



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione



# IDENTIFICAZIONE DEI MODELLI E ANALISI DEI DATI (IMAD)

## Lezione 6: Fondamenti di machine learning

**Corso di Laurea Magistrale in  
INGEGNERIA INFORMATICA**

SPEAKER

Prof. Mirko Mazzoleni

PLACE

Università degli Studi di  
Bergamo

# Syllabus

## Parte I: sistemi statici

### 1. Richiami di statistica

### 2. Teoria della stima

2.1 Proprietà degli stimatori

### 3. Stima a minimi quadrati

3.1 Stima di modelli lineari

3.2 Algoritmo del gradient descent

### 4. Stima a massima verosimiglianza

4.1 Proprietà della stima

4.2 Stima di modelli lineari

### 5. Regressione logistica

5.1 Stima di un modello di regressione logistica

### 6. Fondamenti di machine learning

6.1 Bias-Variance tradeoff

6.2 Overfitting

6.3 Regularizzazione

6.4 Validazione

### 7. Cenni di stima Bayesiana

7.1 Probabilità congiunte, marginali e condizionate

7.2 Connessione con Filtro di Kalman



# IMAD

## Parte I: sistemi statici

## Parte II: sistemi dinamici

### Stima parametrica $\hat{\theta}$

- $\theta$  deterministico

- ***NO assunzioni su ddp dei dati***

- ✓ Stima parametri popolazione
- ✓ Stima modello lineare: minimi quadrati

- ***SI assunzioni su ddp dei dati***

- ✓ Stima massima verosimiglianza parametri popolazione
- ✓ Stima modello lineare: massima verosimiglianza
- ✓ Regressione logistica

- $\theta$  variabile casuale

- ***SI assunzioni su ddp dei dati***

- ✓ Stima Bayesiana

### Machine learning

### Stima parametrica $\hat{\theta}$

- $\theta$  deterministico

- ***NO assunzioni su ddp dei dati***

- ✓ Modelli lineari di pss
- ✓ Predizione
- ✓ Identificazione
- ✓ Persistente eccitazione
- ✓ Analisi asintotica metodi PEM
- ✓ Analisi incertezza stima (numero dati finito)
- ✓ Valutazione del modello



# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati
3. Feasibility of learning
4. Bias-variance tradeoff
5. Learning curves
6. Overfitting
7. Regularizzazione
8. Validazione, cross-validazione e formule di complessità ottima
9. Esercizi con codice



# Outline

## **1. Introduzione al machine learning e alla data science**

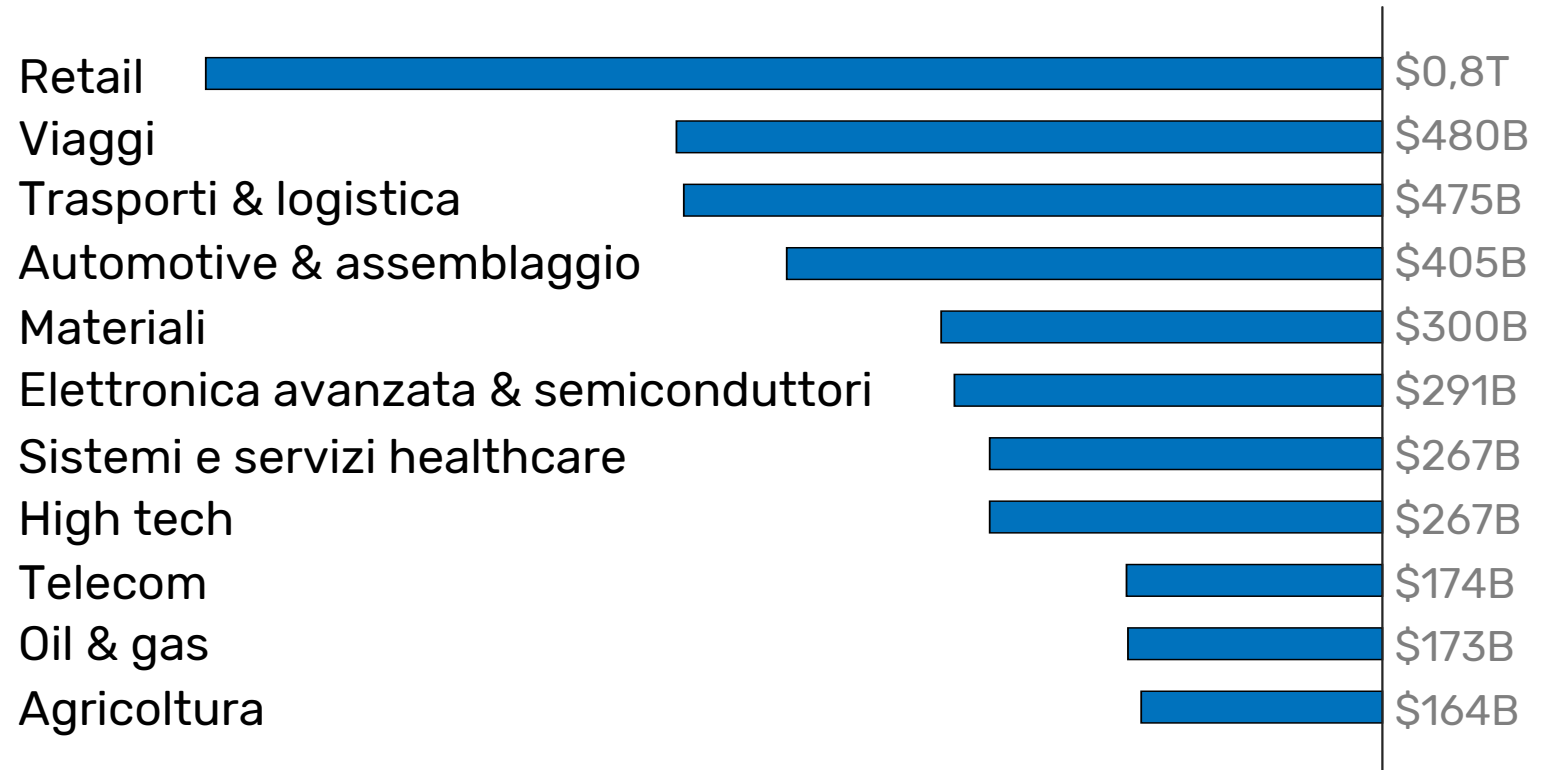
- 2. Problemi supervisionati e non supervisionati
- 3. Feasibility of learning
- 4. Bias-variance tradeoff
- 5. Learning curves
- 6. Overfitting
- 7. Regularizzazione
- 8. Validazione, cross-validazione e formule di complessità ottima
- 9. Esercizi con codice



# Introduzione al machine learning e alla data science

Valore creato dall'Artificial  
Intelligence (AI) entro il  
2030

**\$13**  
**Trillions**  
1 trillion =  $10^{12}$  dollari



- E' **difficile** trovare un settore industriale **che non beneficerà** dell'AI nel prossimo futuro

# Perché proprio ora?

Una soluzione AI funziona quando abbiamo **dati** che riguardano il nostro business:

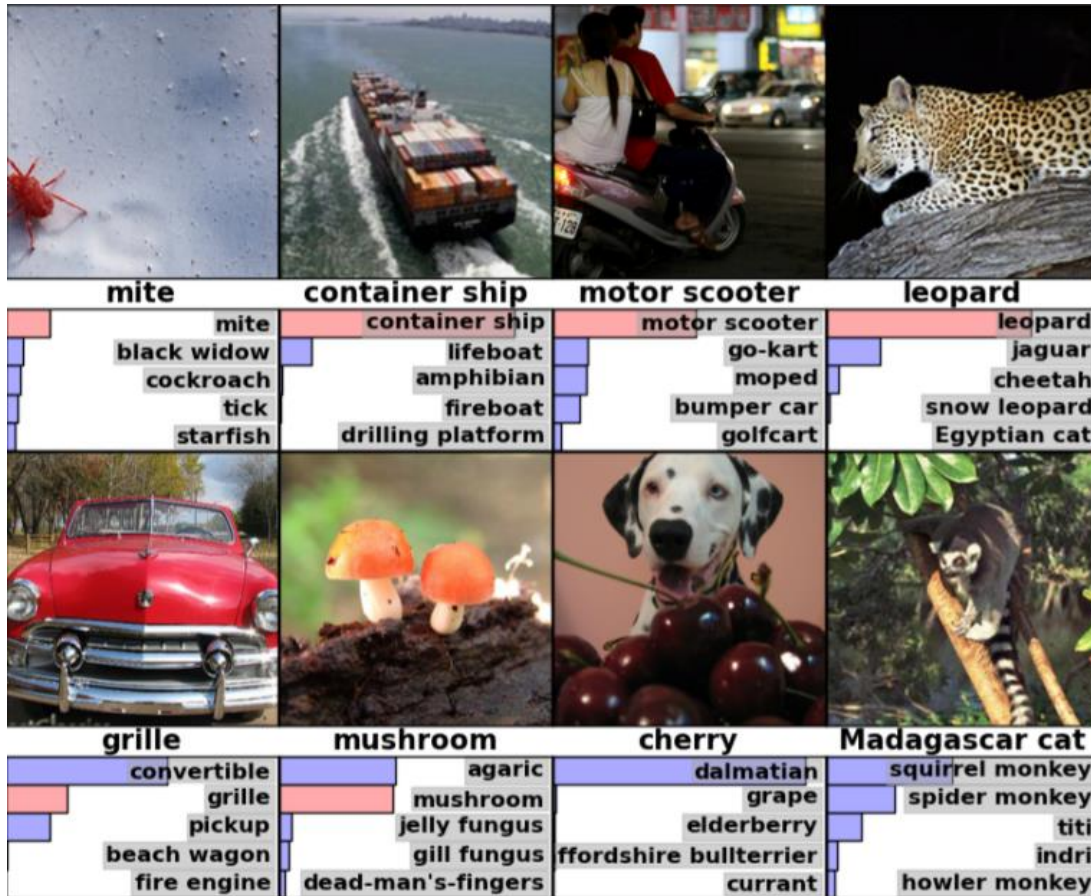
- E' possibile **collezionare dati** da svariati aspetti del business (operations, linea di produzione, supply-chain, parco clienti, campagne di marketing)
- L'informazione collezionata deve essere **analizzata** per ottenere **risultati azionabili**
- Una grande quantità di dati necessita di **specifiche infrastrutture** per essere gestita
- Una grande quantità di dati necessita di **potenza di calcolo** per essere analizzata
- Possiamo far sì che siano i computer a prendere decisioni, basandosi su **esempi**
- Nascita di **specifici lavori** e **specifici titoli di lavoro**





# Esempi di learning

**Ultimi 10 anni:** straordinari passi avanti nelle applicazioni di **visione artificiale**





# In che cosa consiste il learning

Il machine learning ha senso di essere applicato quando

1. Esiste un «pattern» nei dati
2. Non possiamo descriverlo matematicamente (non esiste una funzione analitica)

## **3. Abbiamo dati su di esso**

I presupposti 1. e 2. non sono obbligatori:

- Se un pattern non esiste, non imparerò nulla
- Se posso descrivere un pattern matematicamente, presumibilmente non imparerò la migliore relazione
- Il vero vincolo è l'assunzione 3.

# Machine learning vs. data science

Superficie della casa (feet <sup>2</sup> )	# camere letto	# bagni	Rinnovata di recente	Prezzo (1000\$)
523	1	2	No	115
645	1	3	No	150
708	2	1	No	210
1034	3	3	Si	280

**Inputs**

**Output**


## Machine learning

- Predirre Output dato Input
- Software di AI operative (sito web \ app mobile)



**Output:** Codice e programma

## Data science

- Le case con 3 bagni sono più costose di quelle con 2 bagni della medesima dimensione
- Le case rinnovate di recente costano  **Output:** Presentazione 15% in più



# Organizzare i dati per imparare modelli dai dati

È importante specificare i **criteri di accettazione**: quanto il nostro modello è «buono»?

**Esempio:** si vuole progettare un sistema di visione per controllo qualità



NO DIFETTO



NO DIFETTO



DIFETTO

**Obiettivo:** individuare i difetti con  
un'accuratezza del 95%



È necessario avere un dataset (spesso più di uno)  
su cui valutare le performance del modello

**Dati di validazione**

# Organizzare i dati per imparare modelli dai dati

## Dati di Training\Identificazione



Stimo il  
modello

## Dati di validazione



Valuto il  
modello

Output  
dell'algoritmo

NO DIFETTO

NO DIFETTO

NO DIFETTO

**66,7%**  
**accuratezza**

# Outline

1. Introduzione al machine learning e alla data science
- 2. Problemi supervisionati e non supervisionati**
3. Feasibility of learning
4. Bias-variance tradeoff
5. Learning curves
6. Overfitting
7. Regularizzazione
8. Validazione, cross-validazione e formule di complessità ottima
9. Esercizi con codice



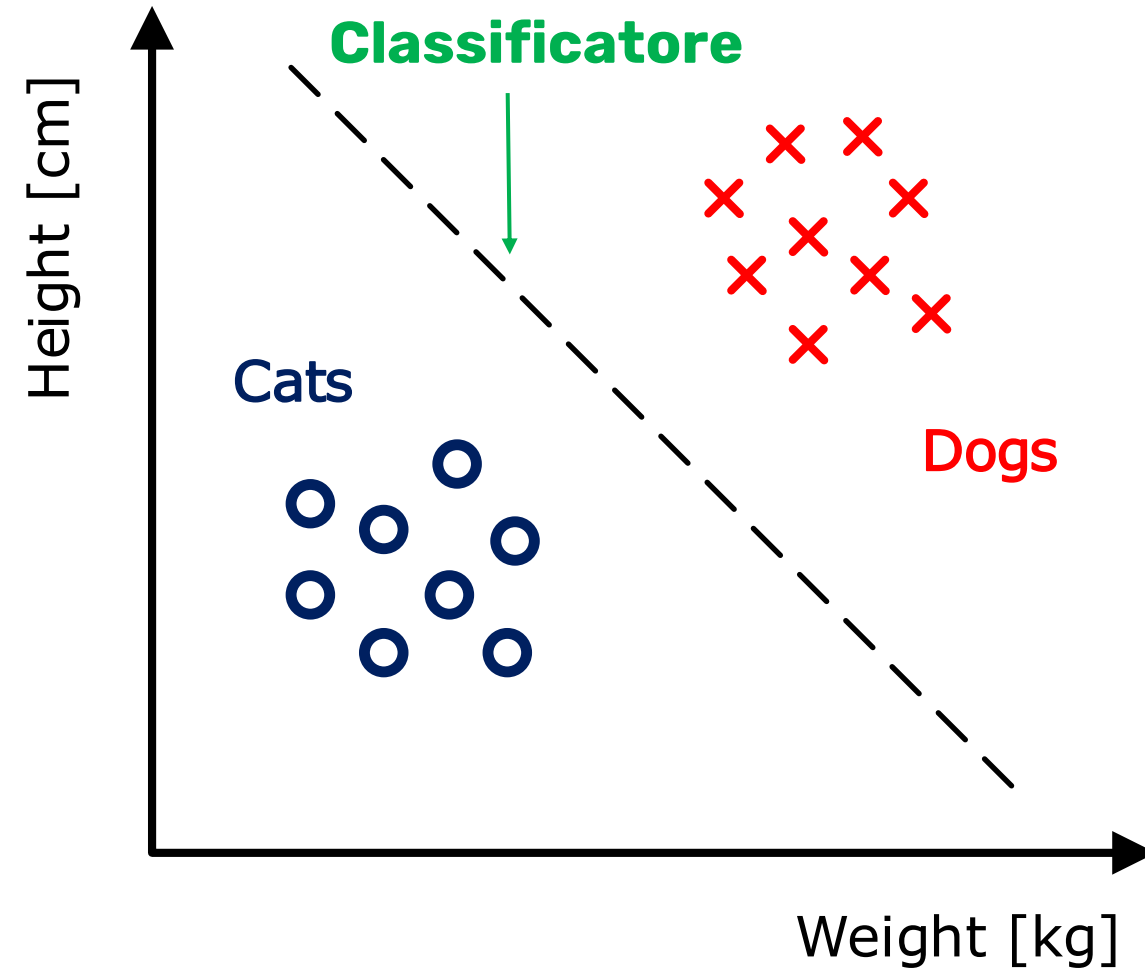
# I componenti del learning

- Input:  $\varphi \in \mathbb{R}^{d \times 1}$  (contenuto testuale della email)  $\rightarrow$  ogni dimensione è un «attributo» della mail
  - Output:  $y$  (spam / non spam?)  $\rightarrow$  la decisione che dobbiamo prendere
  - Funzione target:  $f: \mathbb{R}^{d \times 1} \rightarrow \mathcal{Y}$  (Formula ideale del filtro anti-spam)  $\rightarrow$  ignota, si deve stimare
  - Dati:  $\mathcal{D} = \{(\varphi(1), y(1)), \dots, (\varphi(N), y(N))\}$  (record storico di emails)
    - ✓ Ogni **vettore delle features** (regressori) è costituito da diverse informazioni utilizzate per prevedere la variabile di output
  - Ipotesi scelta:  $g: \mathbb{R}^{d \times 1} \rightarrow \mathcal{Y}, g \in \mathcal{M}$  (formula che viene usata)  $\rightarrow g$  è una **approssimazione** di  $f$
- $\mathcal{M}$  è chiamato **spazio delle ipotesi** (o **set dei modelli**). Insieme all'**algoritmo di learning** forma il **modello di learning**



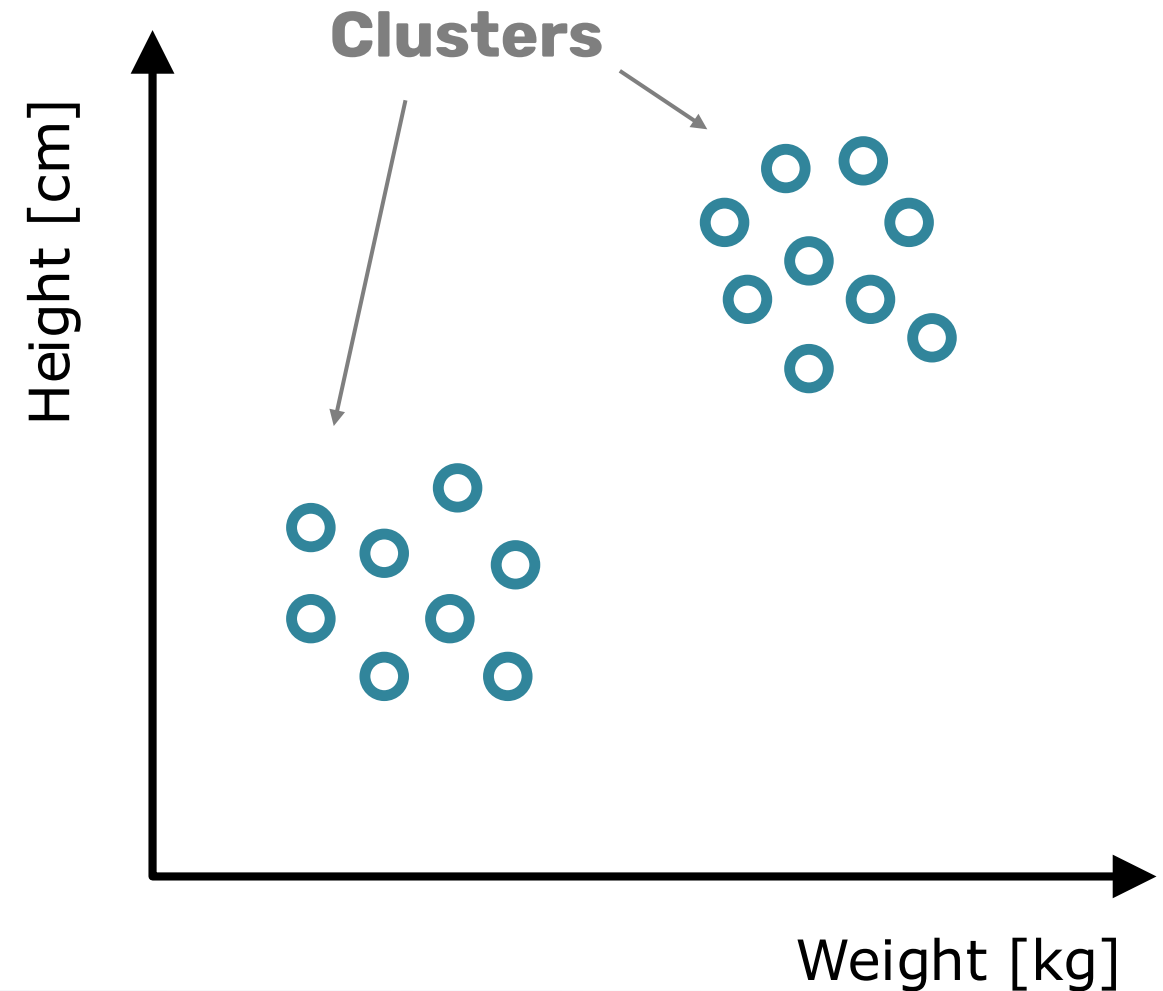
# Apprendimento supervisionato (supervised learning)

