

# Machine learning fundamentals: Lecture 1

Introduction

Feasibility of learning: concepts

Bias - Variance tradeoff

---

Mirko Mazzoleni - Identificazione dei Modelli e Analisi dei Dati (IMAD)

October 2018

University of Bergamo

Department of Management, Information and Production Engineering

*mirko.mazzoleni@unibg.it*

# Resources

These lectures give only a glimpse of the vast machine learning field. Additional material (not required for the exam) can be found using the following resources

## MOOCs

- Learning from data  
(Yaser S. Abu-Mostafa - EDX)
- Machine learning (Andrew Ng - Coursera)
- Deep learning (Andrew Ng - Coursera)
- The analytics edge  
(Dimitris Bertsimas - EDX)
- Statistical learning (Trevor Hastie and Robert Tibshirani - Stanford Lagunita)

## Books

- An Introduction to Statistical Learning, with application in R (Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani)
- Neural Networks and Deep Learning  
(Michael Nielsen)
- Pattern Recognition and Machine Learning (Christopher Bishop)

# Outline

- Introduction
- Components of learning
- Puzzle
- Feasibility of learning
- Bias - variance tradeoff

# Outline

- **Introduction**
- Components of learning
- Puzzle
- Feasibility of learning
- Bias - variance tradeoff

# Why

Machine learning and data science have been deemed as the sexiest jobs of the 21th century

- Virtually every aspect of business is now open to **data** collection
- Collected information need to be analyzed properly in order to get **actionable** results
- A huge amount of data requires specific **infrastructures** to be handled
- A huge amount of data requires **computational** power to be analyzed
- We can let computers to perform decisions given **previous examples**
- Rising of specific job titles
- ... Fun 😊

# Learning examples

Recent years: stunning breakthroughs in computer vision applications



## Learning examples



## Learning examples





## Learning examples

Zebras  $\leftrightarrow$  Horses



## Learning examples

- Spam e-mail detection system
- Credit approval
- Recognize objects in images
- Find the relation between house prices and house sizes
- Predict the stock market
- Market segmentation
- Market basket analysis
- Language models (word2vec)
- Social network analysis
- Movies recommendation
- Low-order data representations

# What learning is about

Machine learning is meaningful to be applied if:

1. A pattern exists
2. We cannot pin it down mathematically
3. We have data on it

Assumption 1. and 2. are not mandatory:

- If a pattern does not exist, I do not learn anything
- If I can describe the mathematical relation, I will not presumably learn the best function
- The real constraint is assumption 3

# Outline

- Introduction
- **Components of learning**
- Puzzle
- Feasibility of learning
- Bias - variance tradeoff

# Components of learning

## Formalization:

- Input:  $\varphi$  (*e-mail textual content*)  $\rightarrow$  each dimension is some e-mail attribute
- Output:  $y$  (*spam/not spam?*)  $\rightarrow$  the decision that we have to take in the end
- Target function:  $f : \mathcal{X} \rightarrow \mathcal{Y}$  (*Ideal spam filter formula*)  $\rightarrow$  unknown, we have to learn it
- Data:  $\mathcal{D} = \{\varphi(1), y(1)\}, \dots, \{\varphi(N), y(N)\}$  (*historical records of e-mail examples*)

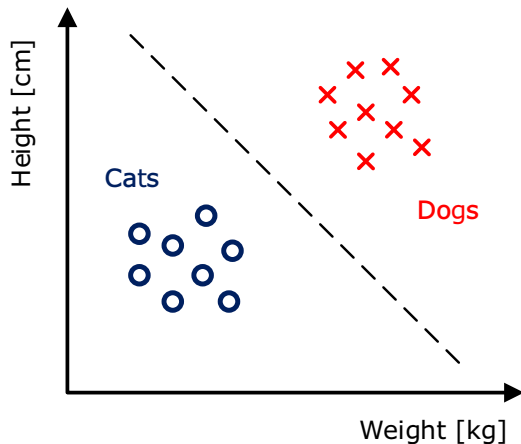
$\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis:  $g : \mathcal{X} \rightarrow \mathcal{Y}, g \in \mathcal{H}$  (*formula to be used*)  $\rightarrow g$  is an approximation of  $f$

$\mathcal{H}$  is called the **Hypothesis space**. This, together with the **Learning algorithm**, form the **learning model**

# Supervised learning

- The “correct answer”  $y$  is given
- Predict  $y$  from a set of inputs  
 $\varphi \in \mathbb{R}^{(d-1) \times 1}$
- Regression: predict continuous output  
 $y \in \mathbb{R}$  (real value)
- Classification: predict discrete categorical output  $y \in \{1, \dots, C\}$   
(class)



## Example: House prices regression

Suppose we want to find a linear function which relates the **measured** regressors  $x_1, x_2, \dots, x_{d-1}$  with the **observed** output  $y$

| Size [feet <sup>2</sup> ] | Number of bedrooms | Number of floors | Age of home [year] | Price [\$]        |
|---------------------------|--------------------|------------------|--------------------|-------------------|
| 2104                      | 5                  | 1                | 45                 | $4.60 \cdot 10^5$ |
| 1416                      | 3                  | 2                | 40                 | $2.32 \cdot 10^5$ |
| 1534                      | 2                  | 1                | 30                 | $3.15 \cdot 10^5$ |
| $\vdots$                  | $\vdots$           | $\vdots$         | $\vdots$           | $\vdots$          |
| $\downarrow$              | $\downarrow$       | $\downarrow$     | $\downarrow$       | $\downarrow$      |
| $x_1$                     | $x_2$              | $x_3$            | $x_4$              | $y$               |

- The number of rows is the number of data points  $N$
- The  $i$ th observation is the vector  $\varphi(i) = [x_1(i) \ x_2(i) \ x_3(i) \ x_4(i)]^T \in \mathbb{R}^{4 \times 1}$
- Each feature vector  $\varphi$  has associated a response  $y \in \mathbb{R}$  that we want to predict for new observations of  $\varphi$

## Example: House prices classification

The components of the features vector are the same. The difference lies in the response variable, which now is a **class** (categorical data type) and not a real value

Suppose that instead of the price value in dollars, we want to classify houses as *expensive* (class  $y = 1$ ) or *cheap* (class  $y = 0$ )

