1. Introduce a basic class of learning methods, namely **local methods**.
2. Discuss the fundamental concept of **bias-variance** trade-off to understand parameter tuning (a.k.a. model selection)
Outline

Learning with Local Methods

From Bias-Variance to Cross-Validation
The problem

What is the price of one house given its area?
The problem

What is the price of one house given its area? Start from data...

<table>
<thead>
<tr>
<th>Area ($m^2$)</th>
<th>Price ($A$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62</td>
</tr>
<tr>
<td>200</td>
<td>135</td>
</tr>
<tr>
<td>300</td>
<td>93</td>
</tr>
<tr>
<td>400</td>
<td>114</td>
</tr>
</tbody>
</table>

Let $S$ the houses example dataset ($n = 100$)

$S = \{(x_1, y_1), ..., (x_n, y_n)\}$

Given a new point $x^*$ we want to predict $y^*$ by means of $S$. 

MLCC 2019
The problem

What is the price of one house given its area? Start from data...

<table>
<thead>
<tr>
<th>Area ($m^2$)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 = 62$</td>
<td>$y_1 = 99,200$</td>
</tr>
<tr>
<td>$x_2 = 64$</td>
<td>$y_2 = 135,700$</td>
</tr>
<tr>
<td>$x_3 = 65$</td>
<td>$y_3 = 93,300$</td>
</tr>
<tr>
<td>$x_4 = 66$</td>
<td>$y_4 = 114,000$</td>
</tr>
</tbody>
</table>

Let $S$ the houses example dataset ($n = 100$)

$$S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$
The problem

What is the price of one house given its area? Start from data...

<table>
<thead>
<tr>
<th>Area (m²)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 = 62$</td>
<td>$y_1 = 99,200$</td>
</tr>
<tr>
<td>$x_2 = 64$</td>
<td>$y_2 = 135,700$</td>
</tr>
<tr>
<td>$x_3 = 65$</td>
<td>$y_3 = 93,300$</td>
</tr>
<tr>
<td>$x_4 = 66$</td>
<td>$y_4 = 114,000$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>

Let $S$ the houses example dataset ($n = 100$)

$$S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

Given a new point $x^*$ we want to predict $y^*$ by means of $S$. 

MLCC 2019 7
Example

Let $x^*$ a $300m^2$ house.
Let $x^*$ a $300m^2$ house.

<table>
<thead>
<tr>
<th>Area ($m^2$)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{93}$</td>
<td>$y_{93} = 274,600$</td>
</tr>
<tr>
<td>$x_{94}$</td>
<td>$y_{94} = 324,900$</td>
</tr>
<tr>
<td>$x_{95}$</td>
<td>$y_{95} = 311,200$</td>
</tr>
<tr>
<td>$x_{96}$</td>
<td>$y_{96} = 515,400$</td>
</tr>
</tbody>
</table>

What is its price?
Nearest Neighbor: $y^*$ is the same of the closest point to $x^*$ in $S$.

$$y^* = 311, 200$$

<table>
<thead>
<tr>
<th>Area ($m^2$)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{93}$ = 255</td>
<td>$y_{93}$ = 274, 600</td>
</tr>
<tr>
<td>$x_{94}$ = 264</td>
<td>$y_{94}$ = 324, 900</td>
</tr>
<tr>
<td>$x_{95}$ = 310</td>
<td>$y_{95}$ = 311, 200</td>
</tr>
<tr>
<td>$x_{96}$ = 480</td>
<td>$y_{96}$ = 515, 400</td>
</tr>
</tbody>
</table>

MLCC 2019
Nearest Neighbors

**Nearest Neighbor:** \( y^* \) is the same of the closest point to \( x^* \) in \( S \).

\[
y^* = 311, 200
\]

