



# The Bias-Complexity Trade-Off

Machine Learning 2023-24

UML Book Chapter 5

Slides P. Zanuttigh (some material F. Vandin)

# Recall: Agnostic PAC Learnability

- ❑ Idea: We drop the requirement of finding a perfect predictor
  - but we do not want to be too far from the best predictor
- ❑ Recall definition of *Agnostic PAC Learnability*:

*A hypothesis class  $\mathcal{H}$  is **agnostic** PAC learnable if there exist a function  $m_{\mathcal{H}}: (0,1)^2 \rightarrow \mathbb{N}$  and a learning algorithm such that **for every**  $\delta, \epsilon \in (0,1)$  and **for every** distribution  $D$  over  $\mathcal{X} \times \mathcal{Y}$ , when running the algorithm on  $m \geq m_{\mathcal{H}}(\epsilon, \delta)$  i.i.d. examples generated by  $D$  the algorithm returns a hypothesis  **$h$**  such that, with probability  $\geq 1 - \delta$  (over the choice of the  $m$  training examples):*

$$L_D(h) \leq \min_{h' \in \mathcal{H}} L_D(h') + \epsilon$$

# A Universal Learner ?

Given a training set  $S$  and a loss function we'd like to find a function  $\hat{h}$  for which  $L_d(\hat{h})$  is small

Pick a learning algorithm  $A$  that given  $S$  produces function  $\hat{h}$

It depends on two components:

1. the hypothesis set  $\mathcal{H}$
2. the procedure to pick  $\hat{h}$  from  $\mathcal{H}$

*Is there a universal learner, i.e., an algorithm  $A$  that predicts the best  $\hat{h}$  for any distribution  $D$  ?*

*What about using the set of **all** functions from  $\mathcal{X}$  to  $\mathcal{Y}$  as the hypothesis class ?*

# No Free Lunch Theorem

## *Theorem (No-Free Lunch)*

Let  $A$  be any learning algorithm for the task of binary classification with respect to the 0-1 loss over a domain  $\mathcal{X}$ . Let  $m$  be any number smaller than  $\frac{|\mathcal{X}|}{2}$ , representing a training set size. Then there exist a distribution  $D$  over  $\mathcal{X} \times \{0,1\}$  such that:

1. There exist a function  $f: \mathcal{X} \rightarrow \{0,1\}$  with  $L_D(f) = 0$
2. With probability of at least  $1/7$  over the choice of  $S \sim D^m$  we have that  $L_D(A(S)) \geq \frac{1}{8}$

## *Corollary (No-Free Lunch)*

Let  $\mathcal{X}$  be an infinite domain set and let  $H$  be the set of all functions from  $\mathcal{X}$  to  $\{0,1\}$ . Then,  $H$  is not PAC learnable

# No Free Lunch: Notes

- ❑ *Key message*: for **every** ML algorithm there exist a task on which it fails even if another ML algorithm is able to solve it
- ❑ *Idea of the proof*: our training set is smaller than half of the domain  
→ no information on what happens on the other half → there exist some target function  $f$  (i.e., target problem/task) that works on the other half in a way that contradicts our estimated labels
  - *Full demonstration not part of the course*
- ❑ Class  $\mathcal{H}$  of all possible functions from  $\mathcal{X}$  to  $\{0,1\}$  (i.e., assuming **no prior knowledge**) is not a good idea
  - It is not PAC learnable

# Corollary: Demonstration

Proceed *by contradiction*

## Corollary (No-Free Lunch)

Let  $\mathcal{X}$  be an infinite domain set and let  $\mathcal{H}$  be the set of all functions from  $\mathcal{X}$  to  $\{0,1\}$ . Then,  $\mathcal{H}$  is not PAC learnable

1. Assume PAC Learnable

- Choose  $\epsilon < \frac{1}{8}$ ,  $\delta < \frac{1}{7}$ , and recall that  $\mathcal{H}$  includes all functions (\*)
- By definition of PAC: there exists an algorithm  $A$  and such that for every distribution  $D$ , if realizable\* (all functions: include any labeling!), when running the algorithm on  $m$  i.i.d. examples generated by  $D$  the algorithm returns a hypothesis  $h$  such that, with probability  $\geq 1 - \delta$ ,  $L_D(A(S)) \leq \epsilon$

