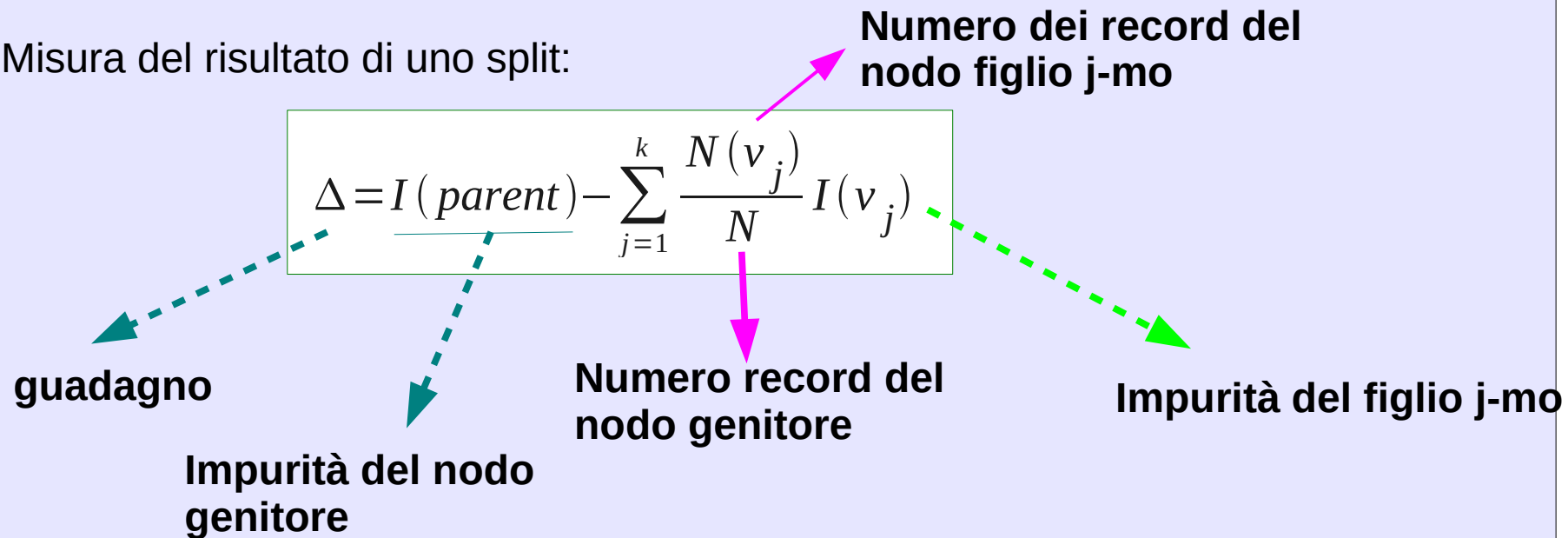


Calcolo del guadagno

Problema della scelta dello split: valuto i diversi split che posso fare, usando attributi diversi, tramite una delle misure viste e scelgo quello che mi restituisce il risultato col minor grado di confusione

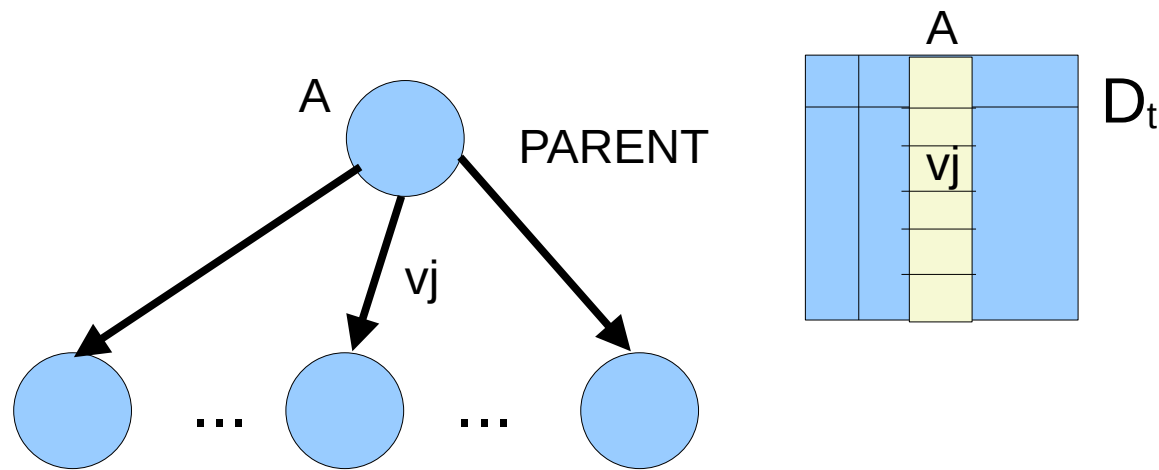
Misura del risultato di uno split:



Dall'impurità del nodo genitore viene sottratta la media pesata delle impurità dei nodi figli. Di solito la misura dell'impurità è scelta in modo tale da minimizzare l'impurità / massimizzare il guadagno

Spiegazione della formula

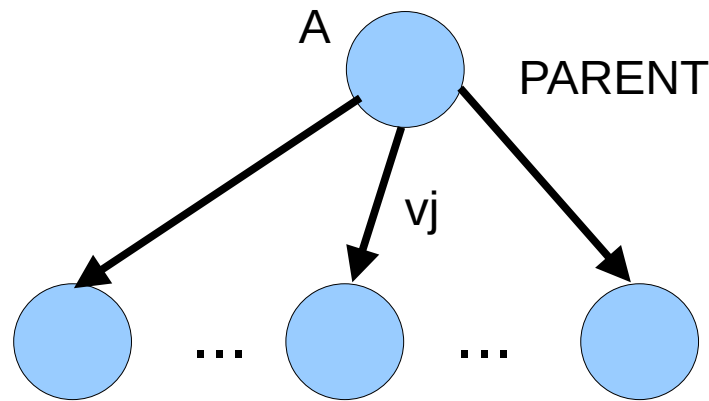
$$\Delta = I(\text{parent}) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j)$$



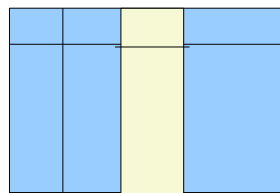
NODI FIGLI, IMMAGINANDO DI FARE LO SPLIT SU A

Spiegazione della formula

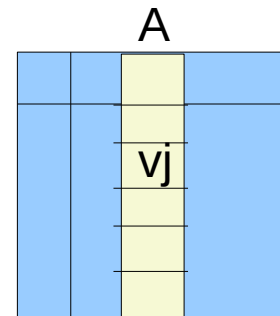
$$\Delta = I(\text{parent}) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j)$$



$N(v_j)$:
cardinalità
di questo
insieme



La porzione di D_t associata al generico nodo figlio j -mo avrà in A valori tutti uguali fra di loro (indichiamo tale valore con v_j)



D_t

N : numero istanze
in D_t

NB: la colonna A contiene valori diversi, alcuni sono uguali a v_j altri no. Questi dati vengono distribuiti fra i figli quando si fa lo split

Information gain

Per **information gain** si intende una misura del guadagno ottenuta usando l'**entropia** come valore dell'impurità dei nodi:

$$\Delta = \text{entropia}(\text{parent}) - \sum_{j=1}^k \frac{N(v_j)}{N} \text{entropia}(v_j)$$

Information gain

Per **information gain** si intende una misura del guadagno ottenuta usando l'**entropia** come valore dell'impurità dei nodi:

$$\Delta = \text{entropia}(\text{parent}) - \sum_{j=1}^k \frac{N(v_j)}{N} \text{entropia}(v_j)$$

Nota: le misure del grado di confusione, come Gini ed entropia tendono a favorire attributi che hanno *molti valori diversi* rispetto ad attributi con *pochi valori* alternativi

Osservazione: un *identificatore univoco* (es. un numero di matricola) annulla l'entropia (ogni nodo figlio conterrà una sola istanza) ma non è un attributo significativo!

Possibile soluzione: usare solo split binari

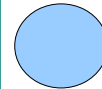
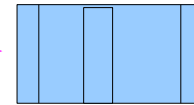
Algoritmo

Dati:

E = learning set

F = attributi descrittivi

attributi



```
CreaAlbero(E, F) {  
  if ( stopping_cond(E, F) ) {  
    Foglia = creaNodo();  
    Foglia.etichetta = classifica(E);  
    Risultato = Foglia;  
  }  
  else {  
    Nodo = creaNodo();  
    Nodo.test = trova_best_split(E, F);  
    V = << insieme dei valori possibili risultanti da Nodo.test >>  
    Foreach ( v ∈ V ) do {  
      Ev = << insieme e ∈ E | Nodo.test(e) == v >>  
      Figlio = CreaAlbero(Ev, F);  
      << aggiungi Figlio ai figli di Nodo, etichettandolo con v >>  
    }  
    Risultato = Nodo;  
  }  
  return Risultato;  
}
```

Algoritmo

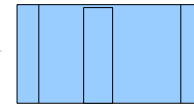
Dati:

E = learning set

F = attributi descrittivi

```
CreaAlbero(E, F) {  
  if ( stopping_cond(E, F) ) {  
    Foglia = creaNodo();  
    Foglia.etichetta = classifica(E);  
    Risultato = Foglia;  
  }  
  else {  
    Nodo = creaNodo();  
    Nodo.test = trova_best_split(E, F);  
    V = << insieme dei valori possibili risultanti da Nodo.test >>  
    Foreach ( v ∈ V ) do {  
      Ev = << insieme  $e \in E \mid \text{Nodo.test}(e) == v$  >>  
      Figlio = CreaAlbero(Ev, F);  
      << aggiungi Figlio ai figli di Nodo, etichettandolo con v >>  
    }  
    Risultato = Nodo;  
  }  
  return Risultato;  
}
```

attributi



Algoritmo

Dati:

E = learning set

F = attributi descrittivi

CreaAlbero(E, F) {

if (stopping_cond(E, F)) {

Foglia = creaNodo();

Foglia.etichetta = classifica(E);

Risultato = Foglia;

}

else {

Nodo = creaNodo();

Nodo.test = trova_best_split(E, F);

V = << insieme dei valori possibili risultanti da *Nodo.test* >>

Foreach (v ∈ V) do {

Ev = << insieme e ∈ E | *Nodo.test(e) == v* >>

Figlio = CreaAlbero(Ev, F);

<< aggiungi Figlio ai figli di *Nodo*, etichettandolo con *v* >>

}

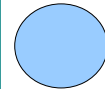
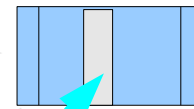
Risultato = *Nodo*;

}

return Risultato;

}

attributi



test

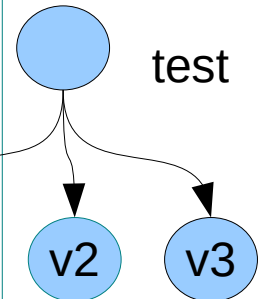
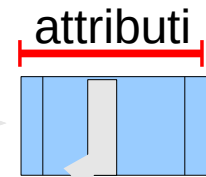
Algoritmo

Dati:

E = learning set

F = attributi descrittivi

```
CreaAlbero(E, F) {  
  if ( stopping_cond(E, F) ) {  
    Foglia = creaNodo();  
    Foglia.etichetta = classifica(E);  
    Risultato = Foglia;  
  }  
  else {  
    Nodo = creaNodo();  
    Nodo.test = trova_best_split(E, F);  
    V = << insieme dei valori possibili risultanti da Nodo.test >>  
    Foreach ( v ∈ V ) do {  
      Ev = << insieme e ∈ E | Nodo.test(e) == v >>  
      Figlio = CreaAlbero(Ev, F);  
      << aggiungi Figlio ai figli di Nodo, etichettandolo con v >>  
    }  
    Risultato = Nodo;  
  }  
  return Risultato;  
}
```



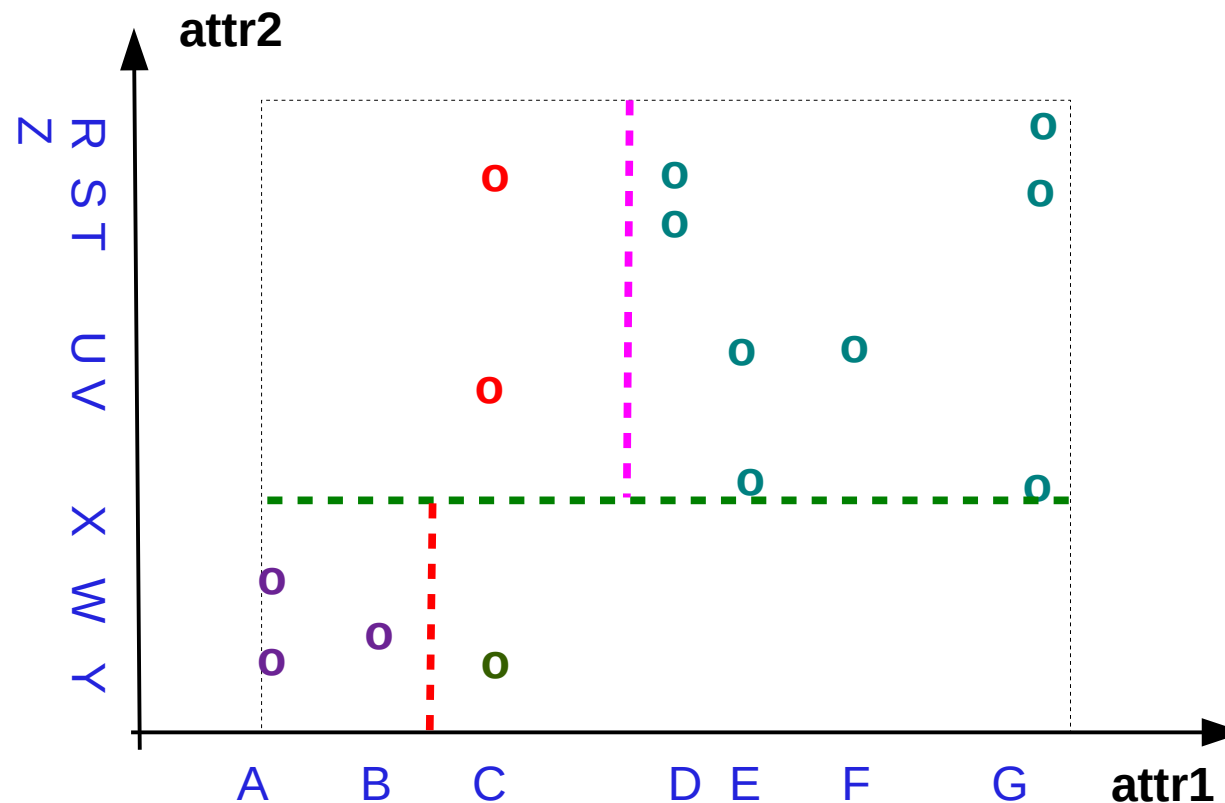
- **trova_best_split**: può essere individuato per esempio tramite il calcolo dell'entropia;
- **classifica**: può per esempio restituire la classe più rappresentata
- **stopping_cond**: può restituire vero per esempio quando tutte le istanze associate al nodo appartengono alla stessa classe oppure quando il numero di istanze è al di sotto di una certa soglia

Partizionamento dello spazio

Supponiamo di poter rappresentare le istanze del learning set come punti in uno spazio multidimensionale: ogni test corrisponde a un **taglio** (una **partizione**) di tale spazio, fatta lavorando su un **singolo attributo**

Partizionamento dello spazio

Supponiamo per semplicità di avere due soli attributi, corrispondenti ai due assi cartesiani. Le lettere rappresentano i valori possibili dei due attributi. Sono stati ordinati con un qualche criterio (anche solo associando numeri a label)



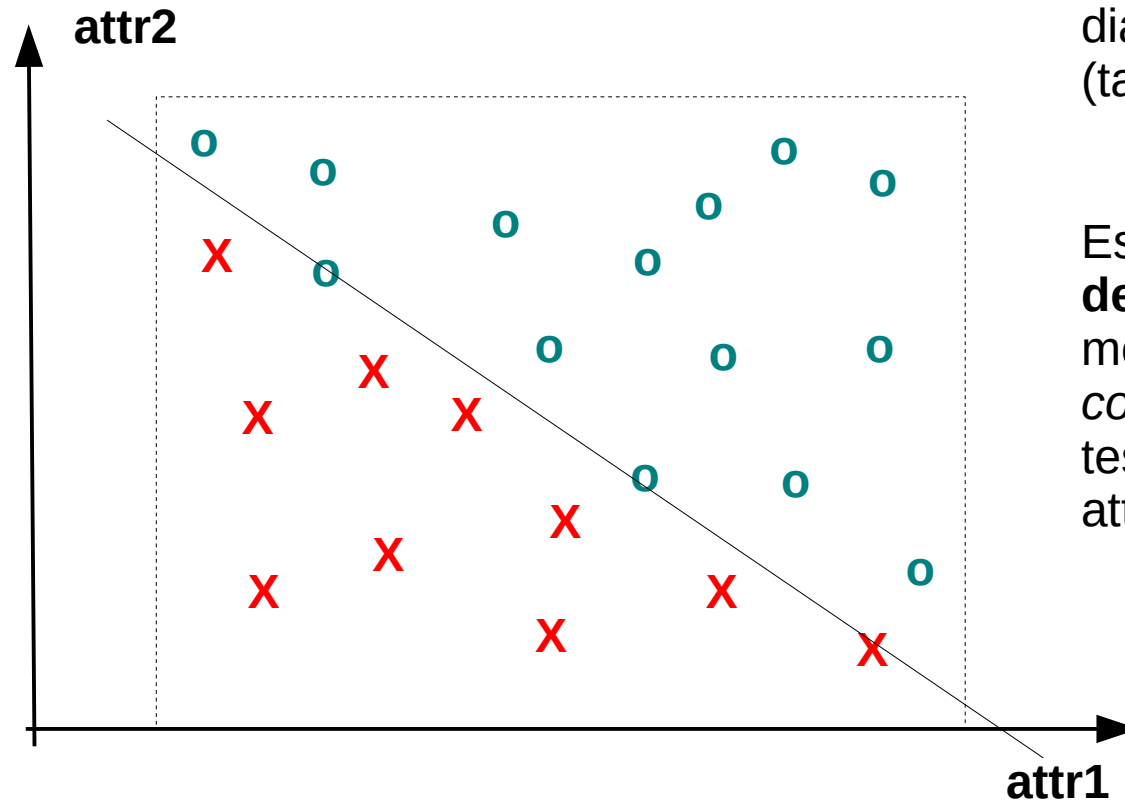
La radice ha associato l'intero learning set, contenuto nel rettangolo

Il primo test (linea verde) partiziona il learning set dividendo gli esempi per cui $[\text{attr2} \geq X]$ da quelli per cui $[\text{attr2} < X]$

Gli altri test partizionano insiemi più piccoli di esempi

Partizionamento dello spazio

E se gli esempi fossero messi così?



Lavorando su un singolo attributo non è possibile effettuare tagli diagonali o più complessi (tagli curvi)

Esistono in letteratura **alberi di decisione obliqui**, prodotti da meccanismi noti come *induzione costruttiva*, in grado di realizzare test basati su composizioni di attributi

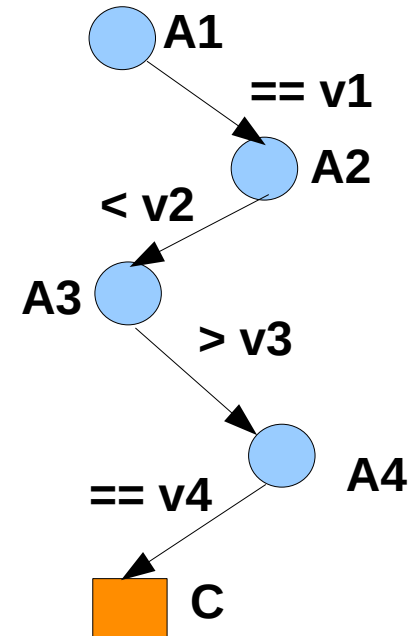
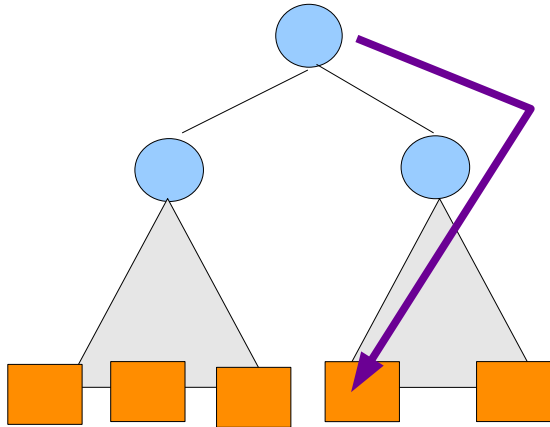
Induzione di alberi di decisione: commenti 1/2

- Gli **algoritmi di induzione** di DT sono **non-parametrici**, non occorrono particolari assunzioni sulle distribuzioni di probabilità
- La costruzione di un albero ottimale è un **problema NP-completo**, solitamente si adottano delle euristiche
- La costruzione di un DT è **computazionalmente poco costosa**; dato un albero, la **classificazione** ha una **complessità nel caso peggiore $O(w)$** , dove w rappresenta la profondità dell'albero
- Un DT è di **semplice interpretazione**, soprattutto se l'albero è piccolo
- I DT **non sono adatti a risolvere certi problemi di tipo booleano**, ad esempio a calcolare la funzione di parità (restituisce 1 se il #1 in una sequenza di bit è pari, 0 altrimenti) – vedere esercizio pag 198
- La presenza di **attributi irrilevanti non influenza negativamente** la costruzione dell'albero

Induzione di alberi di decisione: commenti 2/2

- È possibile incorrere nella **frammentazione dei dati**: procedendo top-down a un certo punto i nodi hanno associato un numero di istanze troppo piccolo per essere statisticamente significativo
- Si può avere **replicazione di sottoalberi**
- Partizionamento dello spazio tramite **tagli rettilinei e paralleli agli assi**
- Poiché molte **misure di impurità sono consistenti le une con le altre**, variare funzione di impurità spesso non modifica sostanzialmente la qualità degli alberi costruiti

Interpretazione di un DT



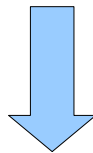
If (A1 == v1 && A2 < v2 && A3 > v3 && A4 == v4) **then** C

Overfitting

Errore di generalizzazione



Learning set



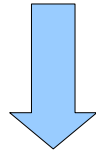
Definizione di sedia:

`(numero_gambe = 4) && (schienale == sì)`

Errore di generalizzazione

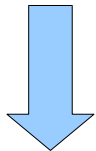


Test set



Sono sedie?

(numero_gambe = 4) && (schienale == sì)

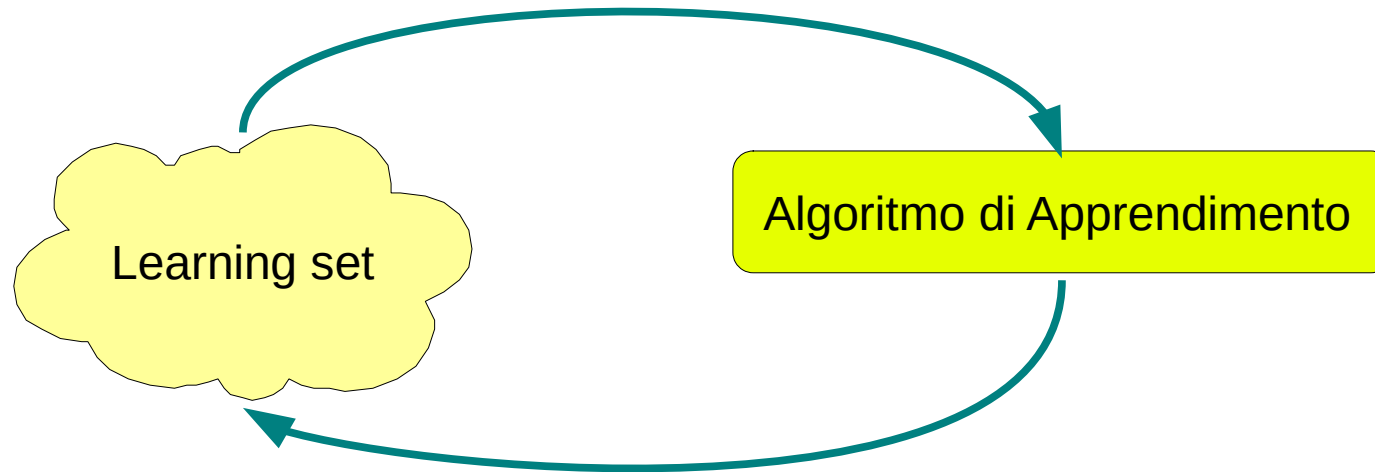


No !!

Il modello appreso è troppo specifico

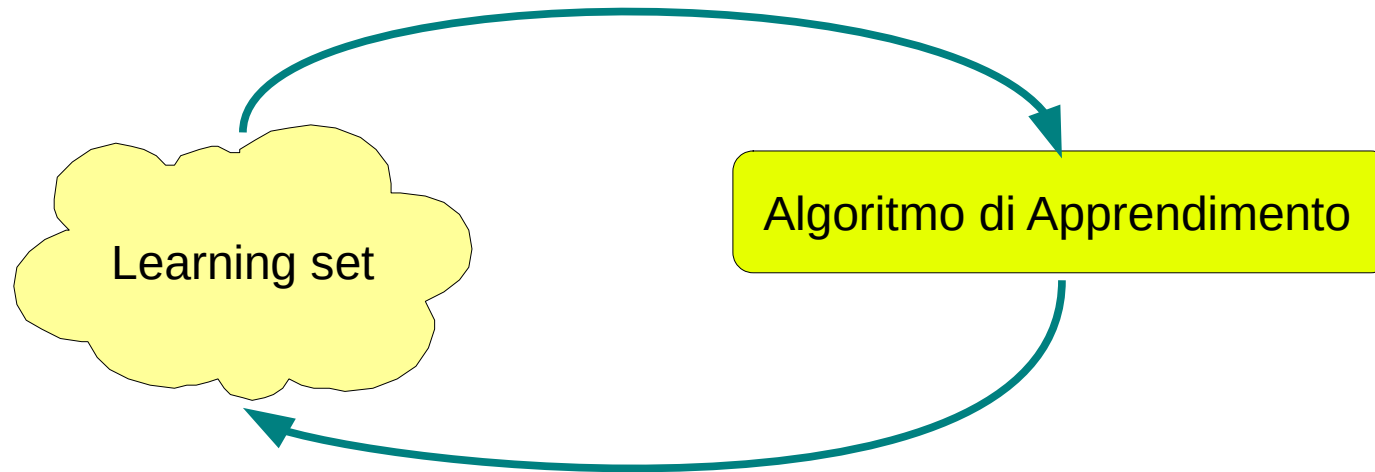
NB: l'esempio è semplice per fornire un'intuizione, di solito si ha overfitting con alberi grandi (modelli complessi)

A cosa è dovuto?



Un possibile condizione di terminazione è: itera l'applicazione dell'algoritmo finché l'errore di classificazione degli esempi di training non scende al di sotto di una certa soglia

A cosa è dovuto?



Un possibile condizione di terminazione è: itera l'applicazione dell'algoritmo finché l'errore di classificazione degli esempi di training non scende al di sotto di una certa soglia

Caso 1: noise
per esempio alcune istanze sono classificate in modo errato

Caso 2: mancanza di esempi
il learning set non rappresenta tutti i casi significativi

Overfitting dovuto a confronti multipli

Consideriamo la costruzione di un albero di decisione: ad ogni iterazione occorre individuare un attributo su cui effettuare il test. Un attributo **viene preso in considerazione** se il guadagno che dà supera una soglia minima

Spesso la procedura di costruzione è **greedy**: cerca di massimizzare il guadagno

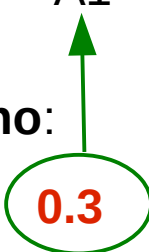
Overfitting dovuto a confronti multipli

Consideriamo la costruzione di un albero di decisione: ad ogni iterazione occorre individuare un attributo su cui effettuare il test. Un attributo **viene preso in considerazione** se il guadagno che dà supera una soglia minima

Spesso la procedura di costruzione è **greedy**: cerca di massimizzare il guadagno

 Nodo corrente

Attributi per cui il guadagno è sufficiente (di solito ci sono più possibilità):

	A1	A2	A3	A4
Guadagno:	 0.3	0.2	0.2	0.1

Guadagno massimo significa **miglior modellazione delle istanze di learning**

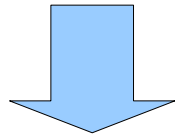
Overfitting

Modello ideale: modello che produce il *minor errore di generalizzazione* possibile. Come poter approssimare il modello ideale quando si ha a disposizione solo un insieme di esempi di learning?

Overfitting

Modello ideale: modello che produce il *minor errore di generalizzazione* possibile. Come poter approssimare il modello ideale quando si ha a disposizione solo un insieme di esempi di learning?

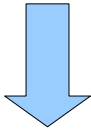
Rasoio di Occam!



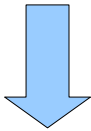
Incorporare una nozione di complessità nel modello
A parità di errore i modelli più semplici sono preferibili

Minimum description length

Implementazione del rasoio di Occam: la migliore ipotesi per la modellazione di un data set è quella che consente la *massima compressione* dei dati



Modello = strumento che consente di rappresentare i dati in modo compatto catturando le loro regolarità



Apprendimento = strumento per catturare regolarità nei dati

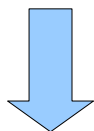
È migliore un modello **accurato** che al contempo è **poco costoso da comunicare** ad un'altra parte che desideri utilizzarlo

FACOLTATIVO

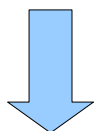
Minimum description length

FACOLTATIVO

Implementazione del rasoio di Occam: la migliore ipotesi per la modellazione di un data set è quella che consente la *massima compressione* dei dati



Modello = strumento che consente di rappresentare i dati in modo compatto catturando le loro regolarità



Apprendimento = strumento per catturare regolarità nei dati

si usa la formulazione di base del MDL detta **two-part code**, in generale:

Siano $H(1)$, $H(2)$, ... dei **modelli candidati**, contenenti **ipotesi**.

L'ipotesi migliore $H \in H(1) \cup H(2) \cup \dots$ per spiegare i dati D è quella che **minimizza la somma** $L(H) + L(D|H)$, dove:

- $L(H)$ è la lunghezza, in bit, della descrizione dell'ipotesi
- $L(D|H)$ è la lunghezza, in bit, delle descrizioni dei dati codificati con l'aiuto dell'ipotesi.

Il modello migliore per spiegare D è il modello più piccolo.

Minimum description length: nota

Nota: in questa formalizzazione un modello $H(1)$ cattura una famiglia di possibili funzioni (nel nostro caso di classificazione) che hanno tutte la stessa forma.

Un'ipotesi è un'istanza della forma di una funzione.

Esempio:

Modello: $y = A \cdot x^2$

Ipotesi: $y = 0.75 \cdot x^2$

FACOLTATIVO

MDL e decision trees

L'MDL deriva dalla *teoria dell'informazione*, la lunghezza in bit indica il costo della trasmissione del modello e dei dati

Esempio di calcolo dell'MDL su alberi di decisione:

$$\text{Costo}(\text{albero}, \text{dati}) = \text{Costo}(\text{albero}) + \text{Costo}(\text{dati} \mid \text{albero})$$

codifica di un nodo: identificatore dell'attributo su cui si fa il test

codifica di una foglia: identificatore della classe associata

Costo(albero): costo della codifica di tutti i suoi nodi

Costo(dati | albero): codifica basata sull'errore di classificazione

FACOLTATIVO

MDL e decision trees

L'MDL deriva dalla *teoria dell'informazione*, la lunghezza in bit indica il costo della trasmissione del modello e dei dati

Esempio di calcolo dell'MDL su alberi di decisione:

$$\text{Costo(albero, dati)} = \text{Costo(albero)} + \text{Costo(dati | albero)}$$

codifica di un nodo: identificatore dell'attributo su cui si fa il test

**supponiamo di avere m attributi,
possiamo rappresentarli con un numero.**

Per codificare un numero *compreso fra 1 e m* occorrono **$\log_2 m$** bit

Es. per codificare un numero fra 1 e 4 occorrono 2 bit

codifica di una foglia: identificatore della classe associata

Costo(albero): costo della codifica di tutti i suoi nodi

Costo(dati | albero): codifica basata sull'errore di classificazione

FACOLTATIVO

MDL e decision trees

L'MDL deriva dalla *teoria dell'informazione*, la lunghezza in bit indica il costo della trasmissione del modello e dei dati

Esempio di calcolo dell'MDL su alberi di decisione:

$$\text{Costo(albero, dati)} = \text{Costo(albero)} + \text{Costo(dati | albero)}$$

codifica di un nodo: identificatore dell'attributo su cui si fa il test

codifica di una foglia: identificatore della classe associata

se abbiamo **k classi** occorrono **$\log_2 k$** bit

Costo(albero): costo della codifica di tutti i suoi nodi

Costo(dati | albero): codifica basata sull'errore di classificazione

FACOLTATIVO

MDL e decision trees

L'MDL deriva dalla *teoria dell'informazione*, la lunghezza in bit indica il costo della trasmissione del modello e dei dati

Esempio di calcolo dell'MDL su alberi di decisione:

$$\text{Costo}(\text{albero}, \text{dati}) = \text{Costo}(\text{albero}) + \text{Costo}(\text{dati} \mid \text{albero})$$

codifica di un nodo: identificatore dell'attributo su cui si fa il test

codifica di una foglia: identificatore della classe associata

Costo(albero): costo della codifica di tutti i suoi nodi

possiamo pensare che sia la **somma dei costi dei suoi nodi**

Costo(dati | albero): codifica basata sull'errore di classificazione

FACOLTATIVO

MDL e decision trees

FACOLTATIVO

L'MDL deriva dalla *teoria dell'informazione*, la lunghezza in bit indica il costo della trasmissione del modello e dei dati

Esempio di calcolo dell'MDL su alberi di decisione:

$$\text{Costo}(\text{albero}, \text{dati}) = \text{Costo}(\text{albero}) + \text{Costo}(\text{dati} \mid \text{albero})$$

codifica di un nodo: identificatore dell'attributo su cui si fa il test

codifica di una foglia: identificatore della classe associata

Costo(albero): costo della codifica di tutti i suoi nodi

Costo(dati | albero): codifica basata sull'errore di classificazione

L'errore è dato fornendo l'istanza classificata erroneamente, quindi per ogni errore viene aggiunto il costo di indicare l'istanza misclassificata, Sia N_E il numero degli errori di classificazione compiuti.

Se il **numero di istanze di training è n** occorrono **$\log_2 n$** bit per rappresentarne ciascuna, quindi $\text{Costo}(\text{dati} \mid \text{albero})$ sarà **$\log_2 n * N_E$**

Si potrebbero tenere molte lezioni sul solo MDL, per approfondimenti:

Tutorial: P.Grünwald, A tutorial introduction to the minimum description length principle. In: Advances in Minimum Description Length: Theory and Applications (edited by P. Grünwald, I.J. Myung, M. Pitt), MIT Press, 2005. (<https://arxiv.org/pdf/math/0406077.pdf> sezione 1.3 in particolare)

FACOLTATIVO

Gestione dell'overfitting durante l'induzione

- **Pruning**: potatura dell'albero \Rightarrow semplificazione del modello \Rightarrow generalizzazione del modello
 - Prepruning
 - Postpruning

Gestione dell'overfitting durante l'induzione

- **Pruning**: potatura dell'albero \Rightarrow semplificazione del modello \Rightarrow generalizzazione del modello
 - **Prepruning**: la costruzione del DT si interrompe prima che l'albero sia completo. Si impone una regola di terminazione più restrittiva
 - **Problema**: definire la nuova condizione di terminazione
 - Postpruning

Gestione dell'overfitting durante l'induzione

- **Pruning**: potatura dell'albero \Rightarrow semplificazione del modello \Rightarrow generalizzazione del modello
 - Prepruning
 - **Postpruning**: prima si costruisce l'albero poi si potano alcuni rami, trasformando alcuni nodi interni in foglie
 - **Problema**: definire la condizione per decidere se un ramo è da potare o meno

Prepruning (early stopping rule)

Regola di interruzione, esempio: non eseguo lo split se il gain è al di sotto di una certa soglia

Vantaggio: evita l'overfitting dei dati di learning

Problema: è difficile scegliere la soglia, se troppo alta si ha *underfitting*

Post-pruning

Questo approccio consiste nel tagliare rami da un albero fatto crescere finché non si hanno più guadagni

Possibili strategie:

- (1) sostituisco un sottoalbero col solo suo cammino usato più di frequente;
- (2) sostituisco un sottoalbero con una foglia la cui classe corrisponde alla classe
Maggiormente rappresentata nelle foglie dell'albero rimosso.

Tende a dare *risultati migliori* del pre-pruning

Problema di efficienza: tagliare rami costruiti significa che il tempo trascorso a costruirli è stato sprecato