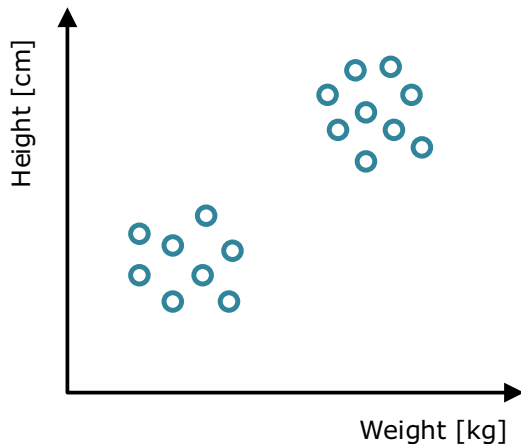
| Size [feet <sup>2</sup> ] | Number of bedrooms | Number of floors | Age of home [year] | Price [class] |
|---------------------------|--------------------|------------------|--------------------|---------------|
| 2104                      | 5                  | 1                | 45                 | 1             |
| 1416                      | 3                  | 2                | 40                 | 0             |
| 1534                      | 2                  | 1                | 30                 | 1             |
| $\vdots$                  | $\vdots$           | $\vdots$         | $\vdots$           | $\vdots$      |
| $\downarrow$              | $\downarrow$       | $\downarrow$     | $\downarrow$       | $\downarrow$  |
| $x_1$                     | $x_2$              | $x_3$            | $x_4$              | $y$           |

The point  $\varphi$  can then be classified to class  $y = 1$  if  $s([1 \ \varphi]) \geq 0.5$ , where  $s(\cdot)$  is the logistic function. This happens when  $[1 \ \varphi]^T \cdot \vartheta \geq 0$  (linear classifier)



# Unsupervised learning

- Instead of **(input, output)** we get **(input, ?)**
- Find properties of the inputs  
 $\varphi \in \mathbb{R}^{(d-1) \times 1}$
- High-level representation of the input
- Elements into the same cluster have similar properties



# Reinforcement learning

- Instead of **(input, output)** we get **(input, output, reward)**
- The algorithm tries to learn what action to take, in order to maximise the reward
- This is called a policy
- Applications in control, robotics, A/B testing



# Learning examples revisited

## Supervised Learning (Classification)

- Spam e-mail detection system
- Credit approval
- Recognize objects in images
- Find the relation between house prices and house sizes
- Predict the stock market

## Unsupervised Learning

- Market segmentation
- Market basket analysis
- Language models (word2vec)
- Social network analysis
- Movies recommendation\*
- Low-order data representations

In this course we will focus on the **supervised** learning case

# Learning examples revisited

## Supervised Learning (**Regression**)

- Spam e-mail detection system
- Credit approval
- Recognize objects in images
- Find the relation between house prices and house sizes
- Predict the stock market

## Unsupervised Learning

- Market segmentation
- Market basket analysis
- Language models (word2vec)
- Social network analysis
- Movies recommendation\*
- Low-order data representations

In this course we will focus on the **supervised** learning case

\* Movie recommendation can be performed both in a supervised and an unsupervised way

## Supervised learning: problem statement

The aim is to learn an unknown function  $f$ , given a dataset  $\mathcal{D}$

- The function is searched in the hypothesis space  $\mathcal{H}$ , where  $h \in \mathcal{H}$  is a specific function
- We want to find a function  $h$  that approximates  $f$  well, on the whole domain  $\mathcal{X}$

What does  $h \approx f$  mean?

- We need to define an error measure or a cost function
- Almost always *pointwise definition*:  $e[f(\varphi), h(\varphi)]$

# Cost functions

## Pointwise error examples

- *Squared error*:  $e(f(\varphi), h(\varphi)) = (f(\varphi) - h(\varphi))^2 \rightarrow$  used for regression
- *Binary error*:  $e(f(\varphi), h(\varphi)) = \mathbb{I}[f(\varphi) \neq h(\varphi)] \rightarrow$  used for classification

It is interesting to look at the *overall error*, which considers all  $N$  examples:

## Overall error examples

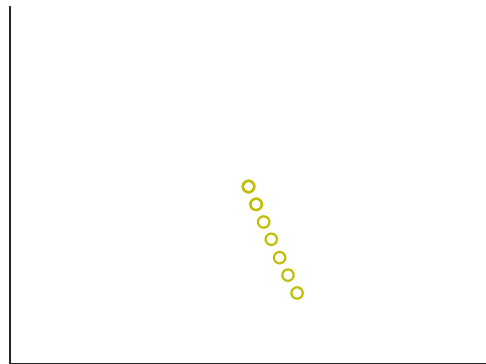
- *In sample error*:  $E_{\text{in}} = J(\cdot) = \frac{1}{N} \sum_{i=1}^N e[f(\varphi(i)), h(\varphi(i))]$
- *Out of sample error*:  $E_{\text{out}} = \mathbb{E}_{\varphi} [e(f(\varphi), h(\varphi))]$

# Outline

- Introduction
- Components of learning
- **Puzzle**
- Feasibility of learning
- Bias - variance tradeoff

## Puzzle

Focus on supervised learning: which are the plausible response values of the unknown function, on positions of the input space that we have not seen?

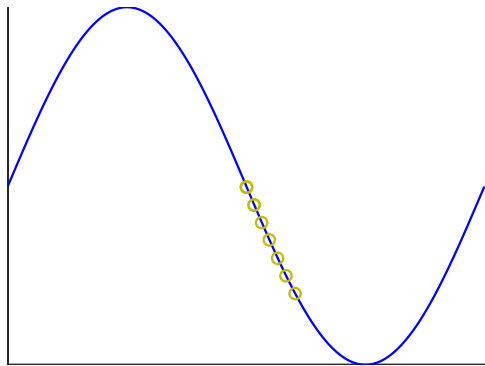


|       |   |   |   |   |   |   |   |   |         |
|-------|---|---|---|---|---|---|---|---|---------|
| ●     | ○ | ● | ● | ● | ○ | ● | ● | ● | $f = 1$ |
| ○     | ● | ○ | ○ | ○ | ● | ○ | ○ | ● | $f = 0$ |
| <hr/> |   |   |   |   |   |   |   |   |         |
| ●     | ○ | ○ |   |   |   |   |   |   | $f = ?$ |



## Puzzle

It is not possible to know how the function behaves outside the observed points  
(*Hume's induction problem*)



● ○ ● ● ● ○ ● ● ●  $f = 1$

○ ● ○ ○ ○ ● ○ ○ ●  $f = 0$

---

● ○ ○  $f = 1$

If first dot is black  $\rightarrow f = 1$

# Outline

- Introduction
- Components of learning
- Puzzle
- **Feasibility of learning**
- Bias - variance tradeoff

# Feasibility of learning

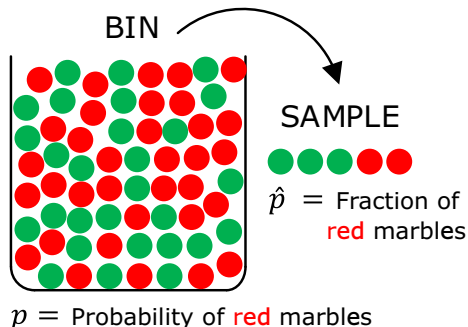
Focus on supervised learning, dichotomous classification case

**Problem:** Learning an unknown function

**Solution:** Impossible ☹. The function can assume any value outside the data we have

## Experiment

- Consider a 'bin' with **red** and **green** marbles
- $\mathbb{P}[\text{picking a red marble}] = p$
- The value of  $p$  is unknown to us
- Pick  $N$  marbles independently
- Fraction of red marbles in the sample =  $\hat{p}$



## Does $\hat{p}$ say something about $p$ ?

No!