- La «risposta corretta» (output label)  $y$  è nota
- Prevedere  $y$  da un set di inputs  $\varphi \in \mathbb{R}^{d \times 1}$   
 $d \times 1$
- **Regressione:** prevedere una variabile continua  $y \in \mathbb{R}$  (**valore reale**)  
 $1 \times 1$
- **Classification:** prevedere una variabile categorica  $y \in \{1, 2, \dots, C\}$  (**classe\categoria**)



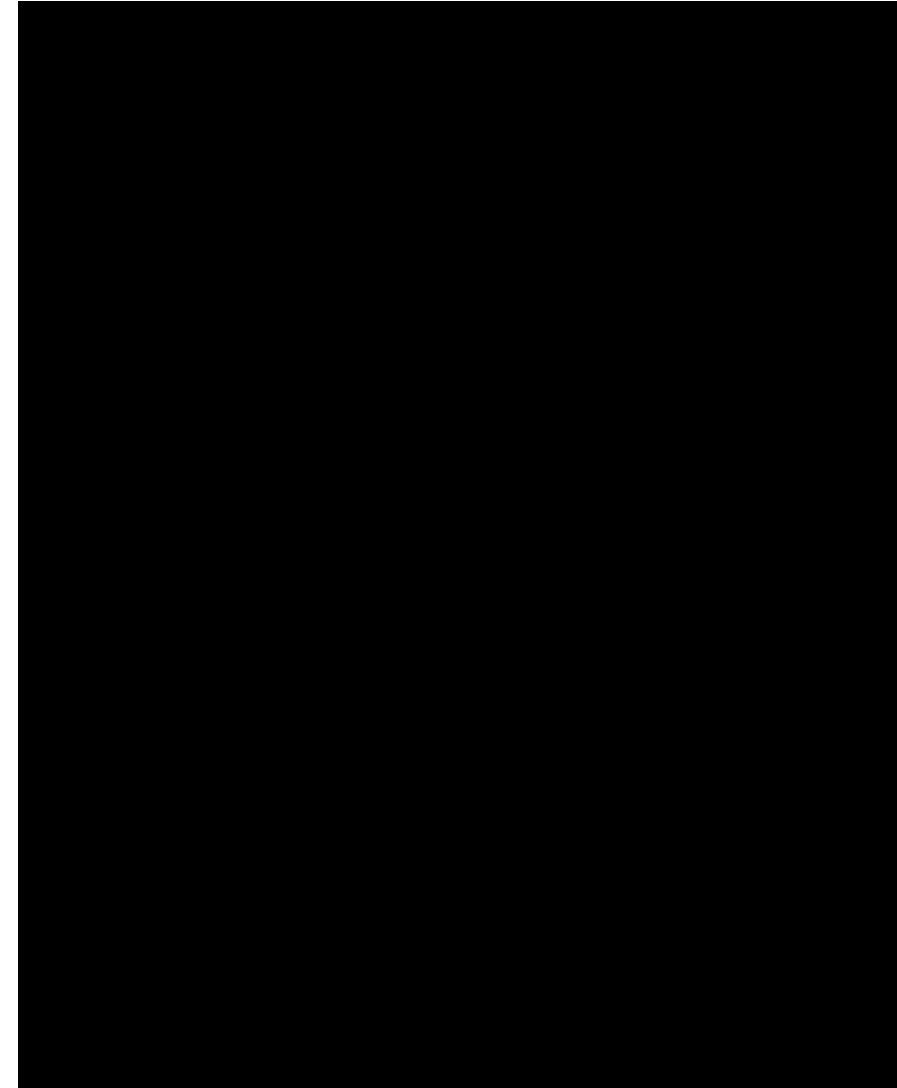
# Apprendimento non supervisionato (unsupervised learning)

- Anzichè **(input, output)** abbiamo **(input, ?)**
- Non c'è una funzione  $f$  da apprendere
- Si vuol esplorare le proprietà di  $\varphi \in \mathbb{R}^{d \times 1}$   
 $d \times 1$
- Rappresentazione «ad alto livello» dell'input
- Gli elementi nello **stesso cluster** hanno proprietà simili



# Apprendimento per rinforzo (reinforcement learning)

- Anzichè **(input, output)** abbiamo **(input, output, ricompensa)**
- L'algoritmo cerca di imparare quale azione intraprendere (policy), al fine di massimizzare la ricompensa
- Applicazioni in controllo, robotica, A/B testing (e.g. multi-armed bandits)



# Esempi di problemi supervisionati e non

## Supervisionati

- Filtro anti-spam **Classificazione**
- Approvazione crediti **Classificazione**
- Riconoscere oggetti in immagini **Classificazione**
- Prevedere i prezzi delle case **Regressione**
- Prevedere the stock market **Regressione**

## Non supervisionati

- Segmentazione mercato **Clustering**
- Market basket analysis **Co-occurrence grouping**
- Modelli del linguaggio (word2vec) **Similarity matching**
- Analisi social networks **Link prediction**
- Low-order data representations **Data reduction**

## **Supervisionato o non supervisionato**

- Recommendation systems **Similarity matching**



# Supervised learning: definizione del problema

Concentriamoci sul problema dell'**apprendimento supervisionato**. Abbiamo già visto due algoritmi di questo paradigma:

- **Regressione lineare:**  $y(i) = f(\varphi(i)) = \underset{1 \times d}{\varphi^T(i)} \underset{d \times 1}{\theta} \Rightarrow$  **regressione**
- **Regressione logistica:**  $y(i) = f(\varphi(i)) = s(\varphi^T(i)\theta) \Rightarrow$  **classificazione**

In entrambi i casi, l'obiettivo era quello di stimare la funzione  $f(\ )$  usando i dati  $\mathcal{D}$ :

- La funzione  $f$  viene cercata, dall'algoritmo di learning, nello spazio delle ipotesi  $\mathcal{M}$
- Vogliamo trovare una funzione  $h \in \mathcal{M}$  che **approssima bene**  $f$ , non solo sui dati  $\mathcal{D}$  a disposizione, ma **sull'intero dominio**  $\mathbb{R}^{d \times 1}$  di  $f$

e.g. «lo spazio di tutte le funzioni lineari»

# Supervised learning: definizione del problema

Cosa vuol dire che  $h \approx f$ ?

- Dobbiamo definire una **misura di errore** o di **costo**. In precedenza, abbiamo già definito delle funzioni di costo  $J(\theta)$  per la regressione lineare e logica

## Misure di errore puntuali

Le misure di errore puntuali  $\ell(\varphi; \theta)$  sono basate su un singolo punto  $\varphi$ . Esempi sono:

- **Errore quadratico:**  $\ell(f(\varphi), h(\varphi; \theta)) = (f(\varphi) - h(\varphi; \theta))^2 \rightarrow$  usata per regressione
- **Errore binario:**  $\ell(f(\varphi), h(\varphi; \theta)) = \mathbb{I}\{f(\varphi) \neq h(\varphi; \theta)\} \rightarrow$  usata per classificazione



# Supervised learning: definizione del problema

## Misure di errore globali

Queste misure considerano tutte le  $N$  osservazioni. È importante distinguere tra **errore in-sample** (errore di train) ed **errore out-of-sample** (errore di validazione o test)

### Errore in-sample

Errore che il modello fa sugli  $N$  **dati osservati a disposizione**, che sono stati usati per stimarlo

$$E_{\text{in}}(h(\boldsymbol{\theta})) \equiv J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N \ell(f(\boldsymbol{\varphi}), h(\boldsymbol{\varphi}; \boldsymbol{\theta}))$$

### Errore out-of-sample

Errore che il modello fa sull'intero dominio di  $f$  (quindi anche **dati che non ho osservato**)

$$E_{\text{out}}(h(\boldsymbol{\theta})) = \mathbb{E}_{\boldsymbol{\varphi}}[\ell(f(\boldsymbol{\varphi}), h(\boldsymbol{\varphi}; \boldsymbol{\theta}))]$$

# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati

## **3. Feasibility of learning**

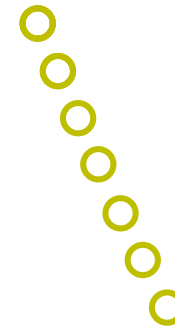
4. Bias-variance tradeoff
5. Learning curves
6. Overfitting
7. Regularizzazione
8. Validazione, cross-validazione e formule di complessità ottima
9. Esercizi con codice



# QUIZ!

**Domanda:** Quale funzione ha generato i punti in figura?

**Risposta:** \_\_\_\_\_



# QUIZ!

**Domanda:** Come vengono classificati i punti in figura?

● ○ ○  $f = ?$

☐  $f = 1$

☐  $f = 0$

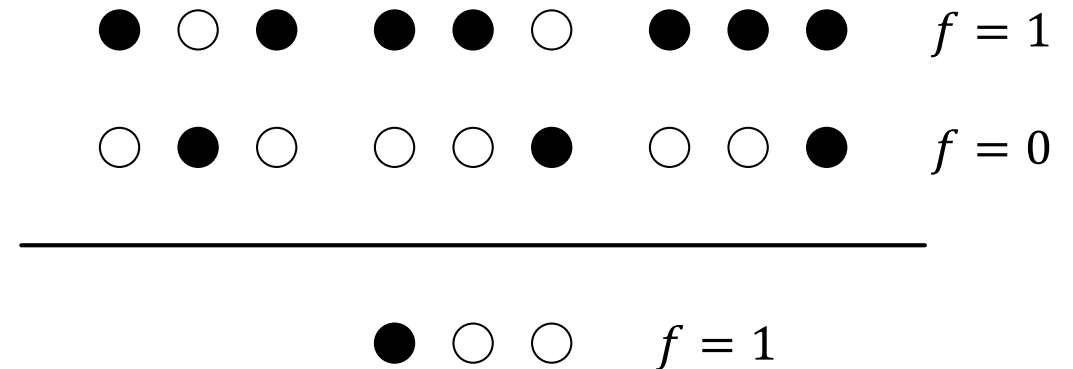
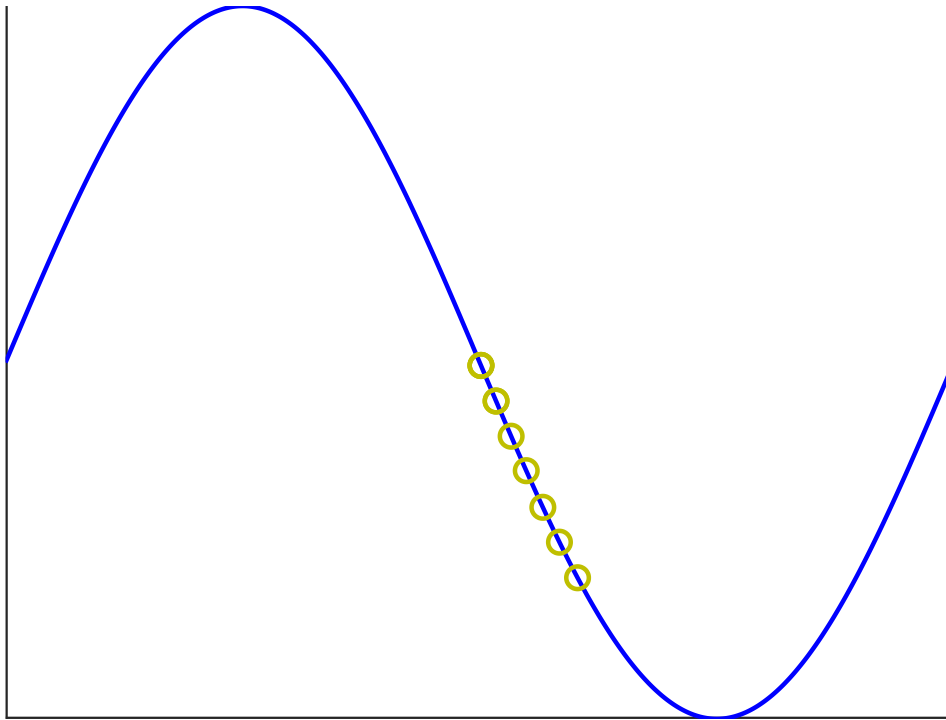
● ○ ● ● ● ○ ● ● ●  $f = 1$

○ ● ○ ○ ○ ● ○ ○ ●  $f = 0$

● ○ ○  $f = ?$

# Feasibility of learning

**Non è possibile** conoscere con certezza come sarà il comportamento della funzione  $f$  su punti che non ho osservato (problema dell'induzione di Hume)



# Feasibility of learning

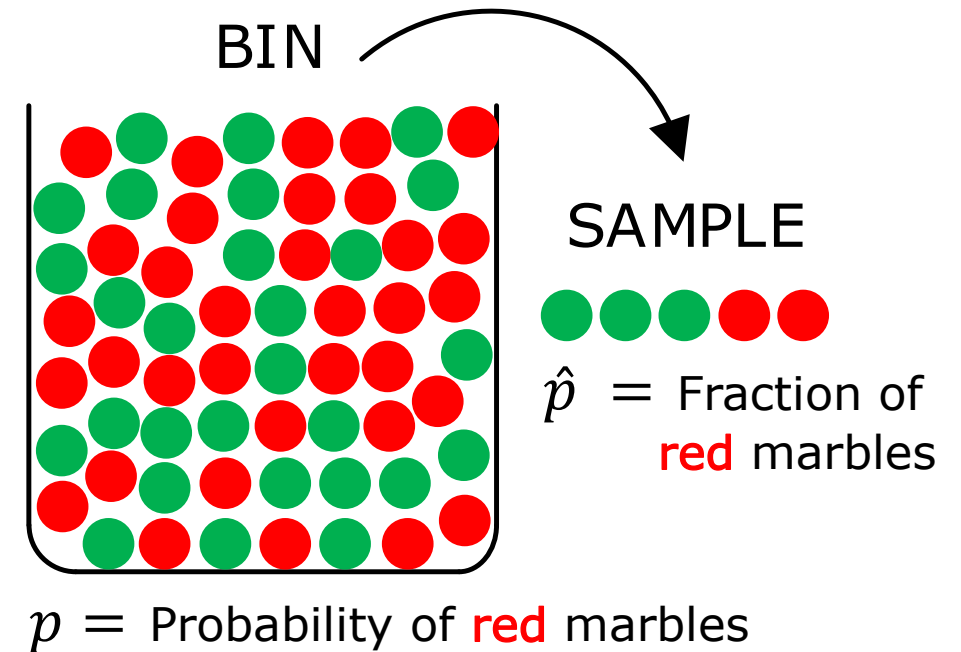
Focalizziamoci sul caso **supervised learning, classificazione binaria**

**Problema:** stimare una funzione ignota  $f$

**Risposta:** Impossibile ☹️. La funzione  $f$  può assumere qualsiasi valore al di fuori dei dati che abbiamo a disposizione

## Consideriamo il seguente esempio

- Prendiamo un'urna con delle biglie **rosse** e **verdi**
- $\mathbb{P}[\text{prendere una biglia rossa}] = p$
- Il valore di  $p$  non è noto
- Estraiamo  $N$  biglie
- Frazione di biglie rosse nel campione estratto =  $\hat{p}$





# $\hat{p}$ ci dice qualcosa riguardo a $p$ ?

**NO!**

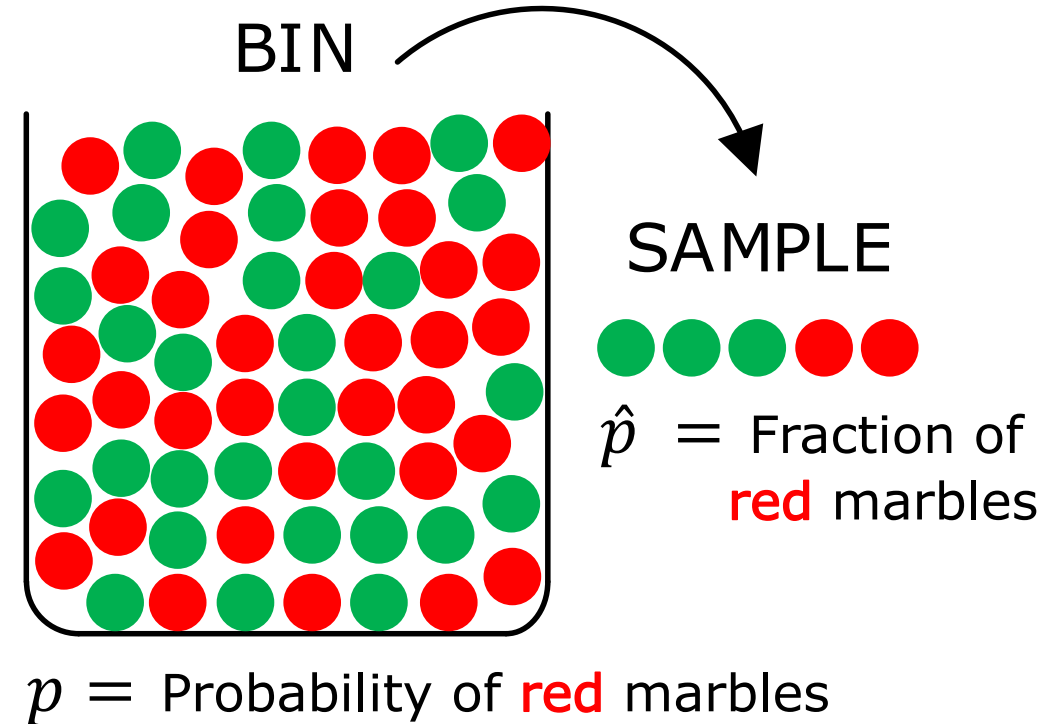
Il campione può essere per la maggior parte **verde** mentre il contenuto dell'urna potrebbe essere per la maggior parte **rosso**

**POSSIBILE**

**SÌ!**

Se estraggo tante biglie, il valore  $\hat{p}$  sarà «vicino» al valore di  $p$

**PROBABILE**



# Connessione con il learning di modelli dai dati

**Urna:** l'incognita è un **numero**  $p$

**Learning:** l'incognita è una **funzione**  $f: \mathbb{R}^{d \times 1} \rightarrow \mathcal{Y}$

Supponiamo che ogni biglia  $\bullet$  rappresenti un punto di input  $\varphi \in \mathbb{R}^{d \times 1}$

Per una **specifica ipotesi**  $h \in \mathcal{M}$  ed un punto  $\varphi$ :

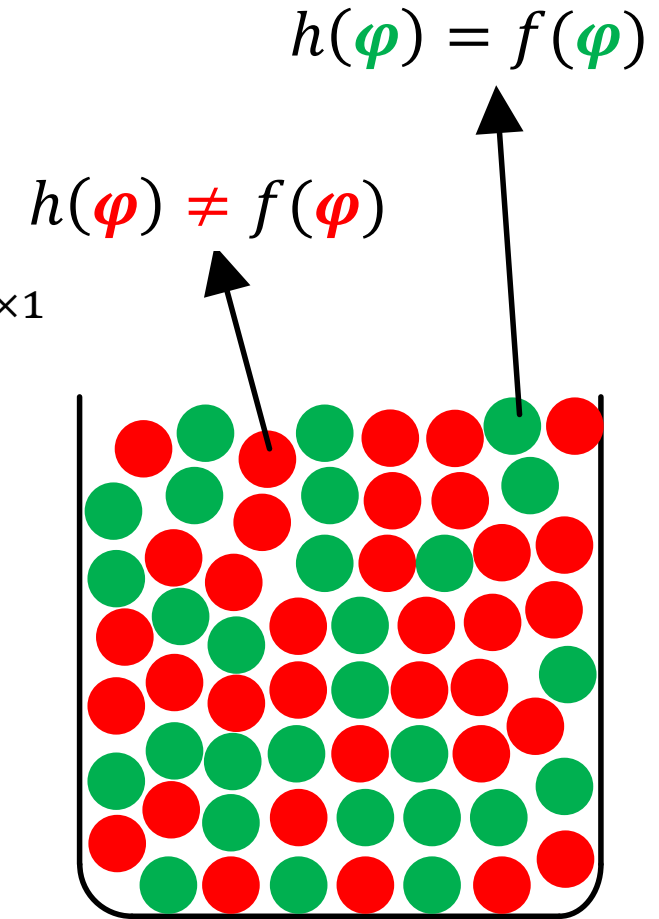
biglia verde  $\bullet \rightarrow h$  **classifica correttamente**  $h(\varphi) = f(\varphi)$

biglia rossa  $\bullet \rightarrow h$  **sbaglia a classificare**  $h(\varphi) \neq f(\varphi)$

Sia  $p$  che  $\hat{p}$  dipendono dalla particolare ipotesi  $h$ :

$\hat{p} \rightarrow$  errore sample in-sample  $E_{\text{in}}(h)$

$p \rightarrow$  errore out-of-sample  $E_{\text{out}}(h)$



# Connessione con il learning di modelli dai dati

Tramite la similitudine delle biglie e dell'urna abbiamo detto che:

$\hat{p} \rightarrow$  errore sample in-sample  $E_{\text{in}}(h)$

$p \rightarrow$  errore out-of-sample  $E_{\text{out}}(h)$

Nel caso delle biglie e dell'urna, ciò che ci interessava veramente stimare era  $p$ , non  $\hat{p}$

Nel caso del learning di modelli, ciò che ci interessa veramente stimare è  $E_{\text{out}}$ , non  $E_{\text{in}}$ , in quanto  $E_{\text{in}}$  **non è un buon indicatore della bontà del modello**

# Connessione con il learning EFFETTIVO di modelli

In uno scenario di learning reale, la funzione  $h$  **non è fissata a-priori**

- *L'algoritmo di learning* è usato per scandagliare lo spazio delle ipotesi  $\mathcal{M}$ , al fine di trovare la miglior  $h \in \mathcal{M}$  che **approssima bene i dati osservati** → chiamiamo questa ipotesi  $g$
- Con tante ipotesi in  $\mathcal{M}$ , c'è un rischio maggiore di trovare una funzione  $g$  **che «fa bene»** sui dati osservati **solo per caso** → la funzione può spiegare benissimo i dati misurati ma fare malissimo su dati nuovi

Esiste quindi **tradeoff** tra **approssimazione** e **generalizzazione**. Si vuole:

- avere un buon modello sui dati **misurati** (training set)
- avere un buon modello su dati **non visti** (e quindi non usati per la stima del modello)

La quantità  $E_{\text{out}}(g) - E_{\text{in}}(g)$  è chiamata **errore di generalizzazione**

# Approssimazione vs. Generalizzazione

L'obiettivo finale è avere un piccolo  $E_{\text{out}}$ : buona approssimazione di  $f$  out-of-sample

Spazio delle ipotesi  $\mathcal{M}$  **PIÙ** complesso  $\Rightarrow$  Migliori possibilità di **approssimare**  $f$  in-sample

Spazio delle ipotesi  $\mathcal{M}$  **MENO** complesso  $\Rightarrow$  Migliori possibilità di **generalizzare**  $f$  out-of-sample

Il caso ideale sarebbe avere uno spazio delle ipotesi  $\mathcal{M}$  che contiene solo la funzione  $f$

$\mathcal{M} = \{ f \}$  Vincere un biglietto della lotteria 😊

# Approssimazione vs. Generalizzazione

L'esempio mostra:

- **Fit perfetto** sui dati di train

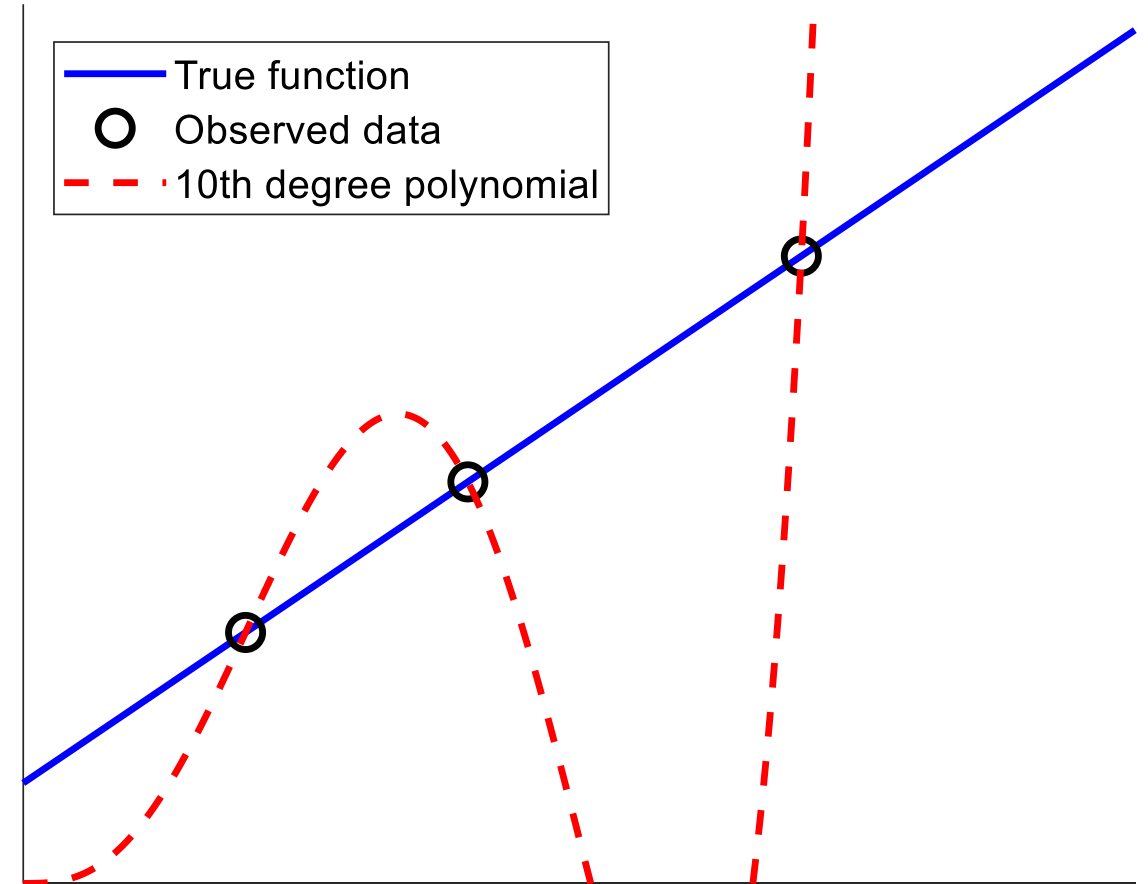
↓

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

- **Fit pessimo** on out of sample (test) data

↓

$$E_{\text{out}} \text{ **enorme**}$$





# Teoria della generalizzazione

Esiste una **teoria della generalizzazione** che studia i casi in cui è **probabile** generalizzare

- Il concetto da portare a casa è che **il learning è fattibile in modo probabilistico**
- Se siamo in grado di affrontare il tradeoff approssimazione-generalizzazione, possiamo **dire con alta probabilità che l'errore di generalizzazione è piccolo**

Un modo per studiare questo tradeoff è valutare i concetti di **bias** e **varianza** di un **modello di learning**

L'approccio **bias-varianza** scompone  $E_{\text{out}}$  in:

1. Quanto bene  $\mathcal{M}$  può approssimare  $f$  → **Bias**
2. Quanto bene riusciamo a scegliere una buona  $h \in \mathcal{M}$ , usando i dati → **Variance**



# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati
3. Feasibility of learning
- 4. Bias-variance tradeoff**
5. Learning curves
6. Overfitting
7. Regularizzazione
8. Validazione, cross-validazione e formule di complessità ottima
9. Esercizi con codice



# Bias e varianza di un modello di learning

Supponiamo di osservare i **dati senza rumore**, cioè che  $y = f(\boldsymbol{\varphi})$ . L'errore out-of-sample può essere espresso come (rendendo esplicita la dipendenza di  $g$  da  $\mathcal{D}$ )

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

L'errore **out-of-sample atteso** del modello è indipendente dalla particolare realizzazione dei dati utilizzati per stimare  $g^{(\mathcal{D})}$ :

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

# Bias e varianza di un modello di learning

Concetriamoci su  $\mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\boldsymbol{\varphi}) - f(\boldsymbol{\varphi}) \right)^2 \right]$ . Definiamo **l'ipotesi «media»**  $\bar{g}(\boldsymbol{\varphi}) = \mathbb{E}_{\mathcal{D}} [g^{(\mathcal{D})}(\boldsymbol{\varphi})]$

Questa «ipotesi media» può essere interpretata come l'ipotesi che deriva dall'usare  $K$  dataset  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$  e costruendola come  $\bar{g}(\boldsymbol{\varphi}) \approx \frac{1}{K} \sum_{k=1}^K g^{(\mathcal{D}_k)}(\boldsymbol{\varphi})$

questo è uno strumento concettuale e  $\bar{g}$  non ha bisogno di appartenere all'insieme delle ipotesi  $\mathcal{M}$

Abbiamo quindi:

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

# Bias e varianza di un modello di learning

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

Quindi;

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

Questo concetto è analogo al concetto di Mean Squared Error (MSE) per uno stimatore parametrico (Lezione 02)

$$= \text{bias}^2 + \text{var}$$

# Bias e varianza di un modello di learning

## Interpretazione

- Il termine di **bias**<sup>2</sup>  $(\bar{g}(\varphi) - f(\varphi))^2$  misura quanto il nostro modello (cioè la nostra funzione stimata  $g$ ) è «lontano» dalla funzione target  $f$

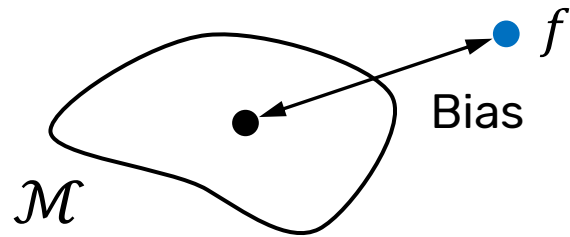
Infatti,  $\bar{g}$  ha il vantaggio di apprendere da un numero illimitato di datasets. Quindi,  $\bar{g}$ , nella capacità di approssimare  $f$ , è limitata solo dai «limiti» di  $\mathcal{M}$

**Esempio:** sia  $\mathcal{M}$  l'insieme delle rette. Per quanti dati possa avere, una retta non potrà mai approssimare bene una curva...

- Il termine **varianza**  $\mathbb{E}_{\mathcal{D}} \left[ \left( g^{(\mathcal{D})}(\varphi) - \bar{g}(\varphi) \right)^2 \right]$  misura quanto  $g^{(\mathcal{D})}$  si «disperde» da  $\bar{g}$ , e può essere pensata come quanto l'ipotesi finale  $g^{(\mathcal{D})}$  differisca dall'ipotesi «migliore» (ovvero quella «media»)

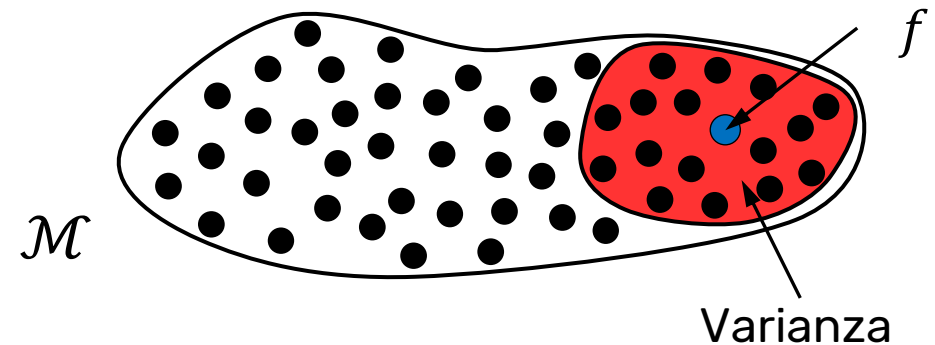
# Bias e varianza di un modello di learning

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



**Set di modelli** (spazio delle ipotesi) **molto PICCOLO**. Poiché esiste una sola ipotesi, sia la funzione media  $\bar{g}$  che l'ipotesi finale  $g^{(\mathcal{D})}$  saranno uguali, per qualsiasi set di dati. Quindi,  $\text{var} = 0$ . Il bias dipenderà esclusivamente da quanto bene questa singola ipotesi si avvicina al target  $f$  e, a meno che non siamo estremamente fortunati, **ci aspettiamo un grande bias**

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



**Set di modelli** (spazio delle ipotesi) **molto GRANDE**. La funzione target sta in  $\mathcal{M}$ . Diversi set di dati porteranno a diverse ipotesi  $g^{(\mathcal{D})}$  che concordano con set di dati a disposizione, e queste ipotesi sono sparse intorno alla regione rossa. Quindi,  $\text{bias} \approx 0$  poichè in media  $g^{(\mathcal{D})}$  è vicina ad  $f$ . **La varianza è grande** (rappresentato euristicamente dalla dimensione della regione rossa)

# Bias e varianza di un modello di learning

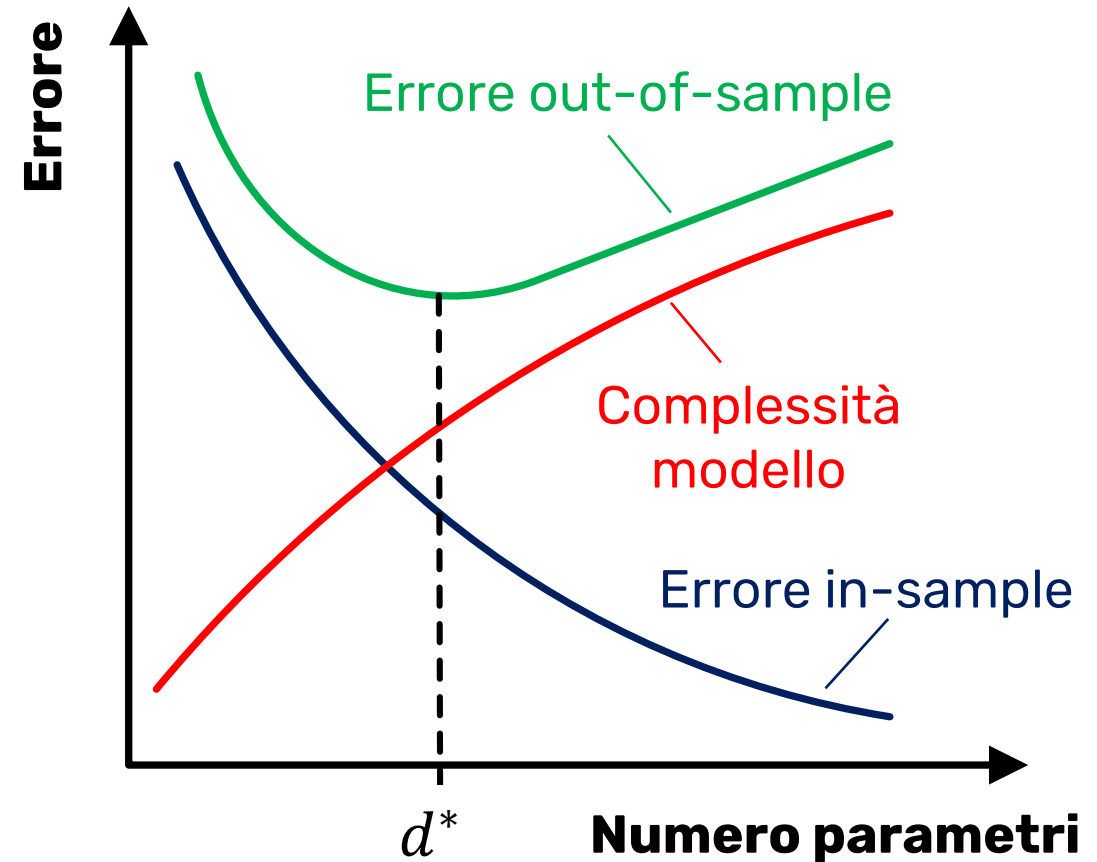
## Regola euristica

Quanti punti  $N$  sono richiesti per assicurarsi un **buona probabilità di generalizzare?**

$$N \geq 10 \cdot \text{numero parametri modello}$$

## Principio generale

La «**complessità del modello**» deve seguire il  
**numero di dati**, non la **complessità della**  
**funzione target**





# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati
3. Feasibility of learning
4. Bias-variance tradeoff

## **5. Learning curves**

6. Overfitting
7. Regularizzazione
8. Validazione, cross-validazione e formule di complessità ottima
9. Esercizi con codice

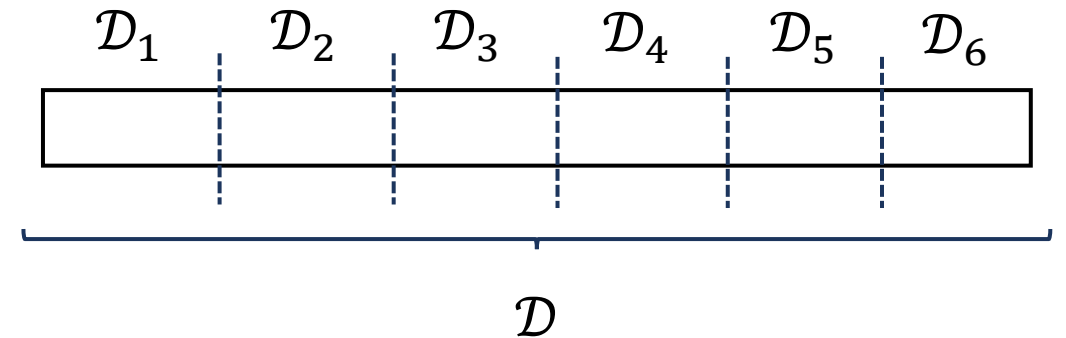


# Learning curves

Le **learning curves** sono uno strumento grafico per capire se un modello di learning soffre di **problemi di bias o varianza**

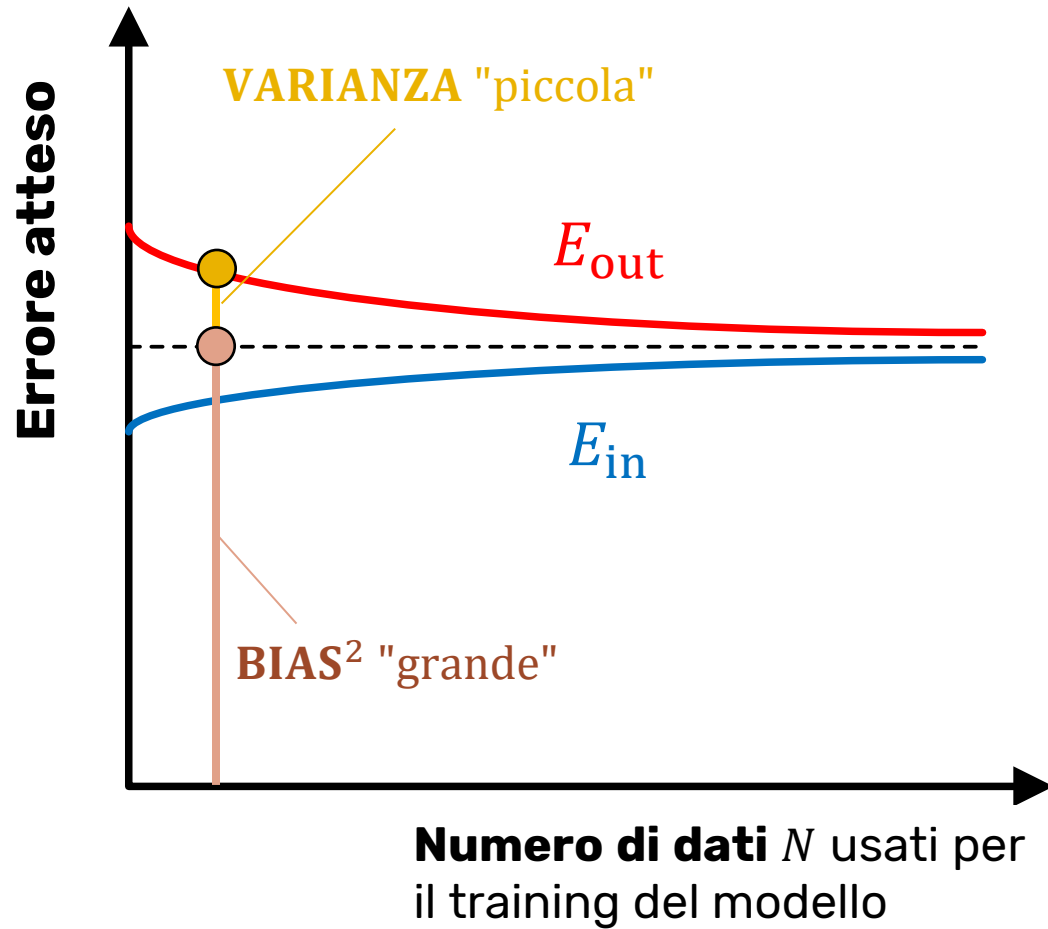
L'idea è di rappresentare, al **variare del numero di dati**  $N$  usati **per stimare** il modello:

- l'errore out-of-sample **atteso**  $\mathbb{E}_{\mathcal{D}}[E_{\text{out}}(g^{\mathcal{D}})]$
- l'errore in-sample **atteso**  $\mathbb{E}_{\mathcal{D}}[E_{\text{in}}(g^{\mathcal{D}})]$

