

## Generalizzazione

Richiamo al concetto di [generalizzazione](#) visto nell'[introduzione](#)

È un punto centrale, come obiettivo del ML, una teoria che supporta in che condizioni ciò si verifica. Guida la *scelta del "best model"* e va *verificato nelle applicazioni*.

### 🔗 Ipotesi di apprendimento induttivo ▾

Tutti gli  $h$  che *approssimano bene*  $f$  sugli esempi di training, approssima bene  $f$  *anche sulle nuove istanze (non viste)*  $x$ .

Fase di <b>Learning</b> ( <i>allenamento, fitting</i> )	Fase <b>Predittiva</b> ( <i>test</i> )
Si costruisce il modello dai dati noti ( <i>dati di training e bias</i> )	Si applica ai nuovi esempi. Prendiamo un input $x'$ e <i>calcoliamo la risposta</i> con il modello. Lo confrontiamo con il <i>target <math>d'</math></i> che il modello non ha mai visto. Qui <i>valutiamo le ipotesi predittive</i> $\Rightarrow$ <i>generalization capability</i>

$$\begin{matrix} \text{performance} \\ \text{in ML} \end{matrix} = \begin{matrix} \text{predictive} \\ \text{accuracy} \end{matrix}$$
  
Stimata dagli errori calcolati sul **Test Set**

### Overfitting

### Premessa: cosa si misura? - recap

Per valutare di solito misuriamo:

- Per la **Classificazione**:
  - **MSE** per la perdita
  - **Accuratezza** o **tasso di errore medio** per esito

- Per la **Regressione**

- **MSE**
- **Root MSE** ( $\$$ )
- **Errore assoluto medio**
- **Errore assoluto massimo**

- Anche *misure statistiche* come  $R$

Un alto errore  $\iff$  <sup>comporta</sup> bassa accuratezza, ad esempio

- fitting povero con un alto errore nell'allenamento
- generalizzazione povera con alto errore nei test

## Validazione: 2 scopi

### Selezione del modello

*Stima della prestazione* (errore di generalizzazione) o *modello di apprendimento diverso* in ordine per scegliere il migliore. Questo include anche la ricerca del *miglior hyperparameters* del modello.

[Ritorna un modello](#)

### Valutazione del modello

Avendo scelto un modello finale, dobbiamo *stimare/valutare* la sua *predizione di errore/rischio* sui nuovi dati di test, misuriamo la qualità/performance degli modelli scelti per ultimi.

[Ritorna una stima](#)

# Resistenza

## Important

- Se la dimensione dell'insieme è sufficiente (es 15% *TR*, 25% *VL*, 25% *TS*, insiemi disgiunti) [image pag. 11 of 3.6]
- *TR*: *Insieme di training* usato per il fit - fase di **training**
- *VL*: *Insieme di Validazione* (o *di selezione*) usato per selezionare il miglior modello tra molti e/o configurazioni di hyperparametri - **Selezione del modello**
- *TR + VL* a volte sono uniti e chiamati *insieme di sviluppo/design*  $\implies$  <sup>quindi</sup> è usato per costruire il modello finale
- *TS*: *Insieme di Test* usato per stimare l'errore di generalizzazione del modello finale - **valutazione del modello**
- 

La stima fatta per la selezione del modello, non è una buona stima per la valutazione del test di fase/rischio. **I risultati dell'insieme di test** non possono essere usati per la selezione del modello.

## Selezione del test o del modello?

Che succede se *l'insieme di test è usato in un ciclo di design ripetuto*?

Facciamo una *selezione del modello* e una *valutazione non affidabile* (stimiamo l'errore di generalizzazione attesa) e *non siamo in grado di farlo negli esempi futuri*. In questo caso *l'insieme di test usato, fornisce una valutazione troppo ottimistica* del vero errore di test.

### GOLD RULE

## Gold Rule

Mantenere la separazione tra gli obiettivi e usare insiemi separati

*TR* (insieme di training) per l'allenamento  
*VL* (insieme di validazione) per la selezione del modello  
*TS* (insieme dei test) per la stima del rischio

## Schema di TR/VL/TS

[image pag 13 of 3.6]

## meta-algoritmo semplice

- Separare gli insiemi *TR*, *VL*, *TS*
- Cercare il *miglior*  $h_{w,\lambda}()$  *cambiando gli hyperparametri*  $\lambda$  del modello
- **For Each** valore diverso di  $\lambda$  (ricerca a griglia)
  - Cercare il *miglior*  $h_{w,\lambda}()$  che minimizzi l'errore/la perdita empirica (*fitting dell'insieme TR*), trovando il *miglior parametro*  $w$ . Per *miglior* si intende il *minimo errore sull'insieme TR*
- Selezionare il *miglior*  $h_{w,\lambda}()$ , dove *miglior* vuol dire *minimo errore sull'insieme VL*
- (**facoltativo**) Ora è possibile fare il fit di  $h_{w,\lambda}(x)$  sull'insieme *TR+VL* con il miglior modello  $\lambda$
- *Valutare* la funzione  $h_{w,\lambda}(x)$  *sull'insieme TS*

Il **for each** è un **doppio ciclo**: Cercare il migliore può essere un **for** su una griglia di valori in un ciclo esterno  
**for each**  $\lambda$  si allena un modello  $h_{w,\lambda}(x)$  (nel ciclo interno) sull'insieme *VL*. Poi si tiene il miglior valore di  $\lambda$   $\implies$  <sup>quindi</sup> il modello con il minimo errore o la massima accuratezza sull'insieme *VL*

## Ricerca su una griglia

## Ricerca su una griglia

Trovare un *valore di un hyperparametro*  $\Rightarrow$  <sup>quindi</sup> un parametro che non è appreso direttamente e che non viene modificato dall'allenamento

La ricerca del miglior hyperparametro può essere un `for` su una griglia di valori candidati. `for each` modello allenato  $h_{w,\lambda}$  si calcola il risultato (l'accuratezza) sull'insieme VL, poi si tiene quello con il minimo errore o la massima accuratezza.

### Example

[image pag. 15 of 3.6]

*Res1* è calcolato nell'insieme VL dal modello con grado polinomiale pari a 1 e  $\lambda = 0.1$  allenata sull'insieme TR.

Il migliore è *Res3*  $\rightarrow$  grado= 4 e  $\lambda = 0.1$  è il vincitore

Possiamo automatizzare questo processo, la parallelizzazione è facile (indipendenza dei tentativi).

## Controesempio

- 20 – 30 esempi
- 1000 variabili di input casuali
- target casuale 0/1

Scelgo *un modello* con *una sola variabile/feature che indovina "per caso"* al 99% sull'insieme di dati, poi faccio lo stesso su qualsiasi split degli insiemi TR, VS, TS.

*Risultato perfetto (modello con accuratezza 99%)? Cosa non va?*

99% non è una buona stima dell'errore di test, quella *corretta è 50%*

1. L'errore stimato sugli insiemi TR o VS per la selezione del modello *non è utile per la stima del rischio*. I dati di TR o VL non vanno usati per scopi di test
2. Usare *tutto il data set* per *feature/model selection* *lede la correttezza della stima*  
Il TS è stato usato implicitamente all'inizio. Il test deve essere separato prima di qualsiasi model selection o design del modello, inclusa la selezione delle features  
Un *TS esterno* fornisce invece la *stima corretta del 50% (random coin result)*. È la correttezza della stima che è in giudizio, non la possibilità di risolvere task.

### Tabella per il controesempio

	1	2	...	26	27	28	...	100	target
1	...	...	...	1	1	1	...	...	1
1				0	0	0			0
3				1	1	1			1
4				0	0	0			0
...				0	0	0			0
...				1	1	1			1
...				0	0	0			0
20				1	1	1			1
TS1				1	0	0			1
TS2				0	1	0			0
TS3				1	1	1			1
Accuracy				100%	33%	66%			

## Resistenza e convalida incrociata K-FOLD

Fare resistenza su CV può fare un uso insufficiente dei dati

### Convalida incrociata K-fold

[image pag. 20 of 3.6]

Si splitta l'insieme dei dati  $D$  in  $k$  sottoinsiemi *mutualmente esclusivi*  $D_1, D_2, \dots, D_k$ . Alleniamo l'algoritmo di apprendimento su  $\frac{D}{D_i}$  e lo testiamo su  $D_i$ . In fine facciamo la *media* di tutti i risultati di  $D_i$  (diagonali).

Questa tecnica può essere usata sia per il VS che per il TS, *usa tutti i dati per fare allenamento, valutazione o testing*.

I **problemi** di questa tecnica sono:

- Troppi fold
- Spesso molto costoso dal punto di vista computazionale
- Combinabile con il VS, doppia k-fold

### Esempio di Selezione e Valutazione del modello con K-Fold

Separiamo i dati in **TR** e **TS** (semplice hold-out o k-fold).

#### Selezione del Modello

Usa K-Fold CV (interno) sul **TR**, ottenendo nuovi **TR** e **VL** in ogni separazione, per trovare i *migliori hyperparametri* del modello, applicando una **grid search** con molti valori possibile dell'hyperparametro.

*quindi*  $\Rightarrow$  per esempio un K-fold-CV per  $\lambda = 0.1$  un k-fold-CV per  $\lambda = 0.01$ , ... poi prendiamo il *miglior  $\lambda$*

Alleniamo con tutto il TR il modello finale

#### Valutazione del modello

Valuta il modello su un **TS esterno**

## Tipico comportamento dell'apprendimento

[image pag. 24 of 3.6]

Sia l'errore di training che quello del test sono alti, in questo caso abbiamo **underfitting**: il modello è troppo semplice riferito alla funzione target sia per i dati noti che per quelli nuovi.

[image pag. 26 of 3.6]

L'errore di training è basso, mentre quello di test è crescente, in questo caso abbiamo **overfitting**: il modello è troppo complesso, si adatta al disturbo dei dati. L'errore di training è molto basso, quello di test può essere alto.

[image pag. 28 of 3.6]

### Dimensione dell'insieme dei dati

$l = 15$

[image pag. 29 of 3.6]

$l = 100$

[image pag- 30 of 3.6]

## Verso SLT

Mettendo tutto insieme:

- La *capacità di generalizzazione* (misurata come l'errore di rischio o di test) del modello, rispettando l'errore di training e le zone di overfitting e underfitting
  1. Il ruolo della complessità del modello
  2. Il ruolo del numero di dati

**Statistical Learning Theory (SLT)**: una teoria generale relativa a questi argomenti

### Impostazione formale della Statistical Learning Theory (SLT) - Semplificata

## Definition

- Approssimare la funzione sconosciuta  $f(x)$ ,  $d$  (or  $y$  or  $t$ ) è il target ( $d = \text{true } f + \text{noise}$ )
- Minimizzare la funzione di rischio  $R = \int L(d, h(x)) dP(x, d)$
- Dati
  - Un valore dal teacher ( $d$ ) e la probabilità di distribuzione  $P(x, d)$
  - Una funzione di perdita (o di costo) ad esempio  $L(h(x), d) = (d - h(x))^2$Cerca  $h$  in  $H$ :  $\text{Min } R$ , ma abbiamo solo l'insieme dei dati finito ( $TR = (x_p, d_p)$ ,  $p = 1, \dots, l$ ).
- Per cercare  $h$ : minimizzare il *rischio empirico* (*errore di training*  $E$ ), trovando il miglior valore per i parametri liberi del modello

$$R_{emp} = \frac{1}{l} \sum_{p=1}^l (d_p - h(x_p))^2$$

- Minimizzazione del rischio empirico (ERM) - Principio induttivo

## Vapnik-Chervonenkis-dim e SLT - una teoria generale

### VC-dim

Data la  $VC - \text{dim}$ , una misura della *complessità di  $H$*  (flessibilità per adattare i dati)

### VC-Bounds

$VC - \text{bounds}$  nella forma: mantiene con probabilità  $1 - \delta$  che garantisce il rischio

$$R \leq R_{emp} + \varepsilon\left(\frac{1}{l}, VC, \frac{1}{\delta}\right)$$

$\varepsilon$  è una *funzione* che *cresce con  $VC$* , che decresce con  $l$  e  $\delta$ . Sappiamo che  $R_{emp}$  *decresce usando modelli complessi* (con  $VC - \text{dim}$  alto).

$\delta$  è la *fiducia*, regola la probabilità che il limite valga.

Ora possiamo vedere come questo può "spiegare" l'*underfitting* e l'*overfitting* e gli aspetti che li controllano

### Intuizione:

- $l$  alto  $\rightarrow$  Fiducia  $VC$  bassa e limite vicino a  $R$
- *modello molto semplice* ( $VC - \text{dim}$  basso) può non essere sufficiente a causa di  $R_{emp}$  alto (*underfitting*)
- $VC - \text{dim}$  alto ( $l$  fissato)  $\rightarrow R_{emp}$  basso ma  $VC - \text{conf}$  e quindi  $R$  può crescere (*overfitting*)

### Minimizzazione del rischio strutturale

[image pag.34 of 3.6]

*flessibilità* - concetto di controllo della complessità del modello: compromesso tra *accuratezza del TR* (fitting) e la *complessità del modello* ( $VC - \text{dim}$ )

## Controllo della complessità - discussione

### Statistical Learning Theory (SLT):

Permette l'*inquadratura formale del problema della generalizzazione* e dell'*underfitting/overfitting*, fornendo limitazioni superiori analitiche e quantitative al rischio  $R$  di predizione su tutti i dati, indipendentemente dal tipo di learning algorithm o dai dettagli del modello.

Il ML è ben formato:

- il *rischio del learning* (e l'errore di generalizzazione) può essere *analiticamente limitato* e solo pochi concetti fondamentali

- Si può trovare una *buona approssimazione della funzione target  $f$*  da esempi, a condizione di avere un *buon numero di dati ed un'adeguata complessità del modello* (misurabile formalmente con la VC-dim)

La SLT porta a *nuovi modelli (SVM)* e altri modelli che direttamente considerino il controllo della complessità nella costruzione del modello. Inoltre fonda uno dei principi induttivi sul *controllo della complessità*  
*Quali (altri) principi ci sono per fondare il controllo della complessità e come operare in pratica?*

- Come misurare la complessità? (flessibilità per il fitting)
- Come trovare il bilanciamento ottimo tra fitting e complessità

## Esempi sul controllo della complessità

### Modelli lineari

- La complessità sembra correlata al numero di parametri liberi  $w$  e alla dimensione dell'input / dell'espansione di base
- Si usano i parametri- $\lambda$  per la versione regolarizzata, usando le tecniche di selezione / validazione del modello per trovare i valori propri di  $\lambda$

#### DT|Decision Tree

- numero di nodi

## Conclusioni

### FLESSIBILITÀ DEI MODELLI DI ML

Usare il potere del ML senza controllo è un modo di produrre *risultati illusori*. Si controlla il *compromesso* tra il fitting e la complessità di un modello. È fondamentale il *ruolo degli approcci di validazione*, per la selezione e la stima del modello.

### IL ML È BEN FONDATO TEORICAMENTE

Vengono fatte domande fondamentali tramite SLT ed altre.