MLCC 2019
Local Methods and Bias Variance Trade-Off

Lorenzo Rosasco
UNIGE-MIT-IIT

About this class

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?

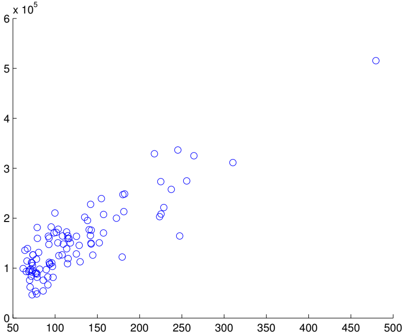
The problem

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

The problem

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

Area (m^2)	Price (€)
$x_1 = 62$	$y_1 = 99,200$
$x_2 = 64$	$y_2 = 135,700$
$x_3 = 65$	$y_3 = 93,300$
$x_4 = 66$	$y_4 = 114,000$
$\vdots$	$\vdots$



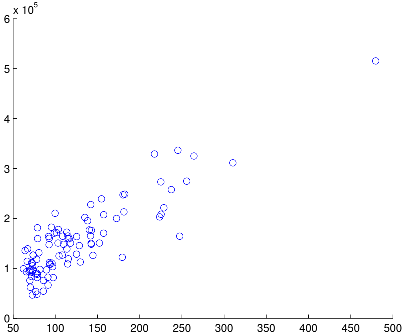
Let S the houses example dataset ($n = 100$)

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

The problem

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

Area (m^2)	Price (€)
$x_1 = 62$	$y_1 = 99,200$
$x_2 = 64$	$y_2 = 135,700$
$x_3 = 65$	$y_3 = 93,300$
$x_4 = 66$	$y_4 = 114,000$
$\vdots$	$\vdots$



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.

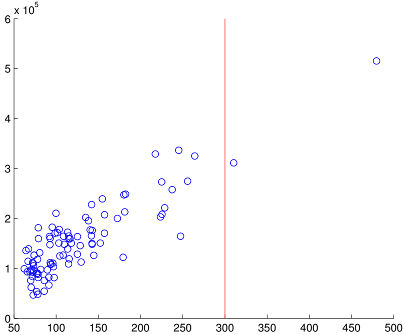
Example

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

Example

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

Area (m^2)	Price (€)
$\vdots$	$\vdots$
$x_{93} = 255$	$y_{93} = 274,600$
$x_{94} = 264$	$y_{94} = 324,900$
$x_{95} = 310$	$y_{95} = 311,200$
$x_{96} = 480$	$y_{96} = 515,400$
$\vdots$	$\vdots$



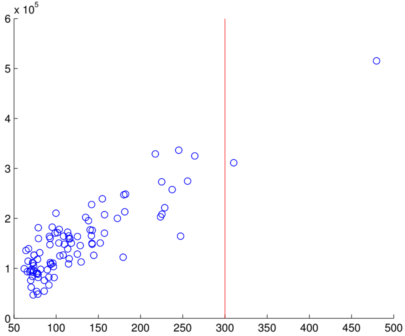
What is its price?

Nearest Neighbors

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

$$y^* = 311,200$$

Area (m^2)	Price (€)
$\vdots$	$\vdots$
$x_{93} = 255$	$y_{93} = 274,600$
$x_{94} = 264$	$y_{94} = 324,900$
$x_{95} = 310$	$y_{95} = 311,200$
$x_{96} = 480$	$y_{96} = 515,400$
$\vdots$	$\vdots$

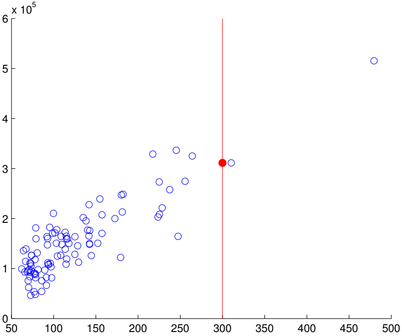


Nearest Neighbors

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

$$y^* = 311,200$$

Area (m^2)	Price (€)
$\vdots$	$\vdots$
$x_{93} = 255$	$y_{93} = 274,600$
$x_{94} = 264$	$y_{94} = 324,900$
$x_{95} = 310$	$y_{95} = 311,200$
$x_{96} = 480$	$y_{96} = 515,400$
$\vdots$	$\vdots$