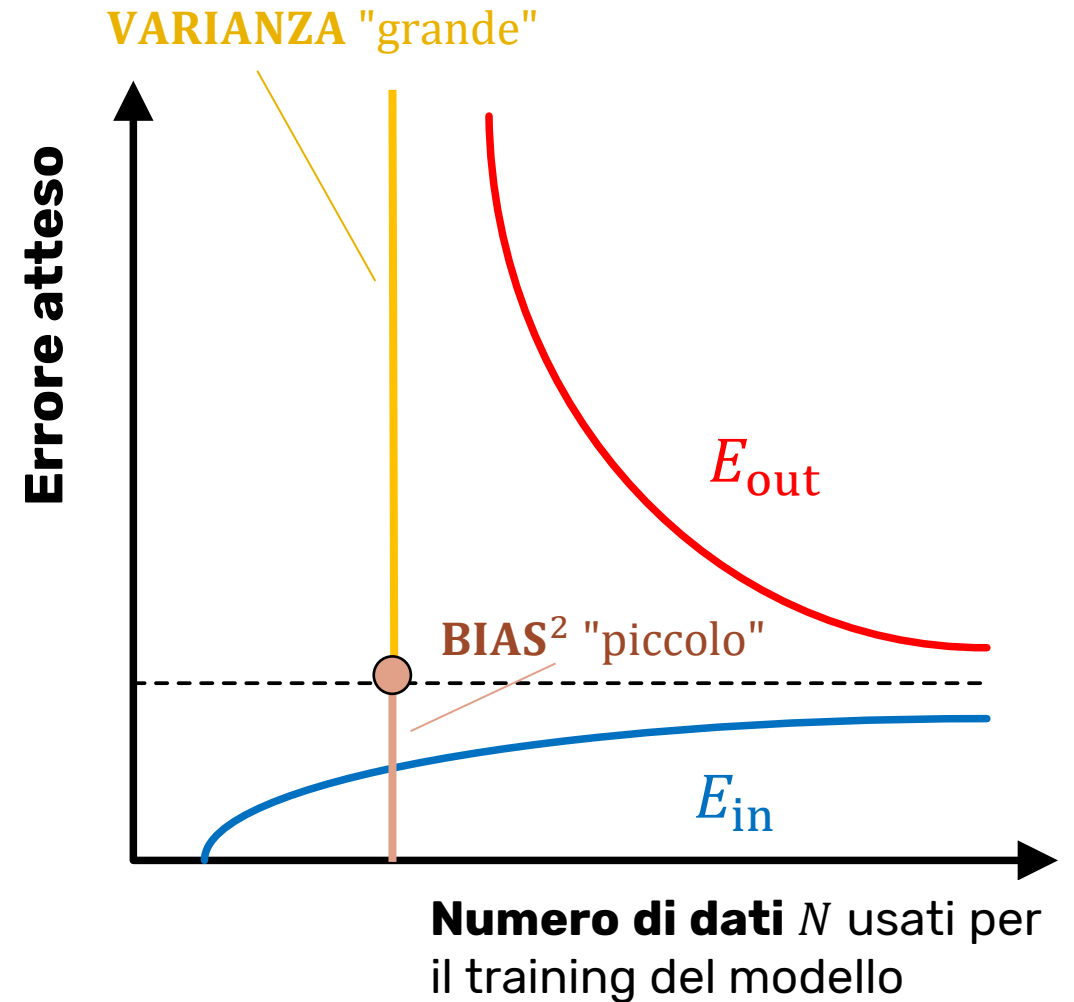


In pratica, le curve vengono calcolate **usando un solo dataset**, oppure dividendolo in più parti e prendendo la «curva media» risultante dai vari sub-datasets

# Learning curves



**Modello «semplice»**



**Modello «complesso»**

# Learning curves

## Interpretazione

- Il **bias** può essere presente quando l'errore atteso è piuttosto elevato e  $E_{in}$  è simile a  $E_{out}$
- Quando è presente **bias**, è improbabile che ottenere più dati aiuti
- La **varianza** può essere presente quando c'è un tanto divario tra  $E_{in}$  e  $E_{out}$
- Quando è presente **varianza**, è probabile che ottenere più dati sia d'aiuto

## Risolvere un problema di **bias**

- Aggiungere features, per esempio combinazioni di features originarie
- Boosting

## Risolvere un problema di **varianza**

- Usare meno features
- Acquisire più dati
- **Usare la regolarizzazione**
- Bagging

# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati
3. Feasibility of learning
4. Bias-variance tradeoff
5. Learning curves
- 6. Overfitting**
7. Regularizzazione
8. Validazione, cross-validazione e formule di complessità ottima
9. Esercizi con codice



# Overfitting

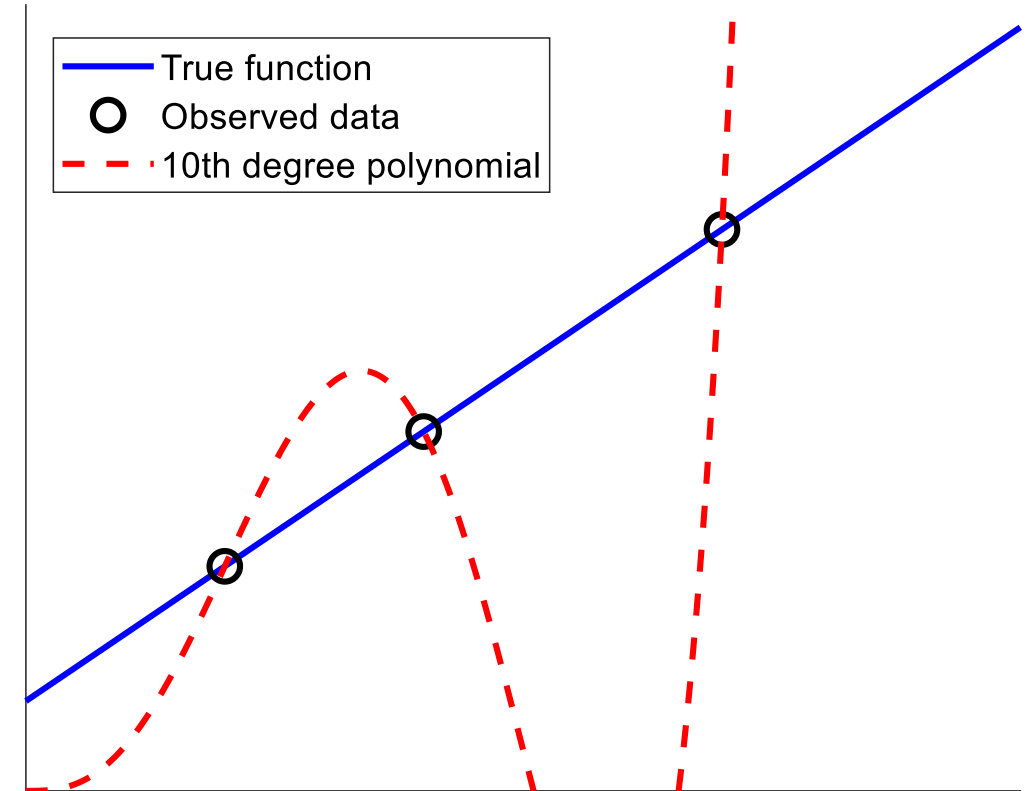


# Overfitting

Abbiamo già incontrato il fenomeno dell'**overfitting** quando abbiamo parlato del **tradeoff approssimazione-generalizzazione**

Abbiamo visto come dobbiamo **usare modelli più semplici se abbiamo pochi dati**, indipendentemente dalla complessità della funzione target

Introduciamo ora un'altra causa di overfitting: il **rumore stocastico sui dati** in uscita  $y$

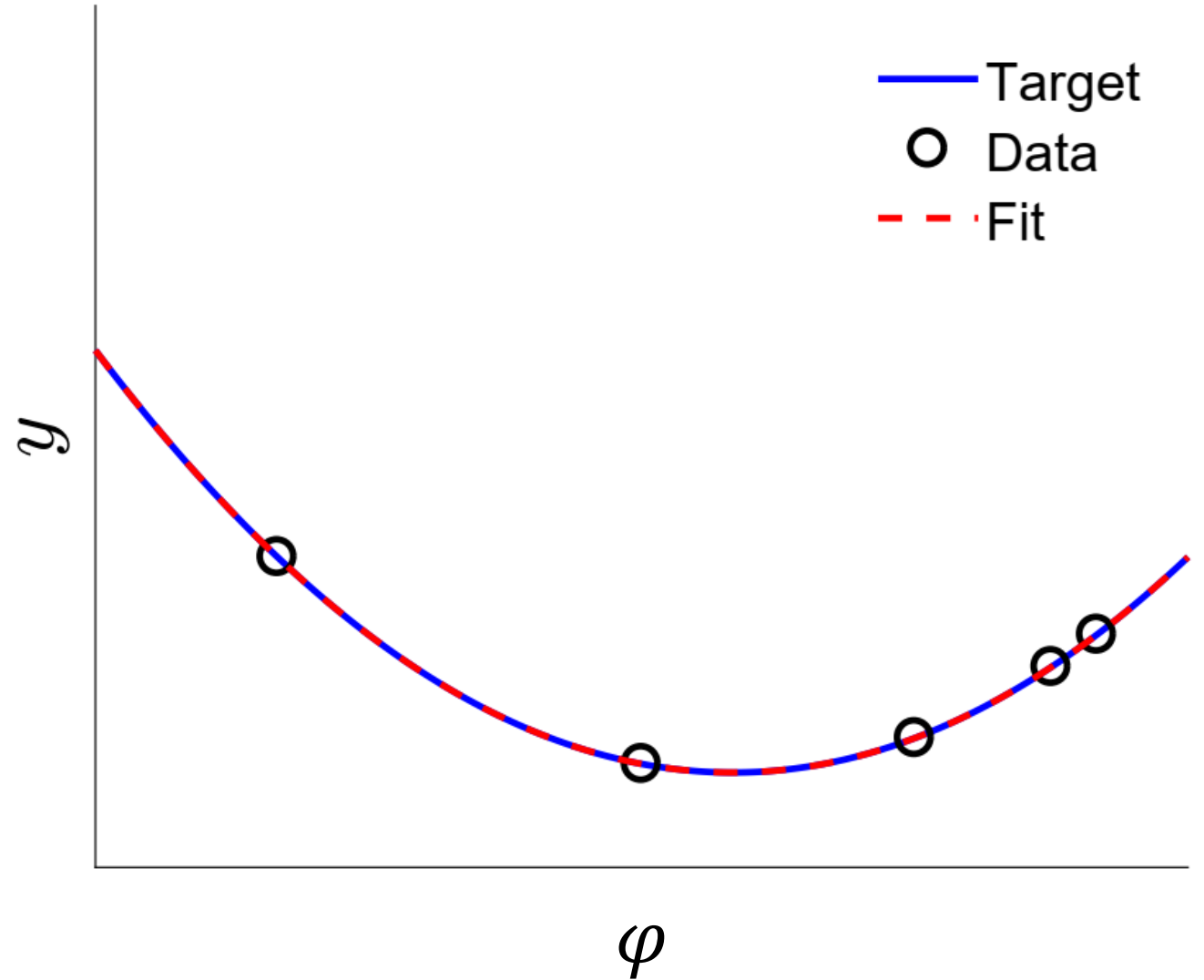


# Esempio di overfitting

Consideriamo il seguente esempio:

- Funzione semplice da imparare
- $N = 5$  punti
- Modello: polinomio del 4° ordine

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

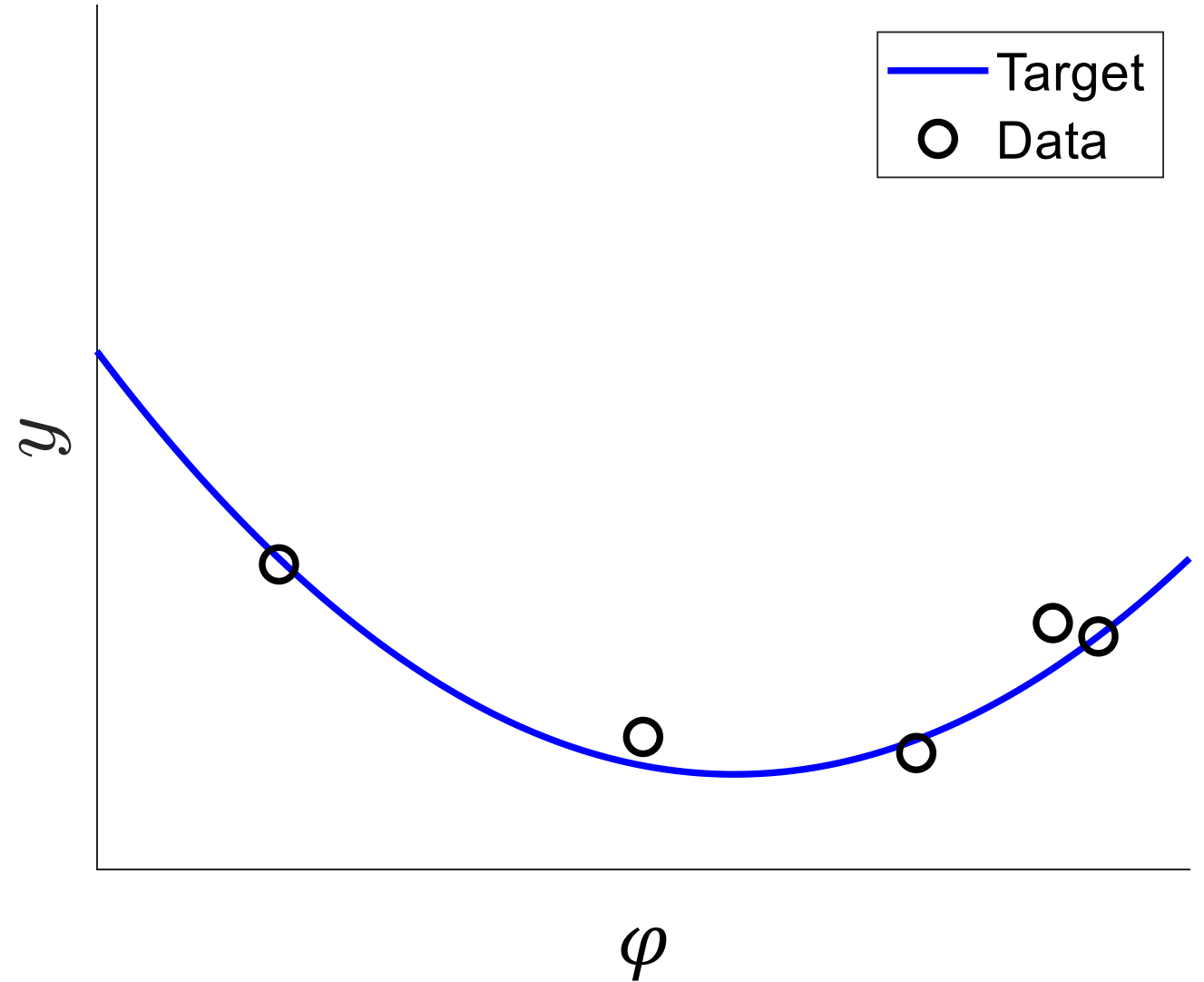




# Esempio di overfitting

Consideriamo il seguente esempio:

- Funzione semplice da imparare
- $N = 5$  punti **rumorosi**
- Modello: polinomio del 4° ordine

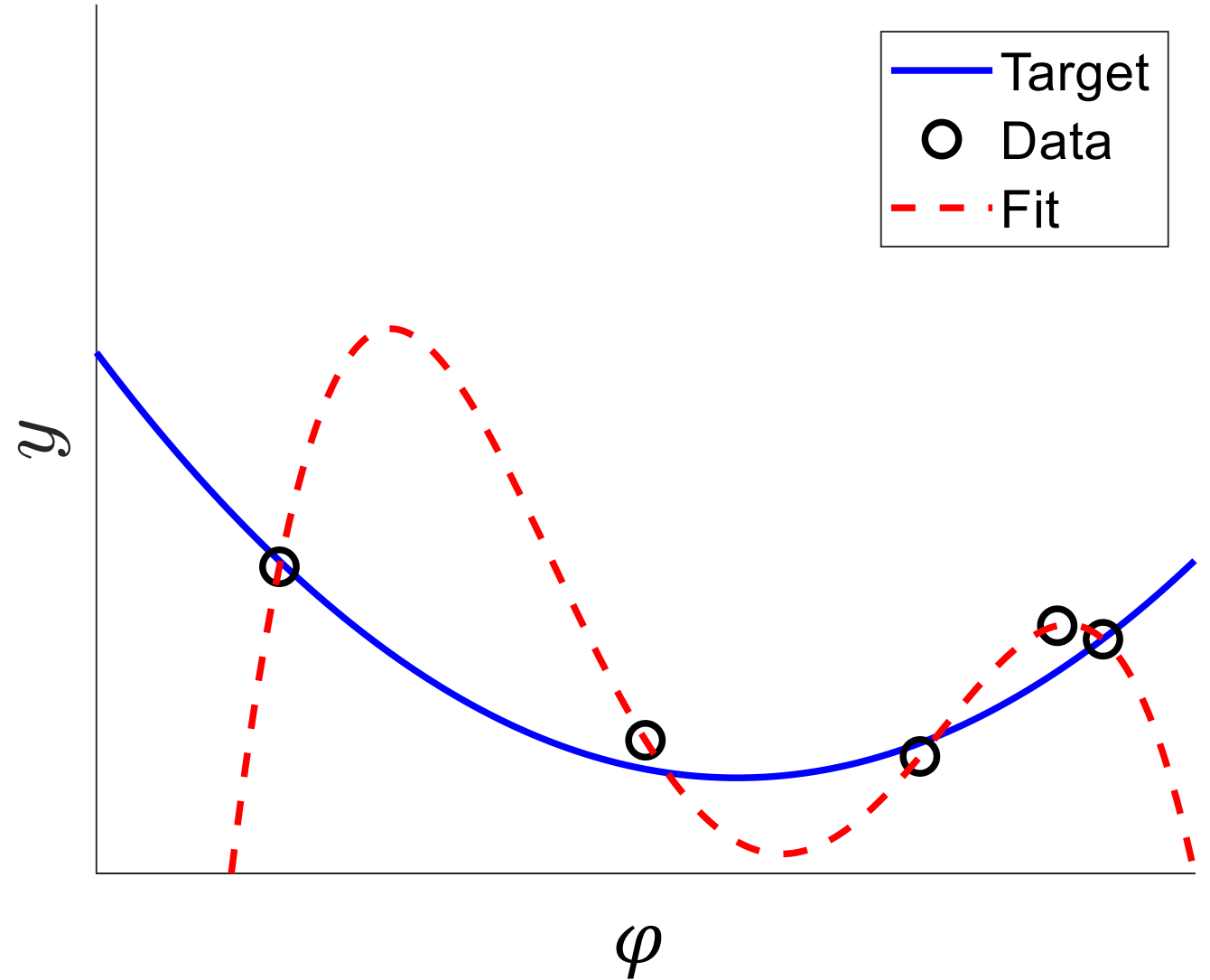


# Esempio di overfitting

Consideriamo il seguente esempio:

- Funzione semplice da imparare
- $N = 5$  punti **rumorosi**
- Modello: polinomio del 4° ordine

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



# Esempio: studente che deve apprendere dei concetti

Per comprendere in modo intuitivo il **fenomeno dell'overfitting**, consideriamo la seguente similitudine