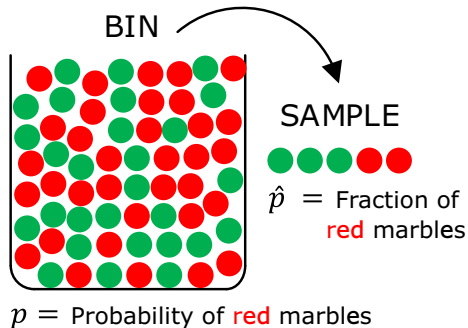
Sample can be mostly **green** while bin is mostly **red**

Possible

Yes!

Sample frequency  $\hat{p}$  is likely close to bin frequency  $p$  (if the sample is sufficiently large)

Probable



## Connection to learning

**Bin:** The unknown is a number  $p$

**Learning:** The unknown is a function  $f : \mathcal{X} \rightarrow \mathcal{Y}$

Each marble  $\bullet$  is a input point  $\varphi \in \mathcal{X} \subset \mathbb{R}^{(d-1) \times 1}$

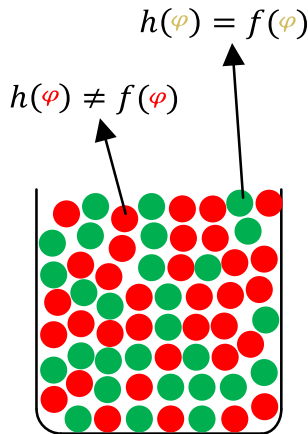
For a specific hypothesis  $h \in \mathcal{H}$ :

- Hypothesis got it right  $\rightarrow h(\varphi) = f(\varphi)$
- Hypothesis got it wrong  $\rightarrow h(\varphi) \neq f(\varphi)$

Both  $p$  and  $\hat{p}$  depend on the particular hypothesis  $h$

$\hat{p} \rightarrow$  In sample error  $E_{\text{in}}(h)$

$p \rightarrow$  Out of sample error  $E_{\text{out}}(h)$



The **Out of sample error**  $E_{\text{out}}(h)$  is the quantity that really matters

## Connection to real learning

In a learning scenario, the function  $h$  is not fixed a priori

- The *learning algorithm* is used to fathom the hypothesis space  $\mathcal{H}$ , to find the **best** hypothesis  $h \in \mathcal{H}$  that **matches the sampled data** → call this hypothesis  $g$
- With many hypotheses, there is more chance to find a good hypothesis  $g$  only by chance → **the function can be perfect on sampled data but bad on unseen ones**

There is therefore an approximation - generalization **tradeoff** between:

- Perform well on the given dataset
- Perform well on unseen data

The quantity  $E_{\text{out}}(g) - E_{\text{in}}(g)$  is called the **generalization error**

# Generalization theory

There is a **generalization theory**, based on the concept of VC-dimension, which studies the cases in which is possible to generalize

- The takeaway concept is that learning is feasible in a probabilistic way
- If we are able to deal with the approximation-generalization tradeoff, we can say with **high probability** that the generalization error is **small**

One way to study the tradeoff is to study the concepts of **bias** and **variance** of a learning model

# Outline

- Introduction
- Components of learning
- Puzzle
- Feasibility of learning
- **Bias - variance tradeoff**



## Approximation vs. generalization

The ultimate goal is to have a small  $E_{\text{out}}$ : good approximation of  $f$  **out of sample**

- More complex  $\mathcal{H} \implies$  better chances of **approximating**  $f$  in sample  $\rightarrow$  if  $\mathcal{H}$  is too simple, we fail to approximate  $f$  and we end up with large  $E_{\text{in}}$
- Less complex  $\mathcal{H} \implies$  better chance of **generalizing** out of sample  $\rightarrow$  if  $\mathcal{H}$  is too complex, we fail to generalize well

## Approximation vs. generalization

The example shows:

- perfect fit on training data

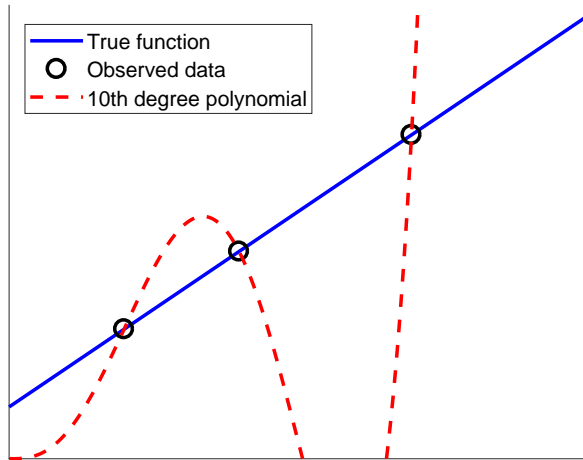
↓

$$E_{\text{in}} = 0$$

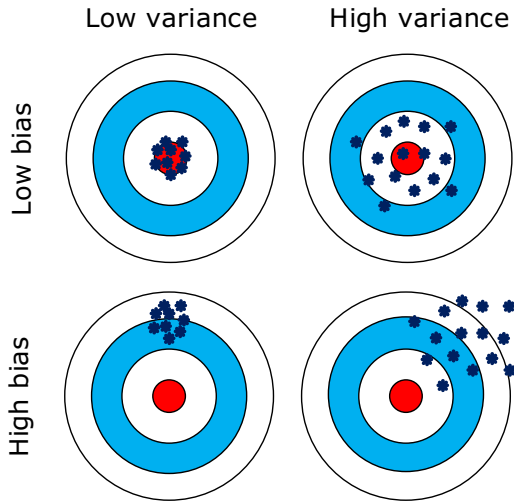
- low fit on test data

↓

$$E_{\text{out}} \text{ high}$$



## Bias and variance



## Bias and variance

Bias-variance analysis decomposes  $E_{\text{out}}$  into two terms:

1. How well  $\mathcal{H}$  can approximate  $f \rightarrow$  **Bias**
2. How well we are able to find a good  $h \in \mathcal{H} \rightarrow$  **Variance**

The out of sample error is (making explicit the dependence of  $g$  on  $\mathcal{D}$ ):

$$E_{\text{out}}(g^{(\mathcal{D})}) = \mathbb{E}_{\varphi} \left[ \left( g^{(\mathcal{D})}(\varphi) - f(\varphi) \right)^2 \right]$$

The **expected** out of sample error of the learning model is independent of the particular realization of data set used to find  $g^{(\mathcal{D})}$ :

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} \left[ E_{\text{out}}(g^{(\mathcal{D})}) \right] &= \mathbb{E}_{\mathcal{D}} \left[ \mathbb{E}_{\varphi} \left[ \left( g^{(\mathcal{D})}(\varphi) - f(\varphi) \right)^2 \right] \right] \\ &= \mathbb{E}_{\varphi} \left[ \mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - f(\varphi) \right)^2 \right] \right] \end{aligned}$$

## Bias and variance

Focus on  $\mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - f(\varphi) \right)^2 \right]$

Define the 'average' hypothesis  $\bar{g}(\varphi) = \mathbb{E}_{\mathcal{D}} [g^{(\mathcal{D})}(\varphi)]$

This average hypothesis can be derived by imagining many datasets  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$ , and building it by  $\bar{g}(\varphi) \approx \frac{1}{N} \sum_{k=1}^K g^{(\mathcal{D}_k)}(\varphi) \rightarrow$  this is a conceptual tool, and  $\bar{g}$  does not need to belong to the hypothesis set

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - f(\varphi) \right)^2 \right] &= \mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) + \bar{g}(\varphi) - f(\varphi) \right)^2 \right] \\ &= \mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) \right)^2 + \left( \bar{g}(\varphi) - f(\varphi) \right)^2 \right. \\ &\quad \left. + 2 \cdot \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) \right) \left( \bar{g}(\varphi) - f(\varphi) \right) \right] \end{aligned}$$

## Bias and variance

$$\mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - f(\varphi) \right)^2 \right] = \underbrace{\mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) \right)^2 \right]}_{\text{var}(\varphi)} + \underbrace{\left( \bar{g}(\varphi) - f(\varphi) \right)^2}_{\text{bias}(\varphi)}$$

Therefore:

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} [E_{\text{out}}(g^{(\mathcal{D})})] &= \mathbb{E}_{\varphi} \left[ \mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - f(\varphi) \right)^2 \right] \right] \\ &= \mathbb{E}_{\varphi} [\text{bias}(\varphi) + \text{var}(\varphi)] \\ &= \text{bias} + \text{var} \end{aligned}$$

# Bias and variance

## Interpretation

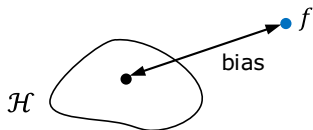
- The **bias** term  $\left(\bar{g}(\varphi) - f(\varphi)\right)^2$  measures how much our learning model is biased away from the target function

In fact,  $\bar{g}$  has the benefit of learning from an unlimited number of datasets, so it is only limited in its ability to approximate  $f$  by the limitations of the learning model itself

- The **variance** term  $\mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) \right)^2 \right]$  measures the variance in the final hypothesis, depending on the data set, and can be thought as how much the final chosen hypothesis differs from the 'mean' (best) hypothesis

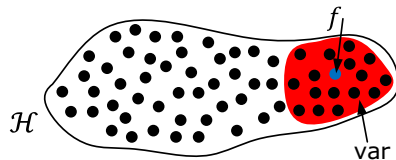
## Bias and variance

$$\text{bias} = \left( \bar{g}(\varphi) - f(\varphi) \right)^2$$



**Very small model.** Since there is only one hypothesis, both the average function  $\bar{g}$  and the final hypothesis  $g^{(\mathcal{D})}$  will be the same, for any dataset. Thus,  $\text{var} = 0$ . The bias will depend solely on how well this single hypothesis approximates the target  $f$ , and unless we are extremely lucky, we expect a large bias

$$\text{variance} = \mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) \right)^2 \right]$$



**Very large model.** The target function is in  $\mathcal{H}$ . Different data sets will lead to different hypotheses that agree with  $f$  on the data set, and are spread around  $f$  in the red region. Thus,  $\text{bias} \approx 0$  because  $\bar{g}$  is likely to be close to  $f$ . The var is large (heuristically represented by the size of the red region)



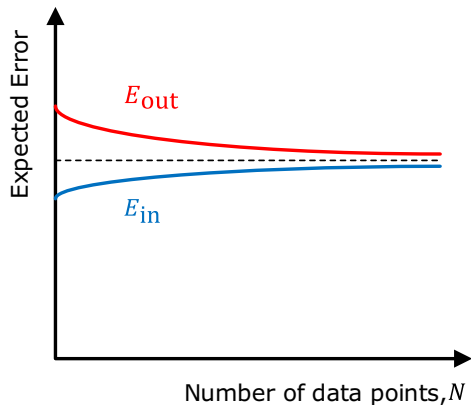
# Learning curves

How it is possible to know if a model is suffering from bias or variance problems?

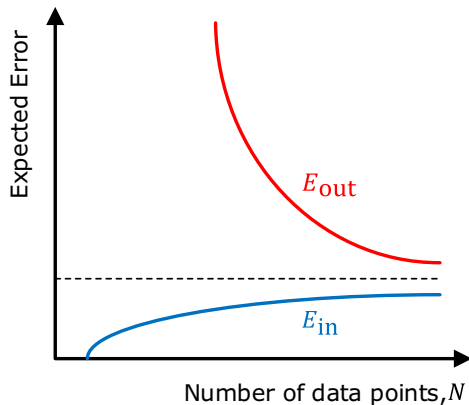
The **learning curves** provide a graphical representation for assessing this, by plotting the *expected out of sample error*  $\mathbb{E}_{\mathcal{D}} [E_{\text{out}}(g^{\mathcal{D}})]$  and the *expected in sample error*  $\mathbb{E}_{\mathcal{D}} [E_{\text{in}}(g^{\mathcal{D}})]$  vs. the number of data  $N$

In the practice, the curves are computed from one dataset, or by dividing it into more parts and taking the mean curve resulting from various datasets