<table>
<thead>
<tr>
<th>Area (( m^2 ))</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{93} = 255 )</td>
<td>( y_{93} = 274, 600 )</td>
</tr>
<tr>
<td>( x_{94} = 264 )</td>
<td>( y_{94} = 324, 900 )</td>
</tr>
<tr>
<td>( x_{95} = 310 )</td>
<td>( y_{95} = 311, 200 )</td>
</tr>
<tr>
<td>( x_{96} = 480 )</td>
<td>( y_{96} = 515, 400 )</td>
</tr>
</tbody>
</table>

MLCC 2019 11
Nearest Neighbors

\[ S = \{(x_i, y_i)\}_{i=1}^{n} \text{ with } x_i \in \mathbb{R}^D, y_i \in \mathbb{R} \]
Nearest Neighbors

- $S = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbb{R}^D, y_i \in \mathbb{R}$
- $x^*$ the new point $x^* \in \mathbb{R}^D$, $y^* = \hat{f}(x^*)$

Computational cost $O(nD)$: we compute $n$ times the distance $\|x - x_i\|$ that costs $O(D)$.
Nearest Neighbors

- \( S = \{(x_i, y_i)\}_{i=1}^{n} \) with \( x_i \in \mathbb{R}^{D}, y_i \in \mathbb{R} \)
- \( x^* \) the new point \( x^* \in \mathbb{R}^{D} \),
- \( y^* \) the predicted output \( y^* = \hat{f}(x^*) \) where

\[
y^* = y_j \quad j = \arg\min_{i=1,\ldots,n} \|x - x_i\|
\]
Nearest Neighbors

- \( S = \{(x_i, y_i)\}_{i=1}^{n} \) with \( x_i \in \mathbb{R}^{D}, y_i \in \mathbb{R} \)
- \( x^* \) the new point \( x^* \in \mathbb{R}^{D} \),
- \( y_* \) the predicted output \( y_* = \hat{f}(x^*) \) where
  \[ y_* = y_j \quad j = \arg \min_{i=1,...,n} ||x - x_i|| \]

Computational cost \( O(nD) \): we compute \( n \) times the distance \( ||x - x_i|| \) that costs \( O(D) \)
Nearest Neighbors

- $S = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbb{R}^D, y_i \in \mathbb{R}$
- $x^*$ the new point $x^* \in \mathbb{R}^D$
- $y_*$ the predicted output $y_* = \hat{f}(x^*)$ where
  \[
y_* = y_j \quad j = \arg\min_{i=1,\ldots,n} \|x - x_i\|
\]

Computational cost $O(nD)$: we compute $n$ times the distance $\|x - x_i\|$ that costs $O(D)$

In general let $d : \mathbb{R}^D \times \mathbb{R}^D$ a distance on the input space, then

$$f(x) = y_j \quad j = \arg\min_{i=1,\ldots,n} d(x, x_i)$$
Extensions

Nearest Neighbor takes $y^*$ is the same of the closest point to $x^*$ in $S$.

<table>
<thead>
<tr>
<th>Area ($m^2$)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{93}$ = 255</td>
<td>$y_{93}$ = 274,600</td>
</tr>
<tr>
<td>$x_{94}$ = 264</td>
<td>$y_{94}$ = 324,900</td>
</tr>
<tr>
<td>$x_{95}$ = 310</td>
<td>$y_{95}$ = 311,200</td>
</tr>
<tr>
<td>$x_{96}$ = 480</td>
<td>$y_{96}$ = 515,400</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Extensions

Nearest Neighbor takes $y^*$ is the same of the closest point to $x^*$ in $S$.