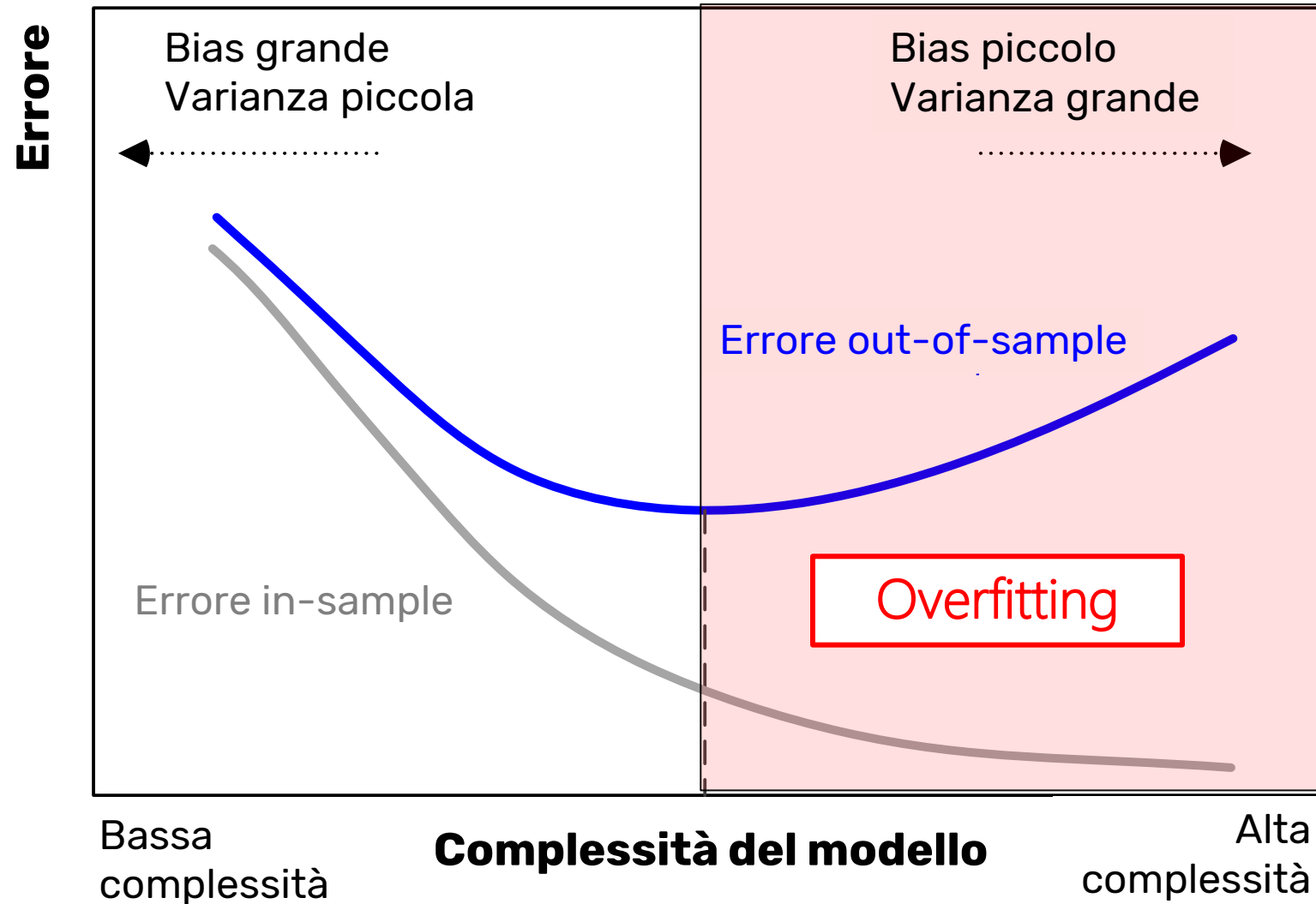
Il docente di un corso fornisce degli esercizi risolti al fine di insegnare a risolvere un problema. Gli esercizi d'esame devono per forza **essere diversi** da quelli forniti a lezione, altrimenti il docente non è in grado di capire se lo studente (o studentessa) ha solo **imparato a memoria** come risolvere gli esercizi o se ha **appreso veramente** i concetti

Nel primo caso (imparare a memoria) lo studente (o studentessa) **non ha veramente imparato**: quando si troverà di fronte un esercizio **simile (ma diverso)** non sarà in grado di risolverlo. Lo studente (o studentessa) ha **overfittato** l'esercizio visto a lezione, **senza averne generalizzato** i concetti e quindi il metodo risolutivo

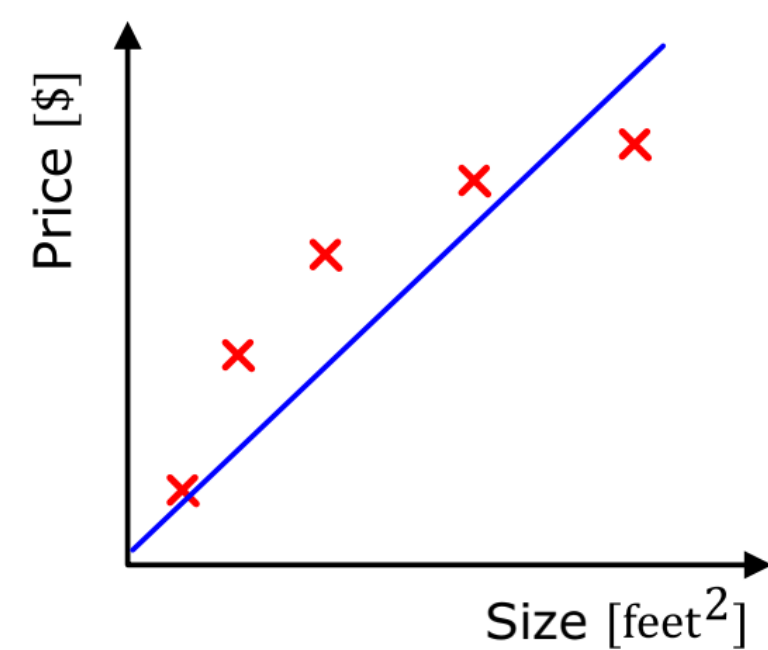


# Overfitting vs. complessità del modello

- Si parla di overfitting quando diminuire  $E_{in}$  porta ad un aumento di  $E_{out}$
- Principale fonte di non funzionamento dei modelli di learning
- L'overfitting porta a una cattiva generalizzazione
- Un modello può mostrare una cattiva generalizzazione anche se non overfitta

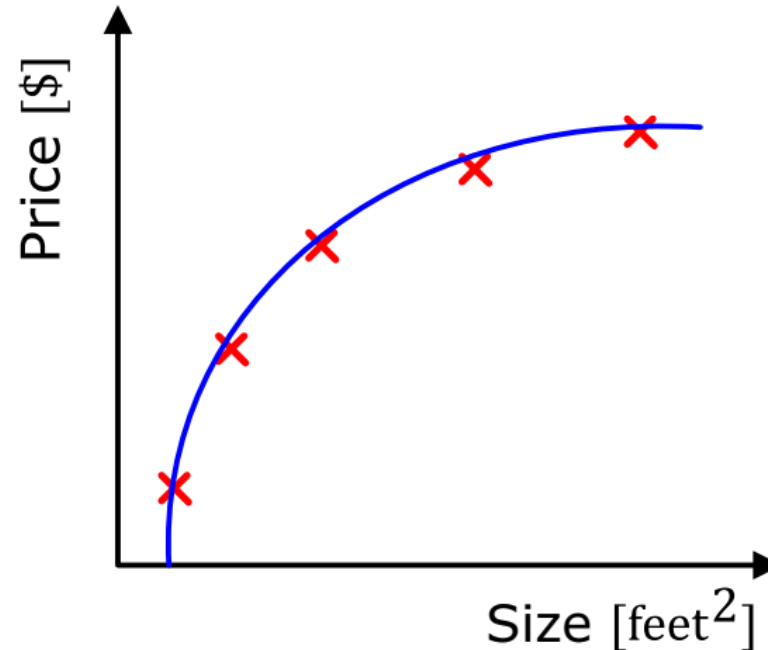


# Overfitting vs. complessità del modello

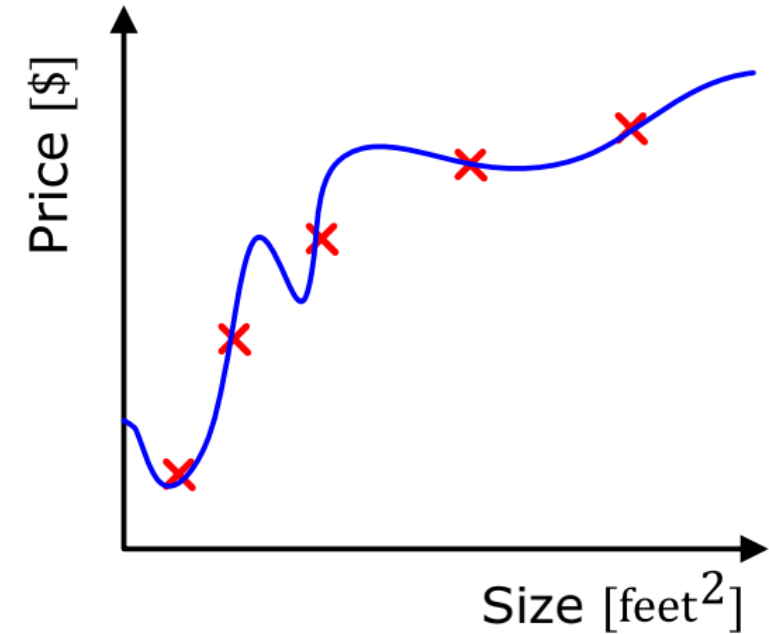


**Underfit**

**Tanto bias**



**OK**



**Overfit**

**Tanta varianza**

# Tradeoff bias – varianza: rivisitazione

Supponiamo che vi sia un **rumore stocastico** (una v.c.)  $\eta$  con media zero e varianza  $\sigma^2$  che affligge le misure, tale che  $y = f(\boldsymbol{\varphi}) + \eta$

$$\mathbb{E}_{\mathcal{D}, \boldsymbol{\varphi}, \eta} \left[ \left( g^{(\mathcal{D})}(\boldsymbol{\varphi}) - (f(\boldsymbol{\varphi}) + \eta(\boldsymbol{\varphi})) \right)^2 \right] =$$

$$= \text{bias}^2 + \text{var} + \sigma^2$$

Anzichè  $f(\boldsymbol{\varphi})$ , osserviamo  
 $y = f(\boldsymbol{\varphi}) + \eta(\boldsymbol{\varphi})$

- L'errore stocastico  $\sigma^2$  non può essere portato a zero
- Il rumore stocastico contribuisce alla varianza dell'ipotesi scelta, **causando overfitting**



**Errore «irriducibile»**

È un po' come quando  
parlavamo del limite di  
Cramer-Rao per gli stimatori...

# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati
3. Feasibility of learning
4. Bias-variance tradeoff
5. Learning curves
6. Overfitting
- 7. Regularizzazione**
8. Validazione, cross-validazione e formule di complessità ottima
9. Esercizi con codice



# Regolarizzazione

La **regolarizzazione** è la prima linea di difesa contro l'overfitting

Abbiamo visto che i **modelli più complessi** sono più inclini all'**overfitting**. Questo perché sono più «potenti» (espressivi) e quindi possono adattarsi anche al rumore

I **modelli semplici** mostrano **meno varianza** a causa della loro espressività limitata. La riduzione della varianza del modello è spesso maggiore dell'aumento del suo bias, per cui, nel complesso, **errore atteso complessivo diminuisce** ( $\text{bias}^2 + \text{var} + \sigma^2$ )

Tuttavia, se ci atteniamo solo a modelli semplici, potremmo non ottenere un'approssimazione soddisfacente della funzione target  $f$

*Come possiamo conservare i vantaggi di **entrambi** i tipi di modello?*



# Regolarizzazione

**Idea:** oltre che minimizzare il funzionale di costo di «fit del modello ai dati»  $E_{\text{in}}(\boldsymbol{\theta}) \equiv J(\boldsymbol{\theta})$ , **minimizziamo** anche la **complessità del modello**

Al posto di  $E_{\text{in}}(\boldsymbol{\theta})$ , minimizziamo un **errore aumentato**  $E_{\text{aug}}(\boldsymbol{\theta})$

$h(\cdot)$  è qualche funzione che rappresenta il nostro modello

$$E_{\text{aug}}(\boldsymbol{\theta}) = \underbrace{\frac{1}{N} \sum_{i=1}^N (y(i) - h(\boldsymbol{\varphi}(i); \boldsymbol{\theta}))^2}_{\text{Quanto male il modello fitta i dati (è un termine di errore)}} + \lambda_{\text{reg}} \cdot \Omega(\boldsymbol{\theta})$$

**Regolarizzatore:** quanto il modello è «complesso»

Il termine  $\lambda_{\text{reg}}$  (iper-parametro) **pesa l'importanza** di minimizzare  $E_{\text{in}} \equiv J(\boldsymbol{\theta})$  rispetto a minimizzare  $\Omega(\boldsymbol{\theta})$

# Esempio: regolarizzazione $L_2$

La regolarizzazione  $L_2$  penalizza la somma del quadrato dei coefficienti  $\boldsymbol{\theta} \in \mathbb{R}^{d \times 1}$

$$E_{\text{aug}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N \left( y(i) - h(\underset{d \times 1}{\boldsymbol{\varphi}(i)}; \underset{d \times 1}{\boldsymbol{\theta}}) \right)^2 + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1} \underset{1 \times 1}{(\theta_j)}^2$$

- Se questa regolarizzazione  $L_2$  viene applicata ad un problema di regressione lineare, il metodo viene chiamato **Ridge regression**
- L'intercetta  $\theta_0$  talvolta non si penalizza. In questo caso  $j$  partirebbe da 1
- Questo problema può anche essere visto come un problema di **ottimizzazione vincolata**

# Esempio: regolarizzazione $L_2$

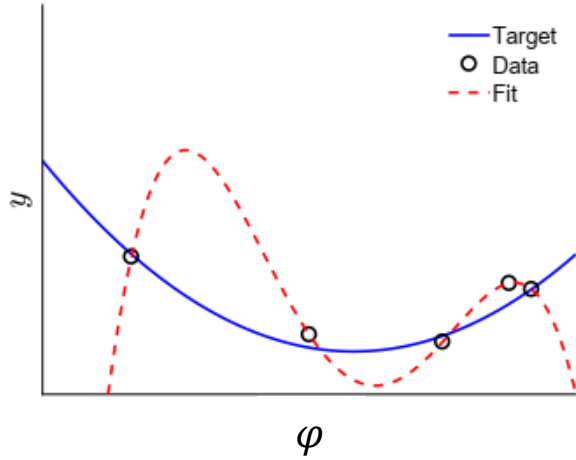
$$\begin{aligned} \text{minimize } E_{\text{in}}(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^N \left( f(\boldsymbol{\varphi}(i)) - h(\boldsymbol{\varphi}(i); \boldsymbol{\theta}) \right)^2 \\ \text{subject to } \boldsymbol{\theta}^\top \boldsymbol{\theta} &\leq c \\ &\quad \begin{matrix} 1 \times d & d \times 1 & 1 \times 1 \end{matrix} \end{aligned}$$

- Con questa interpretazione, stiamo esplicitamente vincolando i coefficienti  $\boldsymbol{\theta}$  a **non assumere valori grandi**
- C'è una relazione tra  $c$  e  $\lambda_{\text{reg}}$  in modo tale che se  $c \uparrow$ , allora  $\lambda \downarrow$

Infatti,  $c$  **più grande** significa che i **pesi** possono essere **maggiori**. Questo è uguale a impostare per un  $\lambda_{\text{reg}}$  **inferiore**, perché il termine di regolarizzazione sarà meno importante e quindi i pesi non verranno ridotti così tanto

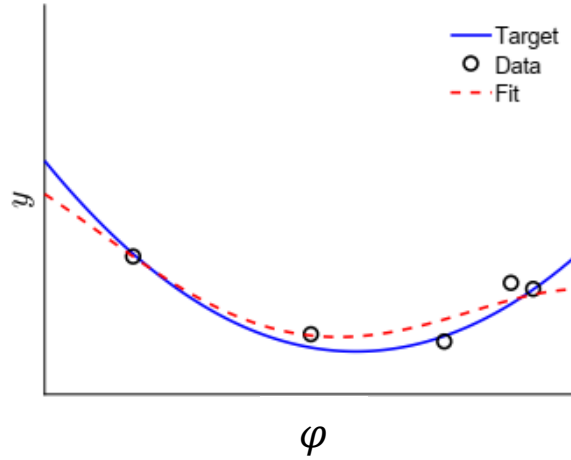
# Effetto dell'iperparametro di regolarizzazione $\lambda$

$$\lambda_{\text{reg}_1}$$

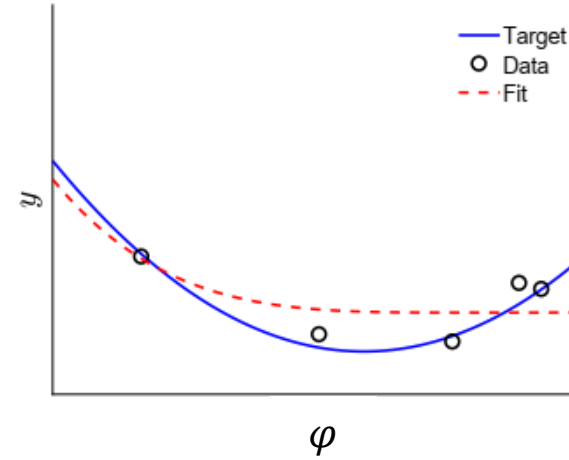


**Overfit**

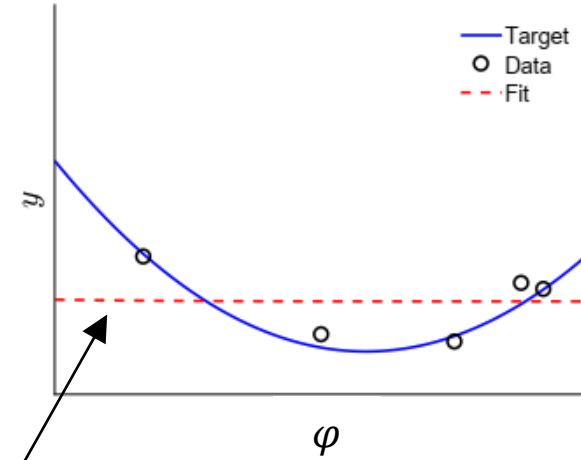
$$\lambda_{\text{reg}_2} > \lambda_{\text{reg}_1}$$



$$\lambda_{\text{reg}_3} > \lambda_{\text{reg}_2}$$



$$\lambda_{\text{reg}_4} > \lambda_{\text{reg}_3}$$

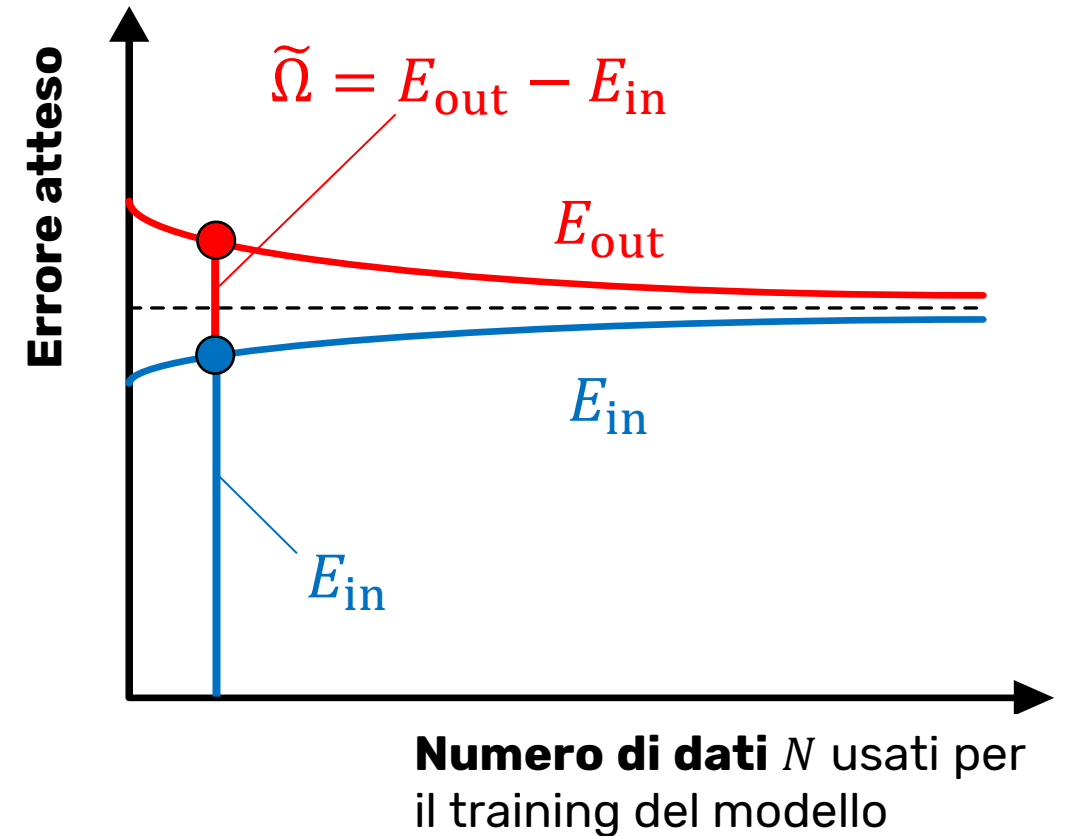
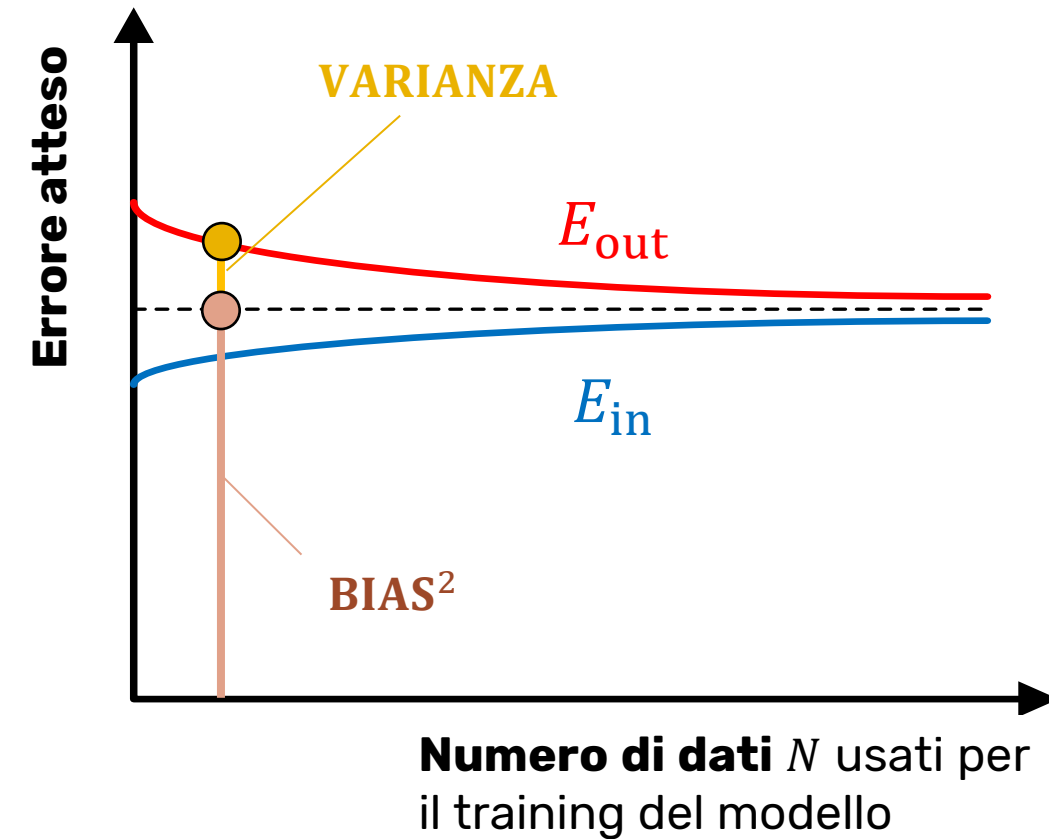