## Learning curves



Simple model



Complex model

# Learning curves

## Interpretation

- **Bias** can be present when the error is quite high and  $E_{\text{in}}$  is similar to  $E_{\text{out}}$
- When **bias** is present, getting more data is not likely to help
- **Variance** can be present when there is a gap between  $E_{\text{in}}$  and  $E_{\text{out}}$
- When **variance** is present, getting more data is likely to help

## Fixing bias

- Try adding more features
- Try polynomial features
- Try a more complex model
- Boosting

## Fixing variance

- Try a smaller set of features
- Get more training examples
- Regularization
- Bagging

# Take home lessons

## Rule of thumb

How many data points  $N$  are required to ensure good generalization?

$$N \geq 10 \cdot (\text{no. of model's parameters})$$

## General principle

Match the 'model complexity' to the **data resources**, not to the **target complexity**

# Machine learning fundamentals: Lecture 2

Overfitting

Regularization

Validation

---

Mirko Mazzoleni - Identificazione dei Modelli e Analisi dei Dati (IMAD)

October 2018

University of Bergamo

Department of Management, Information and Production Engineering

*mirko.mazzoleni@unibg.it*

# Outline

- Overfitting
- Regularization
- Validation
- Model selection
- Cross-validation

# Outline

- **Overfitting**
- Regularization
- Validation
- Model selection
- Cross-validation

# Overfitting

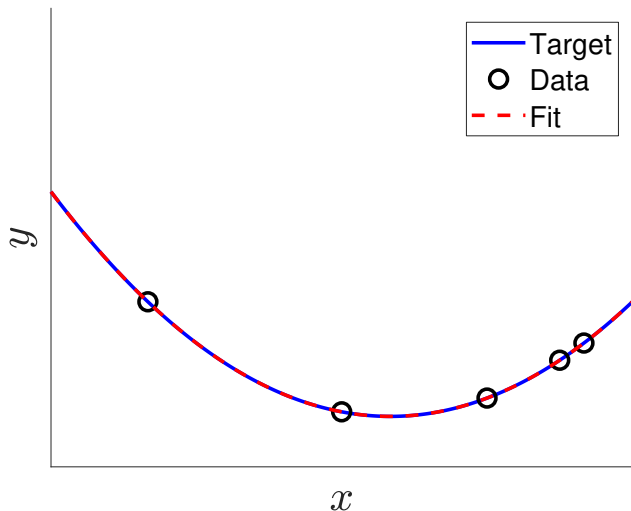




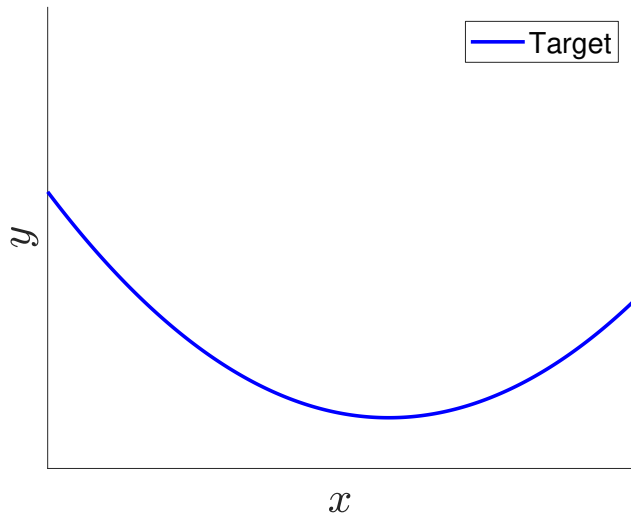
# Overfitting

- Simple target function
- $N = 5$  points
- Fit with 4th order polynomial

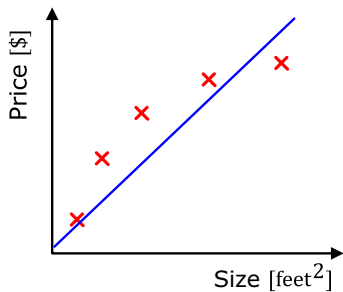
$$E_{\text{in}} = 0, \quad E_{\text{out}} = 0$$



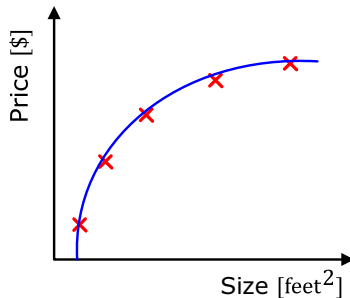
# Overfitting



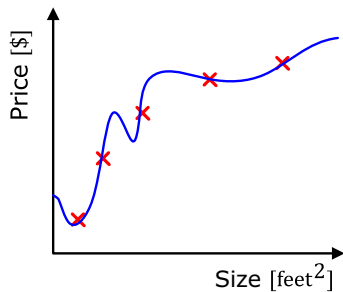
## Overfitting



Underfit



Just right

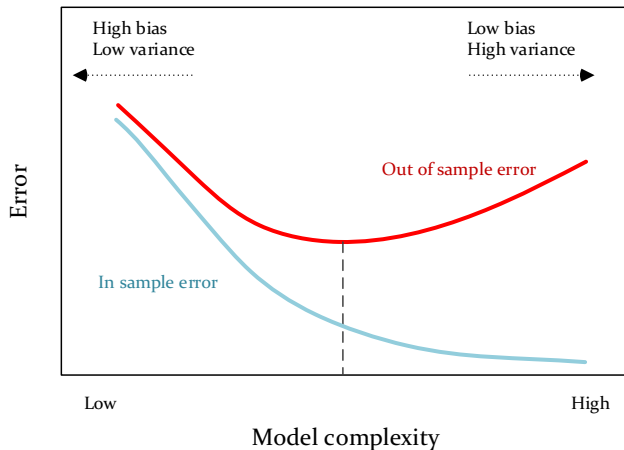


Overfit

# Overfitting

We talk of **overfitting** when decreasing  $E_{\text{in}}$  leads to increasing  $E_{\text{out}}$

- Major source of failure for machine learning systems
- Overfitting leads to bad generalization
- A model can exhibit bad generalization even if it does not overfit





## Bias - variance tradeoff revisited

Let the stochastic noise  $\varepsilon(\varphi)$  be a random variable with mean zero and variance  $\sigma^2$

$$\mathbb{E}_{\mathcal{D}, \varphi, \varepsilon} \left[ \left( g^{(\mathcal{D})}(\varphi) - (f(\varphi) + \varepsilon(\varphi)) \right)^2 \right] =$$
$$\underbrace{\mathbb{E}_{\mathcal{D}, \varphi} \left[ \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) \right)^2 \right]}_{\text{var}} + \underbrace{\mathbb{E}_{\varphi} \left( \bar{g}(\varphi) - f(\varphi) \right)^2}_{\text{bias}} + \underbrace{\mathbb{E}_{\varepsilon, \varphi} \left[ \left( \varepsilon(\varphi) \right)^2 \right]}_{\sigma^2}$$

- Overfitting is caused by **variance**. Variance is, in turn, affected by the noise terms, capturing a model's **susceptibility** to being led astray by the noise
- The variance term depends also on the complexity of  $\mathcal{H}$

# Outline

- Overfitting
- **Regularization**
- Validation
- Model selection
- Cross-validation

## A cure for overfitting

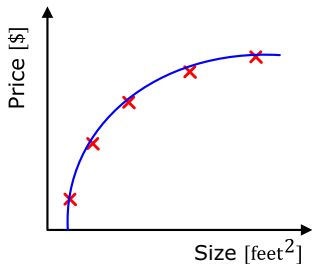
Regularization is the first line of defense against overfitting

- We have seen that **complex** model are more prone to overfitting
- This is because they are more powerful, and thus they can fit the **noise**
- Simple models exhibits **less variance** because of their limited expressiveness. This gain in variance often is greater than their greater bias
- However, if we stick only to **simple** models, we may not end up with a satisfying approximation of the target function  $f$

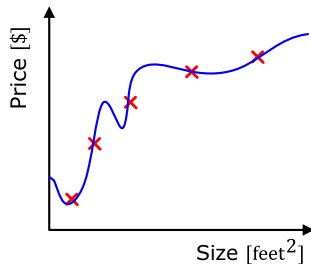
*How can we retain the benefits of **both** worlds?*



## A cure for overfitting



$$\mathcal{H}_2 : \vartheta_0 + \vartheta_1 x + \vartheta_2 x^2$$



$$\mathcal{H}_4 : \vartheta_0 + \vartheta_1 x + \vartheta_2 x^2 + \vartheta_3 x^3 + \vartheta_4 x^4$$

- We can recover the model  $\mathcal{H}_2$  from the model  $\mathcal{H}_4$  by imposing  $\vartheta_3 = \vartheta_4 = 0$

$$\arg \min_{\boldsymbol{\vartheta}} \frac{1}{N} \sum_{i=1}^N \left( h(\boldsymbol{\varphi}(i); \boldsymbol{\vartheta}) - f(\boldsymbol{\varphi}(i)) \right)^2 + 1000 \cdot (\vartheta_3)^2 + 1000 \cdot (\vartheta_4)^2$$

## A cure for overfitting

$$\arg \min_{\vartheta} \frac{1}{N} \sum_{i=1}^N \left( h(\varphi(i); \vartheta) - f(\varphi(i)) \right)^2 + \underbrace{1000 \cdot (\vartheta_3)^2 + 1000 \cdot (\vartheta_4)^2}_{\Omega}$$

- The cost function has been **augmented** with a penalization term  $\Omega(\vartheta_3, \vartheta_4)$
- The minimization algorithm now has to minimize both  $E_{\text{in}}$  and  $\Omega(\vartheta_3, \vartheta_4)$
- Due to the minimization process the value of  $\vartheta_3$  and  $\vartheta_4$  will be shrunk toward a small value → **not exactly zero: soft order constraint**
- If this value is very small, then the contribution of  $x_3$  and  $x_4$  is negligible →  $\vartheta_3$  and  $\vartheta_4$  (very small) multiply  $x_3$  and  $x_4$
- In this way, we ended up with a model that it is like the model  $\mathcal{H}_2$  in terms of complexity → **we can think as like the features  $x_3$  and  $x_4$  were not present**
- It is like we reduced the number of parameters of the  $\mathcal{H}_4$  model

## Regularization

The concept introduced in the previous slides can be extended to the entire model parameters

Instead of minimizing the in-sample error  $E_{\text{in}}$ , minimize the **augmented error**:

$$E_{\text{aug}}(\boldsymbol{\vartheta}) = \frac{1}{N} \sum_{i=1}^N \left( h(\boldsymbol{\varphi}(i); \boldsymbol{\vartheta}) - f(\boldsymbol{\varphi}(i)) \right)^2 + \lambda \sum_{j=1}^{d-1} (\vartheta_j)^2$$

- Usually we do not want to penalize the intercept  $\vartheta_0$ , so  $j$  starts from 1
- The term  $\Omega(h) = \sum_{j=1}^{d-1} (\vartheta_j)^2$  is called **regularizer**
- The regularizer is a penalty term which depends on the hypothesis  $h$
- The term  $\lambda$  weights the importance of minimizing  $E_{\text{in}}$ , with respect to minimizing  $\Omega(h)$

## Regularization

The minimization of  $E_{\text{aug}}$  can be viewed as a **constrained** minimization problem

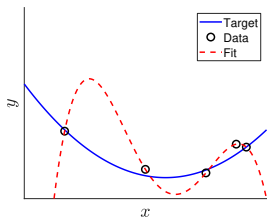
$$\text{Minimize } E_{\text{aug}}(\boldsymbol{\vartheta}) = \frac{1}{N} \sum_{i=1}^N \left( h(\boldsymbol{\varphi}(i); \boldsymbol{\vartheta}) - f(\boldsymbol{\varphi}(i)) \right)^2$$

$$\text{Subject to : } \boldsymbol{\vartheta}^T \boldsymbol{\vartheta} \leq C$$

- With this view, we are explicitly constraining the weights to not have certain large values
- There is a relation between  $C$  and  $\lambda$  in such a way that if  $C \uparrow$  the  $\lambda \downarrow$
- In fact, bigger  $C$  means that the weights can be greater. This is equal to set for a lower  $\lambda$ , because the regularization term will be less important, and therefore the weights will not be shrunk as much

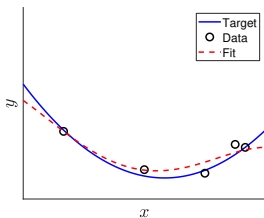
# Effect of $\lambda$

$\lambda_1$



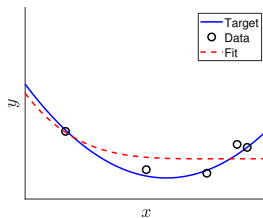
**Overfit**

$\lambda_2 > \lambda_1$



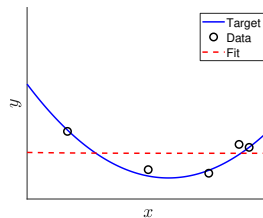
→

$\lambda_3 > \lambda_2$



→

$\lambda_4 > \lambda_3$



**Underfit**

# Augmented error

General form of the augmented error

$$E_{\text{aug}}(\vartheta) = E_{\text{in}}(\vartheta) + \lambda\Omega(h)$$

- $\Omega(h)$  is a measure of complexity of a specific hypothesis  $h \in \mathcal{H}$
- This information about the model complexity is used in the cost function and drives the minimization process

The augmented error  $E_{\text{aug}}$  is **better** than  $E_{\text{in}}$  as a proxy for  $E_{\text{out}}$

## Augmented error

The holy Grail of machine learning would be to have a formula for  $E_{\text{out}}$  to minimize

- In this way, it would be possible to directly minimize the out of sample error instead of the in sample one
- Regularization helps by estimating the quantity  $\Omega(h)$ , which, added to  $E_{\text{in}}$ , gives  $E_{\text{aug}}$ , an estimation of  $E_{\text{out}}$

## Choice of the regularizer

There are many choices of possible regularizers. The most used ones are:

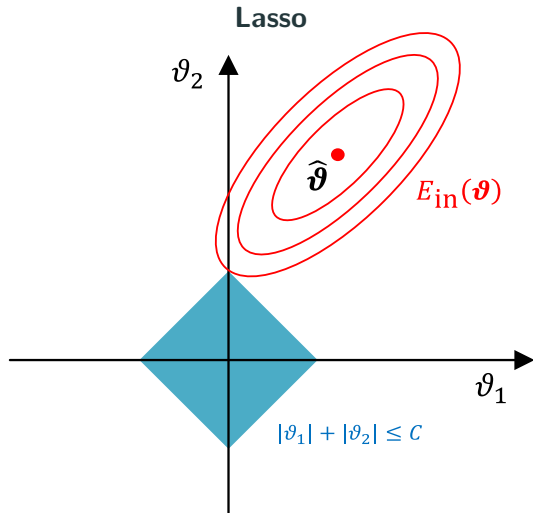
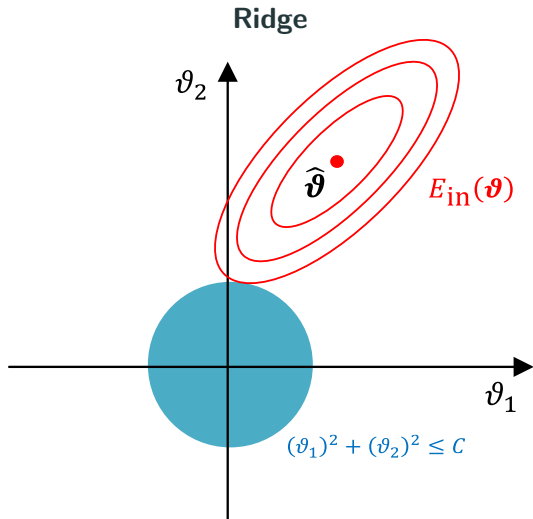
- **$L_2$  regularizer:** also called Ridge regression,  $\Omega(h) = \sum_{j=1}^{d-1} \vartheta_j^2$
- **$L_1$  regularizer:** also called Lasso regression,  $\Omega(h) = \sum_{j=1}^{d-1} |\vartheta_j|$

The different regularizers behaves differently:

- The ridge penalty tends to shrink all coefficients to a lower value
- The lasso penalty tends to set more coefficients exactly to zero



## Geometrical interpretation



## Regularization and bias-variance

The effects of the regularization procedure can be observed in the bias and variance terms

- Regularization trades more **bias** in order to considerably decrease the **variance** of the model
- Regularization strives for **smoother** hypothesis, thereby reducing the opportunities to overfit
- The amount of regularization  $\lambda$  has to be chosen specifically for each type of regularizer
- Usually  $\lambda$  is chosen by cross-validation

# Outline

- Overfitting
- Regularization
- **Validation**
- Model selection
- Cross-validation

## Validation vs. regularization

The out of sample error can be seen as:  $E_{\text{out}}(h) = E_{\text{in}}(h) + \text{overfit penalty}$

### Regularization

$$E_{\text{out}}(h) = E_{\text{in}}(h) + \underbrace{\text{overfit penalty}}_{\text{regularization estimates this quantity}}$$

### Validation

$$\underbrace{E_{\text{out}}(h)}_{\text{validation estimates this quantity}} = E_{\text{in}}(h) + \text{overfit penalty}$$

# Validation set

The idea of a **validation set** is to estimate the model performance out of sample

1. Remove a subset from the training data → this subset is not used in training
2. Train the model on the remaining training data → the model will be trained on less data
3. Evaluate the model's performance on the held-out set → this is an unbiased estimation of the out of sample error
4. Retrain the model on *all* the data

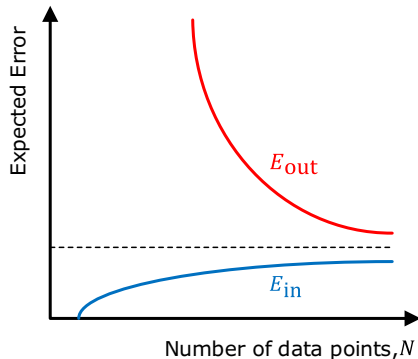
## $K$ is taken out of $N$

Given the dataset  $\mathcal{D} = \{\varphi(1), y(1)\}, \dots, \{\varphi(N), y(N)\}$

$\underbrace{K \text{ points}}_{\mathcal{D}_{\text{val}}} \rightarrow \text{validation}$

$\underbrace{N - K \text{ points}}_{\mathcal{D}_{\text{train}}} \rightarrow \text{training}$

- Small  $K$ : bad estimate of  $E_{\text{out}}$
- Large  $K$ : possibility of learning a bad model (learning curve)



**$K$  is put back into  $N$**

$$\mathcal{D} \rightarrow \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{val}}$$

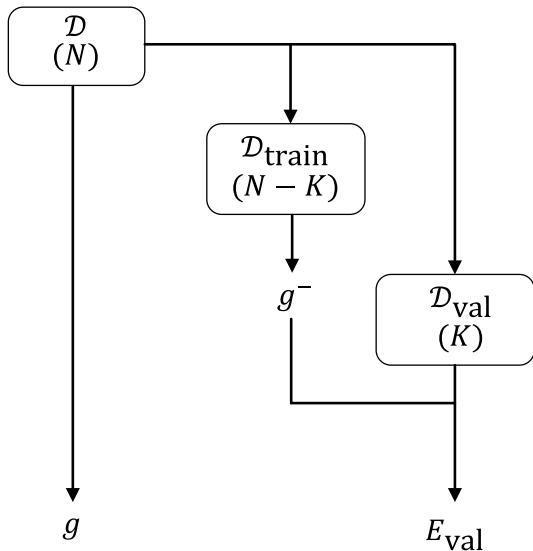
$$\begin{array}{ccc} \downarrow & \downarrow & \downarrow \\ N & N - K & K \end{array}$$

$$\mathcal{D} \Rightarrow g \quad \mathcal{D}_{\text{train}} \Rightarrow g^{-}$$

$$E_{\text{val}} = E_{\text{val}}(g^{-})$$

**Rule of thumb:**

$$K = \frac{N}{5}$$



# Outline

- Overfitting
- Regularization
- Validation
- **Model selection**
- Cross-validation