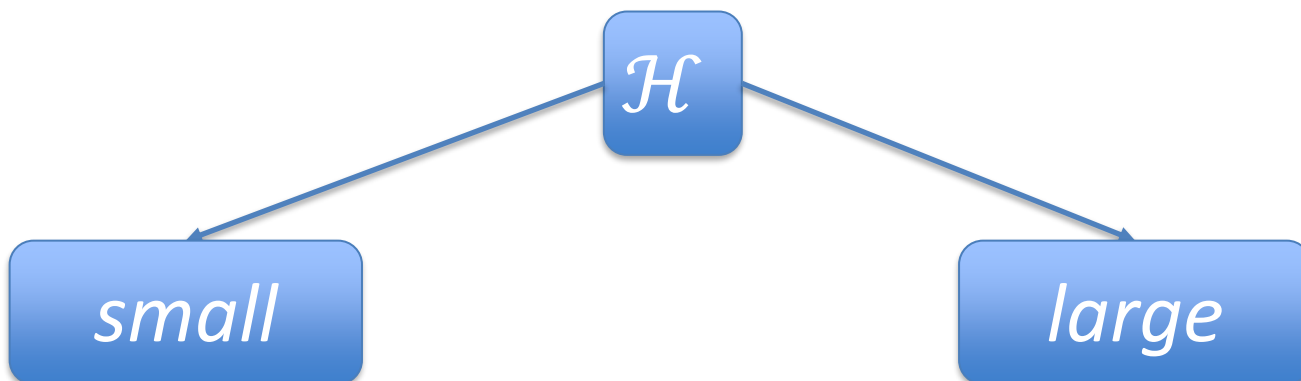
2. Apply No-Free Lunch theorem to  $A$

- $|\mathcal{X}| > 2m$  (it is  $\infty$ ): for any ML algorithm (including  $A$ !) there exist a distribution  $D$  for which with probability  $\geq \frac{1}{7} > \delta$ ,  $L_D(A_S) \geq \frac{1}{8} > \epsilon$

3. The two blue results are in contradiction (sum of probabilities is bigger than one!)



# Choose a Good Hypothesis Set



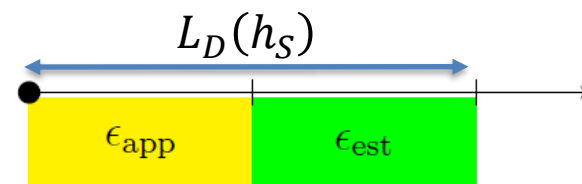
Low approximation capabilities (large  $L_S$ )  
Good generalization properties ( $L_D \sim L_S$ )

Good approximation capabilities (small  $L_S$ )  
Risk of overfitting ( $L_D \gg L_S$ )

- We need to use our prior knowledge about  $D$  to pick a good hypothesis set
- We would like  $\mathcal{H}$  to be large, so that it may contain a function  $h$  with small  $L_S(h)$  and hopefully a small  $L_D(h)$
- No free lunch:  $\mathcal{H}$  cannot be too large!  
→ A too large  $\mathcal{H}$  leads to the risk of overfitting

# Error Decomposition (1)

Consider an  $ERM_{\mathcal{H}}$  hypothesis  $h_S$  :



The true error of  $ERM_{\mathcal{H}}$  can be decomposed as:

$$L_D(h_S) = \epsilon_{app} + \epsilon_{est}$$

$$\epsilon_{app} = \min_{h \in \mathcal{H}} L_D(h)$$

## Approximation error

- Minimum true risk achievable by a predictor in  $\mathcal{H}$
- Depends on the choice of  $\mathcal{H}$  (*but not on  $S$* )
- Once  $\mathcal{H}$  is selected it is fixed
- Larger  $\mathcal{H} \rightarrow$  smaller  $\epsilon_{app}$
- $\epsilon_{app} = 0$  if realizability holds
- Otherwise bounded by Bayes predictor