**Underfit**

Se regolarizzo troppo, imparerò la funzione più semplice possibile, ovvero una retta orizzontale (costante) con intercetta  $\theta_0$

# Intuizione sull'importanza di $E_{\text{aug}}$ rispetto a $E_{\text{in}}$

Minimizzare  $E_{\text{aug}}$  rispetto ad  $E_{\text{in}}$  conduce ad un modello migliore (ovvero un modello con miglior capacità di generalizzare e quindi con  $E_{\text{out}}$  minore)



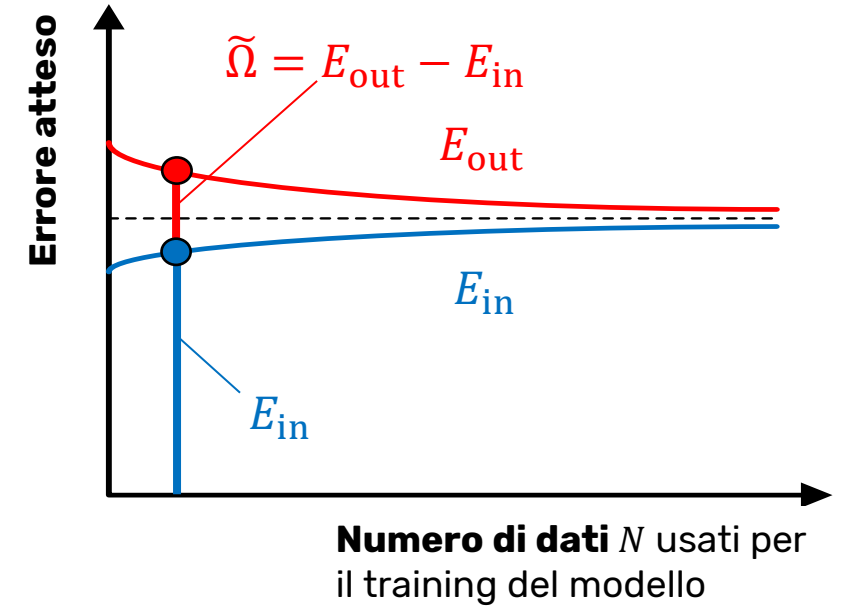
# Intuizione sull'importanza di $E_{\text{aug}}$ rispetto a $E_{\text{in}}$

Dal grafico precedente, oltre che tramite bias e varianza, possiamo interpretare  $E_{\text{out}}$  come la somma di due contributi:

$$E_{\text{out}}(\boldsymbol{\theta}) = E_{\text{in}}(\boldsymbol{\theta}) + \tilde{\Omega}(\boldsymbol{\theta})$$

Ricordando la definizione di  $E_{\text{aug}}$  abbiamo

$$E_{\text{aug}}(\boldsymbol{\theta}) = E_{\text{in}}(\boldsymbol{\theta}) + \lambda_{\text{reg}} \Omega(\boldsymbol{\theta})$$



L'errore  $E_{\text{aug}}$  è **migliore** rispetto ad  $E_{\text{in}}$  come proxy per  $E_{\text{out}}$

# Intuizione sull'importanza di $E_{\text{aug}}$ rispetto a $E_{\text{in}}$

Il Santo Graal del machine learning sarebbe avere  
un'espressione di  $E_{\text{out}}$  da minimizzare

- In questo modo, sarebbe possibile minimizzare direttamente l'errore out-of-sample invece di quello in-sample (o dell'errore aumentato)
- La **regolarizzazione** aiuta nello stimare la quantità  $\Omega(\theta)$ , che, sommata ad  $E_{\text{in}}$ , fornisce  $E_{\text{aug}}$ , il quale è una stima di  $E_{\text{out}}$

# Scelta del termine di regolarizzazione

Esistono diversi tipi di regolarizzazione. I più usati sono:

- **Regolarizzazione  $L_2$** : chiamata anche penalità **Ridge**  $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} (\theta_j)^2$
- **Regolarizzazione  $L_1$** : chiamata anche penalità **Lasso**  $\Omega(\boldsymbol{\theta}) = \sum_{j=0}^{d-1} |\theta_j|$
- **Regolarizzazione elastic-net**:  $\Omega(\boldsymbol{\theta}) = \beta \sum_{j=0}^{d-1} (\theta_j)^2 + (1 - \beta) \sum_{j=0}^{d-1} |\theta_j|$

La penalità **Ridge** tende a ridurre tutti i coefficienti a un **valore inferiore**

La penalità **Lasso** tende a portare più coefficienti **esattamente a zero**



# Scelta del termine di regolarizzazione

## Ridge

Notiamo che  $\theta_1$  e  $\theta_2$  sono «piccoli»

$$E_{\text{in}}(\boldsymbol{\theta}) \equiv J(\boldsymbol{\theta})$$

Curve di livello  
(supponiamo funzione  
di costo  $J(\boldsymbol{\theta})$  convessa)

$$(\theta_1)^2 + (\theta_2)^2 \leq c$$

Vincoli

## Lasso

Notiamo che  $\theta_1 = 0$  e  $\theta_2$  «piccolo»

Stima di  $\boldsymbol{\theta}$  senza  
regolarizzazione

Stima di  $\boldsymbol{\theta}$  con  
regolarizzazione

$$|\theta_1| + |\theta_2| \leq c$$

# Regolarizzazione e bias-varianza tradeoff

Gli effetti della regolarizzazione possono essere osservati nei termini di **bias** e **varianza**:

- La regolarizzazione **aumenta di poco il bias** (perché ottengo un modello più semplice) al fine di **ridurre considerevolmente la varianza** del modello di learning
- La regolarizzazione porta ad avere **ipotesi più «smooth»**, regolari, riducendo il rischio di overfitting
- L'iperparametro di regolarizzazione  $\lambda_{\text{reg}}$  deve essere scelto in modo specifico per ogni tipo di regolarizzatore. Solitamente si usa una procedura come la **validazione** o la **cross-validazione**

# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati
3. Feasibility of learning
4. Bias-variance tradeoff
5. Learning curves
6. Overfitting
7. Regularizzazione
- 8. Validazione, cross-validazione e formule di complessità ottima**
9. Esercizi con codice



# Validazione

L'errore out-of-sample può essere visto come:

$$E_{\text{out}}(\boldsymbol{\theta}) = E_{\text{in}}(\boldsymbol{\theta}) + \text{penalità per la complessità del modello}$$

## Regolarizzazione

$$E_{\text{out}}(\boldsymbol{\theta}) = E_{\text{in}}(\boldsymbol{\theta}) + \underbrace{\text{penalità per la complessità del modello}}$$

**La REGOLARIZZAZIONE stima questa quantità**

## Validazione

$$\underbrace{E_{\text{out}}(\boldsymbol{\theta})}_{\text{La VALIDAZIONE}} = E_{\text{in}}(\boldsymbol{\theta}) + \text{penalità per la complessità del modello}$$

**La VALIDAZIONE stima questa quantità**

# Validazione

L'idea delle procedure di validazione è quella di stimare  $E_{\text{out}}$ , utilizzando **un dataset diverso (validation set)** rispetto a quello usato per la stima del modello **(training \identification set)**

La **regolarizzazione** e la **validazione** sono due tecniche che possono (e devono) essere usate insieme:

- la **regolarizzazione** aiuta a stimare un modello che può generalizzare meglio
- la **validazione** fornisce una stima dell'errore out-of-sample del modello stimato

**Nota:** La regolarizzazione e la validazione non vengono usate solo nell'ambito machine learning! **Sono fondamentali anche nell'identificazione di sistemi dinamici!**

# Set di validazione

L'obiettivo del **set di validazione** è quello di stimare le **performance out-of-sample** del modello. Una procedura comune che si segue è:

- 1. Rimuovo** un subset di dati dai dati totali → questo subset non è usato per il training (stima)
- 2. Stimo** il modello sulla parte di dati rimanente → il modello sarà allenato su *meno* dati
- 3. Valuto** le performance del modello sul subset di dati che ho rimosso al punto 1. → in questo modo ottengo una stima corretta (unbiased) dell'errore out-of-sample
- 4. Ri-alleno** il modello su tutti i dati



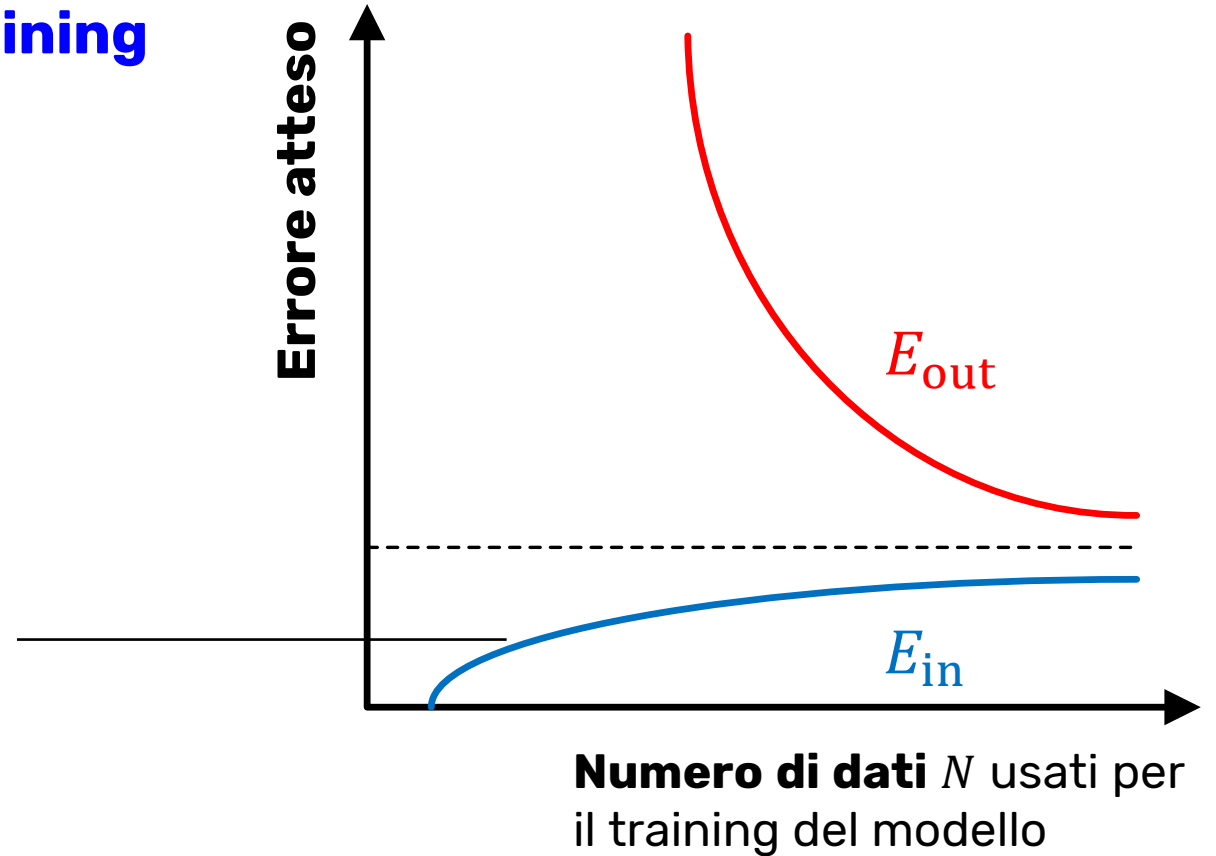
# Set di validazione

Supponiamo di avere il dataset  $\mathcal{D} = \{(\varphi(1), y(1)), \dots, (\varphi(N), y(N))\}$ . Si procede come segue:

$N_{\text{val}}$  dati: **validazione**  
 $\mathcal{D}_{\text{val}}$

$N - N_{\text{val}}$  dati: **training**  
 $\mathcal{D}_{\text{train}}$

- $N_{\text{val}}$  **«piccolo»**: stima di  $E_{\text{out}}$  non buona
- $N_{\text{val}}$  **«grande»**: possibilità di imparare un modello non buono (guardare le learning curves)



# Set di validazione

$$\begin{array}{ccccc} \mathcal{D} & \rightarrow & \mathcal{D}_{\text{train}} & \cup & \mathcal{D}_{\text{val}} \\ \downarrow & & \downarrow & & \downarrow \\ N & & N - N_{\text{val}} & & N_{\text{val}} \end{array}$$

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

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

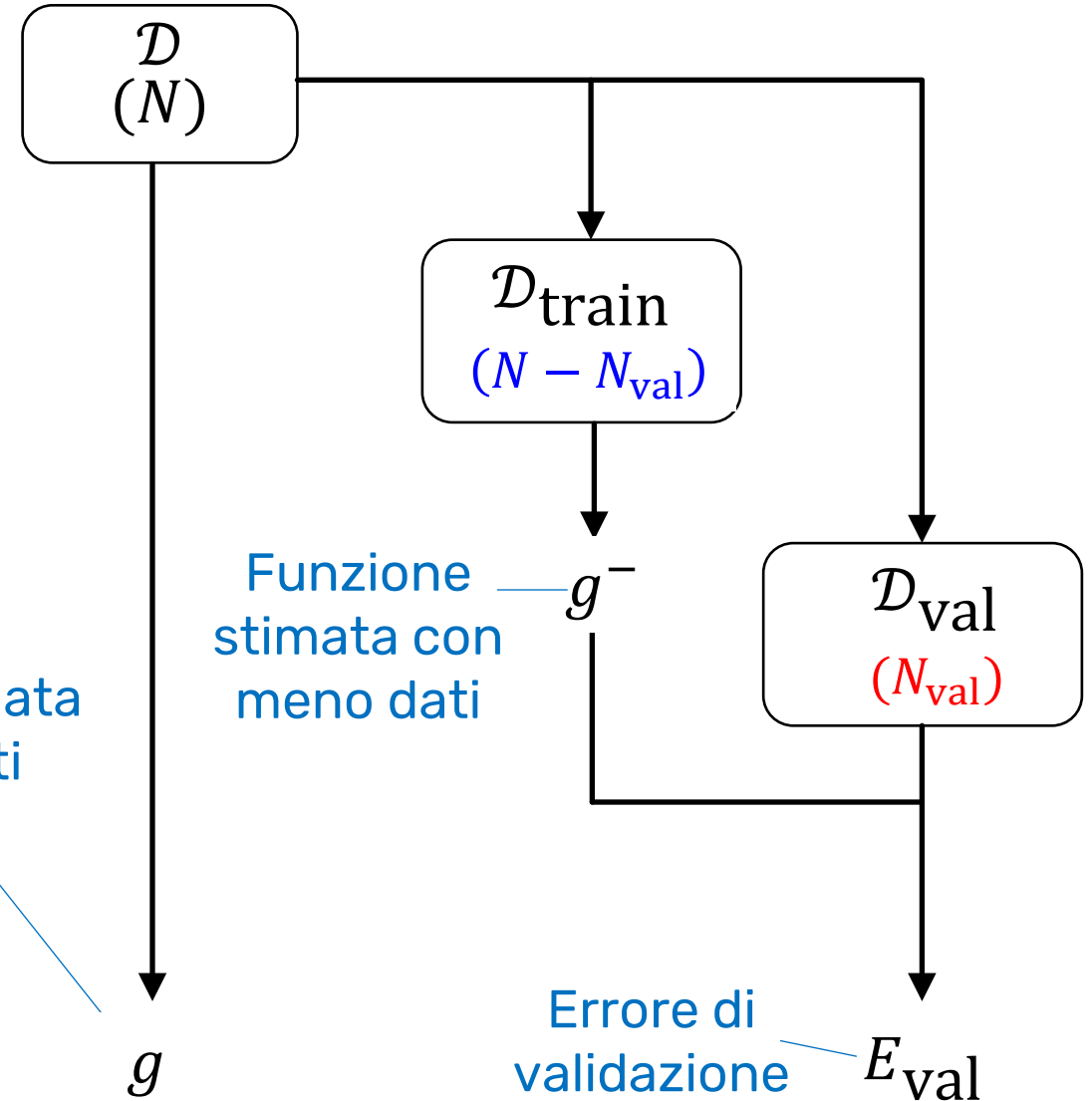
## Rule of thumb

$$N_{\text{val}} = \frac{N}{5}$$

Funzione  
(modello) stimata  
su tutti i dati

Funzione  
stimata con  
meno dati

Errore di  
validazione





# Selezione del modello migliore usando validazione

Le procedure di validazione possono essere utilizzate per due scopi:

1. **Valutare le performance** del modello stimato (e.g. stimare  $E_{\text{out}}$ )
2. **Scegliere il modello migliore** da un insieme di diversi modelli

Per esempio, la scelta del modello migliore include:

- scegliere tra un modello lineare e uno non lineare
- scegliere il numero di regressori da usare
- scegliere il valore del parametro di regolarizzazione  $\lambda_{\text{reg}}$
- ...qualsiasi altra scelta che influisce sull'apprendimento del modello

# Selezione del modello migliore usando validazione

Supponiamo di avere  $N_m$  **set di modelli**

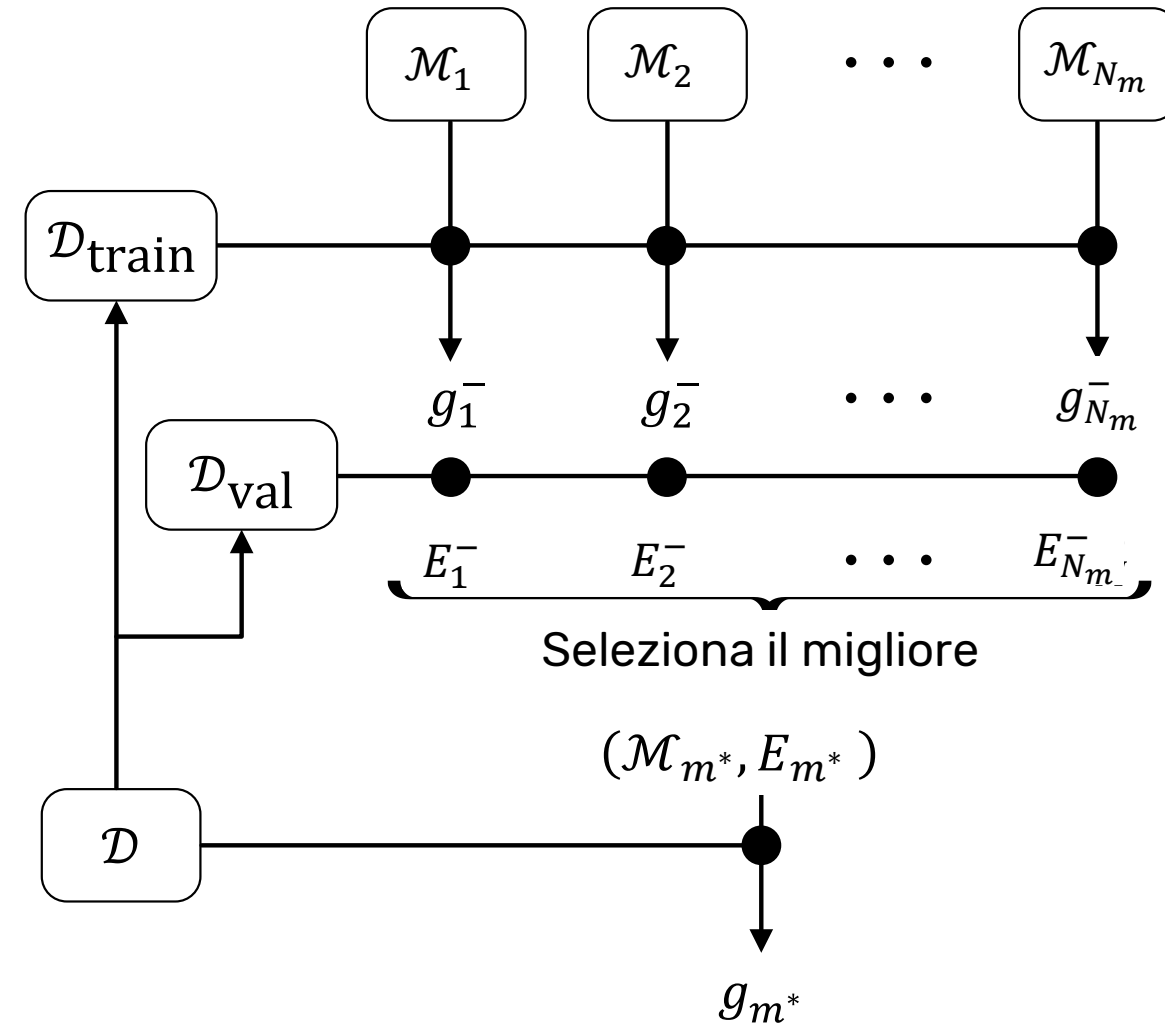
$\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_{N_m}$  tra cui imparare un modello

- **Stimo**  $g_m^-$  usando  $\mathcal{D}_{\text{train}}$  per ogni set di modelli

- **Valuto**  $g_m^-$  usando  $\mathcal{D}_{\text{val}}$

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

- **Seleziono** il modello  $m = m^*$  con l'errore  $E_m^-$  **più basso**



# Selezione del modello migliore usando validazione

**Problema:** se uso il dataset di validazione  $\mathcal{D}_{val}$  «tante volte» per compiere delle scelte (e.g. scegliere tra modelli diversi), allora il dataset di validazione  $\mathcal{D}_{val}$  **non fornisce più una buona stima dell'errore out-of sample**  $E_{out}$

**Intuzione:** usare  $\mathcal{D}_{val}$  per compiere delle scelte su quale modello usare fa sì che **tali scelte siano dipendenti dai particolari valori dei dati** contenuti in  $\mathcal{D}_{val}$ . Chi mi garantisce che con dati diversi avrei compiuto le medesime scelte?

