) = 0$

In general, k controls the flexibility of the k-NN model: $\underline{k \uparrow \Rightarrow \text{flexibility} \downarrow}$

Let's compute the bias and the variance and check how they behave w.r.t. k.

We assumed that $Y = f(X) + \varepsilon \Rightarrow y_i = f(x_i) + \varepsilon_i$ ($\varepsilon_1, \dots \varepsilon_N$ are iid $\mathbb{E}[\varepsilon_i] = 0$, $\mathbb{V}[\varepsilon_i] = \sigma^2$)

Therefore: $\hat{f}_k(x) = \frac{1}{k} \sum_{i : x_i \in N_k(x)} (f(x_i) + \varepsilon_i)$

Assume for simplicity that $x_1, \dots x_N$ in $\mathcal{T}$ are fixed (nonrandom) and randomness arises from $y_1 \dots y_N$ (from $\varepsilon_1 \dots \varepsilon_N$)

- $\mathbb{B}[\hat{f}_k(x)] = \left(f(x) - \mathbb{E}[\hat{f}_k(x)]\right)^2 = \left(f(x) - \frac{1}{k} \sum_{x_i \in N_k(x)} f(x_i)\right)^2$

If flexibility ↑ $\Rightarrow k \downarrow \Rightarrow \mathbb{B}[\hat{f}_k(x)] \downarrow$ if $f(x)$ is reasonably smooth

(the closest neighbor is the most relevant for approximating $f(x)$, 2nd closest is a bit less relative etc)

- $\mathbb{V}[\hat{f}_k(x)] = \mathbb{V}\left[\underbrace{\frac{1}{k} \sum f(x_i)}_{\text{constant}} + \frac{1}{k} \sum \varepsilon_i\right] = \frac{1}{k^2} \sum_i \mathbb{V}[\varepsilon_i] = \frac{\sigma^2}{k}$

If flexibility ↑ $\Rightarrow k \downarrow \Rightarrow \mathbb{V}[\hat{f}_k(x)] \uparrow$

Final Remark: For both regression and classificati... choosing the correct level of flexibility is critical for the success of any learning method.