## Model selection

The most important use of a validation set is **model selection**

- Choose between a linear model and a nonlinear one
- Choice of the order of the polynomial
- Choice of the regularization parameter
- Any other choice that affects the learning of the model

If the validation set is used to perform choices (e.g. to select the regularization parameter  $\lambda$ ), then it **no longer** provides an unbiased estimate of  $E_{\text{out}}$

There is the need of a third dataset: the **test set**, onto which to measure the model's performance  $E_{\text{test}}$

## Using $\mathcal{D}_{\text{val}}$ more than once

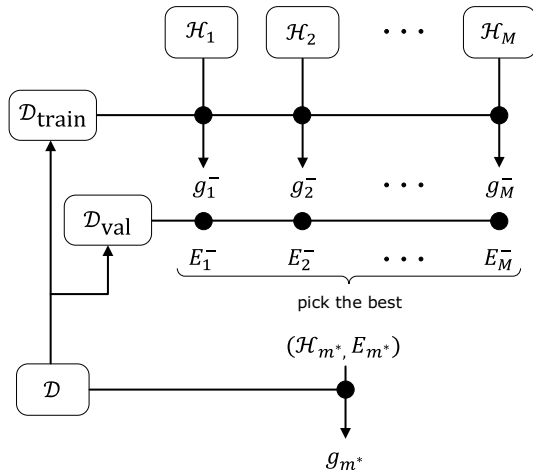
$M$  models  $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$

Use  $\mathcal{D}_{\text{train}}$  to learn  $g_m^-$  for each model

Evaluate  $g_m^-$  using  $\mathcal{D}_{\text{val}}$

$$E_m = E_{\text{val}}(g_m^-) \quad m = 1, \dots, M$$

Pick the model  $m = m^*$  with the smallest  $E_m$



## How much bias

For the  $M$  models  $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$ ,  $\mathcal{D}_{\text{val}}$  is used for “training” on the **finalist model set**:

$$\mathcal{H}_{\text{val}} = \{g_1^-, g_2^-, \dots, g_M^-\}$$

- The validation performance of the final model is  $E_{\text{val}}(g_{m^*}^-)$
- This quantity is biased and not representative of  $E_{\text{out}}(g_{m^*}^-)$ , just as the in sample error  $E_{\text{in}}$  was not representative of  $E_{\text{out}}$
- What happened is that  $\mathcal{D}_{\text{val}}$  has become the “training set” for  $\mathcal{H}_{\text{val}}$
- The risk is to overfit the validation set

# Data contamination

Error estimates:  $E_{\text{in}}$ ,  $E_{\text{val}}$ ,  $E_{\text{test}}$

Contamination: Optimistic bias in estimating  $E_{\text{out}}$

- **Training set:** totally contaminated
- **Validation set:** slightly contaminated
- **Test set:** totally 'clean'

# Outline

- Overfitting
- Regularization
- Validation
- Model selection
- **Cross-validation**

## The dilemma about $K$

The following chain of reasoning:

$$E_{\text{out}}(g) \approx E_{\text{out}}(g^-) \approx E_{\text{val}}(g^-)$$

(small  $K$ )      (large  $K$ )

highlights the dilemma in selecting  $K$

Can we have  $K$  **both** small and large?

## Leave one out cross-validation

Use  $N - 1$  points for training and  $K = 1$  point for validation

$$\mathcal{D}_i = \{\varphi(1), y(1)\}, \dots, \{\cancel{\varphi(i)}, \cancel{y(i)}\}, \dots, \{\varphi(N), y(N)\},$$

where  $\mathcal{D}_i$  is the training set without the point  $i$

The final hypothesis learned from  $\mathcal{D}_i$  is  $g_i^-$

The validation error on the point  $\varphi(i)$  is  $e(i) = E_{\text{val}}(g_i^-) = e(g_i^-(\varphi(i)), y(i))$

It is then possible to define the **cross-validation error**

$$E_{\text{cv}} = \frac{1}{N} \sum_{i=1}^N e(i)$$

## Cross-validation for model selection

Cross-validation can be used effectively to perform model selection by selecting the right regularization parameter  $\lambda$

1. Define  $M$  models by choosing different values for  $\lambda$ :  $(\mathcal{H}, \lambda_1), (\mathcal{H}, \lambda_2), \dots, (\mathcal{H}, \lambda_M)$
2. **for** each model  $m = 1, \dots, M$  **do**
  - 2.1 Use cross-validation to obtain estimates of the out of sample error for each model
3. Select the model  $m^*$  with the smallest cross-validation error  $E_{cv}(m^*)$
4. Use the model  $(\mathcal{H}, \lambda_{m^*})$  and all the data  $\mathcal{D}$  to obtain the final hypothesis  $g_{m^*}$

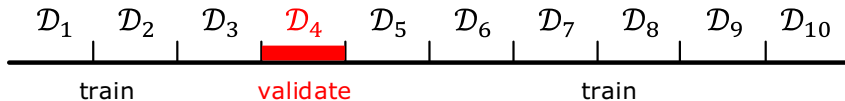


## Leave more than one out

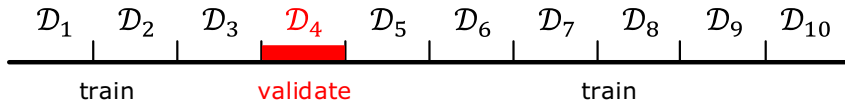
Leave-one-out cross-validation as the disadvantage that:

- It is computationally expensive, requiring a total of  $N$  training sessions for each of the  $M$  models
- The estimated cross-validation error has high variance, since it is based only on one point

It is possible to reserve more points for validation by dividing the training set in “folds”



## Leave more than one out



- This produces  $\frac{N}{K}$  training session on  $N - K$  points each
- A good compromise for the number of folds is 10

**10-fold cross validation:  $K = \frac{N}{10}$**

- Pay attention to not reduce the training set to much
  - Look at the learning curves