<table>
<thead>
<tr>
<th>Area ($m^2$)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{93} = 255$</td>
<td>$y_{93} = 274,600$</td>
</tr>
<tr>
<td>$x_{94} = 264$</td>
<td>$y_{94} = 324,900$</td>
</tr>
<tr>
<td>$x_{95} = 310$</td>
<td>$y_{95} = 311,200$</td>
</tr>
<tr>
<td>$x_{96} = 480$</td>
<td>$y_{96} = 515,400$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Can we do better? (for example using more points)
K-Nearest Neighbor: $y^*$ is the mean of the values of the $K$ closest point to $x^*$ in $S$. If $K = 3$ we have

$$y^* = \frac{274,600 + 324,900 + 311,200}{3} = 303,600$$

<table>
<thead>
<tr>
<th>Area ($m^2$)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{93} = 255$</td>
<td>$y_{93} = 274,600$</td>
</tr>
<tr>
<td>$x_{94} = 264$</td>
<td>$y_{94} = 324,900$</td>
</tr>
<tr>
<td>$x_{95} = 310$</td>
<td>$y_{95} = 311,200$</td>
</tr>
<tr>
<td>$x_{96} = 480$</td>
<td>$y_{96} = 515,400$</td>
</tr>
</tbody>
</table>
K-Nearest Neighbor: \( y^* \) is the mean of the values of the \( K \) closest point to \( x^* \) in \( S \). If \( K = 3 \) we have

\[
y^* = \frac{274,600 + 324,900 + 311,200}{3} = 303,600
\]

<table>
<thead>
<tr>
<th>Area (m²)</th>
<th>Price (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>274,600</td>
</tr>
<tr>
<td>264</td>
<td>324,900</td>
</tr>
<tr>
<td>310</td>
<td>311,200</td>
</tr>
<tr>
<td>480</td>
<td>515,400</td>
</tr>
</tbody>
</table>
K-Nearest Neighbors

- \( S = \{(x_i, y_i)\}_{i=1}^{n} \) with \( x_i \in \mathbb{R}^D, y_i \in \mathbb{R} \)
- \( x^* \) the new point \( x^* \in \mathbb{R}^D \),

Let \( K \) be an integer \( K << n \), defined as
- \( j_1 = \text{arg min}_{i \in \{1, \ldots, n\}} \|x^* - x_i\| \)
- \( j_t = \text{arg min}_{i \in \{1, \ldots, n\} \setminus \{j_1, \ldots, j_{t-1}\}} \|x^* - x_i\| \) for \( t \in \{2, \ldots, K\} \),

predicted output \( y^* = \frac{1}{K} \sum_{i \in \{j_1, \ldots, j_K\}} y_i \)
K-Nearest Neighbors

\[ S = \{(x_i, y_i)\}_{i=1}^{n} \quad \text{with} \quad x_i \in \mathbb{R}^D, y_i \in \mathbb{R} \]

\[ x^* \text{ the new point } x^* \in \mathbb{R}^D, \]

\[ \text{Let } K \text{ be an integer } K << n, \]
K-Nearest Neighbors

- \( S = \{(x_i, y_i)\}_{i=1}^n \) with \( x_i \in \mathbb{R}^D, y_i \in \mathbb{R} \)
- \( x^* \) the new point \( x^* \in \mathbb{R}^D \),
- Let \( K \) be an integer \( K << n \),
- \( j_1, \ldots, j_K \) defined as \( j_1 = \arg \min_{i \in \{1,\ldots,n\}} \|x^* - x_i\| \) and \( j_t = \arg \min_{i \in \{1,\ldots,n\}\setminus\{j_1,\ldots,j_{t-1}\}} \|x^* - x_i\| \) for \( t \in \{2, \ldots, K\} \),
K-Nearest Neighbors

- \( S = \{(x_i, y_i)\}_{i=1}^{n} \) with \( x_i \in \mathbb{R}^D, y_i \in \mathbb{R} \)
- \( x^* \) the new point \( x^* \in \mathbb{R}^D \),
- Let \( K \) be an integer \( K << n \),
- \( j_1, \ldots, j_K \) defined as \( j_1 = \arg\min_{i \in \{1, \ldots, n\}} \|x^* - x_i\| \) and \( j_t = \arg\min_{i \in \{1, \ldots, n\} \setminus \{j_1, \ldots, j_{t-1}\}} \|x^* - x_i\| \) for \( t \in \{2, \ldots, K\} \),
- predicted output

\[
y_* = \frac{1}{K} \sum_{i \in \{j_1, \ldots, j_K\}} y_i
\]
K-Nearest Neighbors (cont.)

\[ f(x) = \frac{1}{K} \sum_{i=1}^{K} y_{j_i} \]
K-Nearest Neighbors (cont.)

\[ f(x) = \frac{1}{K} \sum_{i=1}^{K} y_{ji} \]

**Computational cost** \( O(nD + n \log n) \): compute the \( n \) distances \( \|x - x_i\| \) for \( i = \{1, \ldots, n\} \) (each costs \( O(D) \)). Order them \( O(n \log n) \).
K-Nearest Neighbors (cont.)

\[ f(x) = \frac{1}{K} \sum_{i=1}^{K} y_{j_i} \]

- **Computational cost** \( O(nD + n \log n) \): compute the \( n \) distances \( \|x - x_i\| \) for \( i = \{1, \ldots, n\} \) (each costs \( O(D) \)). Order them \( O(n \log n) \).

- **General Metric** \( d_{f} \) is the same, but \( j_1, \ldots, j_K \) are defined as
  \[ j_1 = \arg \min_{i \in \{1, \ldots, n\}} d(x, x_i) \] and
  \[ j_t = \arg \min_{i \in \{1, \ldots, n\} \setminus \{j_1, \ldots, j_{t-1}\}} d(x, x_i) \] for \( t \in \{2, \ldots, K\} \).
K-NN puts equal weights on the values of the selected points.

Parzen Windows

PARZEN WINDOWS:

\[ \hat{f}(x) = \frac{\sum_{i=1}^{n} y_i k(x, x_i)}{\sum_{i=1}^{n} k(x, x_i)} \]

where \( k \) is a similarity function

\[ k(x, x') \geq 0 \text{ for all } x, x' \in \mathbb{R}^D \]

\[ k(x, x') \to 1 \text{ when } \|x - x'\| \to 0 \]

\[ k(x, x') \to 0 \text{ when } \|x - x'\| \to \infty \]
Parzen Windows

K-NN puts equal weights on the values of the selected points. Can we generalize it?
Parzen Windows

K-NN puts equal weights on the values of the selected points. Can we generalize it? Closer points to $x^*$ should influence more its value.
Parzen Windows

K-NN puts equal weights on the values of the selected points. Can we generalize it?

Closer points to $x^*$ should influence more its value

**PARZEN WINDOWS:**

$$
\hat{f}(x) = \frac{\sum_{i=1}^{n} y_i k(x, x_i)}{\sum_{i=1}^{n} k(x, x_i)}
$$

where $k$ is a *similarity function*

- $k(x, x') \geq 0$ for all $x, x' \in \mathbb{R}^D$
- $k(x, x') \to 1$ when $\|x - x'\| \to 0$
- $k(x, x') \to 0$ when $\|x - x'\| \to \infty$
Examples of $k$

- $k_1(x, x') = \text{sign} \left( 1 - \frac{\|x-x'\|}{\sigma} \right)_{+}$ with a $\sigma > 0$
- $k_2(x, x') = \left( 1 - \frac{\|x-x'\|}{\sigma} \right)_{+}$ with a $\sigma > 0$
- $k_3(x, x') = \left( 1 - \frac{\|x-x'\|^2}{\sigma^2} \right)_{+}$ with a $\sigma > 0$
- $k_4(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}$ with a $\sigma > 0$
- $k_5(x, x') = e^{-\frac{\|x-x'\|}{\sigma}}$ with a $\sigma > 0$
$K$-Nearest neighbor depends on $K$. When $K = 1$
$K$-Nearest neighbor depends on $K$. When $K = 2$
$K$-Nearest neighbor depends on $K$. When $K = 3$, the result changes a lot! How to select $K$?
K-Nearest neighbor depends on $K$.
When $K = 4$
K-Nearest neighbor depends on $K$.

When $K = 5$
$K$-Nearest neighbor depends on $K$.

When $K = 9$, the result changes a lot! How to select $K$?
$K$-Nearest neighbor depends on $K$.
When $K = 15$
$K$-Nearest neighbor depends on $K$.

Changing $K$ the result changes a lot! How to select $K$?
Outline

Learning with Local Methods

From Bias-Variance to Cross-Validation
Optimal choice for the Hyper-parameters

- $S = (x_i, y_i)_{i=1}^n$ training set. Name $Y = (y_1, \ldots, y_n)$ and $X = (x_1^\top, \ldots, x_n^\top)$.

- $K \in \mathbb{N}$ hyperparameter of the learning algorithm

- $\hat{f}_{S,K}$ learned function (depends on $S$ and $K$)

The expected loss $E_K$ is

$$E_K = E_S E_{x,y} \left( y - \hat{f}_{S,K}(x) \right)^2.$$
Optimal choice for the Hyper-parameters

$S = (x_i, y_i)_{i=1}^n$ training set. Name $Y = (y_1, \ldots, y_n)$ and $X = (x_1^\top, \ldots, x_n^\top)$.

$K \in \mathbb{N}$ hyperparameter of the learning algorithm

$\hat{f}_{S,K}$ learned function (depends on $S$ and $K$)

The expected loss $\mathcal{E}_K$ is

$$\mathcal{E}_K = \mathbb{E}_S \mathbb{E}_{x,y} (y - \hat{f}_{S,K}(x))^2$$
Optimal choice for the Hyper-parameters

- \( S = (x_i, y_i)_{i=1}^{n} \) training set. Name \( Y = (y_1, \ldots, y_n) \) and 
  \( X = (x_1^\top, \ldots, x_n^\top) \).
- \( K \in \mathbb{N} \) hyperparameter of the learning algorithm
- \( \hat{f}_{S,K} \) learned function (depends on \( S \) and \( K \))

The expected loss \( \mathcal{E}_K \) is
\[
\mathcal{E}_K = \mathbb{E}_S \mathbb{E}_{x,y} (y - \hat{f}_{S,K}(x))^2
\]

Optimal hyperparameter \( K^* \) should minimize \( \mathcal{E}_K \)
Optimal choice for the Hyper-parameters

$S = (x_i, y_i)_{i=1}^n$ training set. Name $Y = (y_1, \ldots, y_n)$ and $X = (x_1^\top, \ldots, x_n^\top)$.

$K \in \mathbb{N}$ hyperparameter of the learning algorithm

$\hat{f}_{S,K}$ learned function (depends on $S$ and $K$)

The expected loss $\mathcal{E}_K$ is

$$\mathcal{E}_K = \mathbb{E}_S \mathbb{E}_{x,y} (y - \hat{f}_{S,K}(x))^2$$

Optimal hyperparameter $K^*$ should minimize $\mathcal{E}_K$

$$K^* = \arg \min_{K \in \mathbb{K}} \mathcal{E}_K$$
Optimal choice for the Hyper-parameters (cont.)

Optimal hyperparameter $K^*$ should minimize $\mathcal{E}_K$

Ideally! (In practice we don't have access to the distribution)

▶ We can still try to understand the above minimization problem: does a solution exists? What does it depend on?

▶ Yet, ultimately, we need something we can compute!
Optimal choice for the Hyper-parameters (cont.)

Optimal hyperparameter $K^*$ should minimize $\mathcal{E}_K$

$$K^* = \arg \min_{K \in K} \mathcal{E}_K$$
Optimal hyperparameter $K^*$ should minimize $\mathcal{E}_K$

$$K^* = \arg \min_{K \in \mathcal{K}} \mathcal{E}_K$$

Ideally! (In practice we don’t have access to the distribution)

- We can still try to understand the above minimization problem: does a solution exist? What does it depend on?
- Yet, ultimately, we need something we can compute!
Example: regression problem

Define the pointwise expected loss

\[ \mathcal{E}_K(x) = \mathbb{E}_S \mathbb{E}_{y|x} (y - \hat{f}_{S,K}(x))^2 \]
Example: regression problem

Define the *pointwise expected loss*

\[ E_{K}(x) = \mathbb{E}_{S}\mathbb{E}_{y|x}(y - \hat{f}_{S,K}(x))^{2} \]

By definition \( E_{K} = \mathbb{E}_{x}E_{K}(x) \).
Example: regression problem

Define the \textit{pointwise expected loss}

\[ \mathcal{E}_K(x) = \mathbb{E}_y | x (y - \hat{f}_{S,K}(x))^2 \]

By definition \( \mathcal{E}_K = \mathbb{E}_x \mathcal{E}_K(x) \).

Regression setting:

\begin{itemize}
  \item Regression model \( y = f_*(x) + \delta \)
\end{itemize}
Define the *pointwise expected loss*

\[ \mathcal{E}_K(x) = \mathbb{E}_S \mathbb{E}_{y|x} (y - \hat{f}_{S,K}(x))^2 \]

By definition \( \mathcal{E}_K = \mathbb{E}_x \mathcal{E}_K(x).\)

Regression setting:

- Regression model \( y = f_*(x) + \delta \)
- \( \mathbb{E}\delta = 0, \mathbb{E}\delta^2 = \sigma^2 \)
Define the *pointwise expected loss*

\[ \mathcal{E}_K(x) = \mathbb{E}_y \mathbb{E}_{y|x}(y - \hat{f}_{S,K}(x))^2 \]

By definition \( \mathcal{E}_K = \mathbb{E}_x \mathcal{E}_K(x) \).

**Regression setting:**
- Regression model \( y = f_*(x) + \delta \)
- \( \mathbb{E} \delta = 0, \mathbb{E} \delta^2 = \sigma^2 \)

Now \( \mathcal{E}_K(x) = \mathbb{E}_y \mathbb{E}_{y|x}(y - \hat{f}_{S,K}(x))^2 \)
Define the *pointwise expected loss*

\[ \mathcal{E}_K(x) = \mathbb{E}_S \mathbb{E}_{y|x} (y - \hat{f}_{S,K}(x))^2 \]

By definition \( \mathcal{E}_K = \mathbb{E}_x \mathcal{E}_K(x) \).

Regression setting:

- Regression model \( y = f_*(x) + \delta \)
- \( \mathbb{E}\delta = 0, \mathbb{E}\delta^2 = \sigma^2 \)

Now \( \mathcal{E}_K(x) = \mathbb{E}_S \mathbb{E}_{y|x} (y - \hat{f}_{S,K}(x))^2 = \mathbb{E}_S \mathbb{E}_{y|x} (f_*(x) + \delta - \hat{f}_{S,K}(x))^2 \)
Example: regression problem

Define the *pointwise expected loss*

\[
\mathcal{E}_K(x) = \mathbb{E}_S \mathbb{E}_{y|x}(y - \hat{f}_{S,K}(x))^2
\]

By definition \( \mathcal{E}_K = \mathbb{E}_x \mathcal{E}_K(x) \).

Regression setting:

- Regression model \( y = f^*_\langle x \rangle + \delta \)
- \( \mathbb{E}\delta = 0, \mathbb{E}\delta^2 = \sigma^2 \)

Now \( \mathcal{E}_K(x) = \mathbb{E}_S \mathbb{E}_{y|x}(y - \hat{f}_{S,K}(x))^2 = \mathbb{E}_S \mathbb{E}_{y|x}(f^*_\langle x \rangle + \delta - \hat{f}_{S,K}(x))^2 \) that is

\[
\mathcal{E}_K(x) = \mathbb{E}_S(f^*_\langle x \rangle - \hat{f}_{S,K}(x))^2 + \sigma^2
\]

...
Define the *noisyless K-NN* (it is ideal!)

\[ \tilde{f}_{S,K}(x) = \frac{1}{K} \sum_{l \in K_x} f_*(x_l) \]
Define the *noisyless* $K$-NN (it is ideal!)

$$\tilde{f}_{S,K}(x) = \frac{1}{K} \sum_{l \in K_x} f_*(x_l)$$

Note that $\tilde{f}_{S,K}(x) = \mathbb{E}_{y|x} \hat{f}_{S,K}(x)$.
Bias Variance trade-off for K-NN

Define the *noisyless* K-NN (it is ideal!)

\[ \tilde{f}_{S,K}(x) = \frac{1}{K} \sum_{l \in K_x} f^*(x_l) \]

Note that \( \tilde{f}_{S,K}(x) = \mathbb{E}_{y|x} \hat{f}_{S,K}(x) \).

Consider

\[ \mathcal{E}_K(x) = \left( f^*(x) - \mathbb{E}_X \tilde{f}_{S,K}(x) \right)^2 + \mathbb{E}_S ( \tilde{f}_{S,K}(x) - \hat{f}_{S,K}(x) )^2 + \sigma^2 \]

bias

variance

...
Bias Variance trade-off for K-NN

Define the *noisyless* K-NN (it is ideal!)

\[ \tilde{f}_{S,K}(x) = \frac{1}{K} \sum_{l \in K_x} f_*(x_l) \]

Note that \( \tilde{f}_{S,K}(x) = \mathbb{E}_{y|x} \hat{f}_{S,K}(x) \).

Consider

\[ \mathcal{E}_K(x) = (f_*(x) - \mathbb{E}_X \tilde{f}_{S,K}(x))^2 + \frac{1}{K^2} \mathbb{E}_X \sum_{l \in K_x} \mathbb{E}_{y_l|x_l} (y_l - f_*(x_l))^2 + \sigma^2 \]

...
Define the *noisyless* K-NN (it is ideal!)

\[
\hat{f}_{S,K}(x) = \frac{1}{K} \sum_{l \in K_x} f_*(x_l)
\]

Note that \(\hat{f}_{S,K}(x) = \mathbb{E}_{y|x} \hat{f}_{S,K}(x)\).

Consider

\[
\mathcal{E}_K(x) = (f_*(x) - \mathbb{E}_X \hat{f}_{S,K}(x))^2 + \frac{\sigma^2}{K} + \sigma^2
\]

...
Bias Variance trade-off

Errors

Variance

Bias

K
Bias-Variance trade-off is theoretical, but shows that:

▶ an optimal parameter exists and
▶ it depends on the noise and the unknown target function.

How to choose $K$ in practice?
▶ Idea: train on some data and validate the parameter on new unseen data as a proxy for the ideal case.
Bias-Variance trade-off is theoretical, but shows that:

- an optimal parameter exists and

How to choose \( K \) in practice?

Idea: train on some data and validate the parameter on new unseen data as a proxy for the ideal case.
How to choose the hyper-parameters

Bias-Variance trade-off is theoretical, but shows that:

▶ an optimal parameter exists and
▶ it depends on the noise and the unknown target function.
How to choose the hyper-parameters

Bias-Variance trade-off is theoretical, but shows that:

▶ an optimal parameter exists and
▶ it depends on the noise and the unknown target function.

How to choose $K$ in practice?
How to choose the hyper-parameters

Bias-Variance trade-off is theoretical, but shows that:

▶ an optimal parameter exists and
▶ it depends on the noise and the unknown target function.

How to choose $K$ in practice?

▶ Idea: train on some data and validate the parameter on new unseen data as a proxy for the ideal case.
Hold-out Cross-validation
Hold-out Cross-validation

For each $K$

1. shuffle and split $S$ in $T$ (training) and $V$ (validation)
2. train the algorithm on $T$ and compute the empirical loss on $V$

$$
\hat{E}_K = \frac{1}{|V|} \sum_{x,y \in V} (y - \hat{f}_{T,K}(x))^2
$$

3. Select $\hat{K}$ that minimize $\hat{E}_K$.

The above procedure can be repeated to augment stability and $K$ selected to minimize error over trials.

There are other related parameter selection methods (k-fold cross validation, leave-one out...).
Hold-out Cross-validation

For each $K$

1. shuffle and split $S$ in $T$ (training) and $V$ (validation)
Hold-out Cross-validation

For each $K$

1. shuffle and split $S$ in $T$ (training) and $V$ (validation)

2. train the algorithm on $T$ and compute the empirical loss on $V$

$$
\hat{E}_K = \frac{1}{|V|} \sum_{x,y \in V} (y - \hat{f}_{T,K}(x))^2
$$

The above procedure can be repeated to augment stability and $K$ selected to minimize error over trials.

There are other related parameter selection methods (k-fold cross validation, leave-one out...).
For each $K$

1. shuffle and split $S$ in $T$ (training) and $V$ (validation)

2. train the algorithm on $T$ and compute the empirical loss on $V$
\[
\hat{E}_K = \frac{1}{|V|} \sum_{x,y \in V} (y - \hat{f}_{T,K}(x))^2
\]

Select $\hat{K}$ that minimize $\hat{E}_K$.

The above procedure can be repeated to augment stability and $K$ selected to minimize error over trials.

There are other related parameter selection methods (k-fold cross validation, leave-one out...).
Hold-out Cross-validation

For each $K$

1. shuffle and split $S$ in $T$ (training) and $V$ (validation)
2. train the algorithm on $T$ and compute the empirical loss on $V$
   $$\hat{E}_K = \frac{1}{|V|} \sum_{x,y \in V} (y - \hat{f}_{T,K}(x))^2$$
3. Select $\hat{K}$ that minimize $\hat{E}_K$. 

The above procedure can be repeated to augment stability and $K$ selected to minimize error over trials. There are other related parameter selection methods (k-fold cross validation, leave-one out...).
Hold-out Cross-validation

For each $K$

1. shuffle and split $S$ in $T$ (training) and $V$ (validation)
2. train the algorithm on $T$ and compute the empirical loss on $V$
   \[ \hat{E}_K = \frac{1}{|V|} \sum_{x,y \in V} (y - \hat{f}_{T,K}(x))^2 \]
3. Select $\hat{K}$ that minimize $\hat{E}_K$.

The above procedure can be repeated to augment stability and $K$ selected to minimize error over trials.
Hold-out Cross-validation

For each $K$

1. shuffle and split $S$ in $T$ (training) and $V$ (validation)
2. train the algorithm on $T$ and compute the empirical loss on $V$
\[
\hat{E}_K = \frac{1}{|V|} \sum_{x,y \in V} (y - \hat{f}_{T,K}(x))^2
\]
3. Select $\hat{K}$ that minimize $\hat{E}_K$.

The above procedure can be repeated to augment stability and $K$ selected to minimize error over trials.

There are other related parameter selection methods (k-fold cross validation, leave-one out...).
V-Fold Cross-validation
Training and Validation Error behavior

Training vs. validation error
$T = 50\% \, S; \, n = 200$

$\hat{K} = 8$.
Training and Validation Error behavior

$\hat{K} = 8$. 
Wrapping up

In this class we made our first encounter with learning algorithms (local methods) and the problem of tuning their parameters (via bias-variance trade-off and cross-validation) to avoid overfitting and achieve generalization.
Next Class

High Dimensions: Beyond local methods!