Nearest Neighbors

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

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

$$y_* = y_j \quad j = \arg \min_{i=1,\dots,n} \|x - x_i\|$$

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

$$y_* = y_j \quad j = \arg \min_{i=1,\dots,n} \|x - x_i\|$$

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

$$y_* = y_j \quad j = \arg \min_{i=1,\dots,n} \|x - x_i\|$$

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

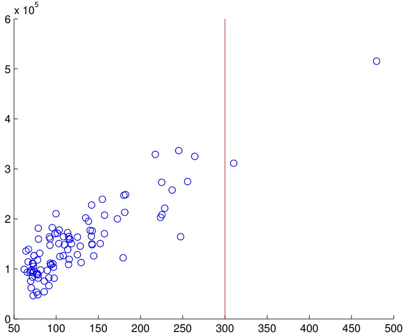
In general let $d : \mathbb{R}^D \times \mathbb{R}^D$ a distance on the input space, then

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

Extensions

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

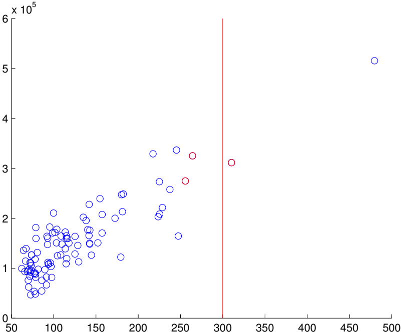
Area (m^2)	Price (€)
$\vdots$	$\vdots$
$x_{93} = 255$	$y_{93} = 274,600$
$x_{94} = 264$	$y_{94} = 324,900$
$x_{95} = 310$	$y_{95} = 311,200$
$x_{96} = 480$	$y_{96} = 515,400$
$\vdots$	$\vdots$



Extensions

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

Area (m^2)	Price (€)
$\vdots$	$\vdots$
$x_{93} = 255$	$y_{93} = 274,600$
$x_{94} = 264$	$y_{94} = 324,900$
$x_{95} = 310$	$y_{95} = 311,200$
$x_{96} = 480$	$y_{96} = 515,400$
$\vdots$	$\vdots$



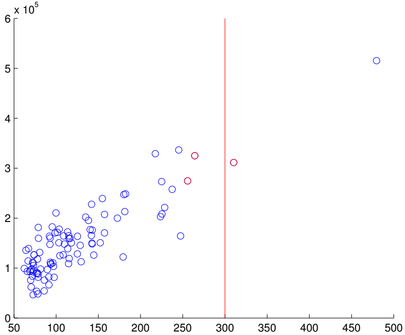
Can we do better? (for example using more points)

K-Nearest Neighbors

K-Nearest Neighbor: y^* is the mean of the values of the K closest point to x^* in S. If $K = 3$ we have

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

Area (m^2)	Price (€)
$\vdots$	$\vdots$
$x_{93} = 255$	$y_{93} = 274,600$
$x_{94} = 264$	$y_{94} = 324,900$
$x_{95} = 310$	$y_{95} = 311,200$
$x_{96} = 480$	$y_{96} = 515,400$
$\vdots$	$\vdots$

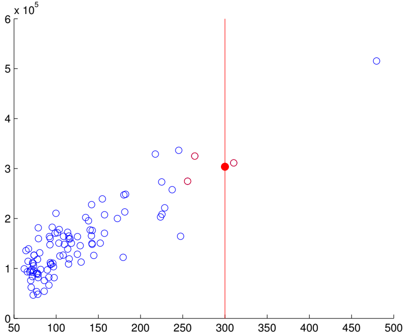


K-Nearest Neighbors

K-Nearest Neighbor: y^* is the mean of the values of the K closest point to x^* in S. If $K=3$ we have

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

Area (m^2)	Price (€)
$\vdots$	$\vdots$
$x_{93} = 255$	$y_{93} = 274,600$
$x_{94} = 264$	$y_{94} = 324,900$
$x_{95} = 310$	$y_{95} = 311,200$
$x_{96} = 480$	$y_{96} = 515,400$
$\vdots$	$\vdots$



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 $x^* \in \mathbb{R}^D$,

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 $K << n$,

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 $K << n$,
- $j_1, \dots, j_K$ defined as $j_1 = \arg\min_{i \in \{1,\dots,n\}} \|x^* - x_i\|$ and $j_t = \arg\min_{i \in \{1,\dots,n\} \setminus \{j_1,\dots,j_{t-1}\}} \|x^* - x_i\|$ for $t \in \{2, \dots, K\}$,

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 $K << n$,
- $j_1, \dots, j_K$ defined as $j_1 = \arg\min_{i \in \{1,\dots,n\}} \|x^* - x_i\|$ and $j_t = \arg\min_{i \in \{1,\dots,n\} \setminus \{j_1,\dots,j_{t-1}\}} \|x^* - x_i\|$ for $t \in \{2, \dots, K\}$,
- predicted output

$$y_* = \frac{1}{K} \sum_{i \in \{j_1,\dots,j_K\}} y_i$$

K-Nearest Neighbors (cont.)

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

K-Nearest Neighbors (cont.)

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

- **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 $O(n \log n)$.

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 $i = \{1, \dots, n\}$ (each costs $O(D)$). Order them $O(n \log n)$.

- **General Metric** d f is the same, but $j_1, \dots, j_K$ are defined as $j_1 = \arg\min_{i \in \{1,\dots,n\}} d(x, x_i)$ and $j_t = \arg\min_{i \in \{1,\dots,n\} \setminus \{j_1,\dots,j_{t-1}\}} d(x, x_i)$ for $t \in \{2, \dots, K\}$

Parzen Windows

K-NN puts equal weights on the values of the selected points.

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:

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

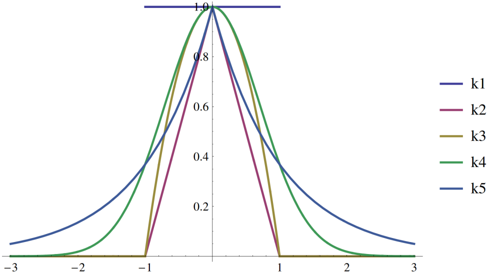
where k is a *similarity function*

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

Parzen Windows

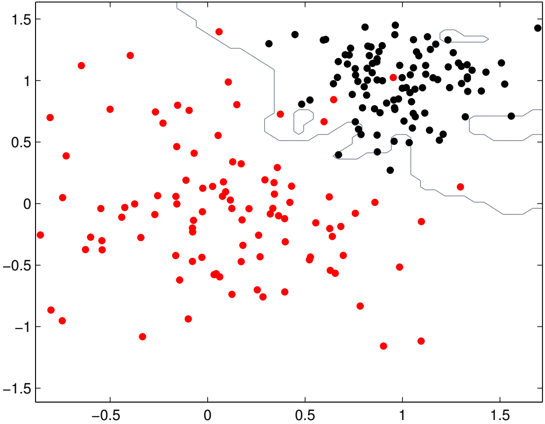
Examples of k

- $k_1(x,x') = \operatorname{sign}\left(1 - \frac{\|x-x'\|}{\sigma}\right)_+$ 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 $\sigma > 0$



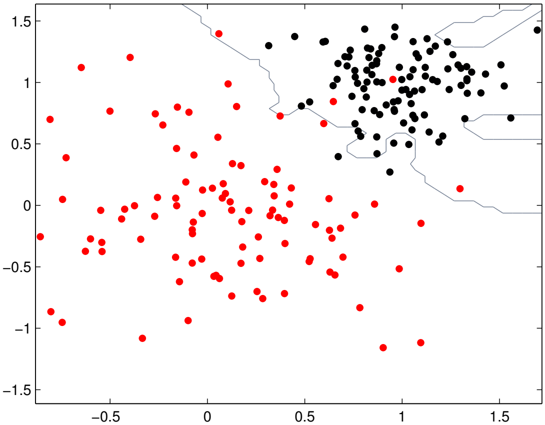
K-NN example

K-Nearest neighbor depends on K.
When $K = 1$



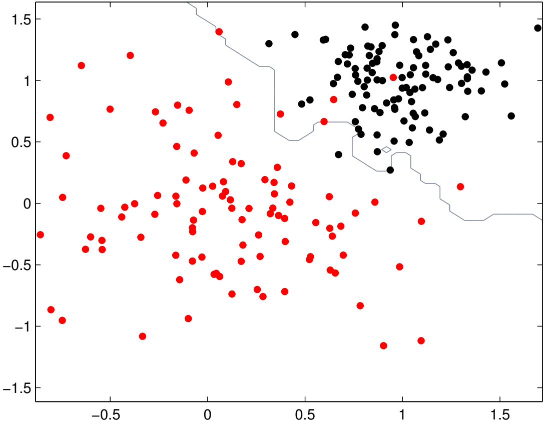
K-NN example

K-Nearest neighbor depends on K.
When $K = 2$



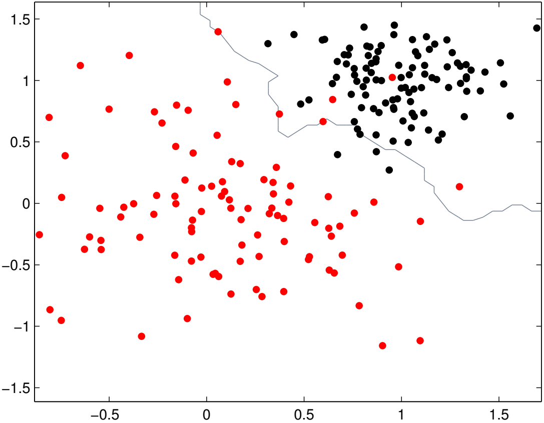
K-NN example

K-Nearest neighbor depends on K.
When $K = 3$



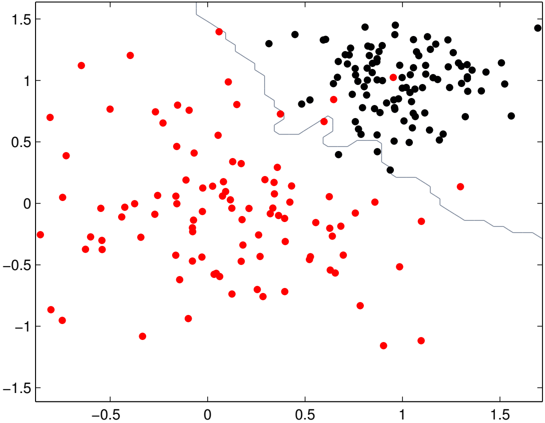
K-NN example

K-Nearest neighbor depends on K.
When $K = 4$



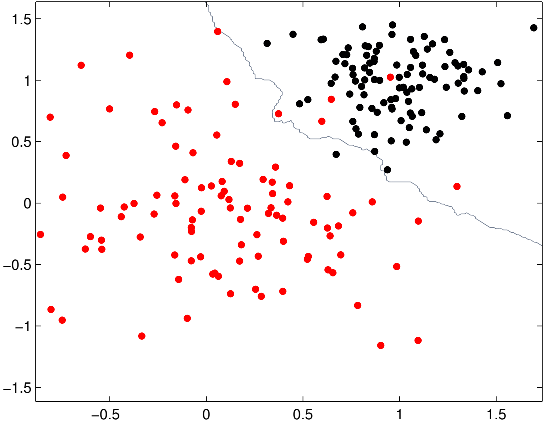
K-NN example

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



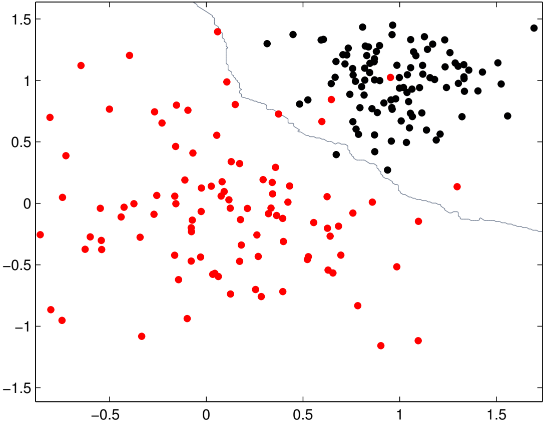
K-NN example

K-Nearest neighbor depends on K.
When $K = 9$



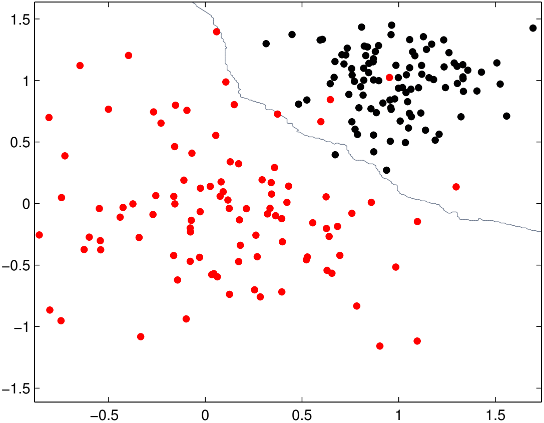
K-NN example

K-Nearest neighbor depends on K.
When $K = 15$



K-NN example

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 $Y = (y_1, \dots, y_n)$ and $X = (x_1^\top, \dots, x_n^\top)$.
- $K \in \mathbb{N}$ hyperparameter of the learning algorithm
- $\hat{f}_{S,K}$ learned function (depends on S and 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 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 $Y = (y_1, \dots, y_n)$ and $X = (x_1^\top, \dots, x_n^\top)$.
- $K \in \mathbb{N}$ 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 $\mathcal{E}_K$

Optimal choice for the Hyper-parameters (cont.)

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 $\mathcal{E}_K$

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

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

- We can still try to understand the above minimization problem: does a solution exists? 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*

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

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

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 $y = f_*(x) + \delta$

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

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

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

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

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

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

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

...

Bias Variance trade-off for K-NN

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

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

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

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

$$\mathcal{E}_K(x) = \underbrace{(f_*(x) - \mathbb{E}_X \tilde{f}_{S,K}(x))^2}_{\text{bias}} + \underbrace{\mathbb{E}_S (\tilde{f}_{S,K}(x) - \hat{f}_{S,K}(x))^2 + \sigma^2}_{\text{variance}}$$

...

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

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

...

Bias Variance trade-off for K-NN

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

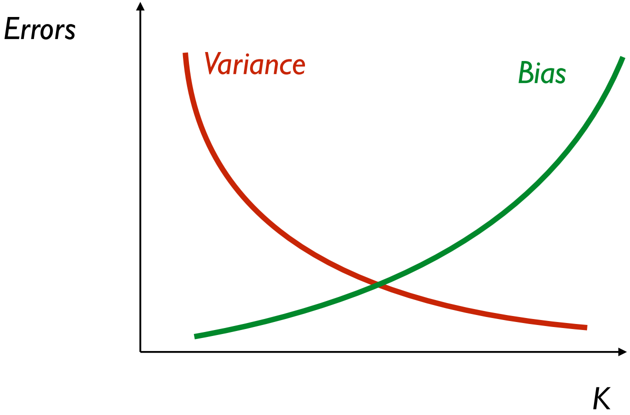
$$\tilde{f}_{S,K}(x) = \frac{1}{K} \sum_{l \in K_x} f_*(x_l)$$

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

$$\mathcal{E}_K(x) = \underbrace{(f_*(x) - \mathbb{E}_X \tilde{f}_{S,K}(x))^2}_{\text{bias}} + \underbrace{\frac{\sigma^2}{K} + \sigma^2}_{\text{variance}}$$

...

Bias Variance trade-off



How to choose the hyper-parameters

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

How to choose the hyper-parameters

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

- an optimal parameter exists and

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

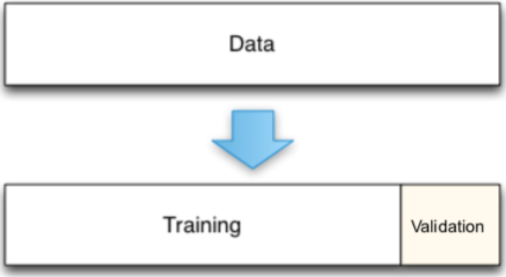
Bias-Variance trade-off is theoretical, but shows that:

- 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

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

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

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

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

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

Hold-out Cross-validation

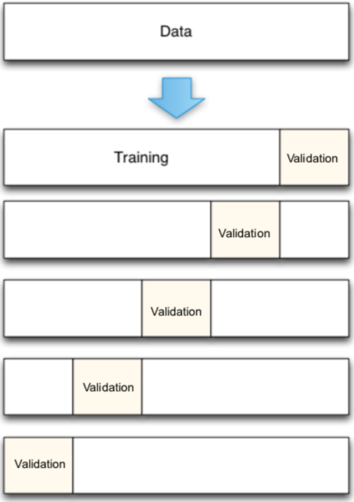
For each K

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

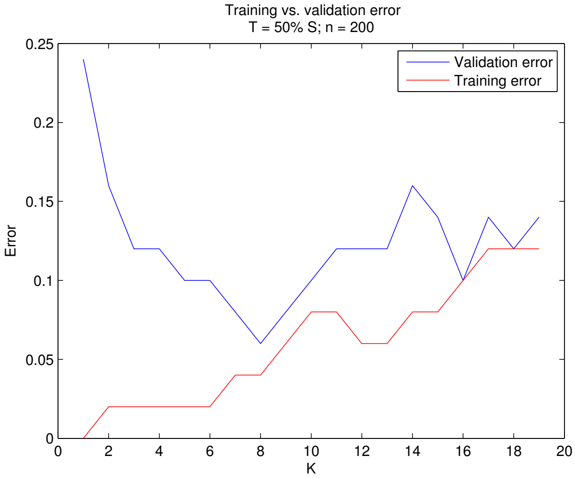
The above procedure can be repeated to augment stability and K selected to minimize error over trials.

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

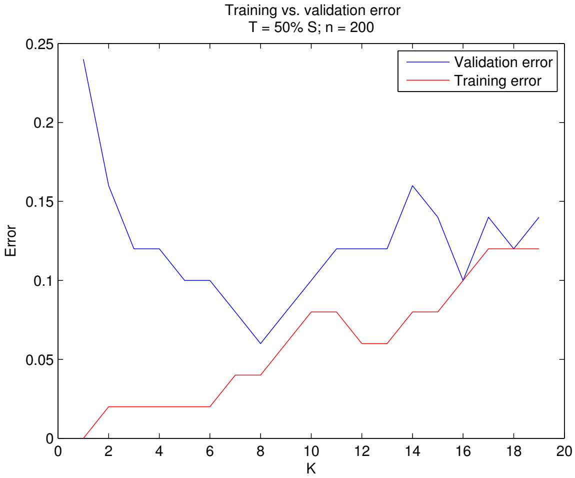
V-Fold Cross-validation



Training and Validation Error behavior



Training and Validation Error behavior



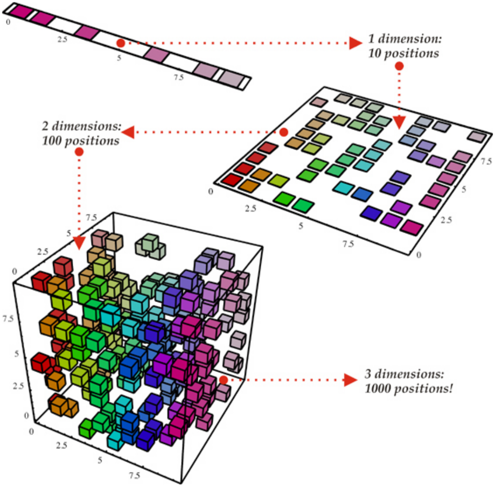
$\hat{K} = 8.$

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!


1 dimension:
10 positions
2 dimensions:
100 positions
3 dimensions:
1000 positions!
0
2.5
5
7.5