Quello che sta succedendo è che stiamo **overfittando il validation set**

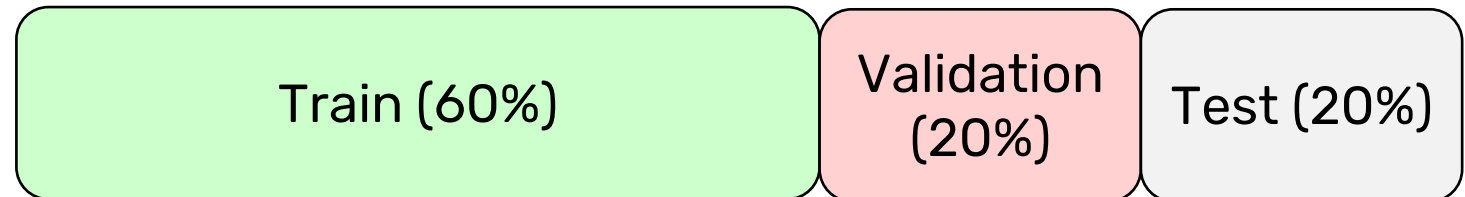
**Soluzione:** c'è bisogno di un terzo dataset. Il **dataset di test**, sul quale calcoleremo l'errore di test  $E_{test}$

# «Contaminazione» dei dataset

Abbiamo finora ottenuto **tre stime** dell'errore  $E_{\text{out}}$

**Contaminazione:** bias ottimistico nello stimare  $E_{\text{out}}$  (e.g. dire che  $E_{\text{out}}$  è più piccolo di quanto è in realtà)

- **Training set:** totalmente contaminato
- **Validation set:** un po' contaminato
- **Test set:** totalmente «pulito»



# Cross-validazione

La divisione del dataset in **tre parti** (train, validation, test) è fattibile se i dati a disposizione sono molti (dove «molti» dipende dal problema...si guardino le learning curves)

In teoria, vorremmo che:

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

( $N_{\text{val}}$  piccolo)      ( $N_{\text{val}}$  grande)

È l'unico che posso calcolare!

- $N_{\text{val}}$  **grande**: in questo caso, il valore di  $E_{\text{val}}$  calcolato usando  $g^-$  sarebbe simile al valore di  $E_{\text{out}}$  ottenuto da  $g^-$ , poichè uso tanti dati  $N_{\text{val}}$  per la validazione. Ricordiamoci che l'obiettivo di  $E_{\text{val}}$  è proprio quello di stimare  $E_{\text{out}}$
- $N_{\text{val}}$  **piccolo**: in questo caso, il valore di  $E_{\text{out}}$  ottenuto  $g^-$  sarebbe simile al valore di  $E_{\text{out}}$  ottenuto da  $g$  (la funzione stimata su tutti i dati), poichè uso tanti dati  $N - N_{\text{val}}$  per il train di  $g^-$ . Questo è il valore che mi interessa ma che non posso calcolare direttamente

# Cross-validazione

La **cross-validazione** permette di «avere  $N_{\text{val}}$  sia grande che piccolo»

## Leave-one-out cross-validation

Usiamo  $N - 1$  dati per il training e  $N_{\text{val}} = 1$  dato per la validazione

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

- Dato rimosso dai dati usati per il train e usato per la validazione

dove  $\mathcal{D}_i$  è il dataset di training senza il dato  $i$ -esimo.

La funzione (il modello) imparata usando  $\mathcal{D}_i$  è  $g_i^-$

# Cross-validazione

L'errore di validazione sul punto «rimosso»  $\varphi(i)$  è  $\ell(i) = E_{\text{val}}(g_i^-) = \ell(y(i), g_i^-(\varphi(i)))$

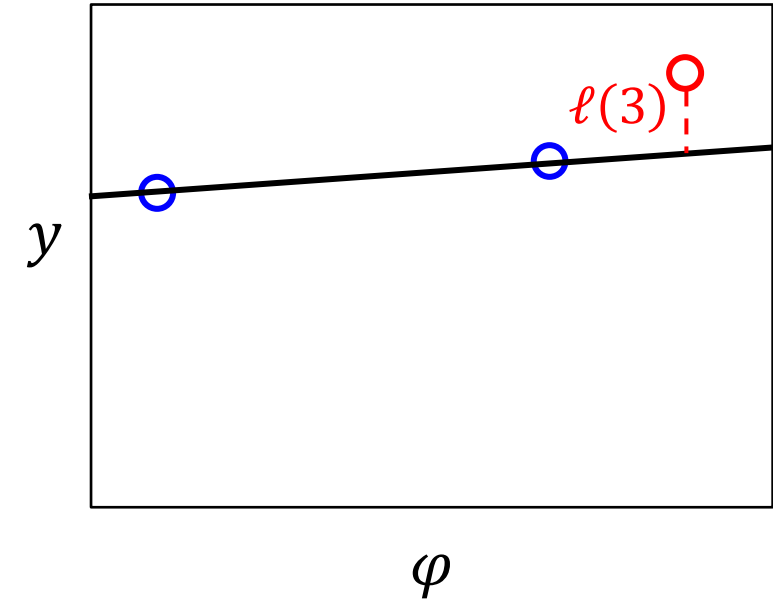
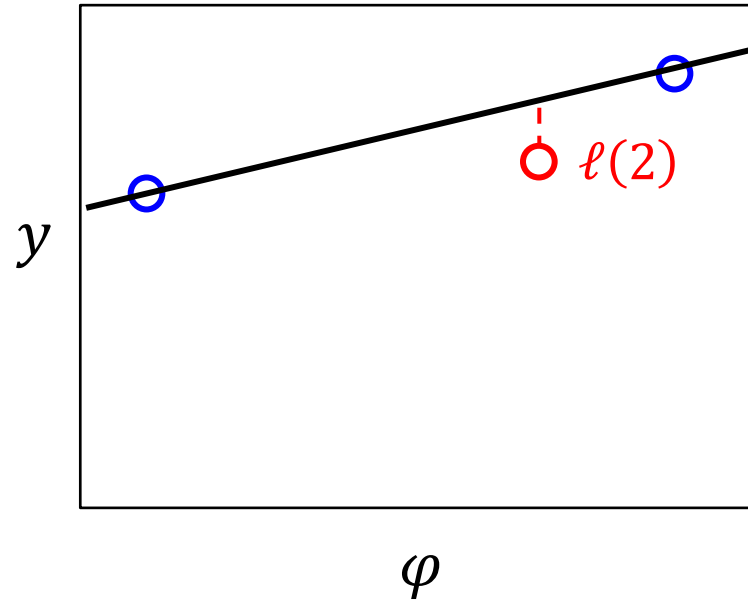
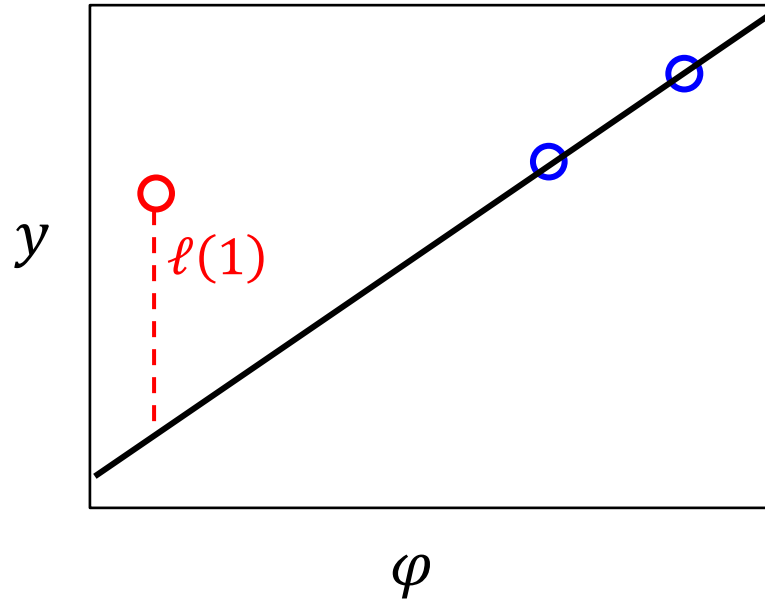
E' possibile definire l'**errore di cross-validazione**  $E_{\text{cv}}$  come:

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

- In questo modo, stimo  $N$  modelli usando  $N - 1$  dati, e li valido usando  $N$  stime dell'errore di validazione (calcolate ognuna su  $N_{\text{val}} = 1$  dati)
- È possibile anche calcolare la deviazione standard (campionaria) dei vari errori  $\ell(i)$ . Se è grande, vuol dire che il modello è molto sensibile ai dati sui quali viene allenato

# Esempio di cross-validazione

**Esempio:** supponiamo di voler imparare un modello lineare usando un regressore e  $N = 3$  osservazioni, utilizzando  $N_{\text{val}} = 1$  per fare la cross-validazione



$$E_{\text{cv}} = \frac{1}{3} (\ell(1) + \ell(2) + \ell(3))$$



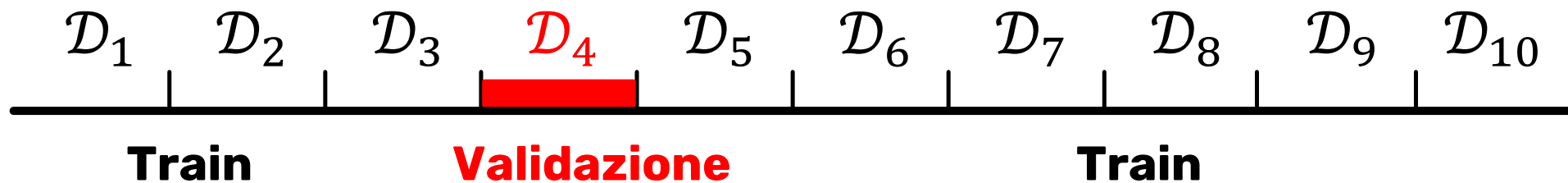
# Cross-validazione $K$ -fold

La cross-validazione con  $N_{\text{val}} = 1$  (leave-one-out cross-validation) ha gli svantaggi che:

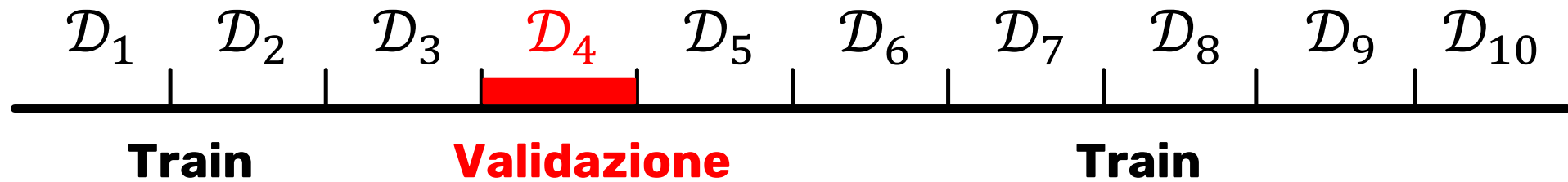
- È **computazionalmente costosa**. Nel caso volessimo usarla per scegliere tra  $M$  modelli, richiederebbe un totale di  $N$  sessioni di training per ciascuno degli  $M$  modelli
- La stima dell'errore  $E_{\text{cv}}$  ha una **varianza elevata**, poiché si basa su un solo dato

E' possibile riservare più punti per la validazione suddividendo il training set in «**folds**».

Per esempio, se  $K = 10$  avremmo



# Cross-validazione $K$ -fold



- La  $K$ -fold cross-validation richiede  $N/N_{\text{val}}$  sessioni di train, ognuna con  $N - N_{\text{val}}$  dati
- Un buon **compromesso** è usare  $K = 10$

$$10\text{-fold cross validation: } N_{\text{val}} = \frac{N}{10}$$

- Attenzione a **non ridurre troppo il training set** (guardare le learning curves)

# Esempio: modo corretto di usare la cross-validazione

Consideriamo un problema di **classificazione** con un **tanti regressori** (features). Una strategia per costruire un modello potrebbe essere la seguente:

1. Trovare un **sottoinsieme di regressori** che mostrano una forte **correlazione** (univariata) con le label
2. **Usando questo sottoinsieme** di predittori, imparare un classificatore
3. Utilizzare la cross-validazione per **stimare gli iperparametri** (e.g. model selection) e per **stimare l'errore out-of-sample**

Si tratta di una **corretta applicazione** della cross-validazione?

**NO!**

# Esempio: modo corretto di usare la cross-validazione

I regressori selezionati per la stima del modello hanno un **vantaggio sleale**, in quanto sono **stati scelti sulla base di tutti i dati** (step 1)

Rimuovere dati per la cross-validazione **dopo che i regressori sono stati selezionati** non imita correttamente l'applicazione del classificatore a un set di test completamente indipendente

I regressori (e quindi il modello) **hanno «già visto» i dati di validazione**

I dati utilizzati per la validazione sono stati già **utilizzati per effettuare una scelta** che ha coinvolto le label di output (questo **non è corretto**)



# Esempio: modo corretto di usare la cross-validazione

Quello che si sarebbe dovuto fare sarebbe stato di **includere la scelta del sottoinsieme dei regressori all'interno della procedura di cross-validazione**, oppure usare la regolarizzazione per la stima del modello

# Formule di complessità ottima

Queste formule permettono di stimare l'errore out-of-sample  $E_{\text{out}}$  **utilizzando solo il dataset di train**. Per questo motivo, si usano quando ho **troppi pochi dati** per poter usare validazione o cross-validazione

L'idea è **simile alla regolarizzazione**: modificare la funzione di costo dell'errore in-sample  $E_{\text{in}}$ , aggiungendo un termine additivo che **penalizza la complessità del modello**

Vedremo le seguenti formule \ criteri di complessità:

- **Akaike Information Criterion (AIC)**. Un indicatore **equivalente** è il Final Prediction Error (FPE) utilizzato per la stima di modelli dinamici
- **Minimum Description Length (MDL)**, derivante dal Bayesian Information Criterion (BIC)

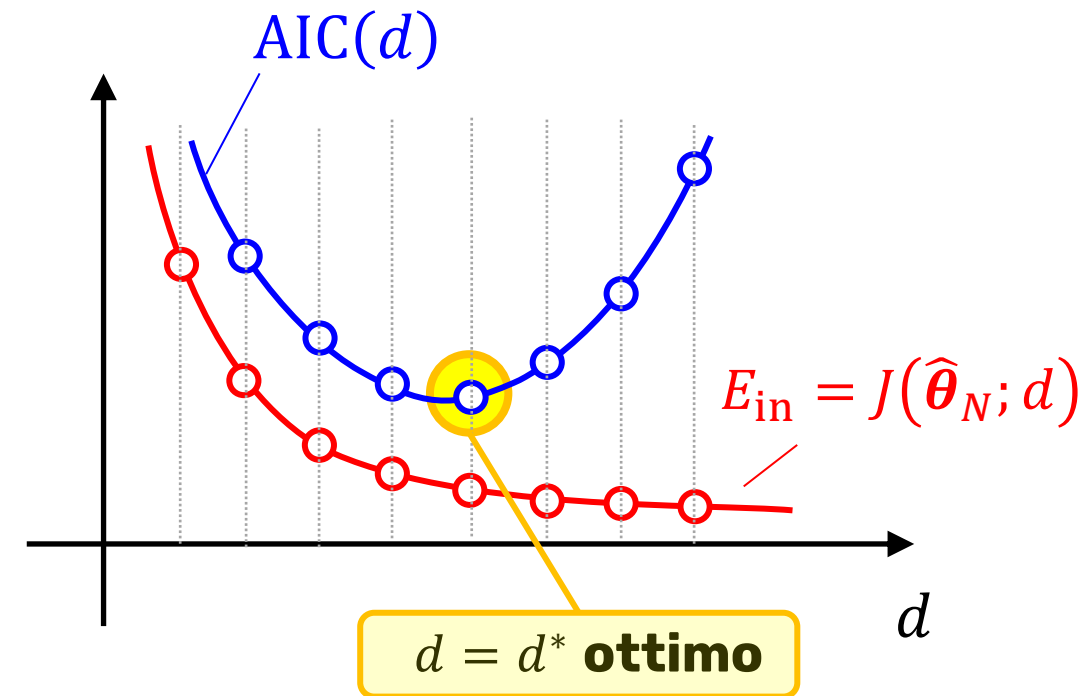
# Formule di complessità ottima

Supponiamo di avere un modello con  $d$  parametri. Indichiamo la stima dei parametri, ottenuta con  $N$  dati, con  $\hat{\theta}_N \in \mathbb{R}^{d \times 1}$ . La stima è ottenuta minimizzando la funzione di costo  $J(\theta; d)$  dove esplicitiamo la dipendenza del costo dal numero di parametri  $d$

## Akaike Information Criterion (AIC)

$$\text{AIC}(d) = 2 \cdot \frac{d}{N} + \ln[J(\hat{\theta}_N; d)]$$

- $d \uparrow \Rightarrow \frac{d}{N} \uparrow$
- $d \uparrow \Rightarrow J(\hat{\theta}_N; d) \downarrow$

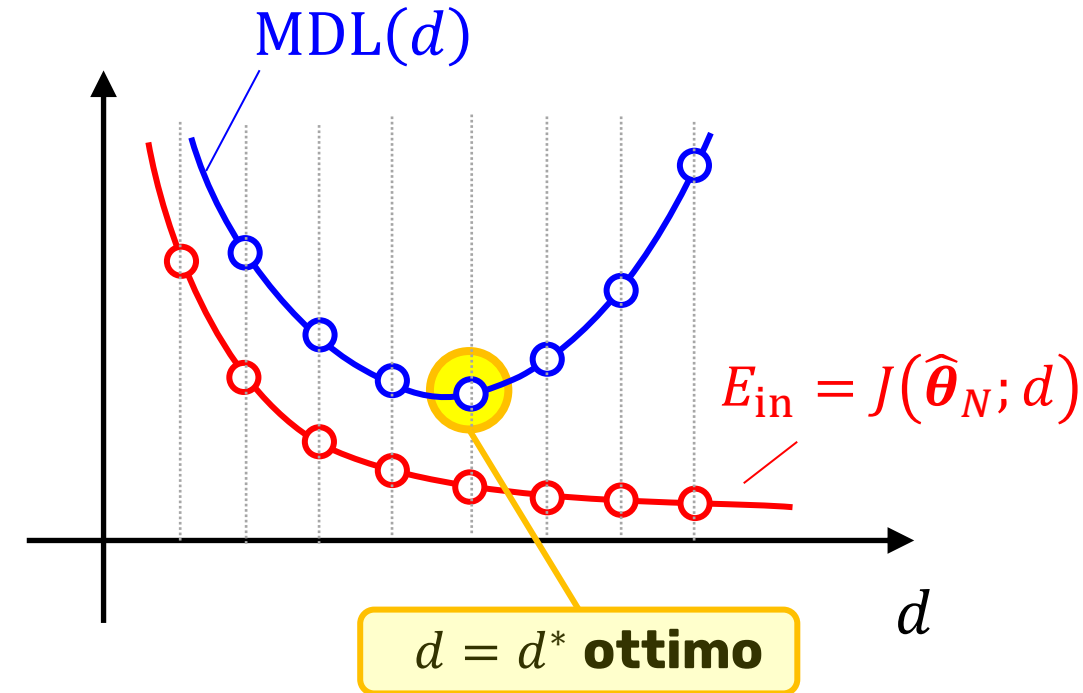


# Formule di complessità ottima

## Minimum Description Length (MDL)

$$\text{MDL}(d) = \ln[N] \cdot \frac{d}{N} + \ln[ J(\hat{\boldsymbol{\theta}}_N; d) ]$$

$$\bullet \quad d \uparrow \Rightarrow \ln[N] \cdot \frac{d}{N} \uparrow \quad \bullet \quad d \uparrow \Rightarrow J(\hat{\boldsymbol{\theta}}_N; d) \downarrow$$