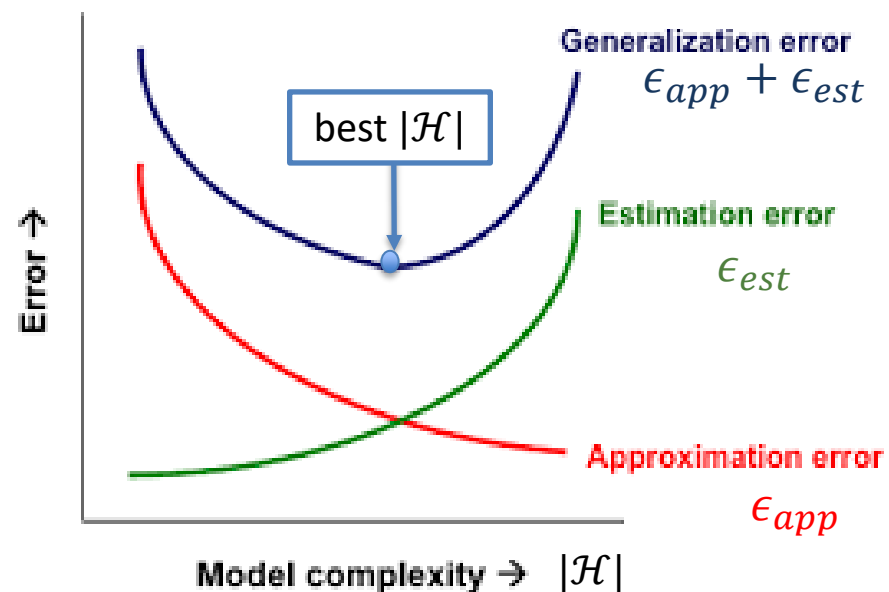
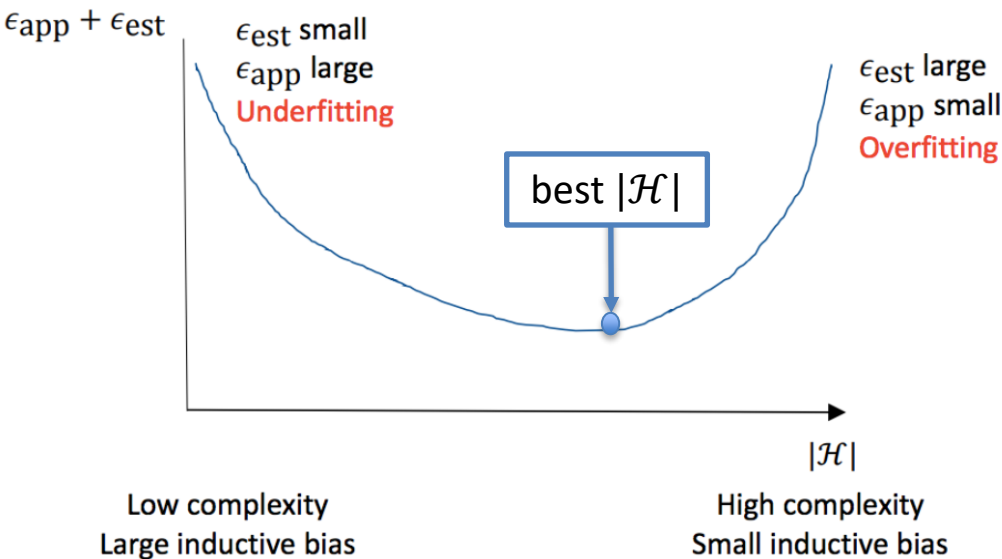
$$\epsilon_{est} = L_D(h_S) - \min_{h \in \mathcal{H}} L_D(h)$$

## Estimation error

- Difference between true error of ERM predictor and approximation error
- Due to the not-optimal ML algorithm not able to find the best  $h^*$  using ERM
- *Depends on  $S$*  (typically smaller for larger  $S$ )
- To decrease we need a smaller  $\mathcal{H}$  so the training error is a good estimate of true error

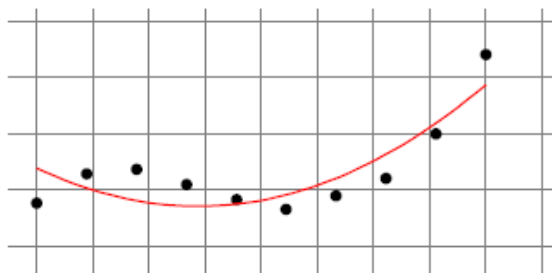


# Error Decomposition (2)



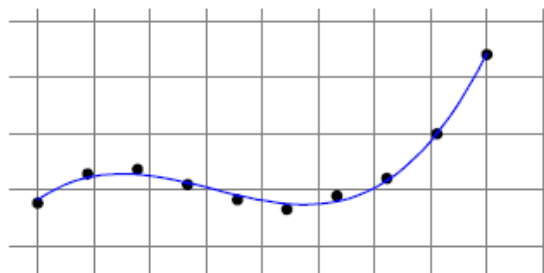
# Example: Polynomial Fitting

A) degree 2



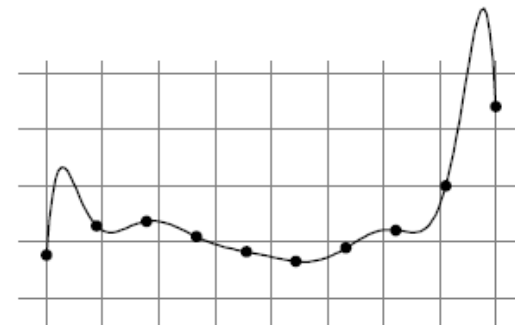
A) *underfitting*

B) degree 3



B) *good approximation*

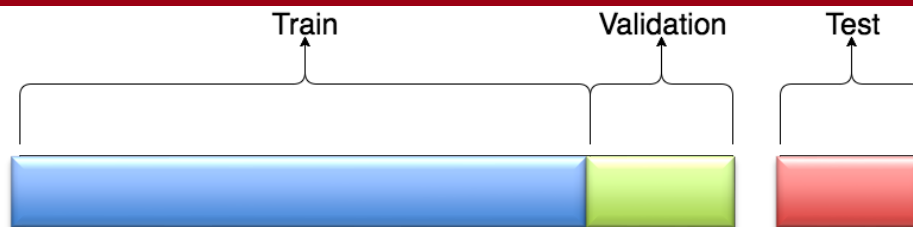
C) degree 10



C) *overfitting*

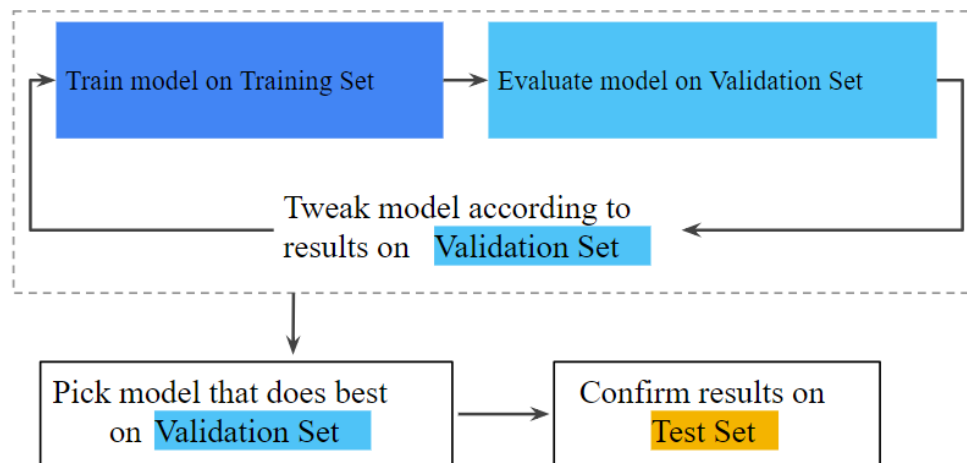
- A. Degree 2: **large**  $\epsilon_{app}$  , **small**  $\epsilon_{est}$  (underfitting)
- B. Degree 10:  $\epsilon_{app} = 0$  , **large**  $\epsilon_{est}$  (overfitting)
- C. Degree 3: good compromise (*best solution ?*)

# Train, Test and Validation Sets



- ❑ We need to **estimate** the generalization error  $L_D(h)$  for a function  $h$  (e.g., the one we selected with ERM)
- ❑ Use a **test set**: a new set of samples not used for picking  $h$ 
  - It must be different (disjoint) from the **training set**
  - More reliable estimation of  $L_D(h)$  (*but still an estimation!!*)
  - The test must not be looked at until we have picked our final hypothesis !
  - In practice: we have 1 set of samples and we split it in **training set** and **test set**
- ❑ Sometimes the training set is further divided into a **training set** and a **validation set**
  - The **validation set** is used for selecting the hyper-parameters of the algorithm
  - Can be used to evaluate error while iterative training procedures are running

# Iterative Training Procedure with Validation Set

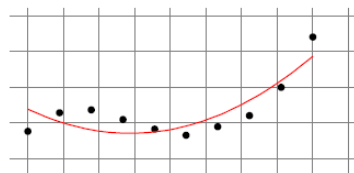


Train a ML algorithm parametrized by some Hyper-Parameters (HP)

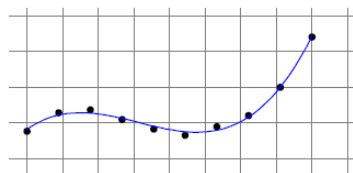
1. Select Hyper-Parameters values
2. Train on the **training set**
3. Evaluate performances on the **validation set**
4. Go back to 1 (select new HP values)
5. Select HP leading to smallest **validation error**
6. Compute error estimation on the **test set**

# Example (with validation set)

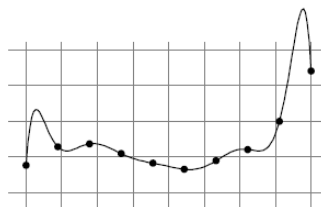
degree 2



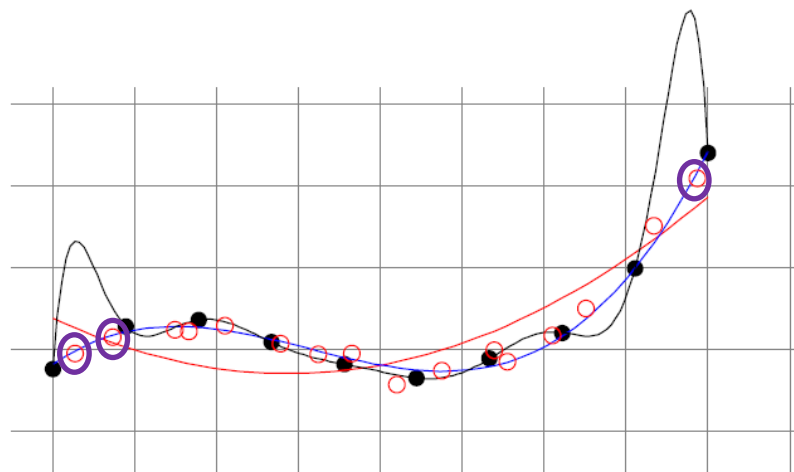
degree 3



degree 10



- ❑ **Training** set  $S$  : black circles
- ❑ Hyper-parameter = degree of the model (2, 3 or 10)
- ❑ Perform ERM minimization over **training** set  $S$ :
  - Degree 10 is the best solution ( $L_S(h_S) = 0$ ) !



- ❑ Introduce a **validation set** (red circles)
- ❑ Train on **training** set for each of the 3 HP values (2,3 and 10)
- ❑ Select solution with lower error on **validation** set
  - Degree 3 is the best solution
  - Degree 10 has low error on **training** set but high on **validation** set

# Occam's Razor



**“All things being  
equal, the simplest  
solution tends to be  
the best one.”**

**William of Ockham**  
(1287-1347)

*A short explanation (that is, a hypothesis that has a short length)  
tends to be more valid than a long explanation*