Il criterio MDL di fatto si comporta come AIC. È però interessante confrontare più nel dettaglio AIC e MDL



# Confronto tra AIC e MDL

$$\text{AIC}(d) = 2 \cdot \frac{d}{N} + \ln[ J(\hat{\boldsymbol{\theta}}_N; d) ] \quad \Leftrightarrow \quad \text{MDL}(d) = \ln[N] \cdot \frac{d}{N} + \ln[ J(\hat{\boldsymbol{\theta}}_N; d) ]$$

Notiamo che la differenza consiste solo nei termini  $2$  e  $\ln[N]$ . Quindi, se  $\ln[N] > 2$  (ovvero se abbiamo più di 8 dati), la formula **MDL suggerisce di usare modelli più parsimoniosi**

**Nota:** Sotto l'assunzione che il meccanismo di generazione dei dati appartenga alla classe di modelli scelta, **FPE e AIC** hanno una probabilità non nulla di **sovrastimare l'ordine** del modello, mentre **MDL** porta ad una **stima asintoticamente corretta** dell'ordine

Dato che raramente l'assunzione è verificata, si **preferisce usare AIC** o FPE, sovrastimando leggermente  $d$

# Riassunto della validazione

- Se disponiamo di **molti dati**, il modo migliore per valutare le performance e selezionare il modello è dividere il **dataset in 3 parti** (training, validazione e test)
- Altrimenti, usiamo **cross-validazione**
- Se i dati sono **davvero pochi**, possiamo utilizzare formule per la scelta della complessità del modello ottimale che utilizzano **solo il training set**:
  - ✓ Akaike Information Criterion (**AIC**), Minimum Description Length (**MDL**)

**Osservazione:** le tecniche di validazione che abbiamo visto si possono usare anche nel caso di identificazione di **modelli dinamici**. Però, in questo caso validazione non può contenere dati estratti randomicamente, altrimenti romperebbe la cronologia temporale dei dati. Per cui, dovrò scegliere dati di validazione **cronologicamente contigui** ai dati di identificazione

# Outline

1. Introduzione al machine learning e alla data science
2. Problemi supervisionati e non supervisionati
3. Feasibility of learning
4. Bias-variance tradeoff
5. Learning curves
6. Overfitting
7. Regularizzazione
8. Validazione, cross-validazione e formule di complessità ottima

## 9. Esercizi con codice



# Regressione lineare con regolarizzazione $L_2$

Consideriamo il modello di **regressione lineare**, con un termine di regolarizzazione  $L_2$   
(**Ridge regression**)

$$\begin{aligned} E_{\text{aug}}(\boldsymbol{\theta}) \equiv J(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^N (y(i) - \underbrace{\boldsymbol{\varphi}^\top(i)}_{1 \times d} \underbrace{\boldsymbol{\theta}}_{d \times 1})^2 + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1} (\theta_j)^2 \\ &= \frac{1}{N} \underbrace{\|Y - X \cdot \boldsymbol{\theta}\|_2^2}_{\substack{N \times 1 \quad N \times d \quad d \times 1}} + \lambda_{\text{reg}} \cdot \|\boldsymbol{\theta}\|_2^2 \end{aligned}$$

Si può dimostrare che la stima in forma chiusa dei parametri si ottiene come:

$$\underbrace{\hat{\boldsymbol{\theta}}_{\text{reg}}}_{d \times 1} = \left( \underbrace{X^\top X}_{d \times d} + \lambda_{\text{reg}} \cdot \underbrace{I_d}_{d \times d} \right)^{-1} \underbrace{X^\top Y}_{\substack{d \times N \quad N \times 1}}$$

# Ridge regression e gradient descent

Supponiamo di avere **solo 2 parametri**  $\theta = [\theta_0, \theta_1]^T \in \mathbb{R}^{2 \times 1}$  per semplicità

$$E_{\text{aug}}(\theta) \equiv J(\theta) = \frac{1}{N} \sum_{i=1}^N (y(i) - \theta_0 - \theta_1 \cdot \varphi_1(i))^2 + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1} (\theta_j)^2$$

E' possibile implementare l'algoritmo del gradient descent come:

**For** {

$$\theta_0 = \theta_0 - \alpha \cdot 2 \left[ \frac{1}{N} \sum_{i=1}^N (y(i) - \theta_0 - \theta_1 \cdot \varphi_1(i)) \cdot (-1) + \lambda_{\text{reg}} \cdot \theta_0 \right]$$

$$\theta_1 = \theta_1 - \alpha \cdot 2 \left[ \frac{1}{N} \sum_{i=1}^N (y(i) - \theta_0 - \theta_1 \cdot \varphi_1(i)) \cdot (-\varphi_1(i)) + \lambda_{\text{reg}} \cdot \theta_1 \right]$$

}

# Regressione logistica con regolarizzazione $L_2$

Consideriamo il modello di **regressione logistica**, con un termine di regolarizzazione  $L_2$

$$E_{\text{aug}}(\boldsymbol{\theta}) \equiv J(\boldsymbol{\theta}) = \sum_{i=1}^N \left( y(i) \cdot \ln \pi(i; \boldsymbol{\theta}) + (1 - y(i)) \cdot \ln[1 - \pi(i; \boldsymbol{\theta})] \right) + \lambda_{\text{reg}} \cdot \sum_{j=0}^{d-1} (\theta_j)^2$$

E' possibile implementare l'algoritmo del gradient descent come:

**For** {

$$\theta_0 = \theta_0 - \alpha \cdot \sum_{i=1}^N 1 \cdot (\pi(i) - y(i)) + 2\lambda_{\text{reg}} \cdot \theta_0$$

$\vdots$

$$\theta_{d-1} = \theta_{d-1} - \alpha \cdot \sum_{i=1}^N \varphi_{d-1}(i) \cdot (\pi(i) - y(i)) + 2\lambda_{\text{reg}} \cdot \theta_{d-1}$$

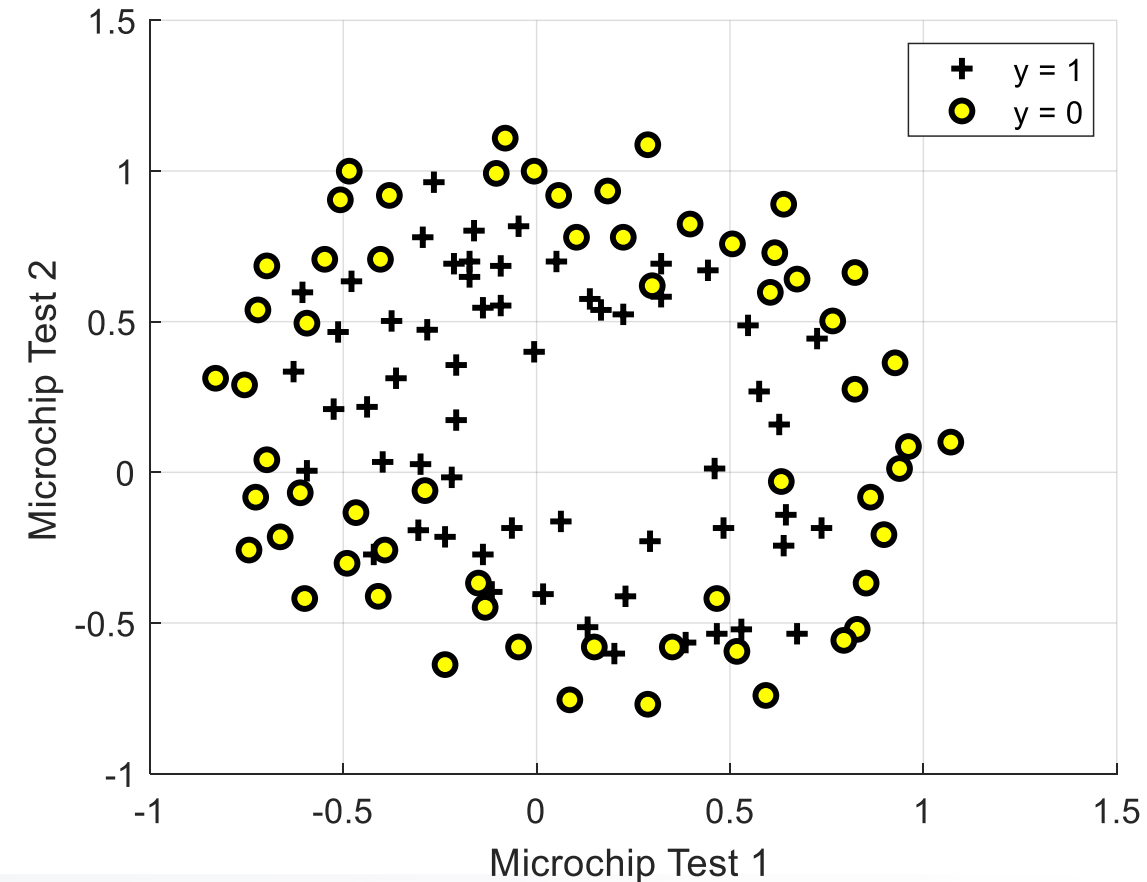
}

$$\pi(i) \equiv \frac{1}{1 + e^{-\boldsymbol{\varphi}^T(i)\boldsymbol{\theta}}} \\ = P(y(i) = 1 | \boldsymbol{\varphi}(i))$$

# Esercizio: classificare microchips difettosi

Vogliamo rilevare se un microchip è **difettoso** in base ai risultati di due test di qualità alla fine della linea di produzione, tramite un modello di **regressione logistica**

- Ogni microchip è descritto dalle seguenti features
  - ✓  $\varphi_1$ : Risultato del test 1
  - ✓  $\varphi_2$ : Risultato del test 2
- Il dataset è costituito da  $N = 118$  microchips

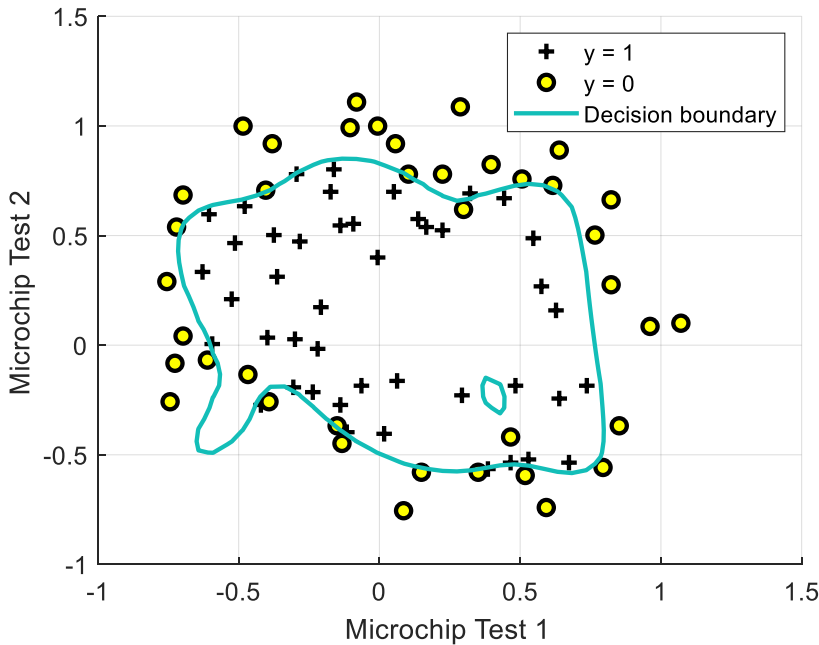


# Esercizio: classificare microchips difettosi

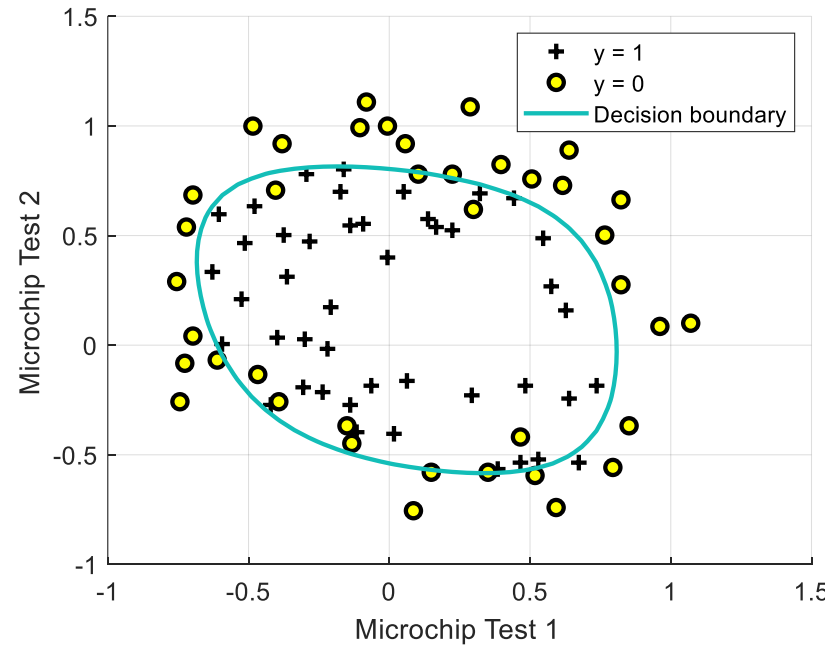
Per ottenere un **confine nonlineare** tramite il classificatore lineare, è possibile usare **features polinomiali**. Ad esempio, usando feature polinomiali di grado 2 otteniamo:

$$\varphi_1^2, \varphi_2^2, \dots, \varphi_{d-1}^2, \quad \varphi_1\varphi_2, \dots, \varphi_1\varphi_{d-1}, \quad \varphi_1\varphi_2^2, \dots, \varphi_1\varphi_{d-1}^2, \quad \varphi_1^2\varphi_2, \dots, \varphi_1^2\varphi_{d-1}, \dots$$

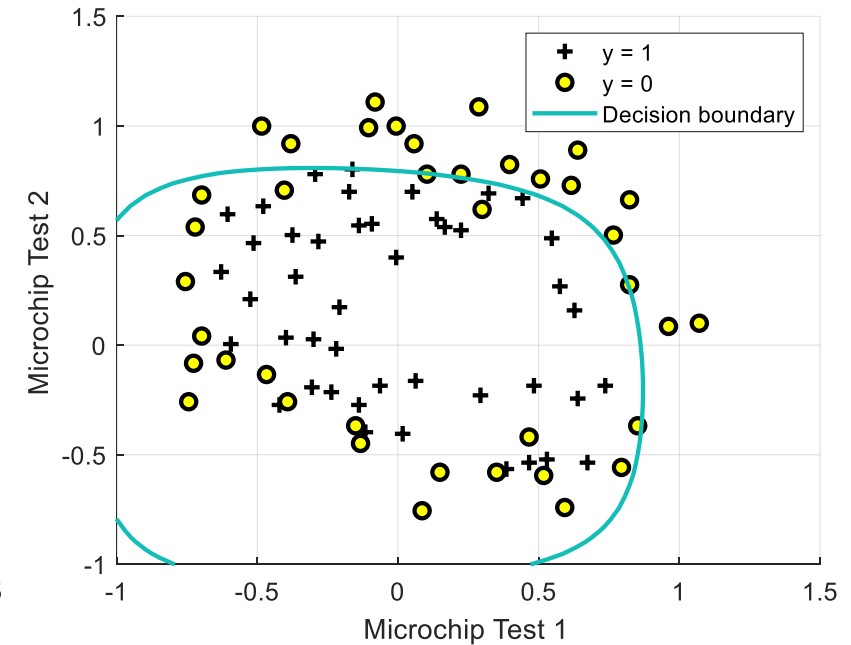
$$\lambda_{\text{reg}} = 0$$



$$\lambda_{\text{reg}} = 0.1$$



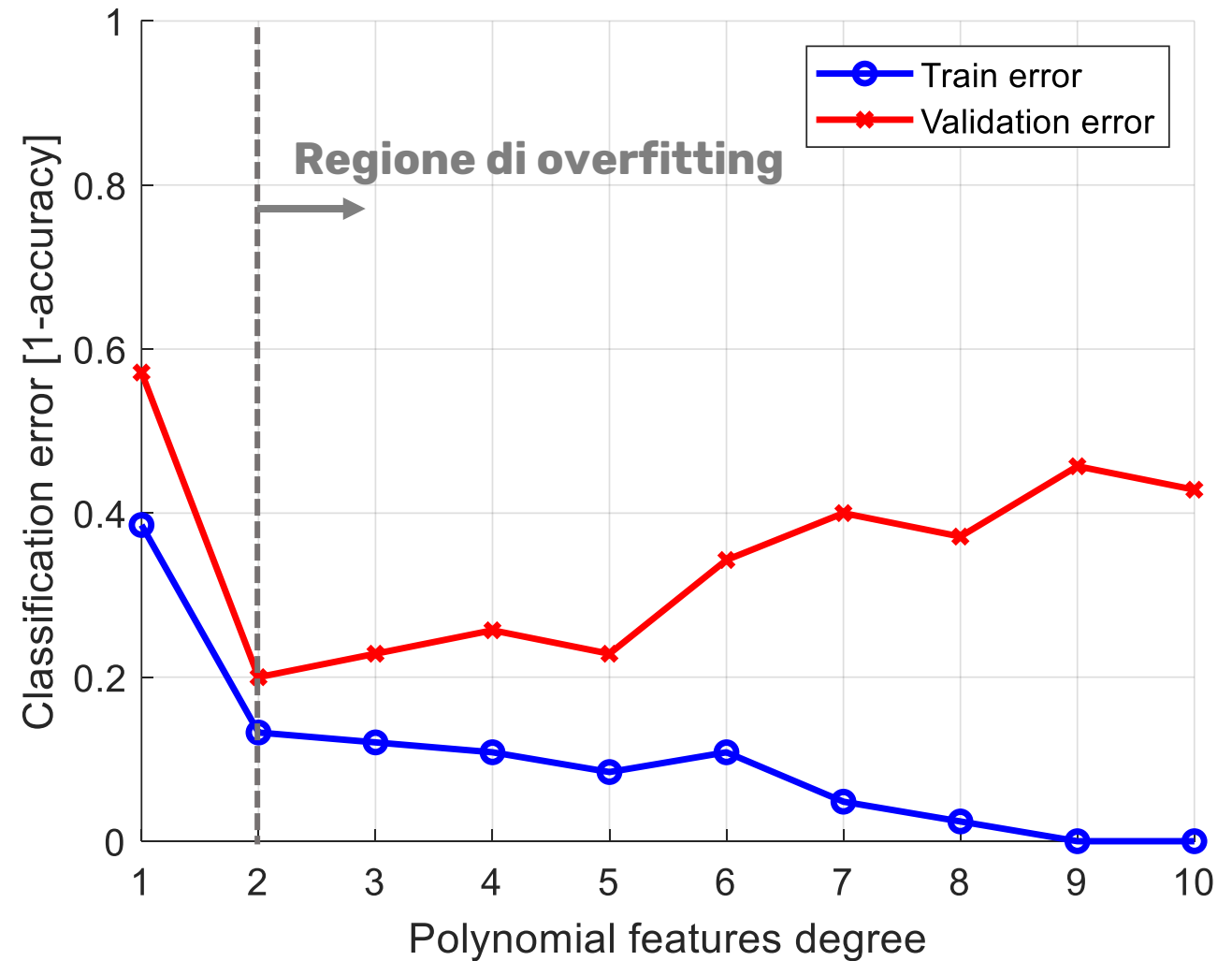
$$\lambda_{\text{reg}} = 10$$





# Esercizio: classificare microchips difettosi

È possibile dividere i dati in training e validation set, in modo da **selezionare l'ordine ottimale** per le features polinomiali tramite **validazione**





**UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO**

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione