

Classificazione II: Tecniche alternative

Classificazione: Definizione

- Data una collezione di record (*training set*)
 - ✓ Ogni record è composto da un insieme di *attributi*, di cui uno esprime la *classe* di appartenenza del record.
- Trova un *modello* per l'attributo di classe che esprima il valore dell'attributo di classe in funzione dei valori degli altri attributi.
- Obiettivo: record non noti devono essere assegnati a una classe nel modo più accurato possibile
 - ✓ Viene utilizzato un *test set* per determinare l'accuratezza del modello. Normalmente, il data set fornito è suddiviso in training set e test set. Il primo è utilizzato per costruire il modello, il secondo per validarlo.
- I classificatori possono essere utilizzati sia a scopo descrittivo sia a scopo predittivo
- Sono più adatti ad attributi nominali (binari o discreti) poichè faticano a sfruttare le relazioni implicite presenti negli attributi ordinali, numerici o in presenza di gerarchie di concetti (es. scimmie e uomini sono primati)

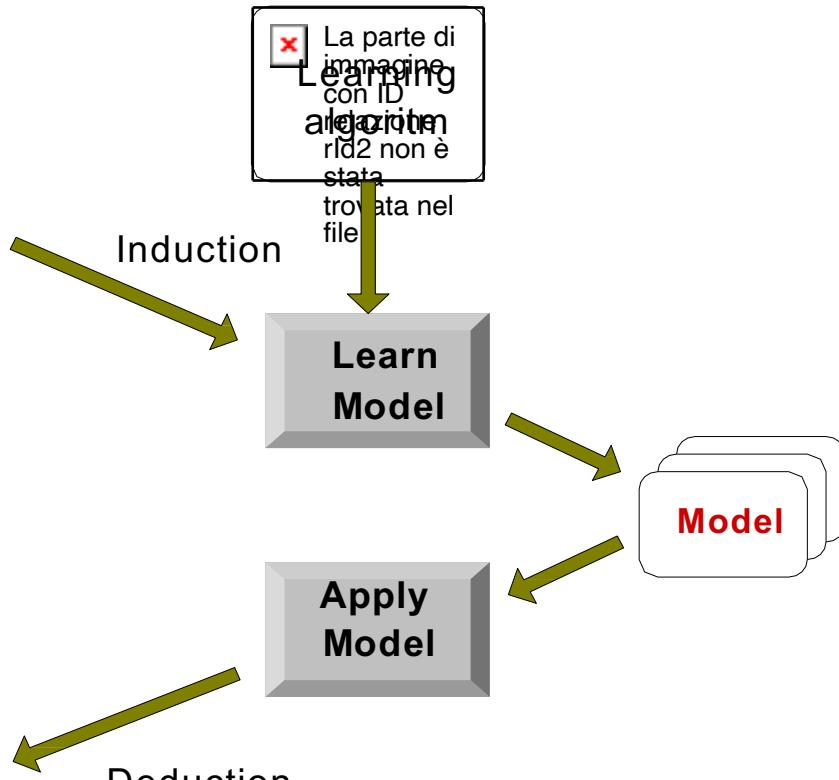
Classificazione: un esempio

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Tecniche di classificazione

- Alberi decisionali o Decision Tree
- **Regole di decisione**
- Nearest-neighbor
- Reti Bayesiane
- Logistic Regression
- Neural Networks
- Support Vector Machines

Rule Based Classification

Classificatori basati su regole

- Classificano i record utilizzando insiemi di **regole del tipo “if...then...”**
- Una regola ha la forma: (*Condizione*) → *y*
 - ✓ *LHS*: antecedente o *Condizione* è una congiunzione di predicati su attributi, *y* è l'etichetta di classe
 - ✓ *RHS*: conseguente o etichetta della classe
 - ✓ Esempi di regole:
 - (Blood Type = Warm) ∧ (Lay Eggs = Yes) → Class = Birds
 - (Taxable Income < 50K) ∧ (Refund = Yes) → Evade = No

Costruire un modello significa identificare un insieme di regole

° La sintassi prevede che la testa della regola contenga solo il valore dell'attributo

Vantaggi e svantaggi

Pro

Espressivi quanto gli alberi decisionali

-  Facili da interpretare
-  Facili da generare
-  Rapidi nella classificazione di nuove istanze

Contro

-  Il costo di costruzione non scala al crescere del training set
-  Risentono fortemente del rumore sui dati

Classificatori basati su regole - Esempio

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

$r1: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds}$

$r2: (\text{Give Birth} = \text{no}) \wedge (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes}$

$r3: (\text{Give Birth} = \text{yes}) \wedge (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals}$

$r4: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles}$

$r5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians}$

Classificatori basati su regole

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

r1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

r2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

r3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

r4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

r5: (Live in Water = sometimes) \rightarrow Amphibians

Classificatori basati su regole

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

r1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

r2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

r3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

r4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

r5: (Live in Water = sometimes) \rightarrow Amphibians

Classificatori basati su regole

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

r1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

r2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

r3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

r4: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

r5: (Live in Water = sometimes) \rightarrow Amphibians

Classificatori basati su regole

- Una regola r copre un'istanza \mathbf{x} se i valori di \mathbf{x} soddisfano l'antecedente di r

$r1$: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

$r2$: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

$r3$: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

$r4$: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

$r5$: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

Classificatori basati su regole

- Una regola r copre un'istanza \mathbf{x} se i valori di \mathbf{x} soddisfano l'antecedente di r

$r1$: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

$r2$: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

$r3$: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

$r4$: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

$r5$: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	Bird
grizzly bear	warm	yes	no	no	?

La regola $r1$ copre hawk \rightarrow Bird

Classificatori basati su regole

- Una regola r copre un'istanza \mathbf{x} se i valori di \mathbf{x} soddisfano l'antecedente di r

$r1$: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

$r2$: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

$r3$: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

$r4$: (Give Birth = no) \wedge (Can Fly = no) \rightarrow Reptiles

$r5$: (Live in Water = sometimes) \rightarrow Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	Mammals

La regola $r3$ copre grizzly bear \rightarrow Mammal

Copertura e accuratezza

- Dato un dataset D e una regola di classificazione A→y definiamo

- Copertura:

✓ Frazione dei record che soddisfano l'antecedente della regola

$$Coverage(r) = \frac{|A|}{|D|}$$

- Accuratezza:

✓ Frazione dei record che, soddisfando l'antecedente, soddisfano anche il conseguente della regola

$$Accuracy(r) = \frac{|A \cap y|}{|A|}$$

RTid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$r = (\text{Status}=\text{Single}) \rightarrow \text{No}$$

$$Coverage(r) = 40\%, \ Accuracy(r) = 50\%$$

Come funzionano le regole?

$r1: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds}$

$r2: (\text{Give Birth} = \text{no}) \wedge (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes}$

$r3: (\text{Give Birth} = \text{yes}) \wedge (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals}$

$r4: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles}$

$r5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians}$

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

Come funzionano le regole?

$r1: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds}$

$r2: (\text{Give Birth} = \text{no}) \wedge (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes}$

$r3: (\text{Give Birth} = \text{yes}) \wedge (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals}$

$r4: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles}$

$r5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians}$

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

Un **lemur** attiva la regola **r3**, quindi è classificato come **mammifero**

Come funzionano le regole?

$r1: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds}$

$r2: (\text{Give Birth} = \text{no}) \wedge (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes}$

$r3: (\text{Give Birth} = \text{yes}) \wedge (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals}$

$r4: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles}$

$r5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians}$

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

Un lemur attiva la regola $r3$, quindi è classificato come mammifero

Una tartaruga attiva sia $r4$ sia $r5$ che determinano classi diverse!

Come funzionano le regole?

$r1: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds}$

$r2: (\text{Give Birth} = \text{no}) \wedge (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes}$

$r3: (\text{Give Birth} = \text{yes}) \wedge (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals}$

$r4: (\text{Give Birth} = \text{no}) \wedge (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles}$

$r5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians}$

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

Un lemur attiva la regola $r3$, quindi è classificato come mammifero

Una tartaruga attiva sia $r4$ sia $r5$ che determinano classi diverse!

Uno **squalo** non attiva nessuna regola!!

Proprietà - Strategia 1

- **Regole mutuamente esclusive:** un insieme di regole R è detto mutuamente esclusivo se nessuna coppia di regole può essere attivata dallo stesso record
 - ✓ Ogni record è coperto al più da una regola
- **Regole esaustive:** un insieme di regole R ha una copertura esaustiva se esiste una regola per ogni combinazione di valori degli attributi
 - ✓ Ogni record è coperto da almeno una regola

Proprietà - Strategia 2

- Non sempre è possibile determinare un insieme di regole esaustive e mutualmente esclusive
- Mancanza di mutua esclusività
 - ✓ Un record può attivare più regole dando vita a classificazioni alternative
 - ✓ Soluzione
 - Definire **un ordine di attivazione** delle regole. Si parla in questo caso di **liste di decisione**
 - Assegnare il record alla classe per la quale vengono attivate più regole (voto)
- Mancanza di esaustività
 - ✓ Un record può non attivare nessuna regola
 - ✓ Soluzione
 - Utilizzare una **classe di default** (“altro”) a cui il record viene associato in caso di non attivazione delle regole

Regole (linearmemente) ordinate

- Le regole sono ordinate secondo una data priorità
 - Un insieme ordinato di regole è anche chiamato lista di decisione
- Quando un record è sottomesso al classificatore
 - È assegnato alla classe della prima regola che viene attivata
 - Se nessuna regola è attivata, è assegnato a una classe di default

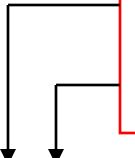
R1: (Give Birth = no) \wedge (Can Fly = yes) \rightarrow Birds

R2: (Give Birth = no) \wedge (Live in Water = yes) \rightarrow Fishes

R3: (Give Birth = yes) \wedge (Blood Type = warm) \rightarrow Mammals

R4: (**Give Birth = no**) \wedge (**Can Fly = no**) \rightarrow Reptiles

R5: (Live in Water = sometimes) \rightarrow Amphibians



Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

Modalità di ordinamento

■ Ordinamento Rule-based

- ✓ Le singole regole sono ordinate in base alla loro qualità

■ Ordinamento Class-based

- ✓ Gruppi di regole che determinano la stessa classe compaiono di seguito nella lista. L'ordinamento rilevante diventa quello tra le classi che può dipendere dall'importanza della classe o dalla gravità di commettere un errore di classificazione per quella classe
- ✓ Il rischio con questa soluzione è che una regola di buona qualità sia superata da una regola di qualità inferiore ma appartenente a una classe importante
- ✓ Soluzione usata dai principali algoritmi di costruzione delle regole RIPPER, C4.5 rules)

Rule-based Ordering

(Refund=Yes) → No

(Refund=No, Marital Status={Single,Divorced},
Taxable Income<80K) → No

(Refund=No, Marital Status={Single,Divorced},
Taxable Income>80K) → Yes

(Refund=No, Marital Status={Married}) → No

Class-based Ordering

(Refund=Yes) → No

(Refund=No, Marital Status={Single,Divorced},
Taxable Income<80K) → No

(Refund=No, Marital Status={Married}) → No

(Refund=No, Marital Status={Single,Divorced},
Taxable Income>80K) → Yes

Esercizio

■ Attributi e valori

- ✓ Air Conditioner = {Working, Broken}
- ✓ Engine = {Good, Bad}
- ✓ Mileage = {High, Medium, Low}
- ✓ Rust = {Yes, No}

■ Insieme di regole

- ✓ Mileage = High → Value = Low
- ✓ Mileage = Low → Value = High
- ✓ Air Conditioner = Working, Engine = Good → Value = High
- ✓ Air Conditioner = Working, Engine = Bad → Value = Low
- ✓ Air Conditioner = Broken → Value = Low

■ Quesiti

- ✓ Le regole sono esaustive?
- ✓ Le regole sono mutualmente esclusive?
- ✓ E' necessario un ordinamento?
- ✓ E' necessaria una classe di default?

Costruzione delle regole

■ Metodi diretti

- ✓ Estraggono le regole direttamente dai dati (es. RIPPER, CN2, Holte's 1R)

■ Metodi indiretti

- ✓ Estraggono le regole dal risultato di altri metodi di classificazione come, ad esempio, i decision tree (C4.5 rules)

Metodo Diretto: Sequential Covering

1. Inizia da una regola vuota
2. Incrementa la regola usando la funzione «Learn-One-Rule»
3. Rimuovi i record di training coperti dalla regola
4. Ripeti i passaggi (2) e (3) fino a quando il criterio di arresto non viene soddisfatto

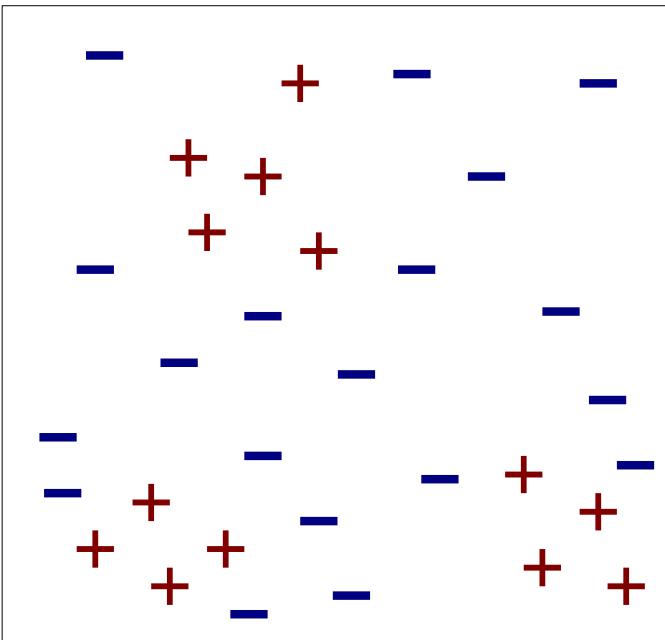
Metodi diretti: sequential covering

Let E be the training set, $A=\{(A_j, v_j)\}$ the set of attribute-value pairs and $Y_0=\{y_1 \dots y_k\}$ the set of classes where y_k is the default class

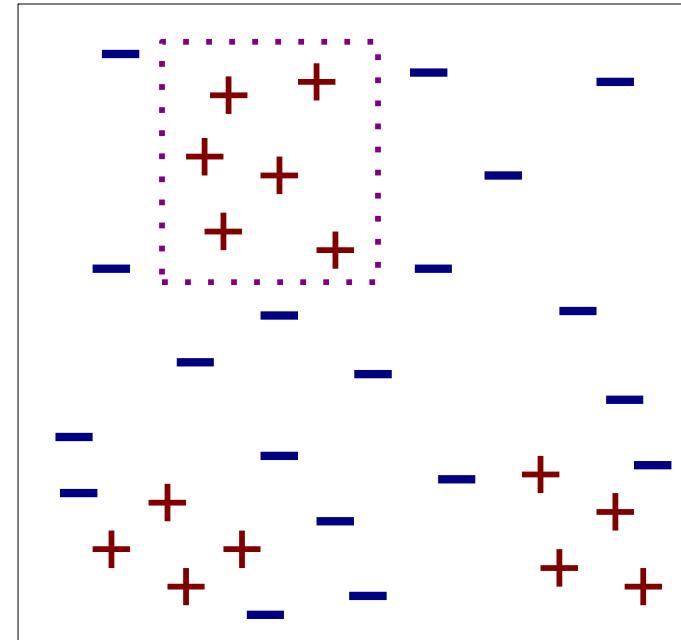
```
set  $R = \emptyset$ 
for each class  $y \in Y_0 - \{y_k\}$  do
    stop = FALSE;
    while !stop do
         $r = \text{Learn-One-Rule}(E, A, y)$ 
        remove from  $E$  training records that are covered by  $r$ 
        if Quality( $r, E$ ) < Threshold then
            Stop = TRUE;
        else
             $R = R \cup r$  // Add  $r$  at the bottom of the rule list
        end while
    end for
 $R = R \cup \{\{\} \rightarrow y_k\}$  // Add the default rule at the bottom of the rule list
PostPruning( $R$ );
```

- L'ordinamento delle regole è determinato dalle classi. Punti di attenzione:
 - ✓ Modalità di creazione delle regole (Learn-One-Rule)
 - ✓ Eliminazione delle istanze
 - ✓ Criterio di stop

Sequential covering: esempio



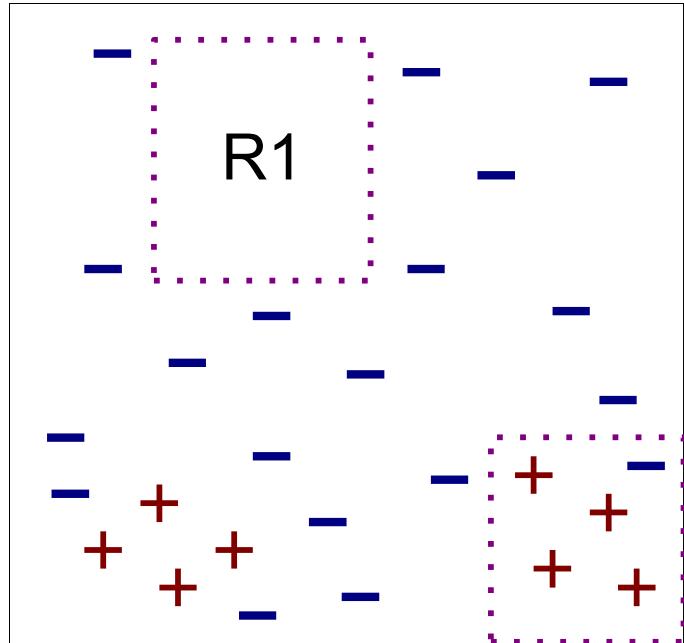
(i) Original Data



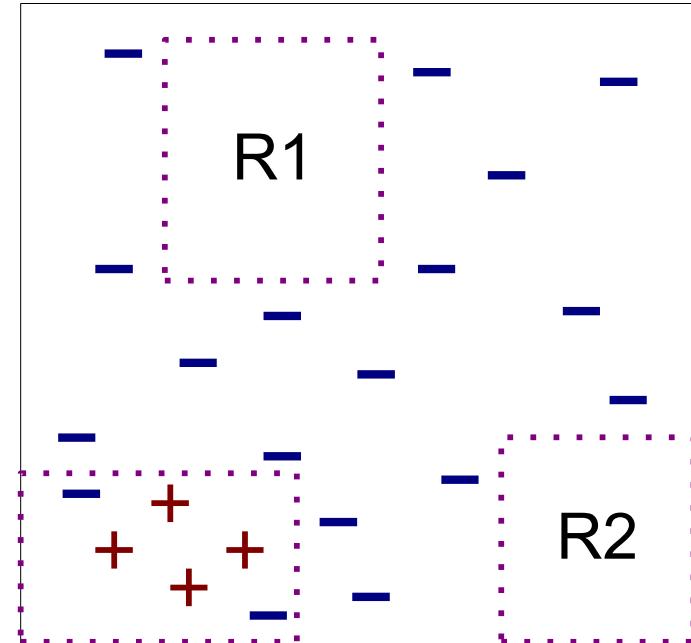
(ii) Step 1

- Chiameremo **esempi positivi** (+) tutte le tuple del training set che appartengono alla classe y . Sono **esempi negativi** (-) tutti gli altri.

Sequential covering: esempio



(iii) Step 2



(iv) Step 3

Eliminazione delle istanze dal training set

- L'eliminazione delle istanze dal training set serve a:

- ✓ Istanze classificate correttamente:
 - ✓ evitare di generare sempre la stessa regola,
 - ✓ evitare di sovrastimare l'accuracy della prossima regola
- ✓ Istanze classificate non correttamente:
 - ✓ evitare di sottostimare l'accuracy della prossima regola

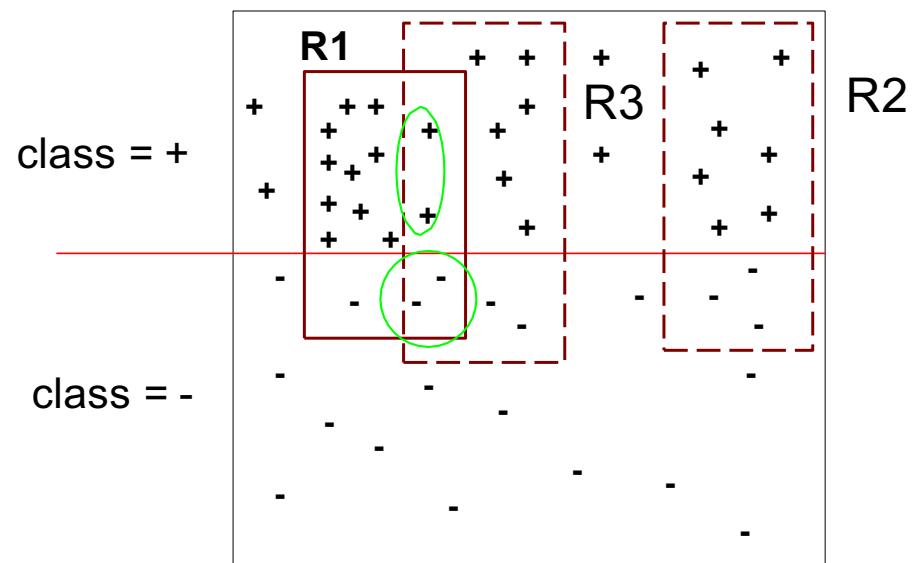
Esempio

- Accuracy(R1)=12/15 (80%)
- Accuracy(R2)=7/10 (70%)
- Accuracy(R3)=8/12 (66.7%)

Le istanze conteggiate per la regola R1 non devono essere conteggiate anche per la regola R3

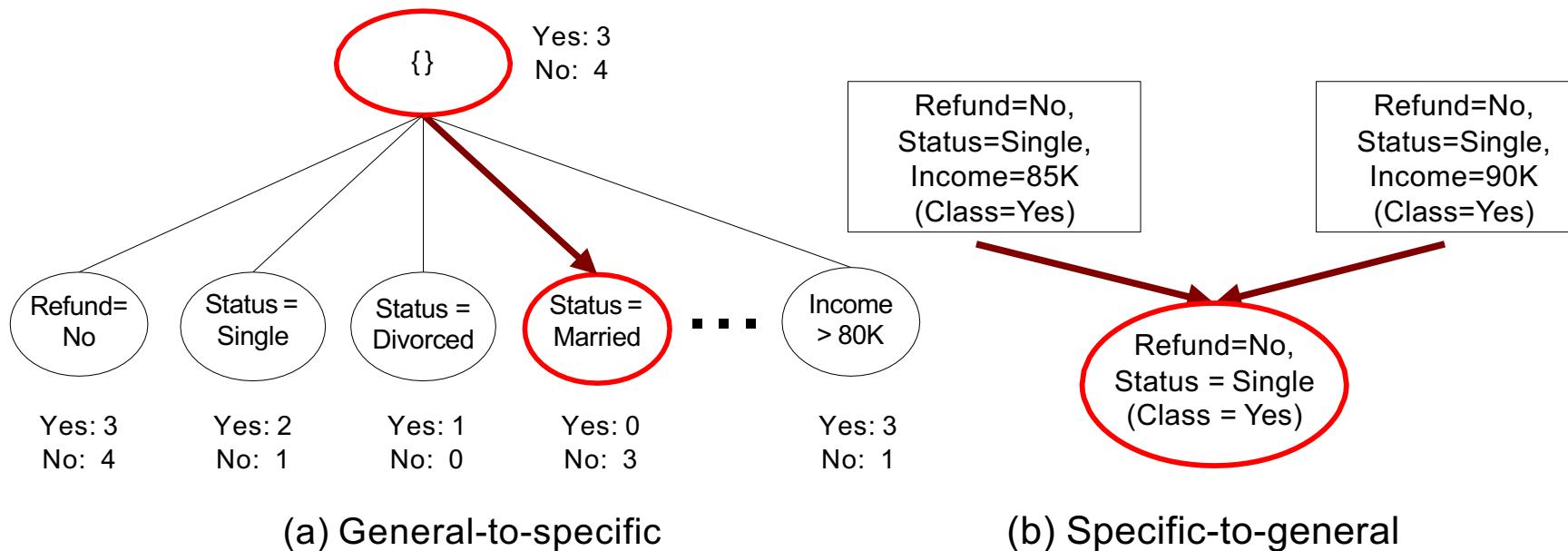
Dopo aver scelto R1

- Accuracy(R2)=7/10 (70%)
- Accuracy(R3)=6/8 (75%)



Learn one rule: 2 diverse strategie

- L'obiettivo dell'algoritmo è trovare una regola che copra quanti più possibili esempi positivi e quanto meno possibili esempi negativi
 - ✓ Il problema è complesso dato che lo spazio di ricerca è esponenziale nel numero di possibili combinazioni dei predicati
- Le regole sono costruite considerando progressivamente un nuovo possibile atomo ossia una possibile coppia (Attributo,Valore)



Valutazione delle regole: Accuracy

- E' necessario un criterio per scegliere quale atomo aggiungere
 - n : numero di istanze coperte dalla regola r (coverage)
 - n_r : numero di istanze correttamente classificate da r
 - k : numero di classi

$$Accuracy(r) = \frac{n_r}{n}$$

Limitata applicabilità poiché non considera la copertura

Valutazione delle regole: Accuracy

Esempio

Dataset con 60 esempi positivi e 100 esempi negativi

- r_1 : copre 50 esempi positivi e 5 esempi negativi
 - r_2 : copre 2 esempi positivi e 0 esempi negativi
-
- $Accuracy(r_1) = \frac{50}{50+5} = 90,9\%$
 - $Accuracy(r_2) = \frac{2}{2+0} = 100\%$

L'elevata accuratezza di r_2 è probabilmente una indicazione non corretta perché la copertura della regola è troppo bassa.
La regola r_1 sembrerebbe essere migliore della regola r_2 .

Valutazione delle regole: Likelihood Ratio

- ✓ Un test statistico che può essere usato per valutare regole che hanno una copertura scarsa è:

$$R = 2 \sum_{i=1}^k f_i \log \frac{f_i}{e_i}$$

- ✓ dove
 - k è il numero di classi,
 - f_i è la frequenza osservata degli esempi di classe i che sono coperti dalla regola,
 - e_i è la frequenza prevista di una regola che effettua previsioni a caso.

Valutazione delle regole: Likelihood Ratio

Esempio

Dataset con 60 esempi positivi e 100 esempi negativi

- r_1 : copre 50 esempi positivi e 5 esempi negativi
- r_2 : copre 2 esempi positivi e 0 esempi negativi

- $e_+(r_1) = 55 \frac{60}{160} = 20,625 \quad e_-(r_1) = 55 \frac{100}{160} = 34,375$
- $e_+(r_2) = 2 \frac{60}{160} = 0,75. \quad e_-(r_2) = 2 \frac{100}{160} = 1,25$
- $R(r_1) = 2 \left[50 \times \log_2 \frac{50}{20,625} + 5 \times \log_2 \frac{5}{34,375} \right] = 99,9$
- $R(r_2) = 2 \left[2 \times \log_2 \frac{2}{0,75} + 0 \times \log_2 \frac{2}{1,25} \right] = 8,37$

La regola r_1 è preferibile alla regola r_2

Valutazione delle regole: Laplace

- ✓ Metriche che considerano la coverage: pesano l'accuracy in base alla coverage

$$Laplace(r) = \frac{f_+ + 1}{n + k}$$

- ✓ dove
 - k è il numero di classi,
 - n è il numero di esempi coperti dalla regola r
 - f_+ è il numero di esempi positivi coperti dalla regola r
- ✓ Note:
 - Se la *copertura* è 0 ($n = 0$ e $f_+ = 0$), assumendo distribuzione uniforme dei dati, l'indice si riduce alla probabilità a priori della classe ($1/k$)
 - Se la *copertura* è alta, l'indice tende asintoticamente al valore dell'*accuracy*
 - Per valori bassi di *copertura* k tende a ridurre il valore dell'indice

Valutazione delle regole: Laplace

Esempio

Dataset con 60 esempi positivi e 100 esempi negativi

- r_1 : copre 50 esempi positivi e 5 esempi negativi
- r_2 : copre 2 esempi positivi e 0 esempi negativi

- $Laplace(r_1) = \frac{50+1}{55+2} = 89,45\%$
- $Laplace(r_2) = \frac{2+1}{2+2} = 75,00\%$

La regola r_1 è preferibile alla regola r_2

Valutazione delle regole: m-estimate

- ✓ Metriche che considerano la coverage: pesano l'accuracy in base alla coverage

$$m - estimate(r) = \frac{f_+ + k p_+}{n + k}$$

- ✓ dove
 - k è il numero di classi,
 - n è il numero di esempi coperti dalla regola r
 - f_+ è il numero di esempi positivi coperti dalla regola r
 - p_+ è la probabilità a priori della classe +

Si noti che per $p_+ = 1/k$, coincide con Laplace

Valutazione delle regole: m-estimate

Esempio

Dataset con 60 esempi positivi e 100 esempi negativi

- r_1 : copre 50 esempi positivi e 5 esempi negativi
- r_2 : copre 2 esempi positivi e 0 esempi negativi ×

- $m - estimate(r_1) = \frac{50 + 2 \times \frac{60}{160}}{55 + 2} = 97,60\%$
- $m - estimate(r_2) = \frac{2 + 2 \times \frac{60}{160}}{2 + 2} = 68,75\%$

La regola r_1 è preferibile a r_2

Valutazione delle regole: FOIL

FOIL (First Order Inductive Learner) misura la variazione dovuta all'incremento della regola con l'aggiunta di un nuovo atomo

- ✓ Supponiamo che la regola $r_o: A \rightarrow +$ copra p_0 esempi positivi e n_0 esempi negativi.
- ✓ Dopo aver aggiunto un nuovo atomo B, la regola estesa $r_1: A, B \rightarrow +$ copre p_1 esempi positivi e n_1 esempi negativi.
- ✓ Il guadagno di informazioni di FOIL della regola estesa è definito come segue:

$$\checkmark \text{FoilGain}(r_o, r_1) = p_1 \left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0} \right)$$

L'indice è proporzionale a p_1 e a $p_1 / (p_1 + n_1)$, quindi tende a favorire regole che hanno elevate *coverage* e *accuracy*

- ✓ Misura alternativa: $v(r_o, r_1) = \frac{p_1 - n_1}{p_1 + n_1} - \frac{p_0 - n_0}{p_0 + n_0}$

Valutazione delle regole: Laplace

Esempio. (FOIL GAIN)

- $r_0 = A \rightarrow +$ copre 50 esempi positivi e 5 esempi negativi
- $r_1 A, B \rightarrow +$ copre 45 esempi positivi e 4 esempi negativi

- $FoilGain(r_0, r_1) = 45 \left(\log_2 \frac{45}{45+4} - \log_2 \frac{50}{50+5} \right) = 0,659$
- $v(r_0, r_1) = \frac{45-4}{45+4} - \frac{50-5}{50+5} = 0,019$

Sequential covering

- Criterio di stop nell'estensione di una regola
 - ✓ Calcola il gain
 - ✓ Se il gain non è significativo stop!
 - Ripeti i passi precedenti fino a quando c'è un miglioramento
- Rule Pruning: ha l'obiettivo di semplificare le regole per migliorare l'errore di generalizzazione delle regole,
 - ✓ può essere utile visto che l'approccio di costruzione è greedy
 - ✓ Possono essere utilizzate le tecniche viste per gli alberi decisionali
 - Minimum Description Length
 - Approccio pessimistico
 - Utilizzo del validation set
 - ✓ Esempio (Reduced error pruning):
 - Rimuovi a turno un atomo dalla regola
 - Elimina l'atomo la cui rimozione comporta il massimo miglioramento dell'error rate sul validation set
 - Ripeti i passi precedenti se c'è un miglioramento

Metodi diretti: RIPPER

- *Repeated Incremental Pruning to Produce Error Reduction*
- Approccio basato sul sequential covering
 - ✓ Una sua versione denominata JRip è implementata in WEKA
- Per problemi a 2 classi, scegli una delle classi come classe positiva e l'altra come classe negativa
 - ✓ Costruisci le regole per la classe positiva
 - ✓ La classe negativa sarà la classe di default
- Per problemi multi-classe
 - ✓ Ordina le classi in base a un criterio di rilevanza della classe (frazione delle istanze che appartengono alla classe)
 - ✓ Costruisci le regole a partire dalla classe più piccola e considerando il resto delle istanze come classe negativa
 - ✓ Ripeti l'operazione con le classi successive

Metodi diretti: RIPPER

- Costruzione del set di regole:

- Usa l'algoritmo “sequential covering”
 - ◆ Trova la regola migliore che copre l'attuale serie di esempi positivi
 - ◆ Elimina sia gli esempi positivi che quelli negativi coperti dalla regola
- Ogni volta che una regola viene aggiunta al set di regole, calcola la nuova lunghezza della descrizione
 - ◆ Interrompi l'aggiunta di nuove regole quando la lunghezza della descrizione è **d** bit più lunga della lunghezza della descrizione più piccola ottenuta finora

Metodi diretti: RIPPER

■ Learn-one-rule:

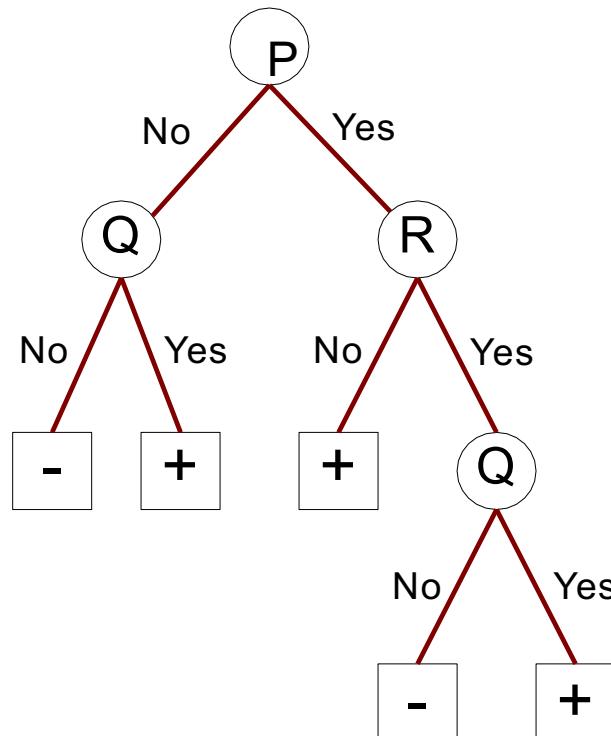
- ✓ Inizia con una regola vuota
- ✓ Aggiungi atomi finchè determinano un miglioramento del FoilGain
- ✓ Interrompi quando la regola non copre più esempi negativi
- ✓ Esegui il pruning utilizzando la tecnica dell'*incremental reduced error pruning*
 - Esegue il pruning dopo ogni passo di growing
- ✓ **Metodo di pruning:** elimina dalla regola gli atomi la cui rimozione massimizza: $v = (p-n)/(p+n)$
 - p: numero di esempi positivi coperti dalla regola nel validation set
 - n: numero di esempi negativi coperti dalla regola nel validation set

Metodi diretti: RIPPER

- Ottimizzazione del set di regole:
 - Per ciascuna regola r nel set di regole R
 - ◆ Considera 2 regole alternative :
 - Regola di sostituzione (r^*): Sviluppa una nuova regola da zero
 - Revised rule(r'): estendi r con l'aggiunta di un atomo
 - ◆ Confronta il set di regole per r con il set di regole per r^* e r'
 - ◆ Scegli l'insieme di regole che minimizzano MDL
 - Ripeti la generazione delle regole e l'ottimizzazione delle regole per gli esempi positivi rimanenti

Metodi indiretti

- E' possibile trasformare un decision tree in un insieme di regole (vedi C4.5rules)
 - ✓ Ad ogni percorso radice-foglia corrisponderà una regola
 - ✓ Saranno poi applicati opportune tecniche di prunini



Rule Set

- $r1: (P=No, Q=No) \rightarrow -$
- $r2: (P=No, Q=Yes) \rightarrow +$
- $r3: (P=Yes, R=No) \rightarrow +$
- $r4: (P=Yes, R=Yes, Q=No) \rightarrow -$
- $r5: (P=Yes, R=Yes, Q=Yes) \rightarrow +$

Metodi Indiretti: C4.5rules

- Estrai le regole da un albero decisionale non potato
- Per ogni regola, $r: A \rightarrow y$,
 - Considera una regola alternative $r': A' \rightarrow y$ dove A' è ottenuto rimuovendo un atomo da A
 - Confronta il tasso di errore pessimistico per r con quello di tutte le regole r'
 - Taglia l'atomo se una delle regole alternative ha un tasso di errore pessimistico inferiore
 - Ripeti fino a quando non è possibile più migliorare l'errore di generalizzazione

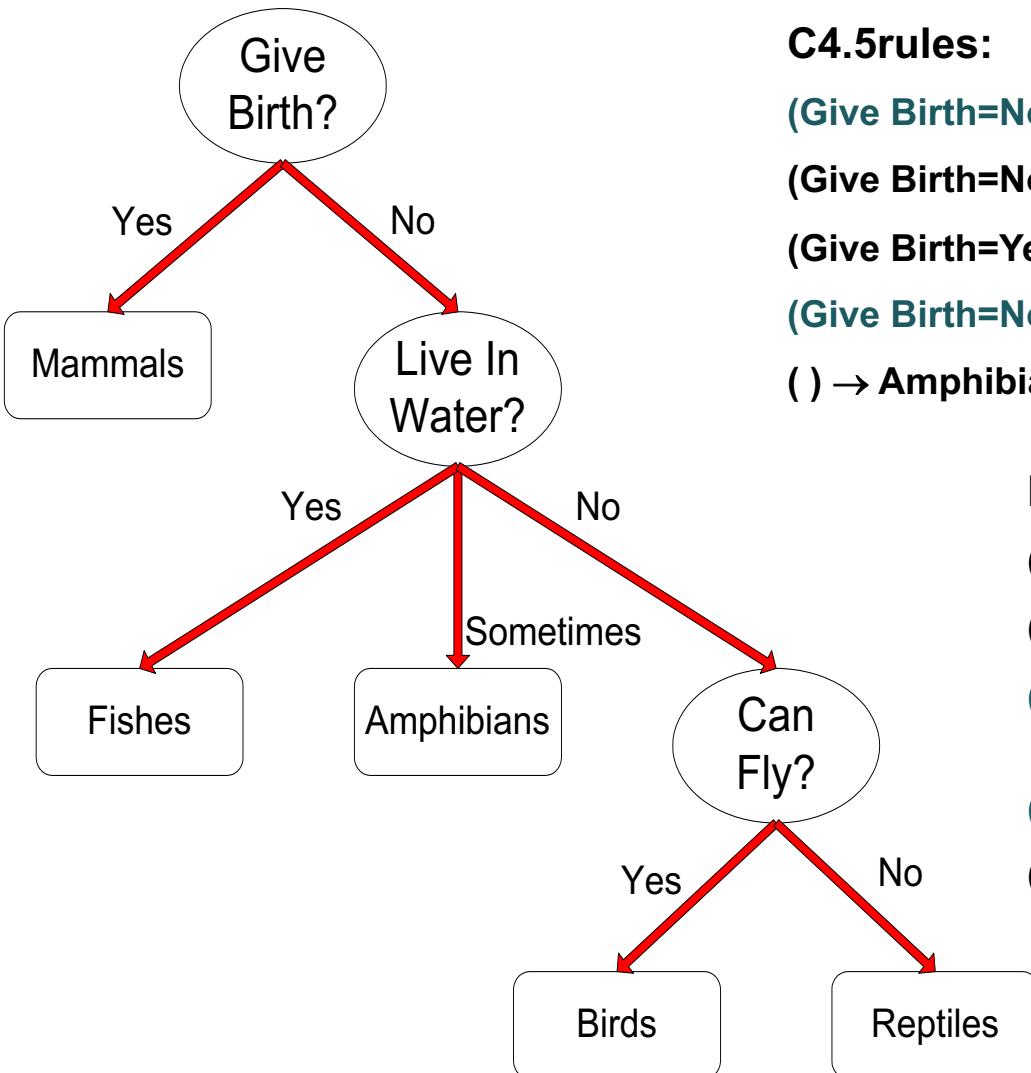
Metodi inderetti: C4.5rules

- Invece di ordinare le regole, ordina sottoinsiemi di regole (**ordinamento di classe**)
 - Ogni sottoinsieme di regole ha lo stesso conseguente (classe)
 - Calcola la *description length* di ciascun subset
 - ◆ Description length = $L(\text{error}) + g L(\text{model})$
 - ◆ g è un parametro che prende in considerazione la presenza di attributi ridondanti in un insieme di regole (valore di default = 0.5)

Esempio

Name	Give Birth	Lay Eggs	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	no	yes	mammals
python	no	yes	no	no	no	reptiles
salmon	no	yes	no	yes	no	fishes
whale	yes	no	no	yes	no	mammals
frog	no	yes	no	sometimes	yes	amphibians
komodo	no	yes	no	no	yes	reptiles
bat	yes	no	yes	no	yes	mammals
pigeon	no	yes	yes	no	yes	birds
cat	yes	no	no	no	yes	mammals
leopard shark	yes	no	no	yes	no	fishes
turtle	no	yes	no	sometimes	yes	reptiles
penguin	no	yes	no	sometimes	yes	birds
porcupine	yes	no	no	no	yes	mammals
eel	no	yes	no	yes	no	fishes
salamander	no	yes	no	sometimes	yes	amphibians
gila monster	no	yes	no	no	yes	reptiles
platypus	no	yes	no	no	yes	mammals
owl	no	yes	yes	no	yes	birds
dolphin	yes	no	no	yes	no	mammals
eagle	no	yes	yes	no	yes	birds

C4.5 rules versus RIPPER



C4.5rules:

$(\text{Give Birth}=\text{No}, \text{Can Fly}=\text{Yes}) \rightarrow \text{Birds}$

$(\text{Give Birth}=\text{No}, \text{Live in Water}=\text{Yes}) \rightarrow \text{Fishes}$

$(\text{Give Birth}=\text{Yes}) \rightarrow \text{Mammals}$

$(\text{Give Birth}=\text{No}, \text{Can Fly}=\text{No}, \text{Live in Water}=\text{No}) \rightarrow \text{Reptiles}$

$() \rightarrow \text{Amphibians}$

RIPPER:

$(\text{Live in Water}=\text{Yes}) \rightarrow \text{Fishes}$

$(\text{Have Legs}=\text{No}) \rightarrow \text{Reptiles}$

$(\text{Give Birth}=\text{No}, \text{Can Fly}=\text{No}, \text{Live In Water}=\text{No}) \rightarrow \text{Reptiles}$

$(\text{Can Fly}=\text{Yes}, \text{Give Birth}=\text{No}) \rightarrow \text{Birds}$

$() \rightarrow \text{Mammals}$

C4.5 rules vs RIPPER

C4.5 rules:

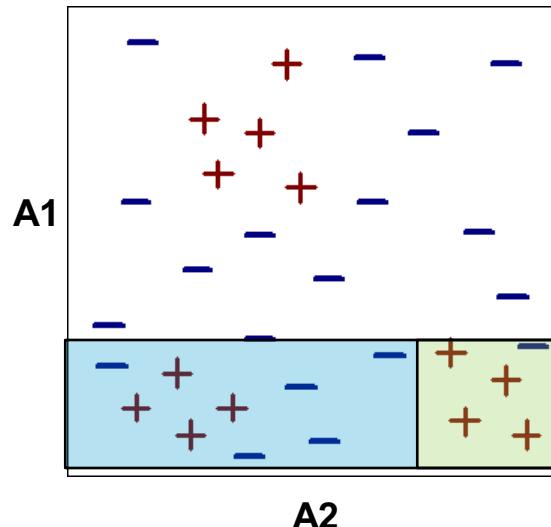
		PREDICTED CLASS				
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL	Amphibians	2	0	0	0	0
CLASS	Fishes	0	2	0	0	1
	Reptiles	1	0	3	0	0
	Birds	1	0	0	3	0
	Mammals	0	0	1	0	6

RIPPER:

		PREDICTED CLASS				
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL	Amphibians	0	0	0	0	2
CLASS	Fishes	0	3	0	0	0
	Reptiles	0	0	3	0	1
	Birds	0	0	1	2	1
	Mammals	0	2	1	0	4

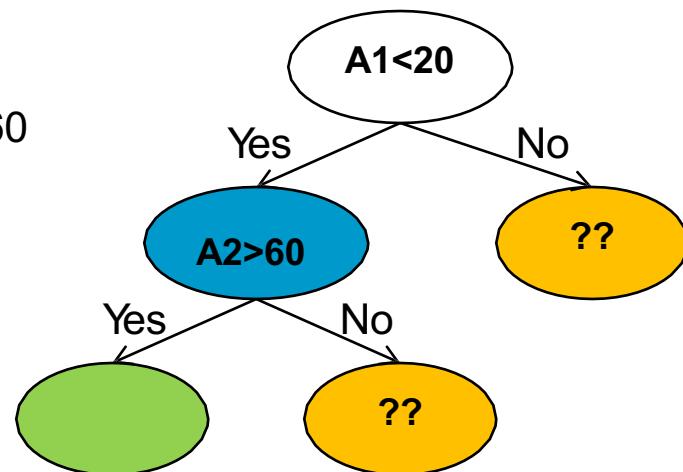
Alberi decisionali vs regole

- Sebbene l'espressività delle due tecniche sia simile l'insieme delle regole prodotte è generalmente diverso
- La differenza fondamentale tra le due tecniche consiste nel fatto che nel procedimento di costruzione:
 - ✓ la bontà del criterio di partizionamento scelto in ogni nodo dell'albero considera la qualità di tutti i figli generati
 - ✓ il criterio con cui viene aggiunto un atomo alla regola valuta solo la bontà della classe che si viene a determinare



R1: $A1 < 20$

R1: $A1 < 20 \text{ and } A2 > 60$



Vantaggi dei classificatori Rule-Based

- Hanno caratteristiche abbastanza simili agli alberi decisionali
 - Altamente espressivi come alberi decisionali
 - Facile da interpretare
 - Prestazioni paragonabili agli alberi decisionali
 - Possono gestire attributi ridondanti
- Più adatti a gestire classi squilibrate
- Gestiscono con difficoltà test set con dati incompleti

Nearest Neighbor

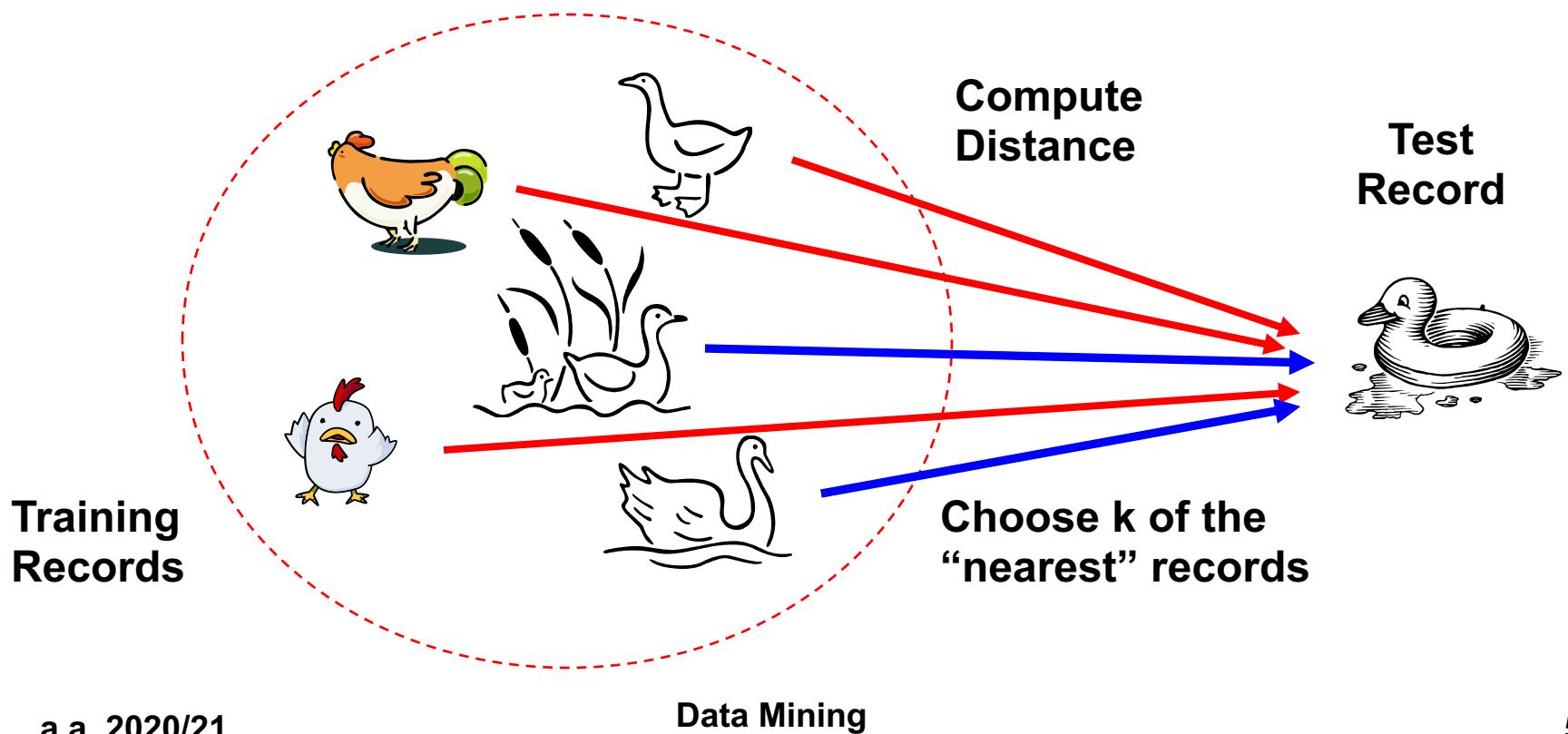
Tell me who your neighbors are, and I'll know who you are

Instance Based Classifiers

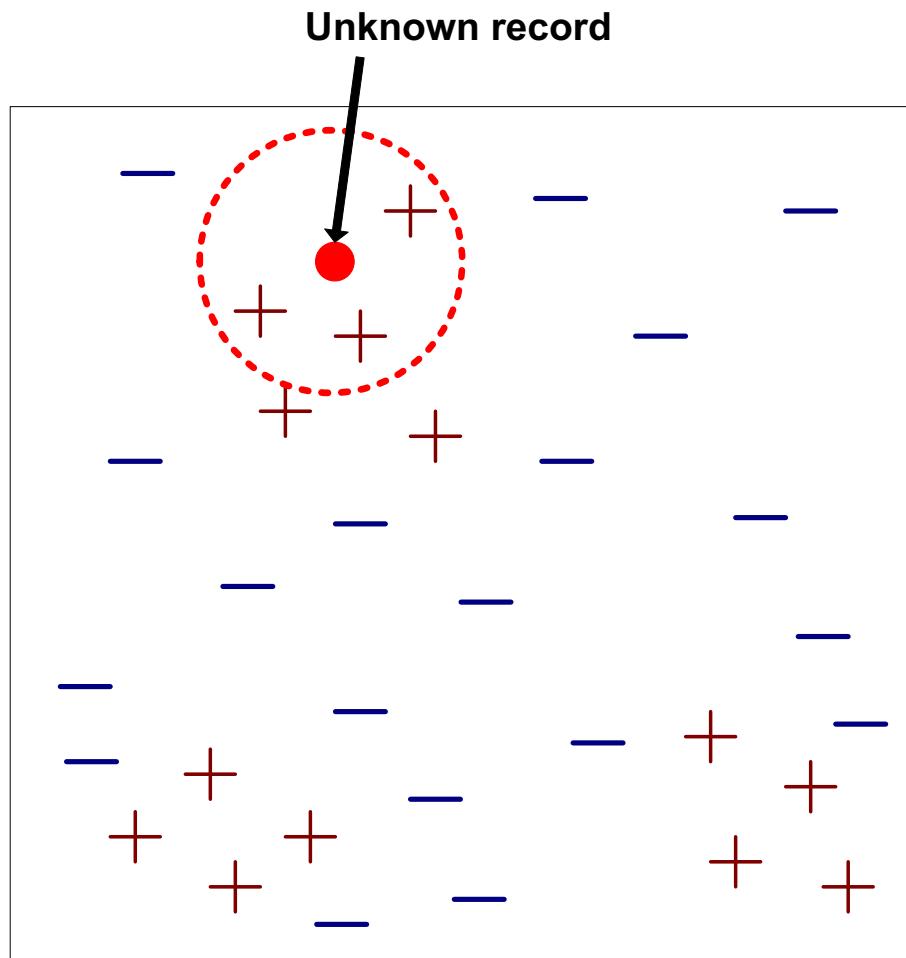
- sono detti *lazy learners* poiché non costruiscono modelli esplicitamente
- Esempi:
 - **Rote-learner**
 - ◆ Memorizza tutti i dati di training ed esegue la classificazione solo se gli attributi del record corrispondono esattamente a uno degli esempi di addestramento
 - **Nearest neighbor**
 - ◆ Usa i k punti “più vicini” (nearest neighbors) per eseguire la classificazione

Classificatori Nearest Neighbor

- Utilizzano i k punti “più vicini” (nearest neighbors) per effettuare la classificazione
- **Idea di base:** Se cammina come un’oca, starnazza come un’oca, allora probabilmente è un’oca

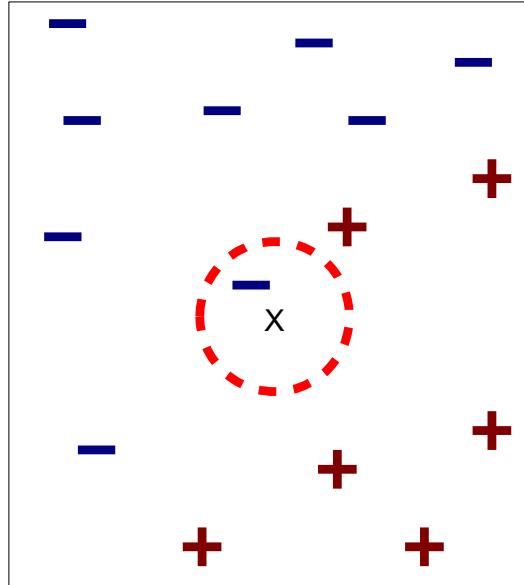


Classificatori Nearest-Neighbor

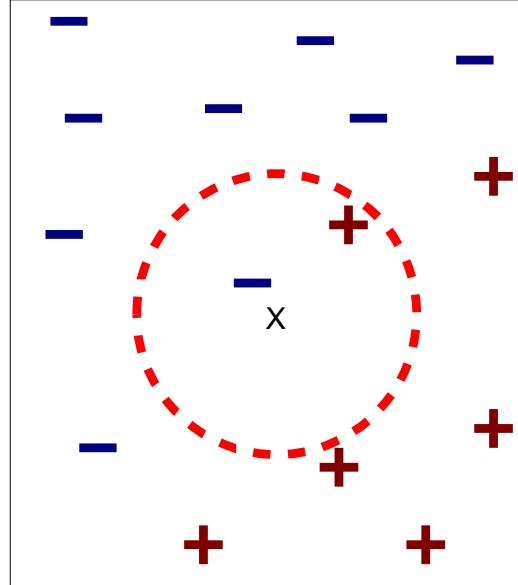


- Richiedono
 - Un training set
 - Una metrica per calcolare la distanza tra i record
 - Il valore di k , ossia il numero di vicini da utilizzare
- Per classificare un nuovo oggetto:
 - Calcola la distanza rispetto ai record nel training set
 - Identifica k nearest neighbors
 - Utilizza le etichette delle classi dei nearest neighbor per determinare la classe del record sconosciuto (es. scegliendo quella che compare con più frequenza)

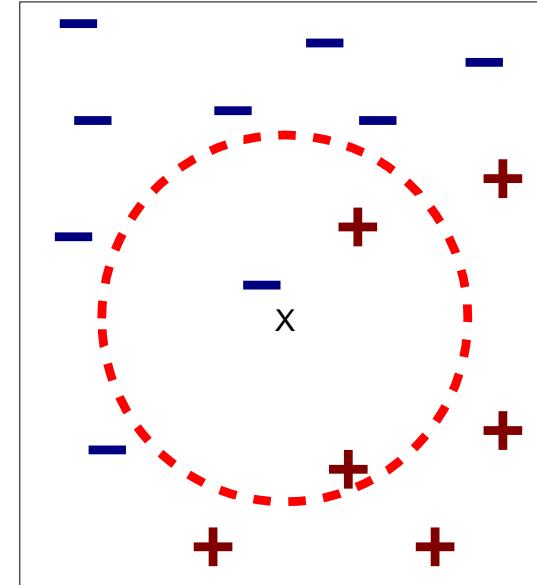
Definizione di Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

I k nearest neighbors di un record x sono i record del training set che hanno le più piccole k distanze da x

Classificazione Nearest Neighbor

- Calcola la distanza tra due punti:

- Distanza euclidea

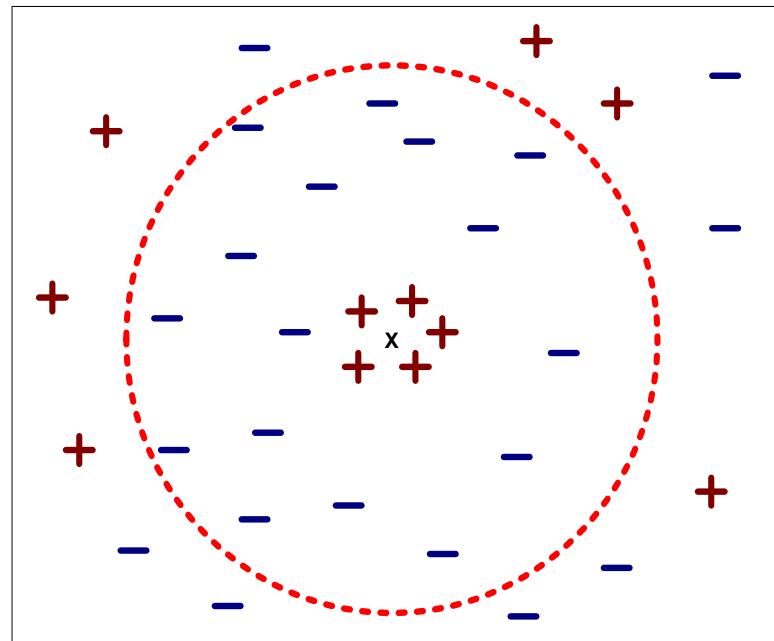
$$d = \sqrt{x^2 + y^2}$$

- Determina la classe dalla lista dei vicini

- Assegna la classe della maggioranza tra i k oggetti che appartengono all'intorno (voto di maggioranza)
 - Pesa il voto secondo la distanza
 - ◆ **weight factor, $w = 1/d^2$**

Classificazione Nearest Neighbor

- La scelta di k è importante perché:
 - ✓ Se k è troppo piccolo, l'approccio è sensibile al rumore
 - ✓ Se k è troppo grande, l'intorno può includere molti esempi appartenenti ad altre classi



Classificazione Nearest Neighbor

● Problemi di scala

- ✓ Per operare correttamente gli attributi devono avere la stessa scala di valori e vanno normalizzati in fase di pre-processing

● Esempio:

- ✓ su quale attributo una differenza di 0.5 vale di più?
 - l'altezza di un adulto varia 1.5 m to 2.1 m
 - il peso di un adulto varia da 40 kg a 150 kg
 - Lo stipendio di una persona varia da 10K € to 1M €
- Normalizzare la scala può non essere sufficiente in presenza di diverse distribuzioni dei dati
 - ✓ Mahalanobis distance

Classificazione Nearest Neighbor

- La selezione della giusta misura di somiglianza è fondamentale:

1 1 1 1 1 1 1 1 1 1 0

vs

0 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 0 0 0

Distanza Euclidea = 1.4142 per entrambe le coppie

K-Nearest Neighbor: Pro & Contro

■ Pro

- ✓ Non richiedono la costruzione di un modello
- ✓ Rispetto ai sistemi basati su regole o decision tree permettono di costruire “contorni” delle classi non lineari e sono quindi più flessibili

■ Contro

- ✓ Richiedono una misura di distanza per valutare la vicinanza
- ✓ Richiedono una fase di pre-processing per normalizzare il range di variazione degli attributi
- ✓ La classe è determinata localmente e quindi è suscettibile al rumore dei dati
- ✓ Sono molto sensibili alla presenza di attributi irrilevanti o correlati che falseranno le distanze tra gli oggetti
- ✓ Il costo di classificazione può essere elevato e dipende linearmente dalla dimensione del training set in mancanza di opportune strutture ad indice

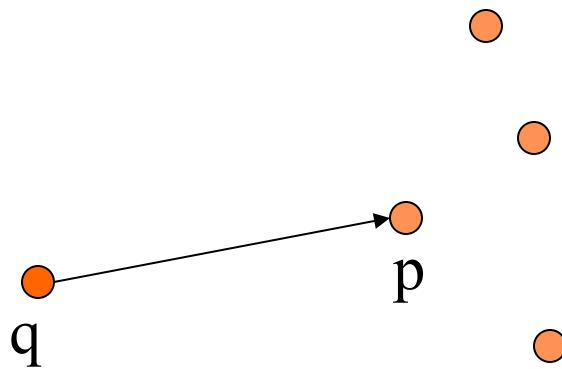
Ottimizzare il calcolo di KNN

- Evitare di calcolare la distanza da tutti gli oggetti del training set
 - Multi-dimensional access methods (k-d trees)
 - Fast approximate similarity search
 - Locality Sensitive Hashing (LSH)
- Condensazione
 - Determina un set di oggetti più piccolo che offre le stesse prestazioni
- Modifica
 - Rimuovi oggetti per migliorare l'efficienza

Index structures for Nearest Neighbor Problems

Ricerca dei vicini (Nearest Neighbor Search)

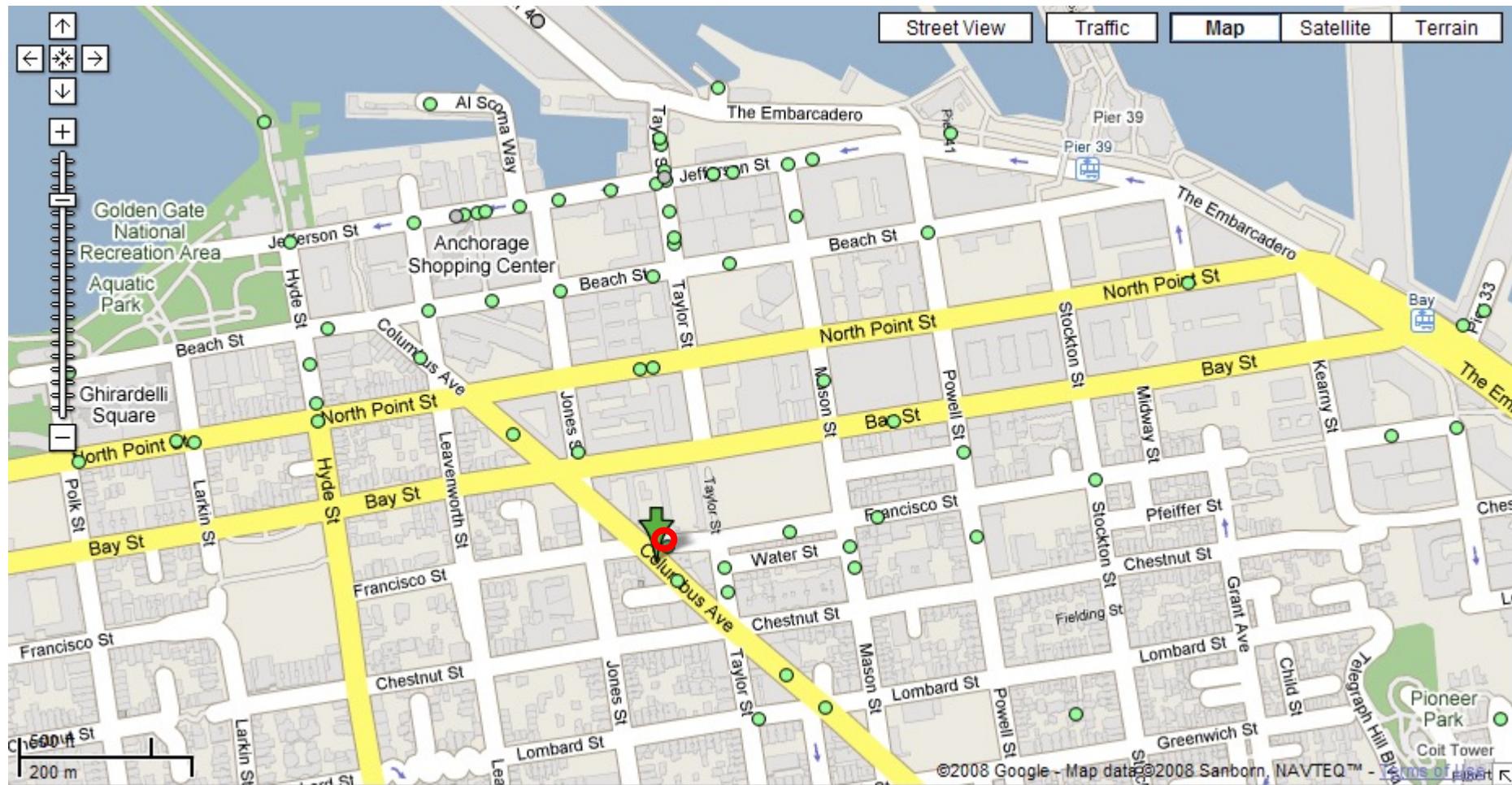
- Dato: un insieme P di n punti in R^d
- Goal: una struttura dati, che data una query (punto) q , trovi il punto più vicino (*nearest neighbor*) p di q in P



Nearest Neighbor Search

Problema: trova il punto in P con distanza minima da q

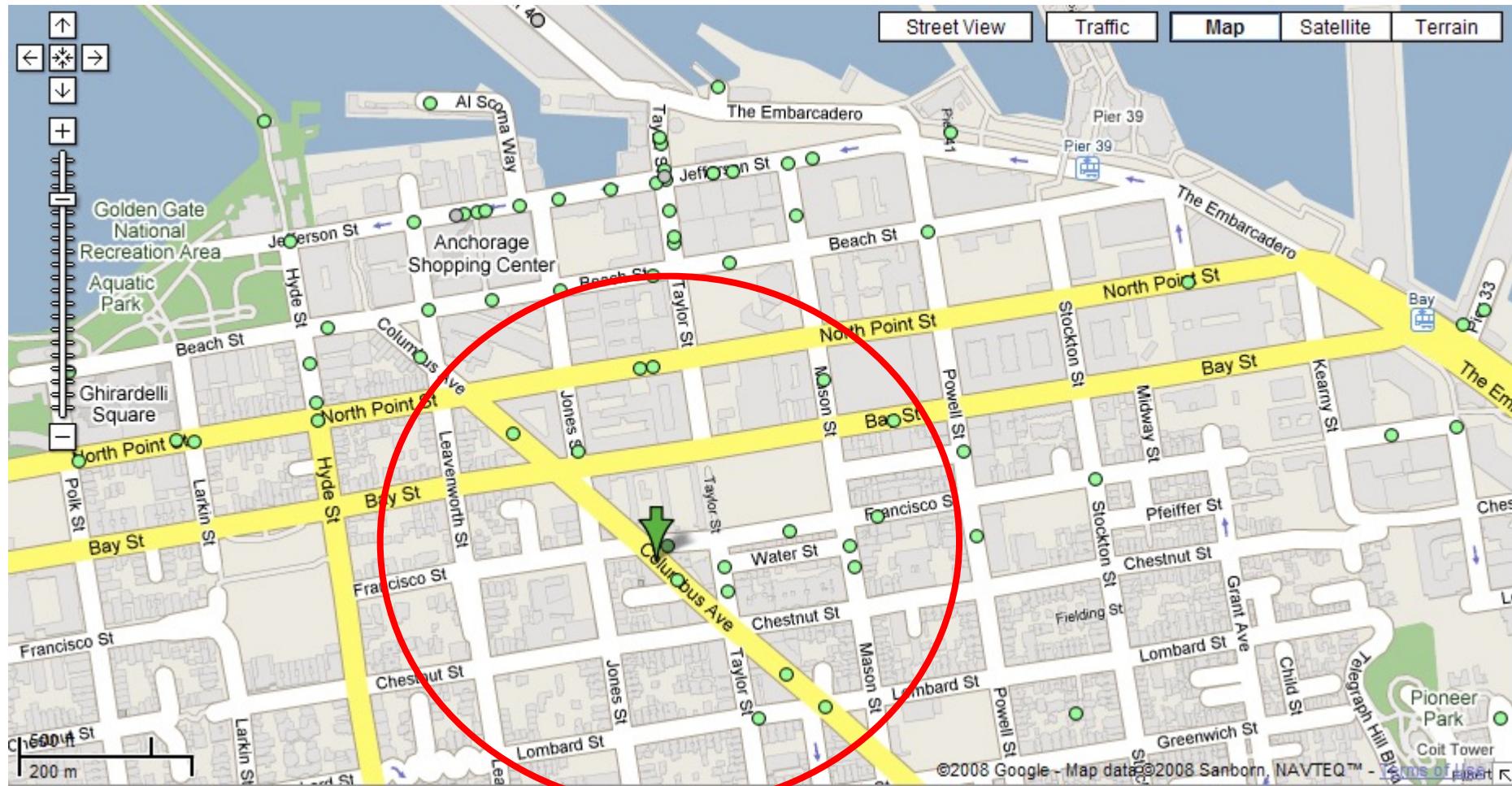
Esempio: qual'è il ristorante più vicino al mio albergo?



Near neighbor (range search)

Problema: trova uno/tutti i punti in P con distanza $\leq r$ da q

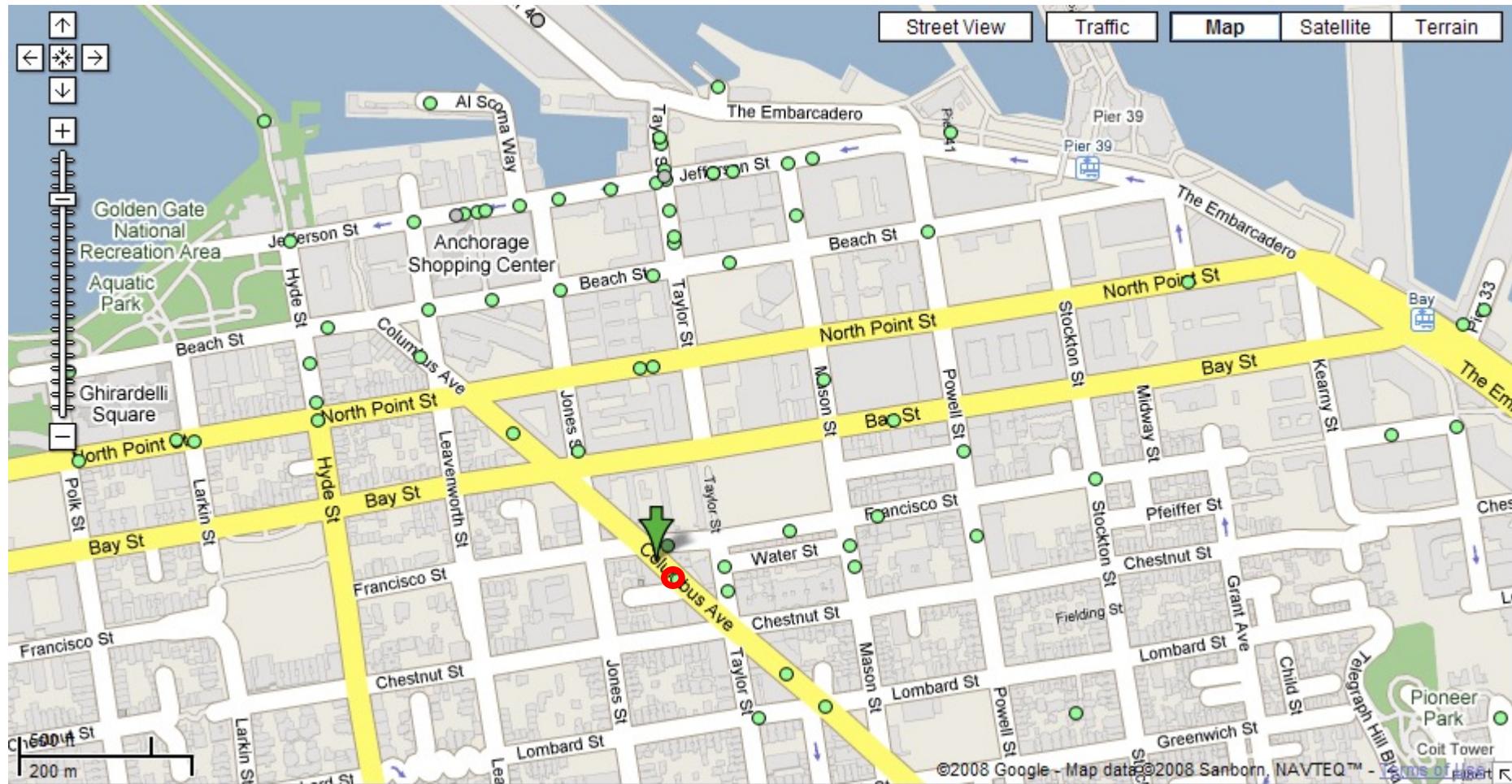
Esempio: trova tutti i ristoranti nel raggio di 400m dal mio albergo



Approximate Near neighbor

Problema: trova uno/tutti i punti p in P , con distanza massima da q $(1+\epsilon)$ volte la distanza da q al punto più vicino

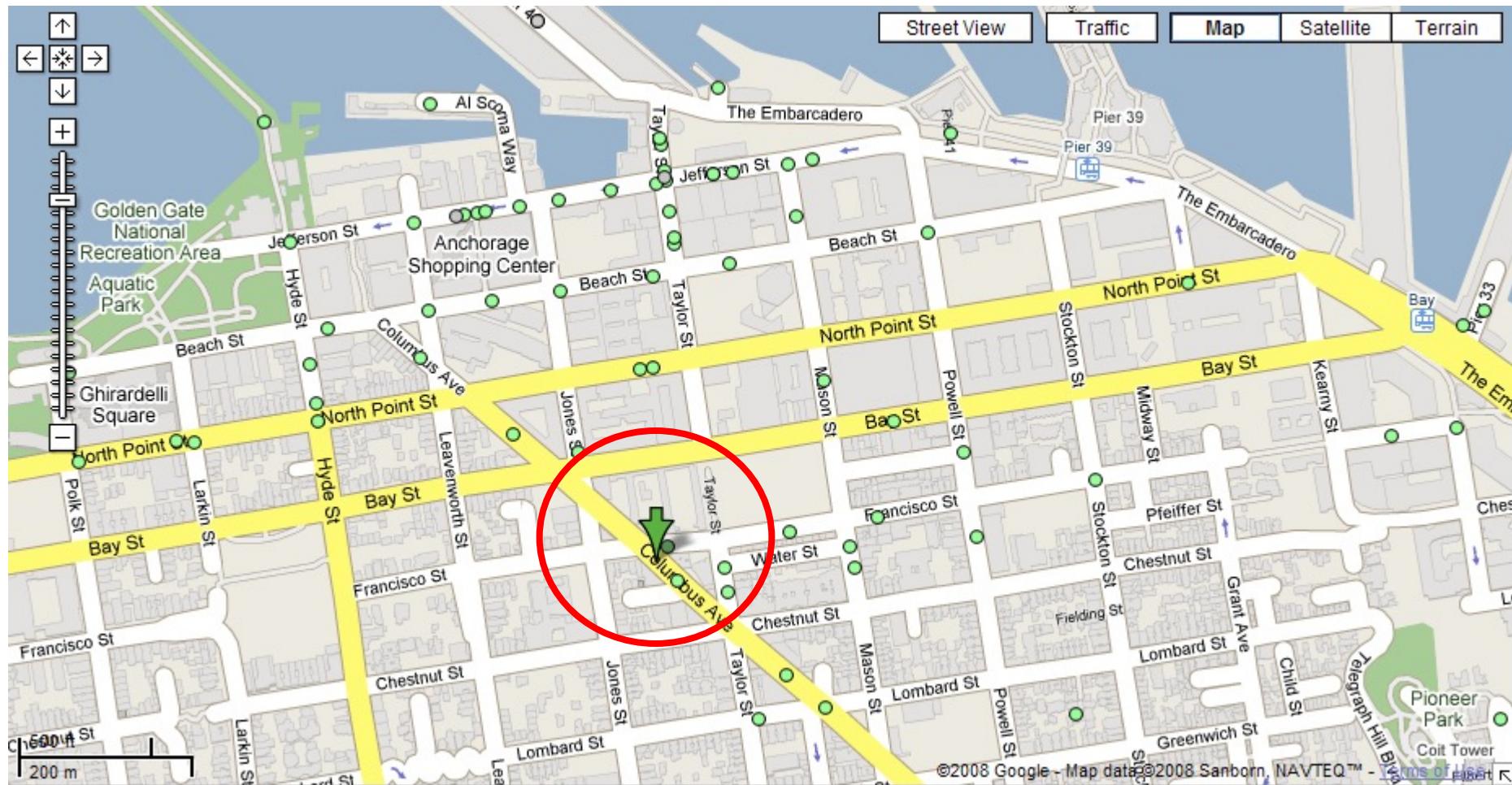
Esempio: trova i ristoranti più vicini al mio albergo



K-Nearest-Neighbor

Problema: trova i K punti più vicini a q

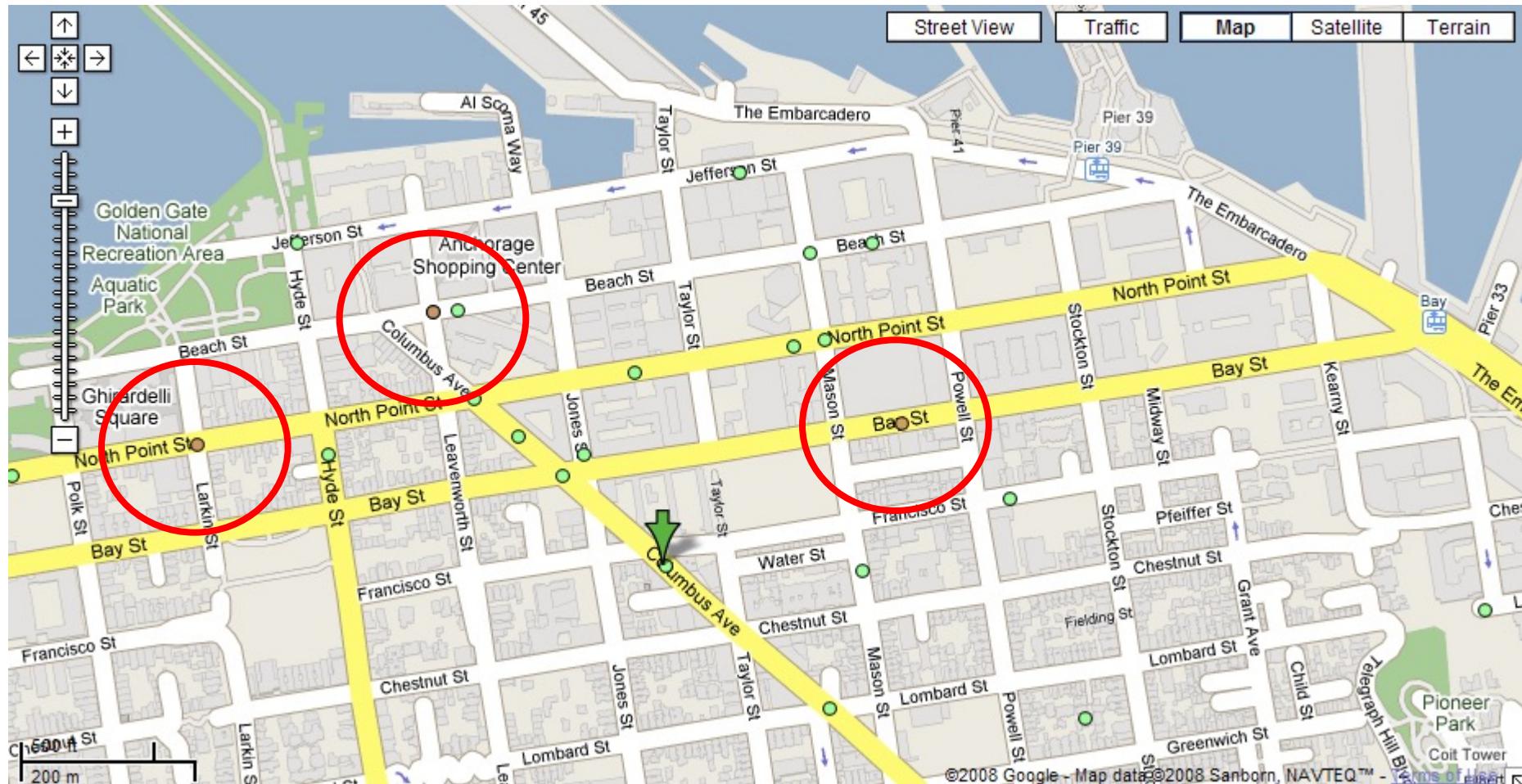
Esempio: trova i 4 ristoranti più vicini al mio albergo.



Spatial join

Problema: dati due insiemi P , Q , trova tutte le coppie p in P , q in Q , tali che p ha una distanza $\leq r$ da q

Esempio: Trova coppie albergo/ristorante e che distano al massimo 200m



Motivatione

Il problema nearest neighbor search si incontra in numerosi campi applicativi, tra i quali:

Pattern recognition
Computer vision
Coding theory
Internet marketing
Spell checking
Copyright violation detection

Statistical classification
Databases
Data compression
DNA sequencing
Plagiarism detection
.....

Algoritmi

- Main memory (Computational Geometry)

- linear scan
- tree-based:
 - ◆ quadtree
 - ◆ kd-tree
- hashing-based: Locality-Sensitive Hashing

- Secondary storage (Databases)

- R-tree (and numerous variants)
- Vector Approximation File (VA-file)

KNN e Grafi di Prossimità

- Voronoi diagram
- Grafi di prossimità
 - un grafico in cui due vertici sono collegati da un arco se e solo se i vertici soddisfano particolari requisiti geometrici
 - nearest neighbor graphs,
 - minimum spanning trees
 - Delaunay triangulations
 - relative neighborhood graphs
 - Gabriel graphs

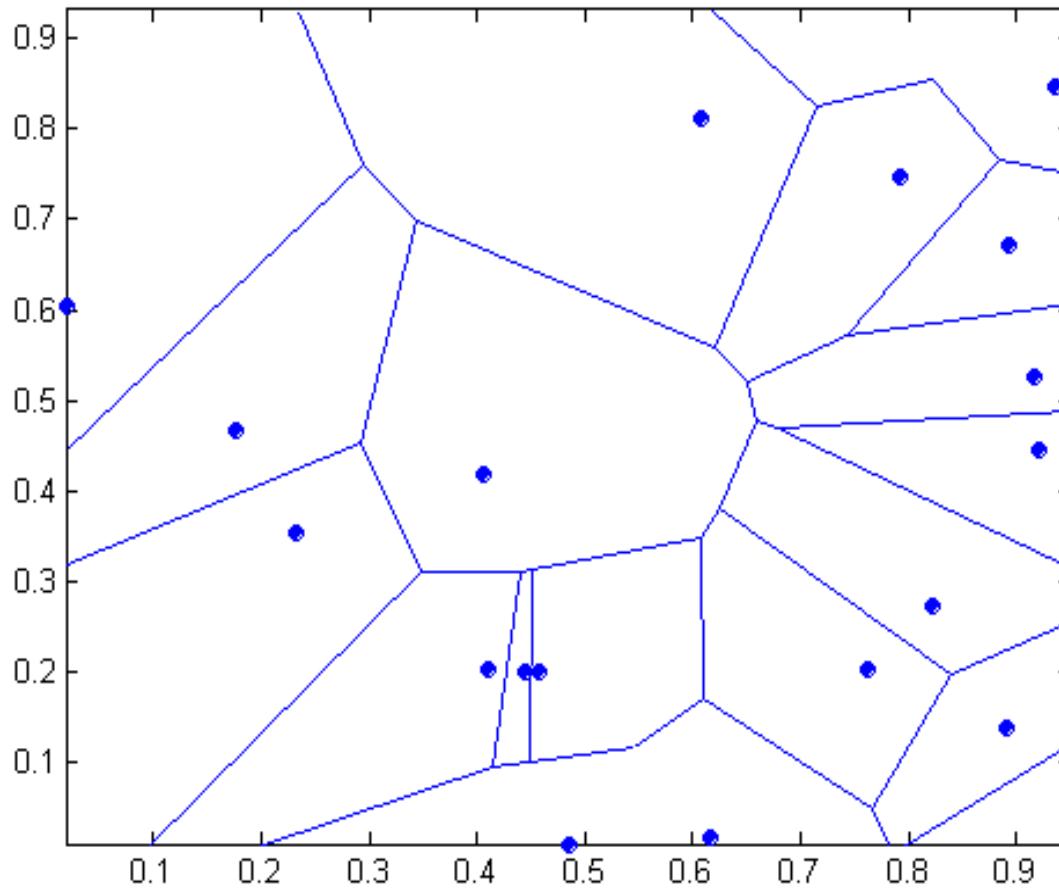
Linear scan (approccio Naïve)

La soluzione più semplice al problema NNS

- Calcola la distanza dal punto interrogazione a ogni altro punto nel database.
- Funziona bene per piccoli database ma diventa rapidamente intrattabile quando la dimensione o la dimensionalità del problema diventano grandi.
- Running time $O(N d)$.

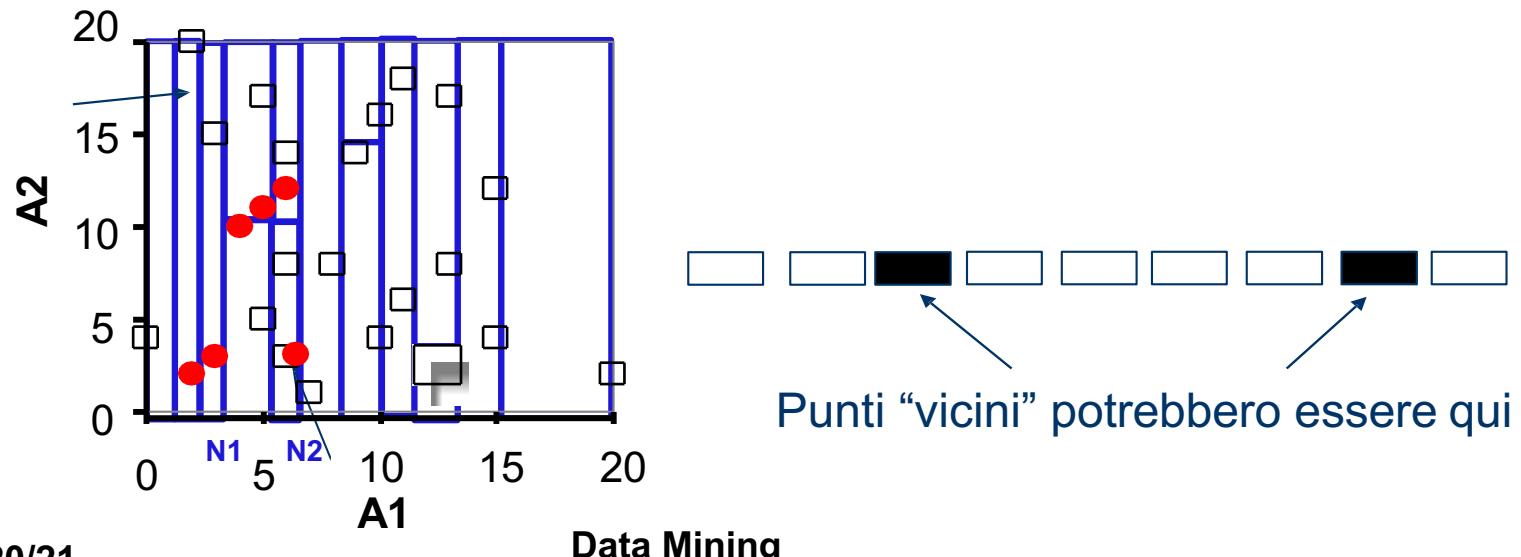
1 nearest-neighbor

Voronoi Diagram and Delaunay triangulation



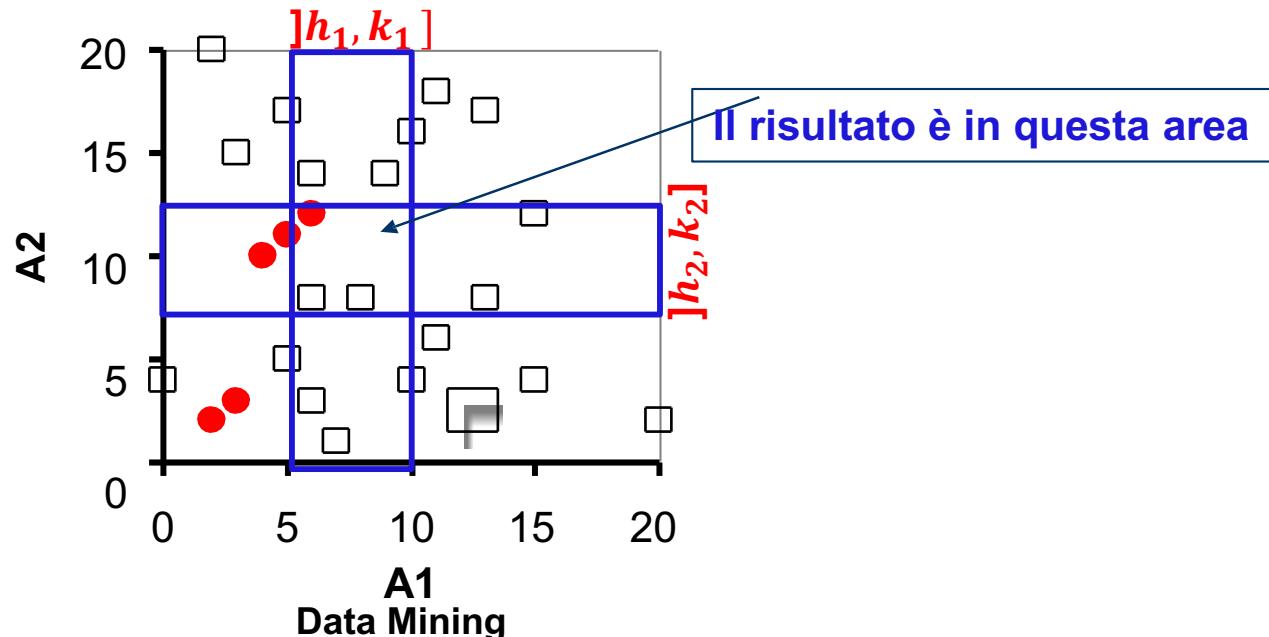
Utilizzo di indici per query spaziali

- Dati n attributi A_1, \dots, A_n e in assenza di strutture dati specifiche si potrebbe pensare di costruire un **B⁺-tree multi-attributo**, che organizzi (ordini) le tuple in base ai valori di A_1, \dots, A_n
- Considerando i nodi foglia del B⁺-tree: $N1 \rightarrow N2 \rightarrow N3 \rightarrow \dots$
 - ✓ La prima foglia, $N1$, conterrà le tuple con i più piccoli valori di A_1 in base alla capacità della foglia stessa
 - ✓ La seconda foglia inizierà con i valori seguenti, e così via
- Qualsiasi ordinamento si scelga, la ricerca di un k-NN di un punto richiederà l'accesso a molti nodi poichè non è una distanza spaziale



Un'altra soluzione basata sui B+-tree

- Assumiamo di conoscere, per esempio dalle statistiche del DB, che i k-NN di un punto q stiano nell'(iper) rettangolo con lati $[h_1, k_1] \times [h_2, k_2] \times \dots$
- Allora possiamo utilizzare n query di range indipendenti $A_i \text{ BETWEEN } h_i \text{ AND } k_i$ su n indici distinti A_1, \dots, A_n e intersecare il risultato
 - ✓ Oltre a richiedere di conoscere i valori dei range questa tecnica richiederebbe comunque molto lavoro poichè leggerà una quantità di tuple proporzionale all'unione degli n risultati a meno della loro intersezione

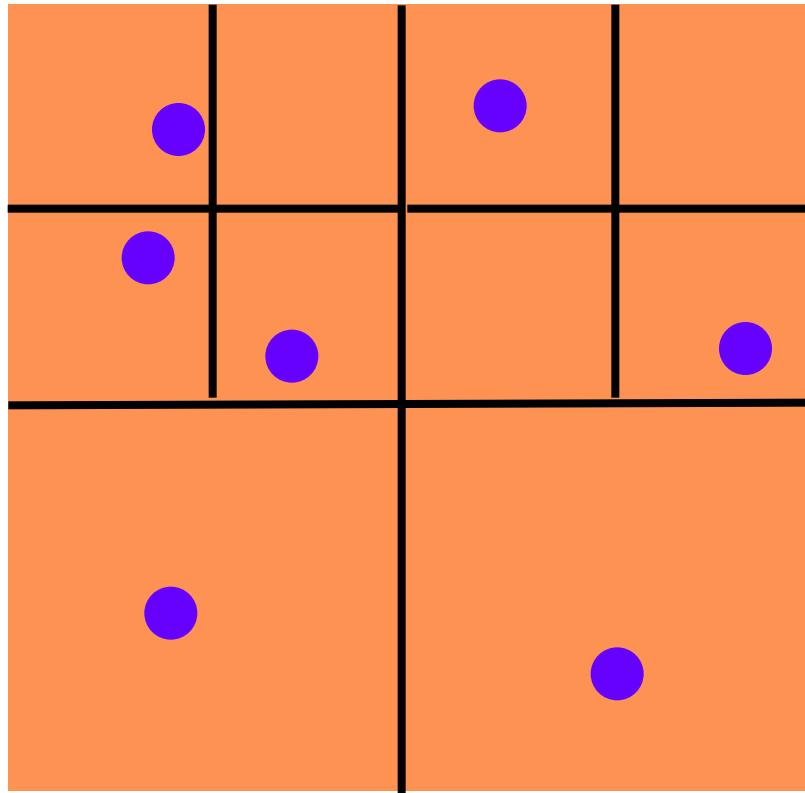


Indici multi-dimensionali (spaziali)

- Il B⁺-tree multi-attributo mappa punti di $A \subseteq \mathbb{R}^D$ in punti di \mathbb{R}
- Questa “linearizzazione” forzatamente favorisce uno degli attributi in base al modo in cui questi sono organizzati nel B⁺-tree
 - ✓ Un B⁺-tree su (X,Y) favorisce query su X, e non può essere utilizzato per query che non specifichino il valore di X
- Per questo motivo, dobbiamo utilizzare un modo che al contrario preservi la “prossimità spaziale”
- Il tema dello “spatial indexing” è stato studiato a partire dagli anni '70 a causa della sua importanza (cartografia, GIS, VLSI, CAD)
- Più recentemente (anni '90), è tornato di “moda” grazie a nuove tipologie di applicazioni come quelle multimediali

Quad-tree

Una struttura dati semplice



Dividi lo spazio in 2d sotto-quadri uguali

Ripeti fino a quando:

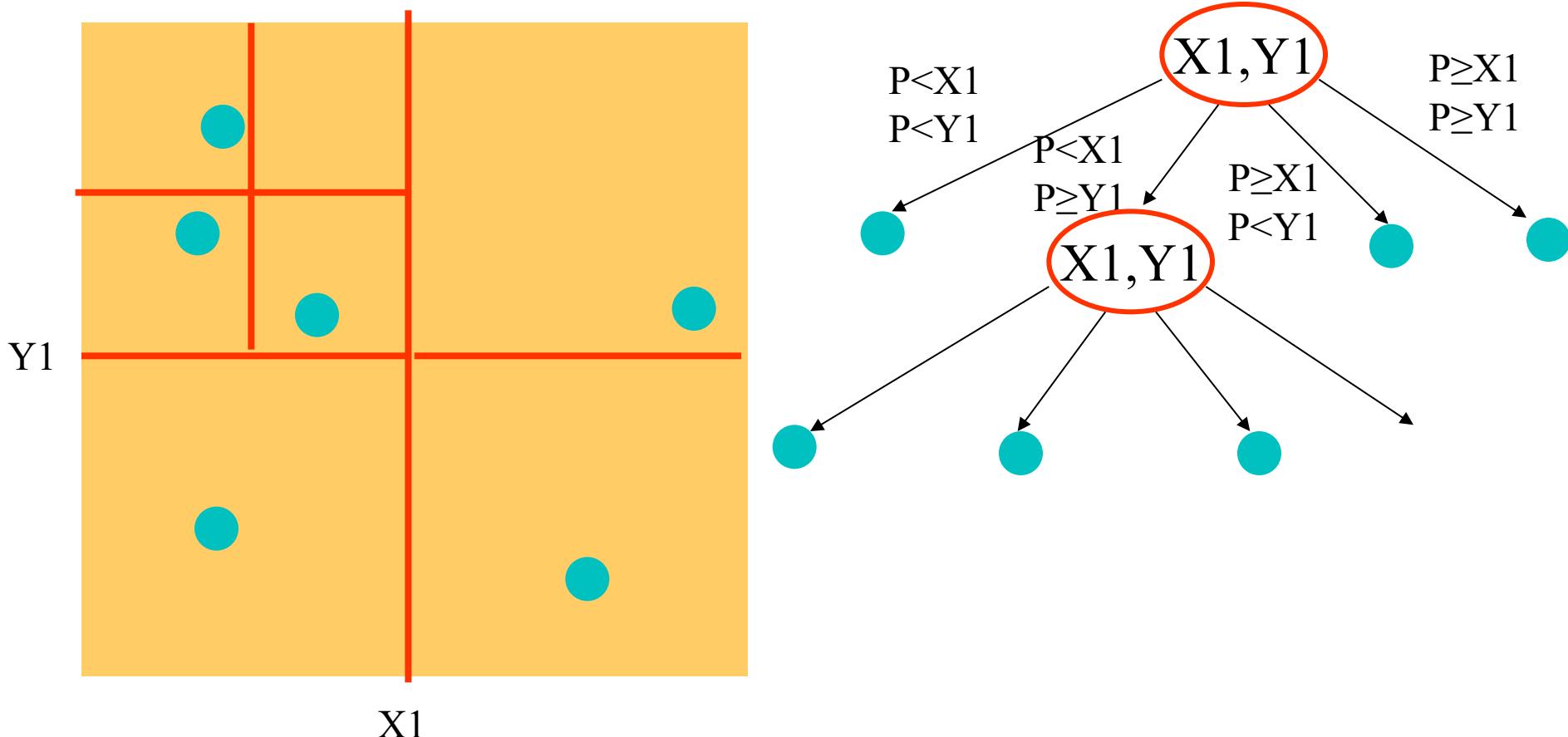
- resta solo un pixel
- resta solo un punto
- restano solo pochi punti

Range query

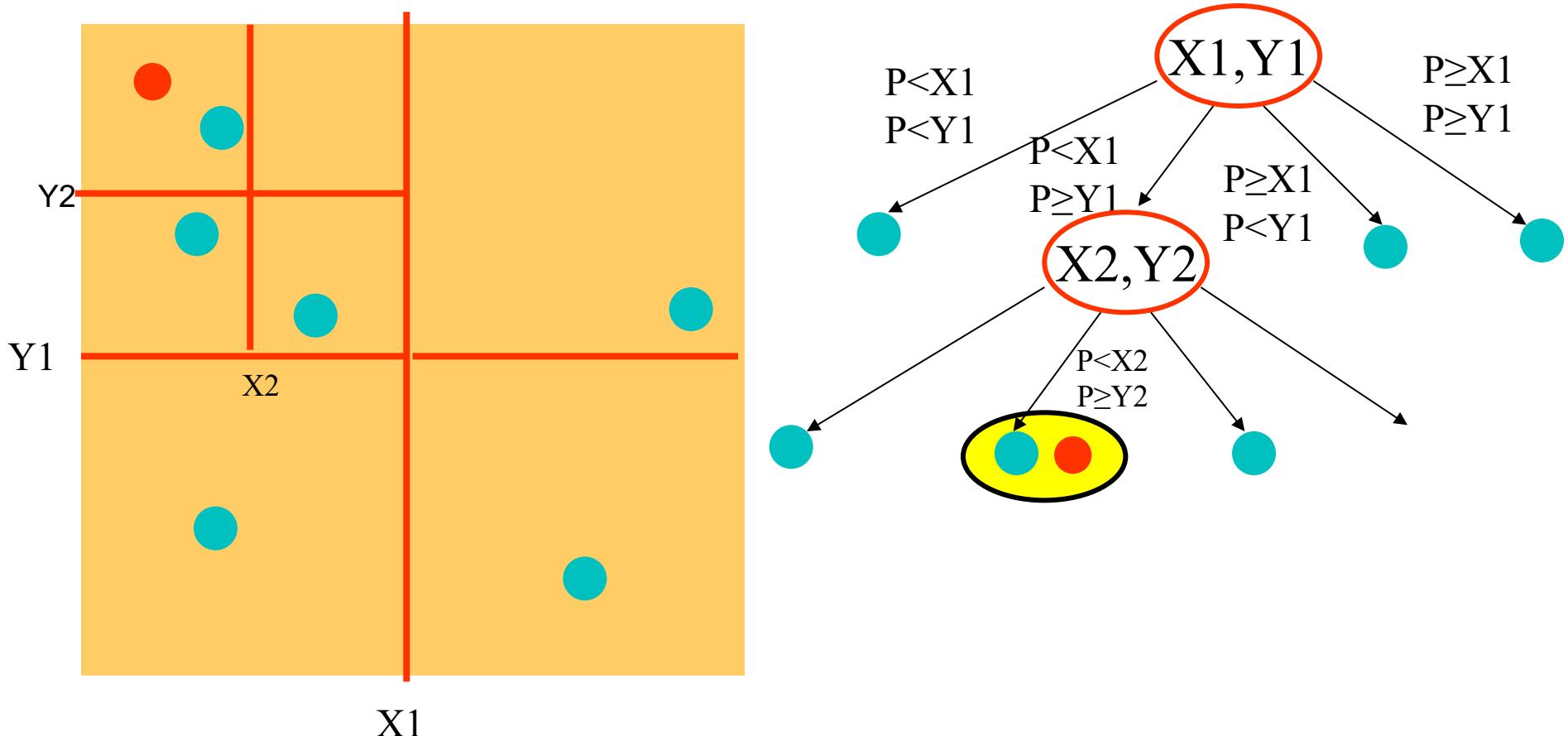
- Near neighbor (range search):

- Inserisci il nodo radice nello stack
- ripeti
 - ◆ Estrai il prossimo nodo T dallo stack
 - ◆ Per ogni figlio C di T :
 - se C è un nodo foglia esamina i punti di C
 - se C si interseca con la sfera del raggio r e centro q , aggiungi C allo stack

Struttura del Quad-tree



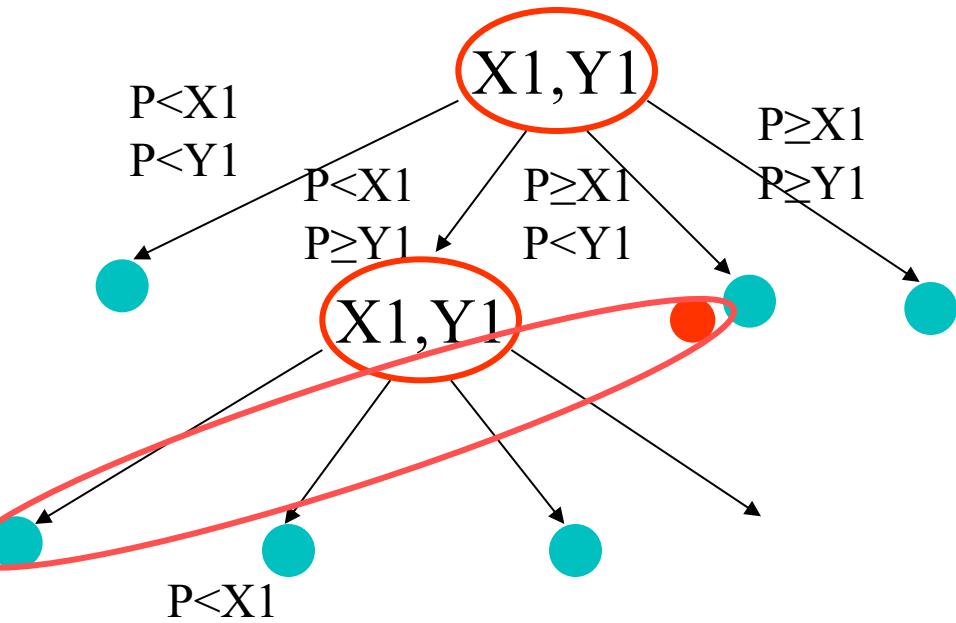
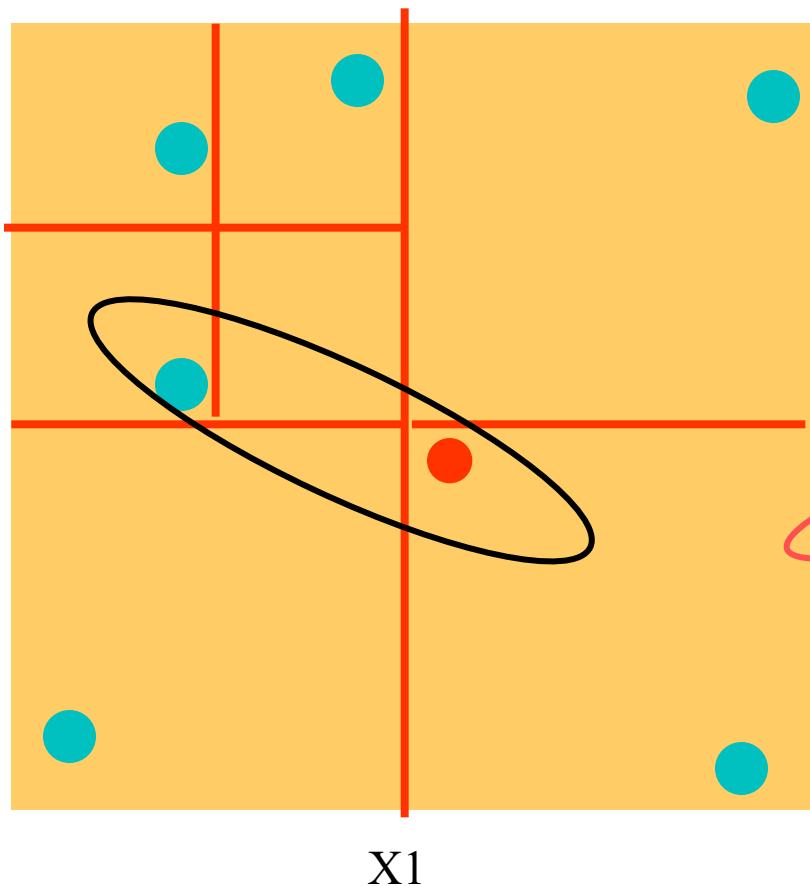
Quad-tree - Query



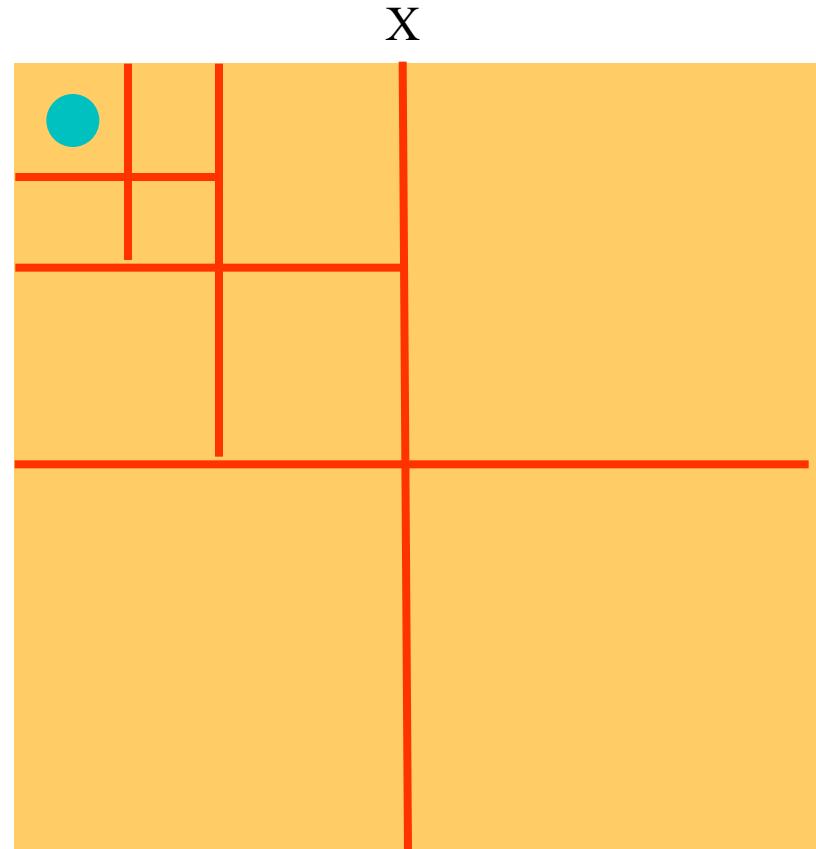
Quad-tree

- Struttura dati molto semplice
- Qual è il rovescio della medaglia?

Quad-tree – Problema 1



Quad-tree – Problema 2



Running Time:
 $O(n \times 2^d)$

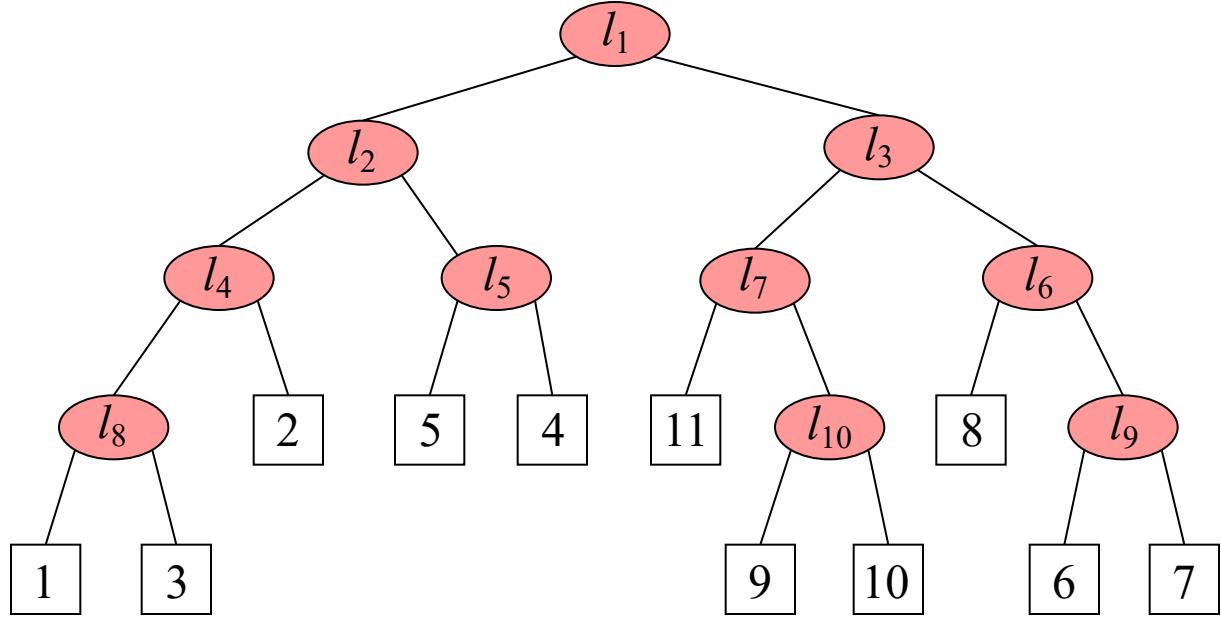
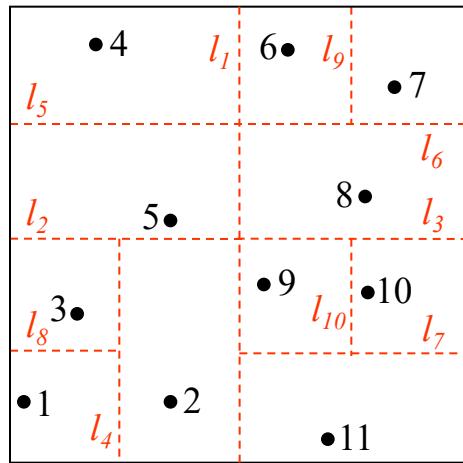
Space and Time Exponential in dimensions

Kd-trees [Bentley'75]

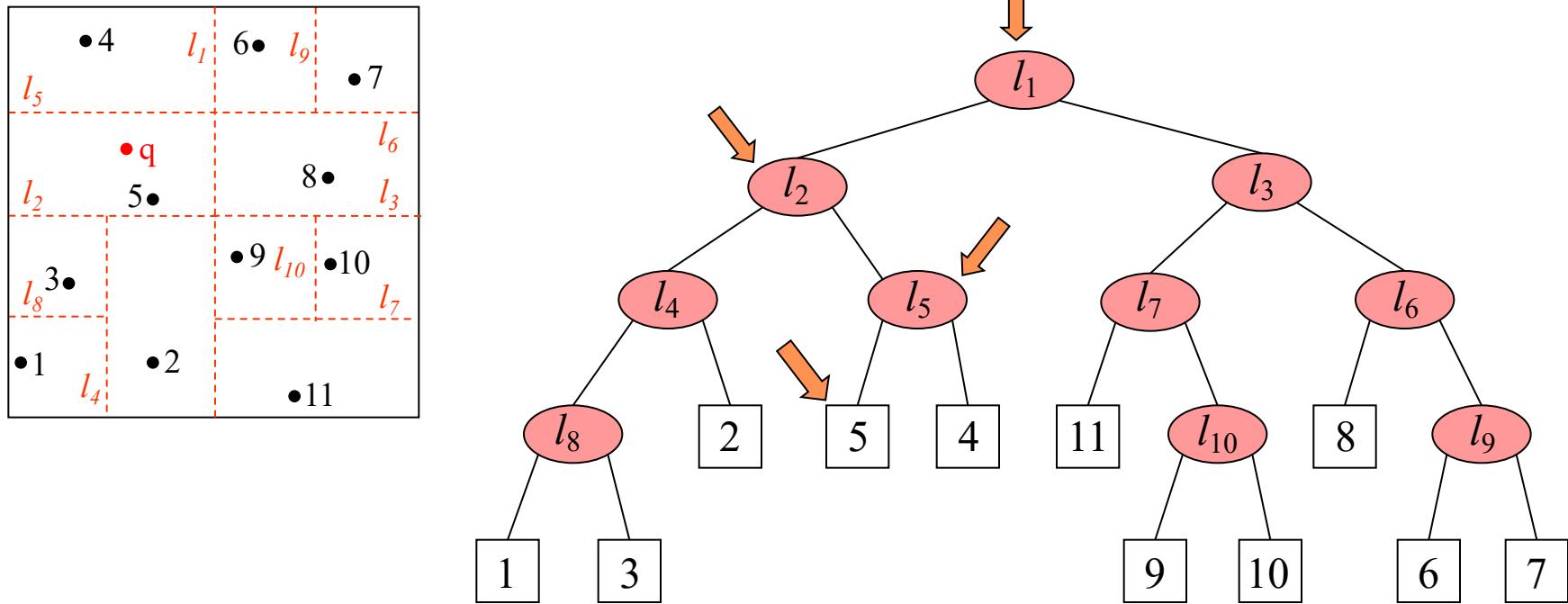
- Main ideas:

- Partizionamenti di una sola dimensione alla volta (e non di 2d dimensioni);
- invece di dividere in mezzo, scegli la suddivisione "attentamente" (molte varianti)
- near(est) neighbor queries: come per quad-trees

Costruzione del Kd-Tree



Kd-Trees Query



Kd-trees

● Pro:

- Nessun spazio vuoto (o meno spazi vuoti);
- Spazio lineare (ciascun nodo foglia contiene almeno un punto).

● Contro:

- Il tempo di esecuzione della query potrebbe continuare ad essere esponenziale
- Tuttavia se non facciamo qualcosa di veramente stupido, il tempo di interrogazione è al massimo d n
- Il problema è hard quando la dimensione è elevata (e.g. maggiore di 20).

Calcolo approssimato

- E' possibile interrompendo la ricerca in anticipo [Arya et al'94]
- Essenzialmente, dopo ogni passo di ricerca, verifica se sei sufficientemente vicino. Se è così fermati.
- Non funziona per query esatte
- E' possibile utilizzare tecniche di hashing (LSH / Locality-Sensitive Hashing)

Memoria secondaria

- Le tecniche di NN sono usate in molti contesti.
- In alcuni di questi sono presenti grandi quantità di dati che necessitano l'uso della memoria secondaria.
- Il raggruppamento dei dati è fondamentale
- Approcci diversi:
 - nella memoria principale, qualsiasi riduzione del numero di punti di ispezione è buona
 - su disco, questo non succede!



Strutture Disk-based

- R-tree [Guttman'84]

- punto di partenza per molte varianti
- oltre 600 citazioni! (secondo CiteSeer)
- Approccio "ottimistico": prova a rispondere alle domande in tempo logaritmico

- Vector Approximation File [WSB'98]

- approccio "pessimistico": se è necessario analizzare l'intero set di dati, è meglio farlo velocemente

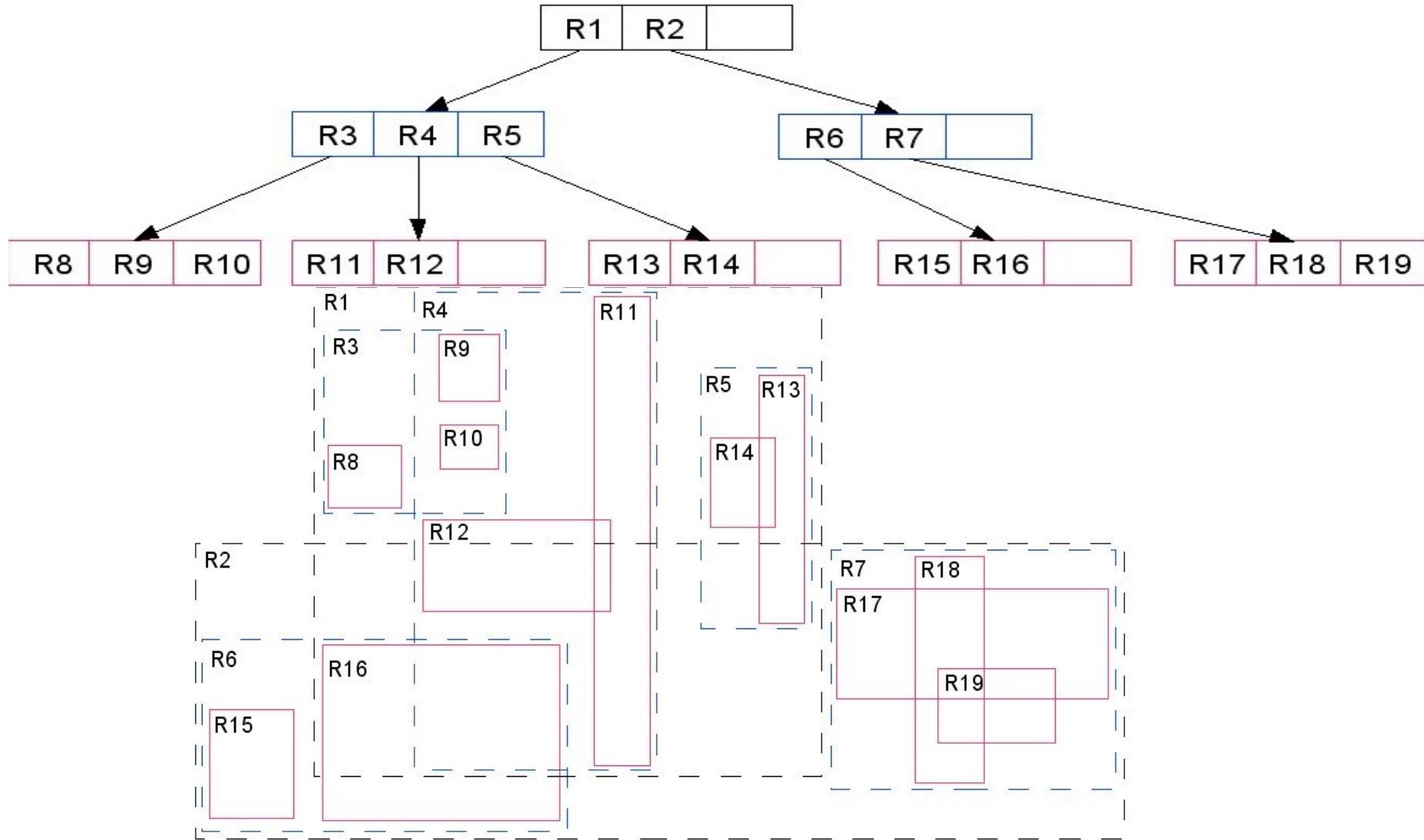
Gli R-tree (Guttman, 1984)

- Gli R-tree sono estensioni dei B⁺-tree a spazi multi-dimensionali
- I B⁺-tree organizzano gli oggetti in
 - ✓ Un insieme di intervalli mono-dimensionali non sovrapposti
 - ✓ Applicano questo principio ricorsivamente dalle foglie alla radice
- Gli R-tree organizzano gli oggetti in
 - ✓ Un insieme di intervalli multi-dimensionali sovrapposti (iperrettangoli)
 - ✓ Applicano questo principio ricorsivamente dalle foglie alla radice
- Gli R-tree sono oggi disponibili in alcuni DBMS commerciali quali Oracle9i

R-tree

- Approccio "bottom-up" (kd-tree è "top-down"):
 - Inizia con un insieme di punti / rettangoli;
 - Partiziona l'insieme in gruppi di cardinalità piccola;
 - Per ogni gruppo, trova il rettangolo minimo che li contiene;
 - Ripeti.

R-tree



R-trees

- **Pro:**

- Supportano near(est) neighbor search (simile a prima)
- Funzionano per punti e rettangoli
- Evitano gli spazi vuoti
- Molte varianti: X-tree, SS-tree, SR-tree ecc
- Funzionano bene per le dimensioni ridotte

- **Contro:**

- Non funzionano molto bene per dimensioni elevate

Classificatori Bayesiani

Classificatori Bayesiani

- Rappresentano un **approccio probabilistico** per risolvere problemi di classificazione
 - ✓ In molte applicazioni la relazione tra i valori degli attributi e quelli della classe non è deterministica
 - Rumore dei dati
 - Presenza di caratteristiche del fenomeno non modellate dagli attributi
 - Difficoltà nel quantificare certi aspetti del fenomeno
 - ✓ Per esempio, predire se una persona è a rischio cardiaco dipende fortemente dalla sua dieta e dalla sua attività fisica ma, persone che hanno un'alimentazione sana e si allenano regolarmente possono avere comunque problemi di cuore
 - Esistono altri fattori quali l'ereditarietà, l'abuso di alcool
 - E' difficile dire quanto una dieta sia "sana" e l'allenamento "adeguato"
 - ✓ Tutto ciò introduce **incertezza** sull'esito della previsione
- I classificatori Bayesiani **modellano relazioni probabilistiche tra gli attributi e l'attributo di classificazione**

Classificazione Bayesiana

- Definisce uno schema probabilistico per risolvere problemi di classificazione

- Probabilità condizionale: $P(Y | X) = \frac{P(X,Y)}{P(X)}$

$$P(X | Y) = \frac{P(X,Y)}{P(Y)}$$

- **Teorema di Bayes:**

$$P(Y | X) = \frac{P(X | Y) P(Y)}{P(X)}$$

- $P(Y | X) = \frac{P(Y,X)}{P(X)} = \frac{P(X|Y) P(Y)}{P(X)}$

Teorema di Bayes – Esempio 1

- Dati:
 - Un medico sa che la meningite provoca il torcicollo nel 50% dei casi ($P(T|M)=0,5$).
 - La probabilità che un paziente abbia la meningite è $1/50.000$ ($P(M)=1/50.000$)
 - La probabilità che un paziente abbia il torcicollo è $1/20$ ($P(T) = 1/20$)
- Se un paziente ha il torcicollo, qual è la probabilità che abbia anche la meningite ($P(M|T)$)?

$$P(M | T) = \frac{P(T | M) \times P(M)}{P(T)} = \frac{0,5 \times 1/50.000}{1/20} = 0,0002$$

Teorema di Bayes – Esempio 2

- Sia X la variabile casuale che rappresenta la squadra che **ospita** la partita e Y la variabile casuale che rappresenta il **vincitore** della partita. Sia X che Y possono assumere valori dall'insieme $\{0,1\}$ (identificatori squadra).
- Si assumano le seguenti probabilità:
 - Probabilità Squadra 0 vinca: $P(Y = 0) = 0,65$.
 - Probabilità Squadra 1 vinca: $P(Y = 1) = 1 - P(Y=0) = 0,35$.
 - Probabilità Squadra 1 ospiti la partita che ha vinto: $P(X=1 | Y=1)=0,75$.
 - Probabilità Squadra 1 ospiti la partita vinta dalla squadra 0: $P(X=1 | Y=0) = 0,3$.
- Calcolare la probabilità (condizionata) che la squadra 1 vinca la prossima partita che giocherà in casa, cioè $P(Y=1 | X=1)$. Usando il teorema di Bayes, otteniamo

$$\begin{aligned} \bullet P(Y = 1 | X = 1) &= \frac{P(X=1|Y=1)\times P(Y=1)}{P(X=1)} = \frac{P(X=1|Y=1)\times P(Y=1)}{P(X=1,Y=1)+P(X=1,Y=0)} \\ &= \frac{P(X=1|Y=1)\times P(Y=1)}{P(X=1|Y=1)\times P(Y=1)+P(X=1|Y=0)\times P(Y=0)} = \frac{0,75\times 0,35}{0,75\times 0,35 + 0,3\times 0,65} = 0,5738 \end{aligned}$$

Teorema di Bayes e Classificazione

- Considera ciascun attributo ed etichetta di classe come variabili casuali
- Dato un record con attributi (X_1, X_2, \dots, X_d)
 - Obiettivo è prevedere la classe Y
 - Nello specifico, vogliamo trovare il valore di Y che massimizza $P(Y | X_1, X_2, \dots, X_d)$
- Possiamo a stimare $P(Y | X_1, X_2, \dots, X_d)$ direttamente dai dati?

Esempio

Predizione della classe di un record con i seguenti valori

$$X = (\text{Refund}=\text{No}, \text{Status}=\text{Divorced}, \text{Income}=120k)$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Possiamo stimare

$P(\text{Evade} = \text{Yes} | X)$ e $P(\text{Evade} = \text{No} | X)$?

In modo da stabilire la classe dell'oggetto:

- ◆ Se $P(\text{Evade}=\text{No}|X) > P(\text{Evade}=\text{Yes}|X) \rightarrow \text{No}$
- ◆ Se $P(\text{Evade}=\text{Yes}|X) > P(\text{Evade}=\text{No}|X) \rightarrow \text{Yes}$

Nel seguito sono usate diverse semplificazioni, e.g.:

Evade = Yes con Yes, e

Evade = No con No

Teorema di Bayes e Classificazione

- Approccio:

- Calcola la probabilità $P(Y | X_1, X_2, \dots, X_d)$ usando il teorema di Bayes

$$P(Y | X_1, \dots, X_d) = \frac{P(X_1, \dots, X_d | Y) P(Y)}{P(X_1, \dots, X_d)}$$

- *Maximum a-posteriori*: Scegli Y che massimizzi $P(Y | X_1, \dots, X_d)$
 - Equivalente a scegliere il valore di Y che massimizzi $P(X_1, \dots, X_d | Y) P(Y)$

- Come calcolare $P(X_1, \dots, X_d | Y)$?

Esempio

Dato il seguente record del test dataset:

$$X = (\text{Refund}=\text{No}, \text{Status}=\text{Divorced}, \text{Income}=120\text{k})$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Usando il Teorema di Bayes

$$P(\text{Yes} | X) = \frac{P(X | \text{Yes}) P(\text{Yes})}{P(X)}$$

$$P(\text{No} | X) = \frac{P(X | \text{No}) P(\text{No})}{P(X)}$$

Come calcolare $P(X | \text{Yes})$ e $P(X | \text{No})$?

Classificatore Naïve Bayes

- Si assume l'indipendenza tra gli attributi X_i quando la classe è nota (indipendenza condizionale):
 - $P(X_1, \dots, X_d | Y) = P(X_1 | Y) P(X_2 | Y) \dots P(X_d | Y)$
 - Possiamo calcolare $P(X_i | Y_j)$ per tutti gli X_i e Y_j del training dataset
 - Il nuovo oggetto è classificato come Y_j se $P(Y_j) \prod_i^d P(X_i | Y_j)$ è massimo.

Indipendenza condizionale

- Princípio di indipendenza condizionale: date le variabili aleatorie \mathbf{X} , \mathbf{Y} e \mathbf{Z} , si dice che \mathbf{X} , \mathbf{Y} sono condizionatamente indipendenti rispetto a \mathbf{Z} se:

$$\mathbf{P}(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}) = \mathbf{P}(\mathbf{X} \mid \mathbf{Z})$$

- Dal principio di indipendenza deriva che

$$\begin{aligned}\mathbf{P}(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z}) &= \frac{\mathbf{P}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})}{\mathbf{P}(\mathbf{Z})} = \frac{\mathbf{P}(\mathbf{X}, \mathbf{Y}, \mathbf{Z})}{\mathbf{P}(\mathbf{Y}, \mathbf{Z})} \times \frac{\mathbf{P}(\mathbf{Y}, \mathbf{Z})}{\mathbf{P}(\mathbf{Z})} \\ &= \mathbf{P}(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}) \times \mathbf{P}(\mathbf{Y} \mid \mathbf{Z}) = \mathbf{P}(\mathbf{X} \mid \mathbf{Z}) \mathbf{P}(\mathbf{Y} \mid \mathbf{Z})\end{aligned}$$

- Quindi:

- $\mathbf{P}(\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_d \mid \mathbf{Y}_j) = \mathbf{P}(\mathbf{X}_1 \mid \mathbf{Y}_j) \mathbf{P}(\mathbf{X}_2 \mid \mathbf{Y}_j) \dots \mathbf{P}(\mathbf{X}_d \mid \mathbf{Y}_j)$

Classificazione Naïve Bayes – Esempio 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Classificazione Naïve Bayes – Esempio 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A | M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0,06$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Classificazione Naïve Bayes – Esempio 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A | M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0,06$$

$$P(A | N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0,0042$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Classificazione Naïve Bayes – Esempio 1

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0,06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0,0042$$

$$P(A|M)P(M) = 0,06 \times \frac{7}{20} = 0,021$$

$$P(A|N)P(N) = 0,004 \times \frac{13}{20} = 0,0027$$

$$P(M|A) = \frac{P(A|M)P(M)}{P(A)} = \frac{0,021}{0,15} = 0,14$$

$$P(N|A) = \frac{P(A|N)P(N)}{P(A)} = \frac{0,0027}{0,15} = 0,018$$

$P(A|M)P(M) > P(A|N)P(N)$

→ Mammals

Classificatore Naïve Bayes – Esempio 2

Dato il seguente record del test dataset:

$$X = (\text{Refund}=\text{No}, \text{Status}=\text{Divorced}, \text{Income}=120\text{k})$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- $P(X | \text{Yes}) =$
 $P(\text{Refund} = \text{No} | \text{Yes}) \times$
 $P(\text{Divorced} | \text{Yes}) \times$
 $P(\text{Income} = 120\text{K} | \text{Yes})$

- $P(X | \text{No}) =$
 $P(\text{Refund} = \text{No} | \text{No}) \times$
 $P(\text{Divorced} | \text{No}) \times$
 $P(\text{Income} = 120\text{K} | \text{No})$

Stima della probabilità

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Classe: $P(Y) = N_c/N$

- e.g., $P(\text{No}) = 7/10$,
 $P(\text{Yes}) = 3/10$

- Per attributi categorici:

$$P(X_i | Y_k) = |X_{ik}| / N_c$$

- dove $|X_{ik}|$ è il numero di istanze con valore X_{ik} per l'attributo X_i che appartengono alla classe Y_k

- Esempi:

$$P(\text{Status}=\text{Married} | \text{No}) = 4/7$$
$$P(\text{Refund}=\text{Yes} | \text{Yes})=0$$

Stima della probabilità dai dati

Dato il seguente record del test dataset:

$$X = (\text{Refund}=\text{No}, \text{Status}=\text{Divorced}, \text{Income}=120k)$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naïve Bayes Classifier:

$$P(\text{No}) = 7/10$$

$$P(\text{Yes}) = 3/10$$

$$P(\text{Refund} = \text{Yes} \mid \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} \mid \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} \mid \text{Yes}) = 0/3 = 0$$

$$P(\text{Refund} = \text{No} \mid \text{Yes}) = 3/3 = 1$$

$$P(\text{Marital Status} = \text{Single} \mid \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} \mid \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} \mid \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} \mid \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} \mid \text{Yes}) = 0$$

Stima della probabilità (attributi continui)

- Nel caso l'attributo A sia continuo non è possibile stimare la probabilità per ogni suo valore
 - ✓ Discretizzare l'attributo in intervalli creando un attributo ordinale
 - se si usano troppi intervalli il limitato numero di eventi del training set per intervallo rende inaffidabile la previsione
 - se si usano pochi intervalli uno di essi può aggregare valori associabili a classi diverse e determinare quindi un decision boundary sbagliato
 - ✓ Associare all'attributo una funzione di densità e stimare i parametri della funzione dal training set:
 - ✓ Stima della densità di probabilità:
 - ◆ Si assume che i valori dell'attributo seguano una distribuzione normale
 - ◆ Si utilizzano i dati per stimare i parametri di distribuzione (ad es. Media e deviazione standard)
 - ◆ Una volta che la distribuzione di probabilità è nota, usala per stimare la probabilità condizionale $P(X_i | Y)$

Stima della probabilità dai dati

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

● Distribuzione normale

$$P(X_i | Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

— Ciascuna coppia (X_i, Y_j)

- Per (Income, Class=No):
Se Class=No

media = 110

varianza = 1910

dev. st. = 43,7

$$P(\text{Income} = 120 | \text{No}) = \frac{1}{\sqrt{2\pi\cdot 1910}} e^{-\frac{(120-110)^2}{2\cdot 1910}} = 0,0588$$

Classificatore Naïve Bayes – Esempio 2

Dato il seguente record del test dataset:

$$X = (\text{Refund}=\text{No}, \text{Status}=\text{Divorced}, \text{Income}=120k)$$

Naïve Bayes Classifier:

$$P(\text{No}) = 7/10$$

$$P(\text{Yes}) = 3/10$$

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

$$\text{P(Refund = No | No)} = 4/7$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0/3 = 0$$

$$\text{P(Refund = No | Yes)} = 3/3 = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$$

$$\text{P(Marital Status = Divorced | No)} = 1/7$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$\text{P(Marital Status = Divorced | Yes)} = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0$$

Per Taxable Income:

Se class = No:

$$\text{media} = 110$$

$$\text{varianza} = 1910$$

Se class = Yes:

$$\text{media} = 90$$

$$\text{varianza} = 2070$$

$$\text{P}(120 | \text{No}) = 0,0588$$

$$\text{P}(120 | \text{Yes}) = 0,0071$$

Classificatore Naïve Bayes – Esempio 2

Dato il seguente record del test dataset:

$$X = (\text{Refund}=\text{No}, \text{Status}=\text{Divorced}, \text{Income}=120\text{k})$$

Naïve Bayes Classifier:

- $P(X | \text{No}) = P(\text{Refund}=\text{No} | \text{No}) \times P(\text{Divorced} | \text{No}) \times P(\text{Income}=120\text{K} | \text{No})$
 $= 4/7 \times 1/7 \times 0.0588 = 0.0048$
- $P(X | \text{Yes}) = P(\text{Refund}=\text{No} | \text{Yes}) \times P(\text{Divorced} | \text{Yes}) \times P(\text{Income}=120\text{K} | \text{Yes})$
 $= 3/3 \times 1/3 \times 0.0071 = 0.0024$
- Poiché $P(X|\text{No}) P(\text{No}) > P(X|\text{Yes}) P(\text{Yes})$
 $4,8 \times 10^{-3} \times 7 \times 10^{-1} > 2,4 \times 10^{-3} \times 3 \times 10^{-1}$
- Allora $P(\text{No}|X) > P(\text{Yes}|X)$ → **Class = No**

Classificatore Naïve Bayes - Problemi

Dato il seguente record del test dataset:

$$X = (\text{Refund}=\text{Yes}, \text{Status}=\text{Divorced}, \text{Income}=120k)$$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- $P(\text{No}) = 7/10$
- $P(\text{Yes}) = 3/10$
- $P(\text{No} | \text{Divorced}) = 1/7 \times 7/10 / 2/10 = 1/2$
- $P(\text{Yes} | \text{Divorced}) = 1/3 \times 3/10 / 2/10 = 1/2$
- $P(\text{No} | \text{Refund}=\text{No}) = 4/7 \times 7/10 / 7/10 = 4/7$
- $P(\text{Yes} | \text{Refund}=\text{No}) = 3/3 \times 3/10 / 7/10 = 3/7$
- $P(\text{No} | \text{Married}) = 4/7 \times 7/10 / 4/10 = 1$
- $P(\text{Yes} | \text{Married}) = 0 \times 3/10 / 4/10 = 0$

Classificatore Naïve Bayes - Problemi

Dato il seguente record del test dataset:

$X = (\text{Refund}=\text{Yes}, \text{Status}=\text{Divorced}, \text{Income}=120k)$

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- $P(\text{No}) = 7/10 \quad P(\text{Refund} = \text{No}) = 7/10$
 $P(\text{Yes}) = 3/10 \quad P(\text{Divorced}) = 2/10$
- $P(\text{Yes} | \text{Refund} = \text{No, Divorced}) =$**
 $P(\text{Refund}=\text{No}, \text{Divorced} | \text{Yes}) \times P(\text{Yes}) / P(\text{Refund}=\text{No}, \text{Divorced}) =$
 $(1/3 \times 3/10) / (1/10) = 1$
 $P(\text{Refund}=\text{No}|\text{Yes}) \times P(\text{Divorced}|\text{Yes}) \times P(\text{Yes}) / P(\text{R}=\text{No, Div.}) =$
 $(1 \times 1/3 \times 3/10) / (1/10) = 1$
- $P(\text{No} | \text{Refund} = \text{No, Divorced}) =$**
 $P(\text{Refund}=\text{No}, \text{Divorced} | \text{No}) \times P(\text{No}) / P(\text{Refund}=\text{No}, \text{Divorced}) =$
 $(0/7 \times 7/10) / (1/10) = 0$
 $P(\text{Refund}=\text{No}|\text{No}) \times P(\text{Divorced}|\text{No}) \times P(\text{No}) /$
 $P(\text{Refund}=\text{No}) \times P(\text{Divorced}) =$
 $(4/7 \times 1/7 \times 7/10) / (1/10) = 4/7$
- $P(\text{R}=\text{N, Divorced} | \text{No}) \neq P(\text{R}=\text{No} | \text{No}) \times P(\text{Divorced}|\text{No})$**

Classificatore Naïve Bayes - Problemi

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Classificazione Naïve Bayes :

$$P(\text{Refund} = \text{Yes} | \text{No}) = 3/7$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/7$$

$$P(\text{Refund} = \text{Yes} | \text{Yes}) = 0/3 = 0$$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 3/3 = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/7$$

$$P(\text{Marital Status} = \text{Divorced} | \text{No}) = 1/7$$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/7$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$$

For Taxable Income:

$$\text{Se class} = \text{No: } \text{media} = 110$$

$$\text{varianza} = 2995$$

$$\text{Se class} = \text{Yes: } \text{media} = 90$$

$$\text{varianza} = 25$$

Given $X = (\text{Refund} = \text{Yes}, \text{Divorced}, 120\text{K})$

$$P(X | \text{No}) = 3/7 \times 1/7 \times 0,0072 = 0,0004$$

$$P(\text{No}) = 7/10$$

$$P(X | \text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$$

$$P(\text{Yes}) = 3/10$$

Classificatore Naïve Bayes - Problemi

Si consideri la tabella senza il record 7

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Classificazione Naïve Bayes :

$$P(\text{Refund} = \text{Yes} | \text{No}) = 2/6$$

$$P(\text{Refund} = \text{No} | \text{No}) = 4/6$$

→ $P(\text{Refund} = \text{Yes} | \text{Yes}) = 0/3 = 0$

$$P(\text{Refund} = \text{No} | \text{Yes}) = 3/3 = 1$$

$$P(\text{Marital Status} = \text{Single} | \text{No}) = 2/6$$

→ $P(\text{Marital Status} = \text{Divorced} | \text{No}) = 0$

$$P(\text{Marital Status} = \text{Married} | \text{No}) = 4/6$$

$$P(\text{Marital Status} = \text{Single} | \text{Yes}) = 2/3$$

$$P(\text{Marital Status} = \text{Divorced} | \text{Yes}) = 1/3$$

$$P(\text{Marital Status} = \text{Married} | \text{Yes}) = 0/3$$

For Taxable Income:

Se class = No: media = 91

$$\text{varianza} = 685$$

Se class = Yes: media = 90

$$\text{varianza} = 25$$

Given X = (Refund = Yes, Divorced, 120K)

$$P(X | \text{No}) = 2/6 \times 0 \times 0.0083 = 0$$

$$P(X | \text{Yes}) = 0 \times 1/3 \times 1.2 \times 10^{-9} = 0$$

Naïve Bayes non è in grado di classificare X come Yes o No!

Classificatori Naïve Bayes - Problemi

- Se una delle probabilità condizionali è zero, allora l'intera espressione diventa zero
- È necessario utilizzare altre stime delle probabilità condizionali rispetto alle frazioni semplici
- Stima della probabilità:

$$\text{Original : } P(A_i | C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace : } P(A_i | C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m - estimate : } P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

c: numero di classi

p: probabilità a priori della classe C

m: parametro

N_c : numero di istanze nella classe C

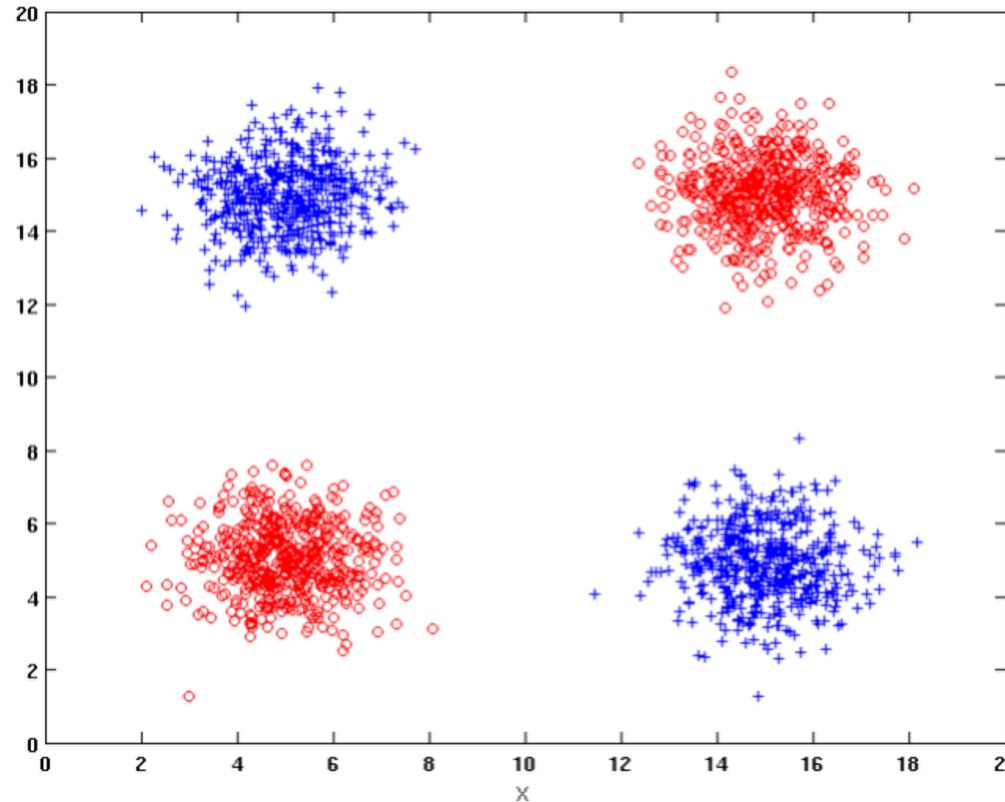
N_{ic} : numero di istanze con valore dell'attributo A_i nella classe C

Naïve Bayes (Summary)

- Robusto rispetto al rumore (punti isolati)
- Gestisce i valori mancanti ignorando l'istanza durante i calcoli delle stime di probabilità
- Robusto rispetto agli attributi irrilevanti
- L'assunzione di indipendenza potrebbe non essere valida per alcuni attributi
 - Usa tecniche alternative come Bayesian Belief Networks (BBN)

Naïve Bayes

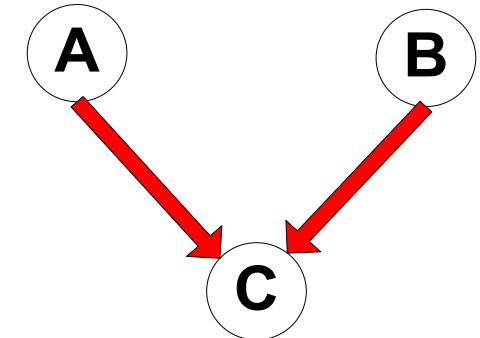
- Come si comporta Naïve Bayes sul seguente set di dati?



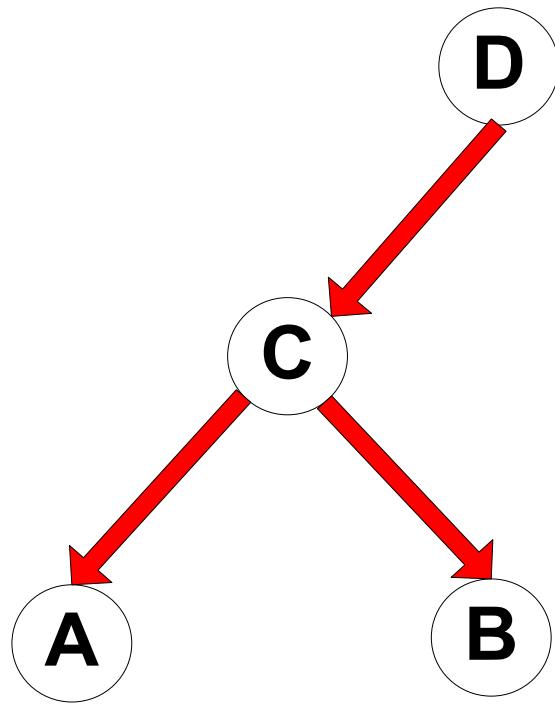
L'indipendenza condizionale degli attributi è violata

Bayesian Belief Networks

- Forniscono la rappresentazione grafica delle relazioni probabilistiche tra un insieme di variabili casuali
- Consiste di:
 - Un grafo diretto aciclico (dag)
 - ◆ Un nodo corrisponde ad una variabile
 - ◆ Un arco corrisponde alla relazione di dipendenza tra una coppia di variabili
 - Una tabella di probabilità che associa ciascun nodo al suo genitore immediato



Indipendenza Conditionale



D è padre di C

A è figlio di C

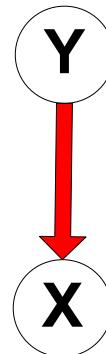
B è discendente di D

D è antenato di A

- Un nodo in una rete bayesiana è condizionalmente indipendente da tutti i nodi non-discendenti, se i suoi genitori sono noti

Tabelle probabilistiche

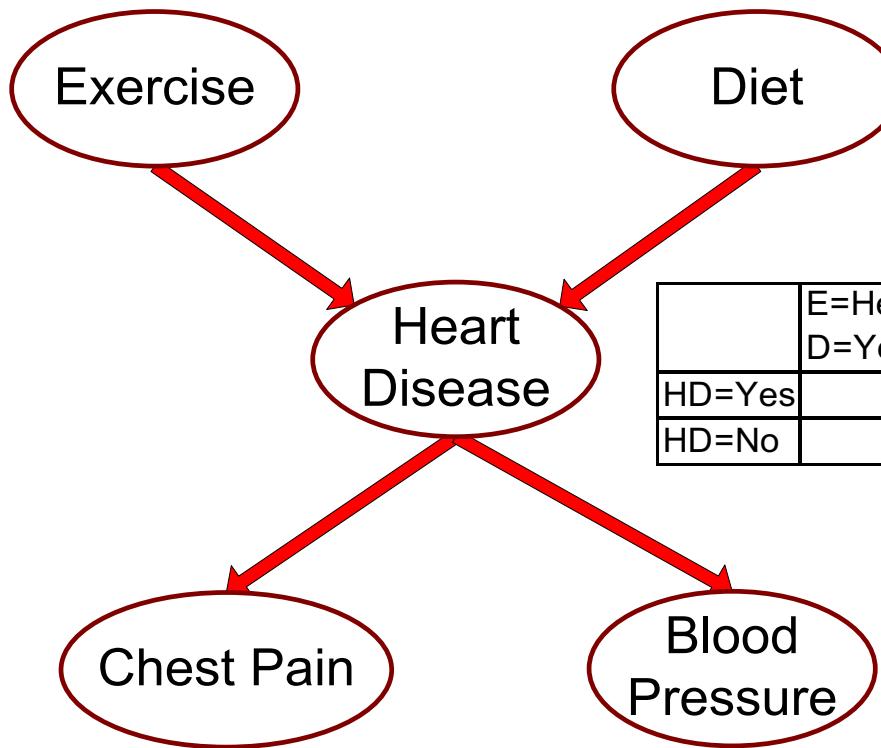
- Se X non ha genitori, la tabella contiene la probabilità a priori $P(X)$
- Se X ha un solo genitore (Y), la tabella contiene la probabilità condizionale $P(X | Y)$
- Se X ha più genitori (Y_1, Y_2, \dots, Y_k), la tabella contiene la probabilità condizionale $P(X | Y_1, Y_2, \dots, Y_k)$



Bayesian Belief Network - Esempio

Exercise=Yes	0.7
Exercise>No	0.3

Diet=Healthy	0.25
Diet=Unhealthy	0.75



	E=Healthy D=Yes	E=Healthy D=No	E=Unhealthy D=Yes	E=Unhealthy D=No
HD=Yes	0.25	0.45	0.55	0.75
HD=No	0.75	0.55	0.45	0.25

	HD=Yes	HD=No
CP=Yes	0.8	0.01
CP>No	0.2	0.99

	HD=Yes	HD=No
BP=High	0.85	0.2
BP=Low	0.15	0.8

Bayesian Belief Network - Esempio

- Dato: $X = (Ex=No, D=Yes, CP=Yes, BP=High)$

– Calcola $P(HD | Ex, D, CP, BP)$?

- $P(HD=Yes | Ex=No, D=Yes) = 0.55$

$$P(CP=Yes | HD=Yes) = 0.8$$

$$P(BP=High | HD=Yes) = 0.85$$

– $P(HD=Yes | Ex=No, D=Yes, CP=Yes, BP=High)$

$$\propto 0.55 \times 0.8 \times 0.85 = 0.374 >$$

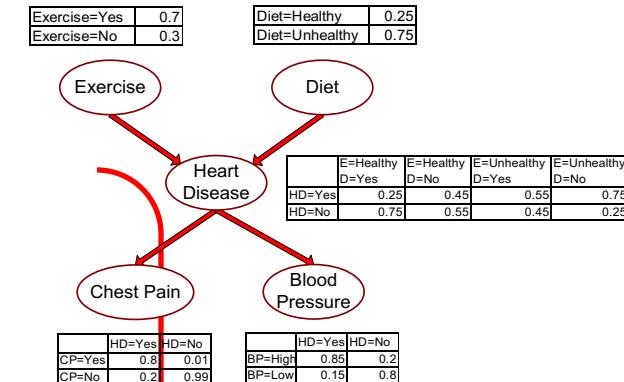
- $P(HD=No | Ex=No, D=Yes) = 0.45$

$$P(CP=Yes | HD=No) = 0.01$$

$$P(BP=High | HD=No) = 0.2$$

– $P(HD=No | Ex=No, D=Yes, CP=Yes, BP=High)$

$$\propto 0.45 \times 0.01 \times 0.2 = 0.0009$$



classifica
X come
Yes

Reti Neurali (Artificial Neural Networks)

Storia

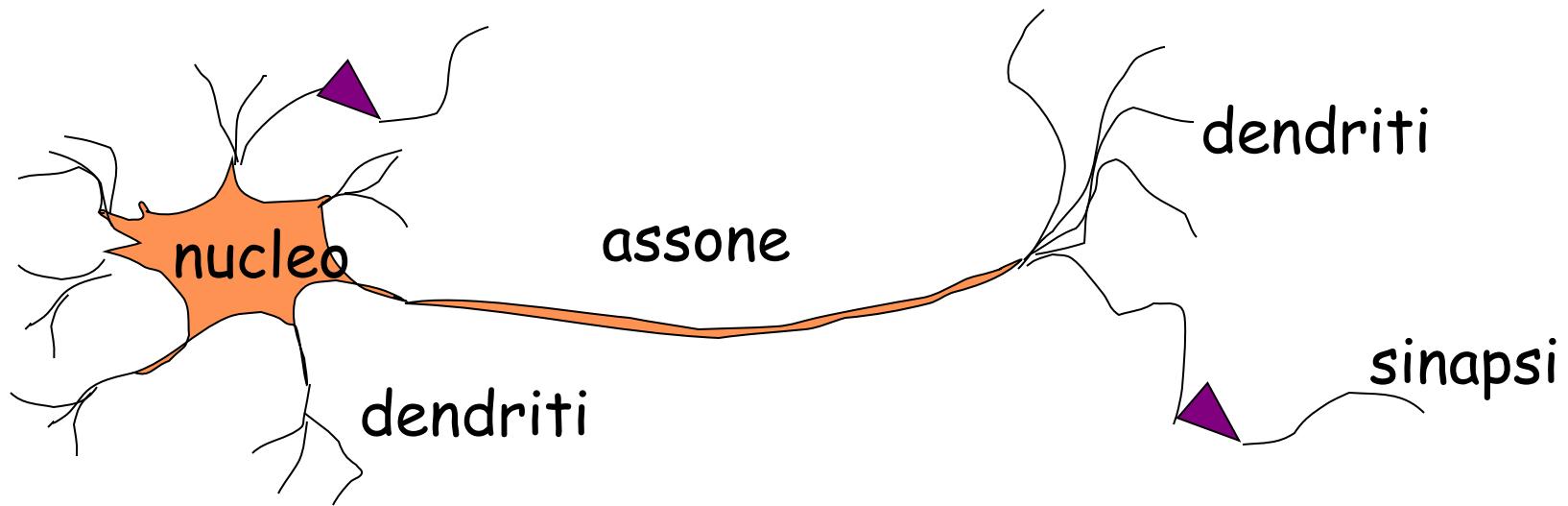
- Le reti neurali artificiali (**Artificial Neural Networks** o **ANN**) sono una simulazione astratta del nostro sistema nervoso, che contiene una collezione di neuroni i quali comunicano fra loro mediante connessioni dette assoni
- Il modello ANN ha una certa somiglianza con gli assoni e dendriti in un sistema nervoso
- Il primo modello di reti neurali fu proposto nel 1943 da **McCulloch** e **Pitts** nei termini di un modello computazionale dell'attività nervosa.
 - A questo modello sono seguiti altri proposti da John Von Neumann, Marvin Minsky, Frank Rosenblatt e molti altri

Due categorie di modelli

- **Biologico**: ha l'obiettivo di imitare sistemi neurali biologici, come le funzionalità auditive e visive.
 - Obiettivo: verifica di ipotesi riguardo ai sistemi biologici
- **Guidato dalle applicazioni**: meno interessato a “mimare” funzioni biologiche
 - Le architetture sono ampiamente condizionate dalle necessità applicative
 - Questi modelli sono anche denominati **architetture connessioniste**
 - Modello trattato nel corso di Machine Learning

Neuroni

- Molti neuroni possiedono strutture arboree chiamate **dendriti** che ricevono segnali da altri neuroni mediante giunzioni dette **sinapsi**
- Alcuni neuroni comunicano mediante poche sinapsi, altri ne posseggono migliaia



Funzionamento di un Neurone

- Si stima che il cervello umano contenga oltre 100 miliardi di **neuroni** e che un neurone può avere oltre 1000 sinapsi in ingresso e in uscita
- Tempo di commutazione di **alcuni millisecondi** (assai più lento di una porta logica), ma la connettività centinaia di volte superiore
- Un neurone trasmette informazioni agli altri neuroni tramite il proprio **assone**
 - L'assone trasmette impulsi elettrici, che dipendono dal suo potenziale
 - L'informazione trasmessa può essere **eccitatoria** oppure **inibitoria**
- Un neurone riceve in ingresso segnali di varia natura, che vengono sommati
- Se l'influenza eccitatoria è predominante, il neurone si attiva e genera messaggi informativi verso le sinapsi di uscita

Struttura di una Rete Neurale

- Una **rete neurale** è costituta da:
 - un insieme di nodi (i **neuroni**) o unità connesse da collegamenti
 - Un insieme di **pesi** associati ai collegamenti
 - Un insieme di **soglie** o livelli di attivazione
- La **progettazione** di una rete neurale richiede:
 - La scelta del numero e del tipo di unità
 - La determinazione della struttura morfologica
 - Codifica degli esempi di addestramento, in termini di ingressi e uscite dalla rete
 - L'inizializzazione e l'addestramento per la determinazione dei pesi delle interconnessioni, attraverso il training set

Problemi Risolvibili con le Reti Neurali

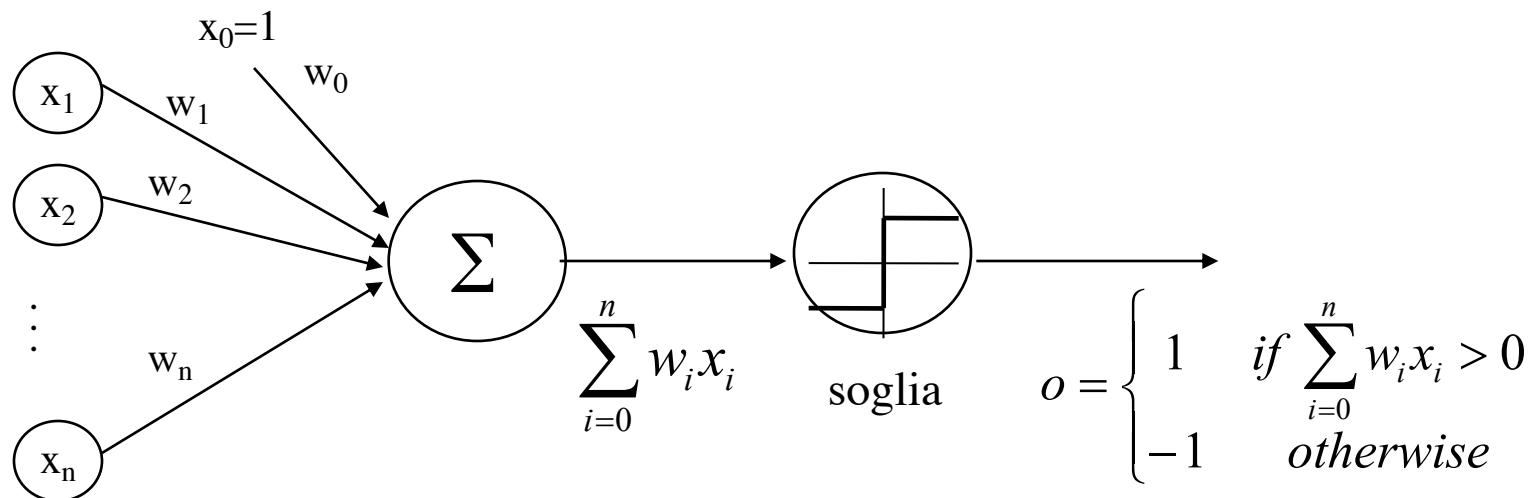
- Caratteristiche delle reti:

- Le istanze sono rappresentate mediante **molte feature a molti valori, anche reali**
- La funzione obiettivo può essere a **valori reali**
- Gli esempi possono essere rumorosi
- I tempi di addestramento possono essere lunghi
- La valutazione della rete appresa deve poter essere effettuata velocemente
- Non è cruciale capire la **semantica** della funzione attesa

- Applicazioni: robotica, image understanding, sistemi biologici, predizioni finanziarie, ecc.

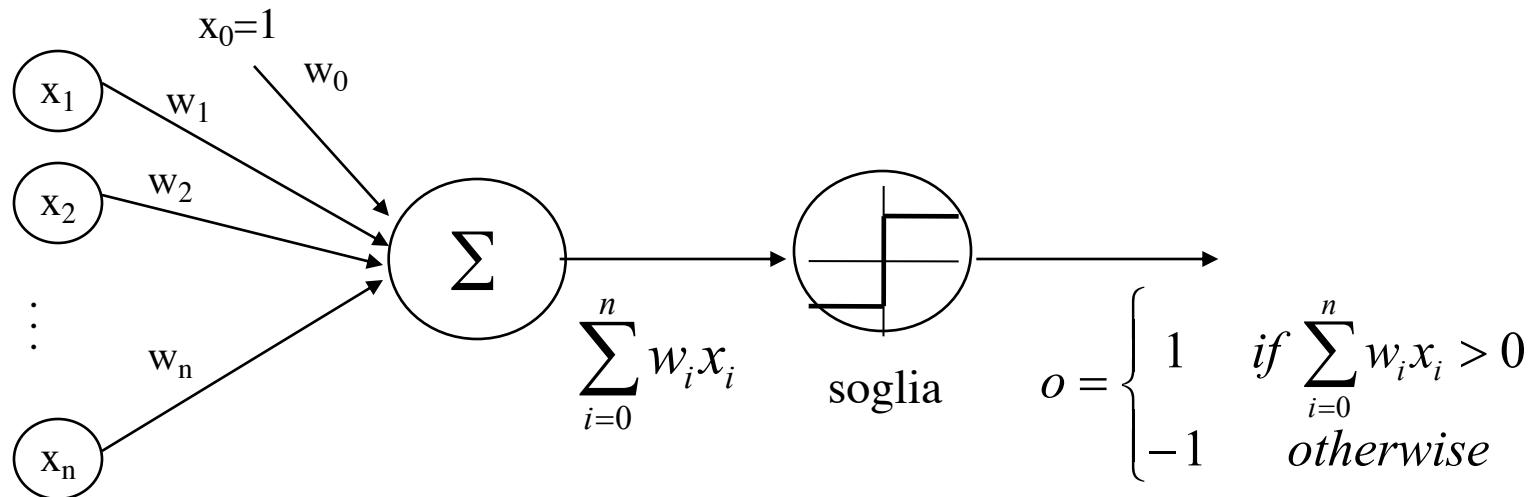
Il Percettrone (perceptron)

- Il percettrone è la rete neurale più semplice
- Nasce da un'idea di Rosenblatt (1962)
- Cerca di simulare il funzionamento del singolo neurone (biologico).



Il Percettrone

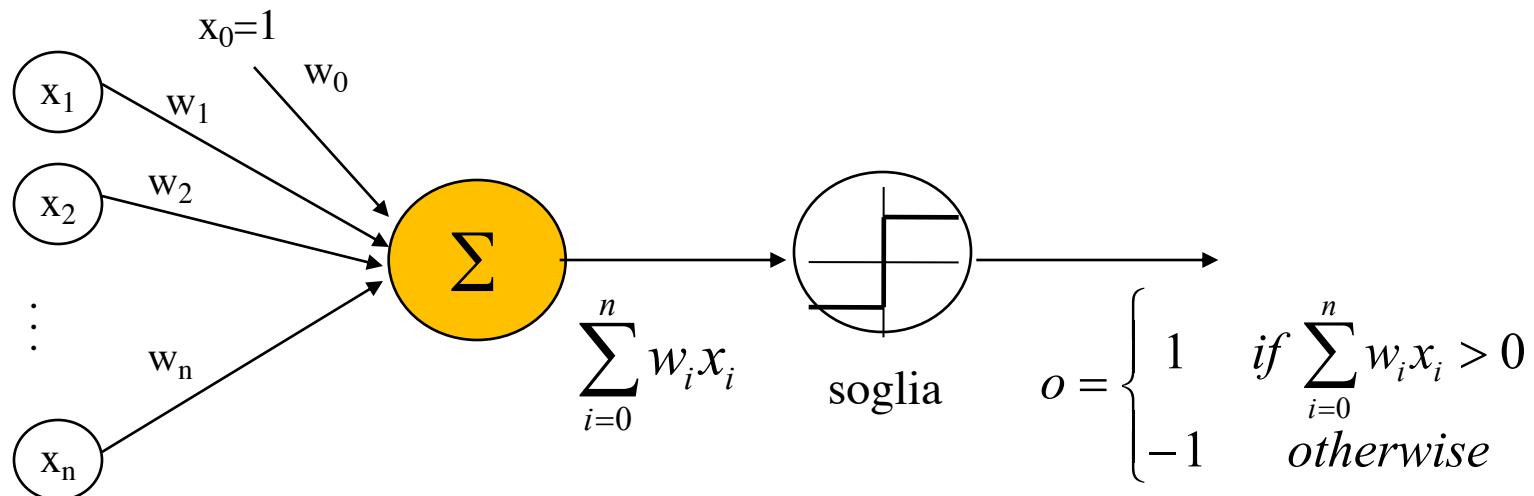
- I valori di uscita sono booleani: -1 oppure 1
- Gli ingressi x_i e i pesi w_i sono valori reali positivi o negativi
- Tre elementi: **ingressi, somma, soglia**
- L'apprendimento consiste nel selezionare **pesi e soglia**



Funzioni somma e soglia (1)

- La funzione d'ingresso (lineare, somma delle componenti di input di $x = (x_1, \dots, x_n)$)

$$w_0 + w_1 x_1 + \dots + w_n x_n = \sum_{i=0}^n w_i x_i = \vec{w} \cdot \vec{x}$$

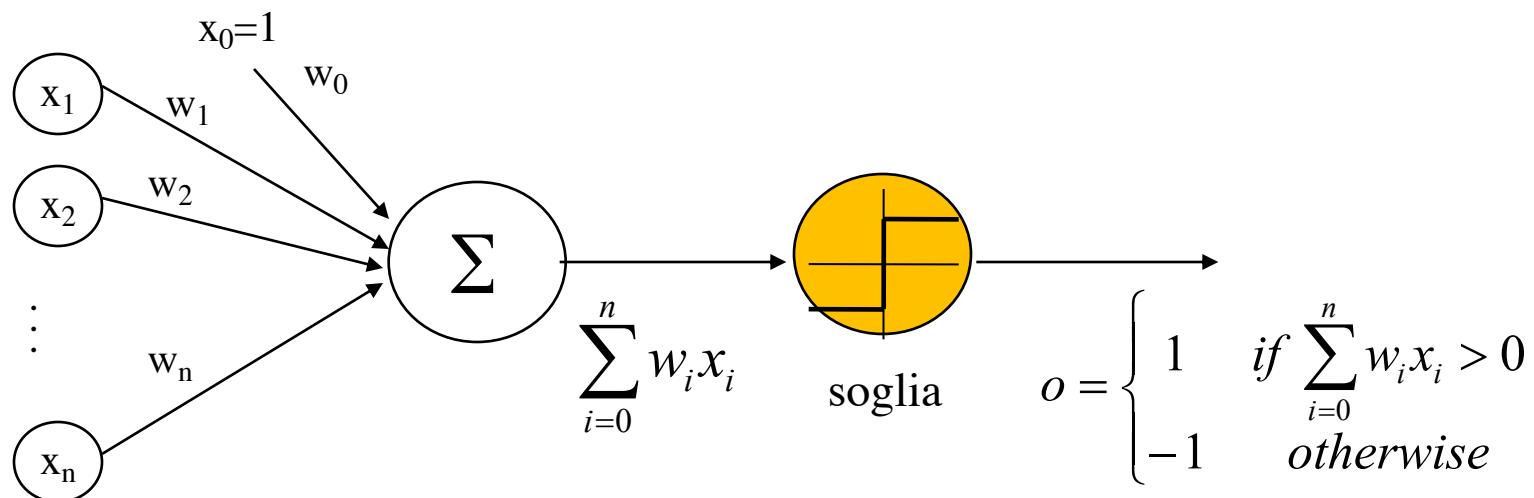


Funzioni somma e soglia (2)

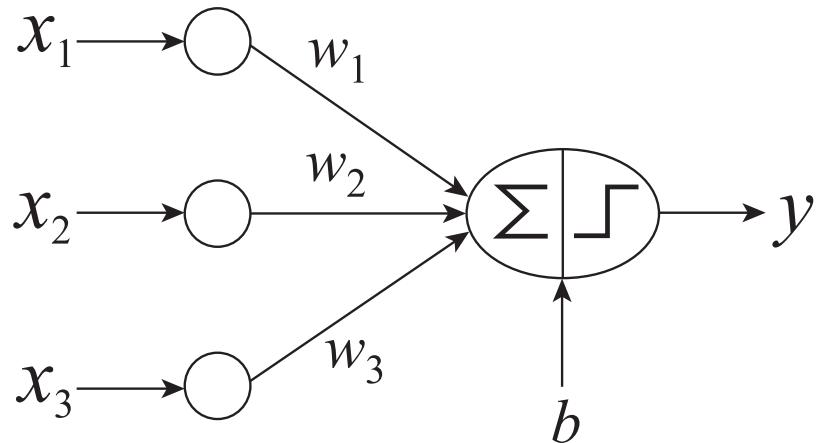
● La funzione di attivazione (non lineare, soglia)

$$o(x_1, \dots, x_n) = g\left(\sum_{i=0}^n w_i x_i\right)$$

- Vogliamo l'unità percettore **attiva** (vicino a +1) quando gli input corretti sono forniti e **inattiva** altrimenti
- E' preferibile che g sia **non lineare**, altrimenti la rete neurale diventa una semplice funzione lineare dell'input



Percettrone – Architettura di base



$$y = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x} + b > 0. \\ -1, & \text{otherwise.} \end{cases}$$

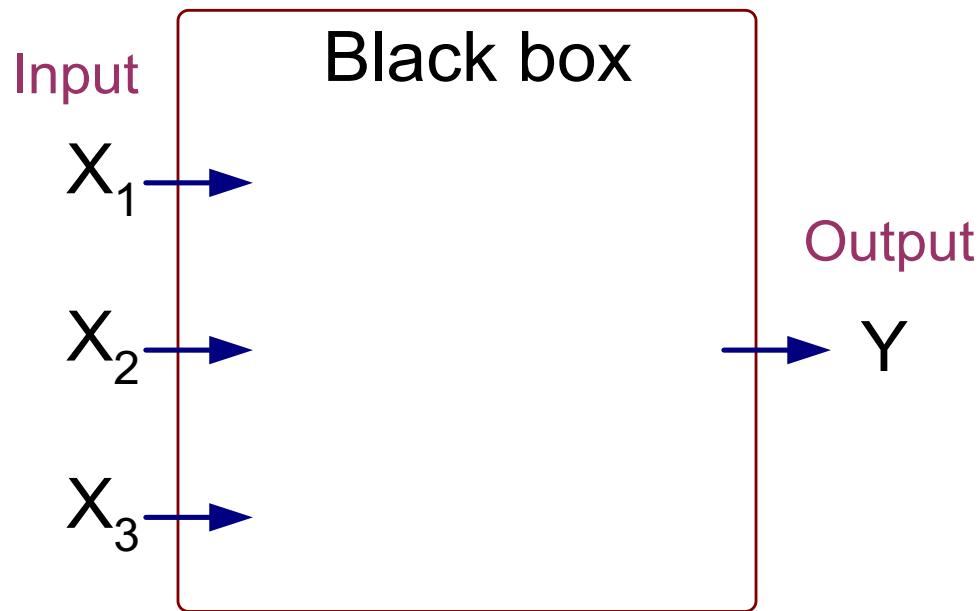
$$\tilde{\mathbf{w}} = (\mathbf{w}^T \ b)^T \quad \tilde{\mathbf{x}} = (\mathbf{x}^T \ 1)^T$$

$$\hat{y} = sign(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})$$

Funzione di attivazione

Artificial Neural Networks (ANN)

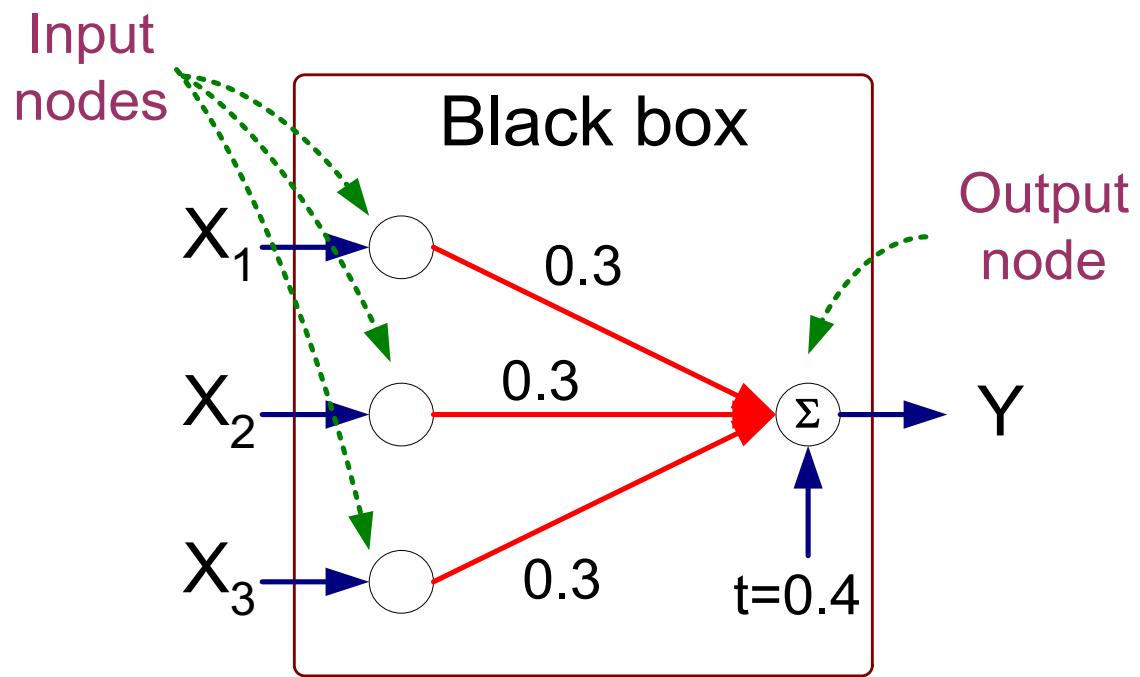
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



L'output Y è 1 se almeno due dei tre valori di input sono uguali a 1.

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

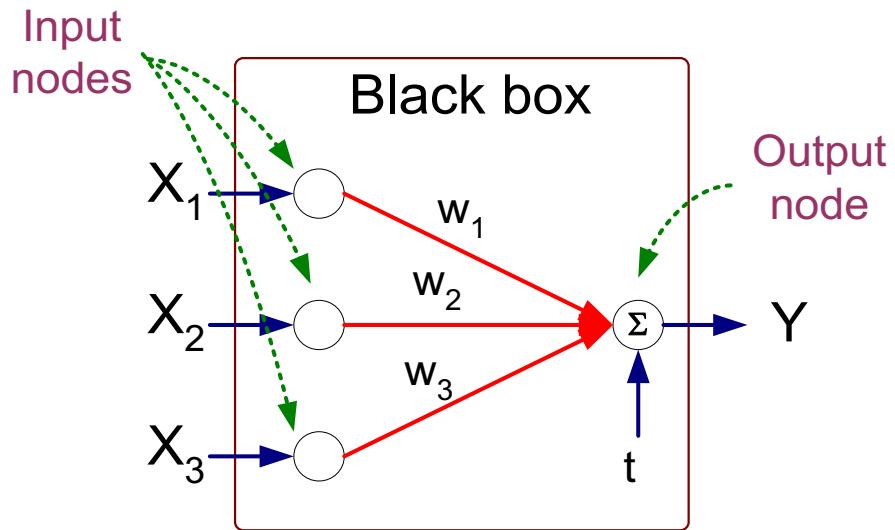


$$Y = \text{sign}(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

where $\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$

Artificial Neural Networks (ANN)

- Il modello è un insieme di nodi interconnessi con collegamenti «pesati».
- Il nodo di output somma ciascun valore di input in base ai pesi dei suoi collegamenti
- Confronta il nodo di output con una soglia t



Modello del Percettrone (Perceptron)

$$\begin{aligned} Y &= \text{sign}\left(\sum_{i=1}^d w_i X_i - t\right) \\ &= \text{sign}\left(\sum_{i=0}^d w_i X_i\right) \end{aligned}$$

Il Percettrone

- Rete a un solo livello
 - Contiene solo nodi di ingresso e di uscita
- Funzione di attivazione: $f = \text{sign}(w \cdot x)$
- L'applicazione del modello è banale, e.g.

$$Y = \text{sign}(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

where $\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$

- $X_1 = 1, X_2 = 0, X_3 = 1 \rightarrow y = \text{sign}(0.2) = 1$

Addestramento del Percettrone (learning)

- Inizializza i pesi (w_0, w_1, \dots, w_d)
- Ripeti
 - Per ciascun esempio del training set (x_i, y_i)
 - ◆ Calcola $f(w, x_i)$
 - ◆ Calcola l'errore $E = y_i - f(w, x_i)$
 - ◆ Aggiorna i pesi:
$$w^{k+1} = w^k + \lambda E x_i = w^k + \lambda(y_i - f(w^k, x_i))x_i$$
- Fino a quando la condizione di uscita è verificata

Addestramento del Percettrone (learning)

- Aggiornamento dei pesi:

$$w^{k+1} = w^k + \lambda E x_i = w^k + \lambda(y_i - f(w^k, x_i))x_i$$

λ = learning rate

- Intuizione:

- Aggiorna i pesi basandoti sull'errore: $E = y_i - f(w^k, x_i)$
- Se $y=f(x, w)$, $E=0$: nessun aggiornamento è necessario
- Se $y>f(x, w)$, $E=2$: il peso deve essere incrementato in modo che $f(x, w)$ cresca
- Se $y<f(x, w)$, $E=-2$: il peso deve essere decrementato in modo che $f(x, w)$ decresca

Perceptron Learning - Esempio

$$w^{(k+1)} = w^{(k)} + \lambda [y_i - f(w^{(k)}, x_i)] x_i$$

$$Y = sign(\sum_{i=0}^d w_i X_i)$$

$$\lambda = 0.1$$

X ₁	X ₂	X ₃	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	w ₀	w ₁	w ₂	w ₃
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

Epoch	w ₀	w ₁	w ₂	w ₃
0	0	0	0	0
1	-0.2	0	0.2	0.2
2	-0.2	0	0.4	0.2
3	-0.4	0	0.4	0.2
4	-0.4	0.2	0.4	0.4
5	-0.6	0.2	0.4	0.2
6	-0.6	0.4	0.4	0.2

Classificazione e ANN

- Problema di apprendimento:
 - dati insiemi di punti su uno spazio n-dimensionale, classificarli in due gruppi (positivi e negativi)
 - inoltre dato un nuovo punto x' decidere a quale gruppo appartiene
- Il primo problema è di **classificazione**, mentre per risolvere il secondo è richiesta capacità di **generalizzazione**, come l'apprendimento di concetti

Problema di classificazione

- Il problema è quindi ridotto alla determinazione dell'insieme dei pesi (w_0, w_1, \dots, w_n) migliore per minimizzare gli errori di classificazione
- Quindi lo spazio delle ipotesi H è infinito ed è dato da tutte le possibili assegnazioni di valori agli $n+1$ pesi (w_0, w_1, \dots, w_n) :
- Si tratta di ottimizzare la funzione

$$o(x) = sign(w \cdot x)$$

Esempio per due attributi

- Con $x = (x_1, x_2)$ in ingresso, si ha:

$$o(x) = \text{sign}(w_0 + w_1 x_1 + w_2 x_2)$$

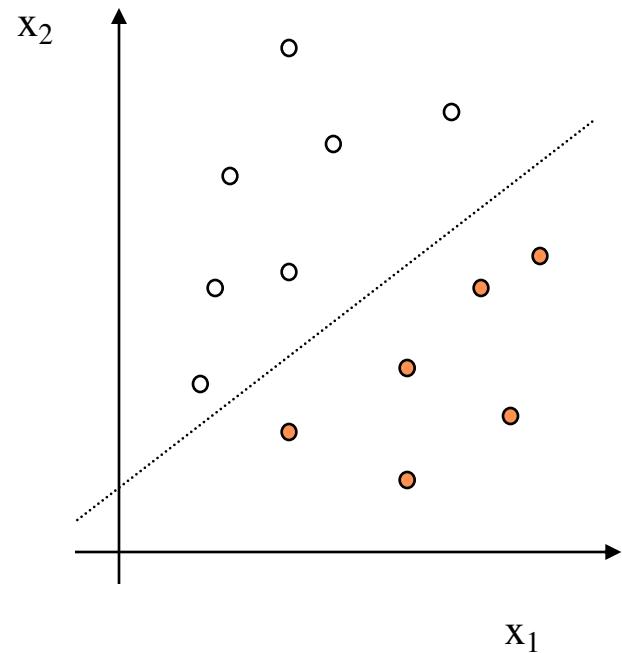
- La retta di separazione è data da:

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$

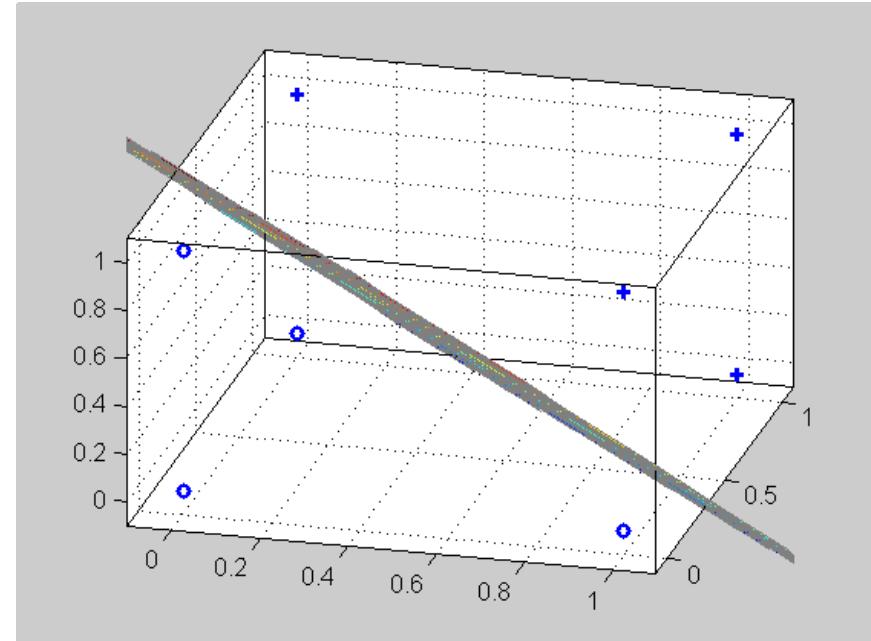
- Nel caso di n attributi, quel che si apprende è un **iperpiano di separazione** dato da:

$$w \cdot x + w_0 = 0$$



Addestramento del Percettrone (learning)

- Poiché $f(w, x)$ è una combinazione lineare di variabili di input, la separazione è definita da un iperpiano
- Per i problemi separabili non linearmente, l'algoritmo di apprendimento del percettrone fallirà perché nessun iperpiano lineare può separare perfettamente i dati



Funzioni rappresentabili con il percettrone

- I percetroni possono rappresentare tutte le funzioni booleane primitive AND, OR, NAND e NOR
 - $AND(x_1, x_2) = sign(-0.8 + 0.5x_1 + 0.5x_2)$
 - $OR(x_1, x_2) = sign(-0.3 + 0.5x_1 + 0.5x_2)$
 - ecc. (provate a scrivere le altre per esercizio)

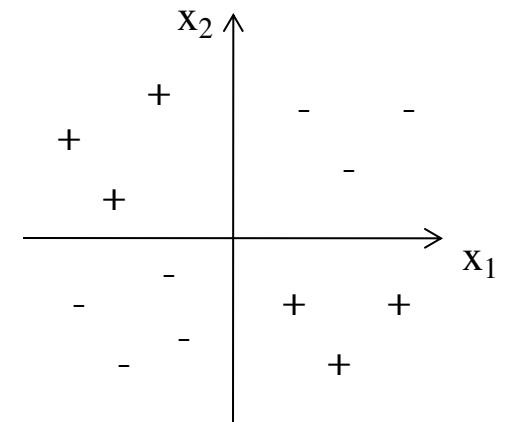
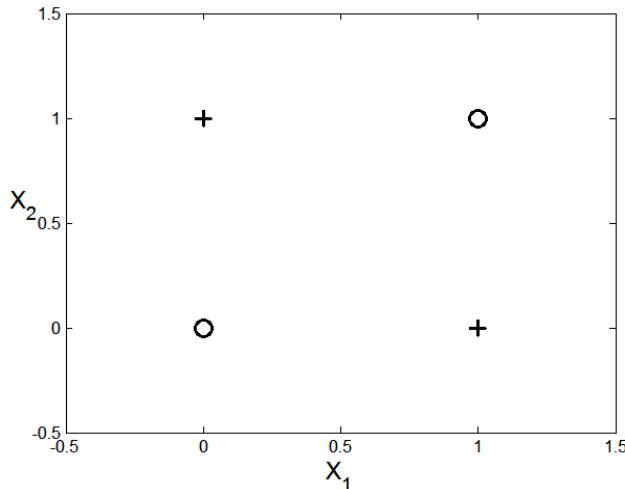
Convergenza del Percettrone

- Il **teorema di convergenza del percettrone** (Rosemblatt, 1962) assicura che il percettrone riuscirà a delimitare le 2 classi se il sistema è *linearmente separabile*
- In altre parole, nell'ottimizzazione non esistono minimi locali

Funzioni non rappresentabili

- Alcune funzioni booleane non possono essere rappresentate dal modello percettrone
 - es. la funzione XOR (vale 1 se e solo se $x_1 \neq x_2$):

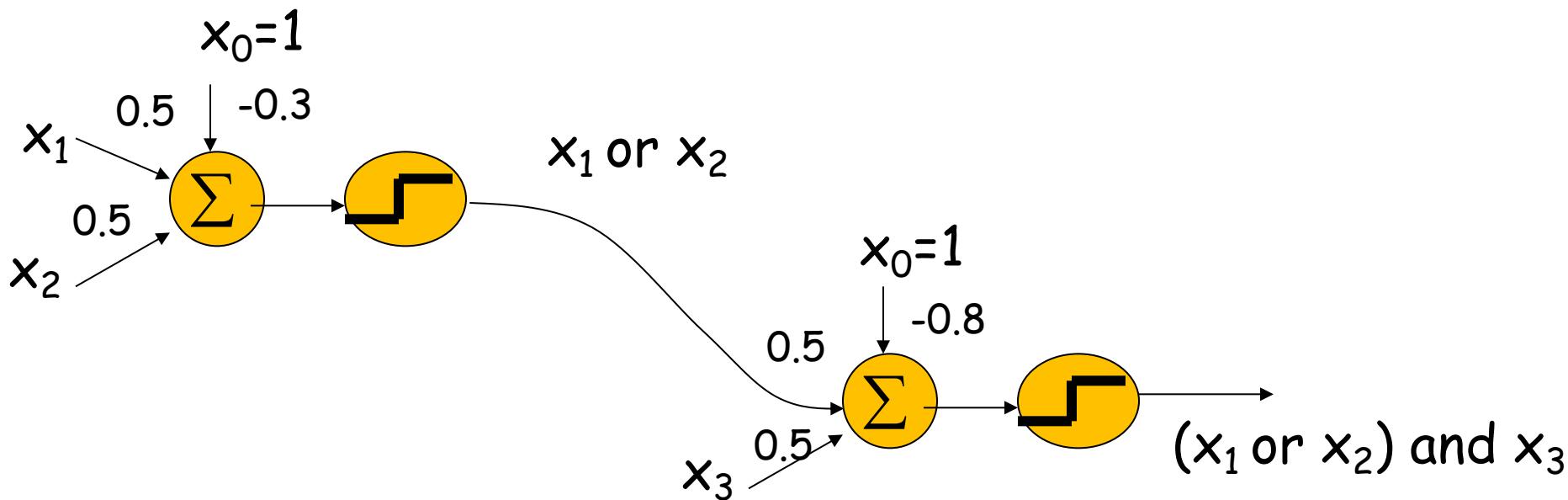
x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1



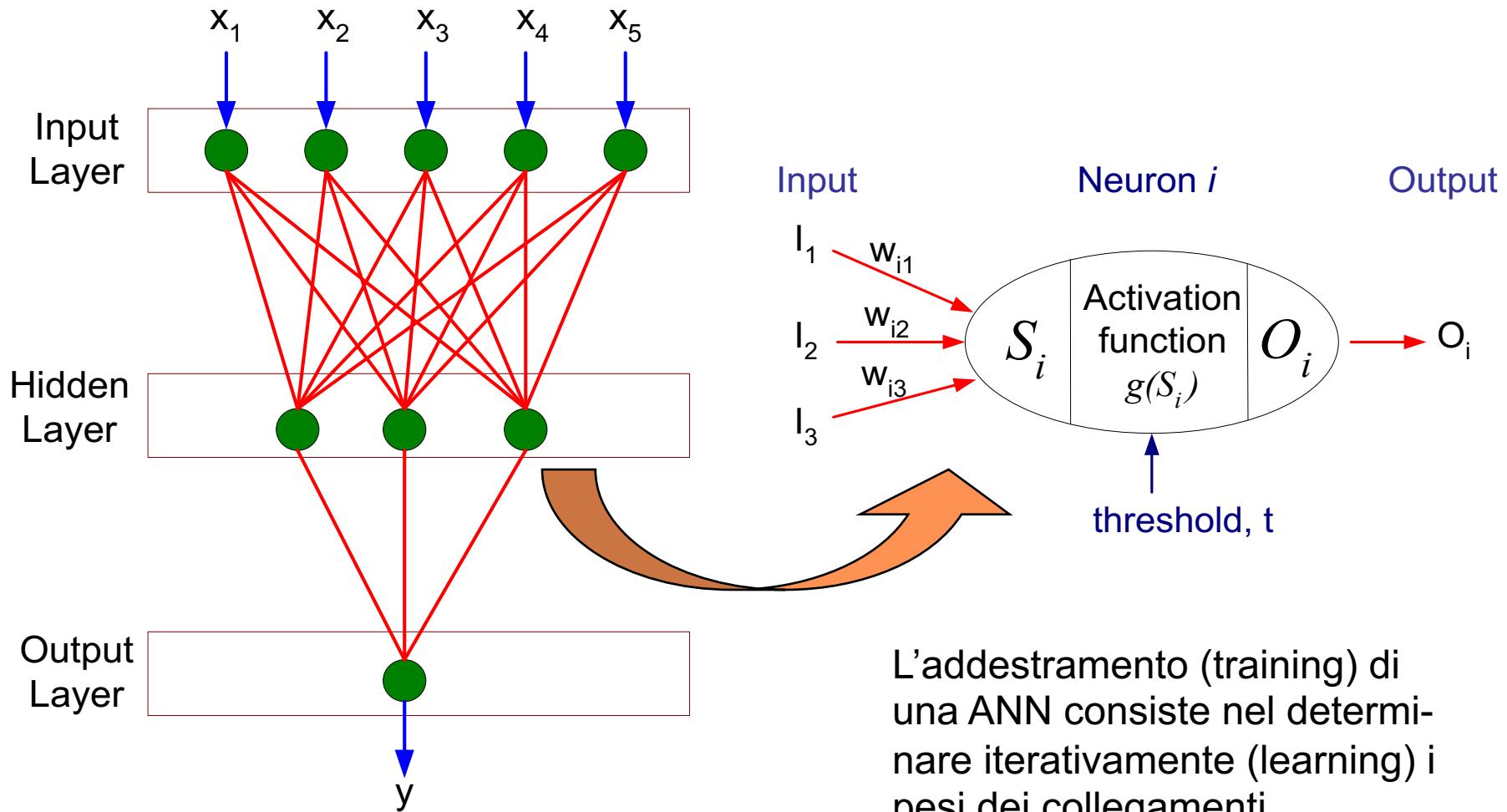
$$y = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$$

Funzioni booleane complesse (multilivello)

- Livelli nascosti
 - Livelli intermedi tra i livelli di ingresso e di uscita
- Funzioni di attivazione più generali (sigmoidea, lineare, etc)



Struttura generale di una ANN

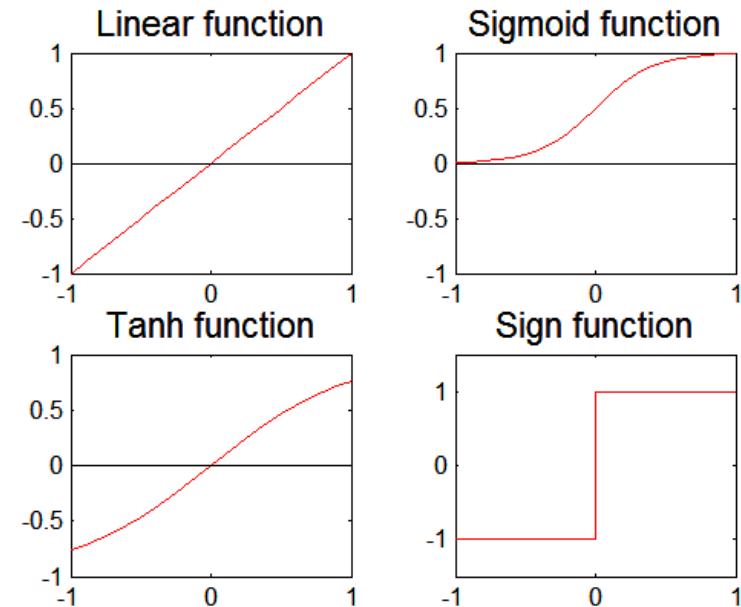


Artificial Neural Networks (ANN)

- Diverse topologie di rete neurale
 - rete a singolo strato (percettrone) contro rete a più strati
 - Feed-forward (rete acicliche) contro rete ricorrente

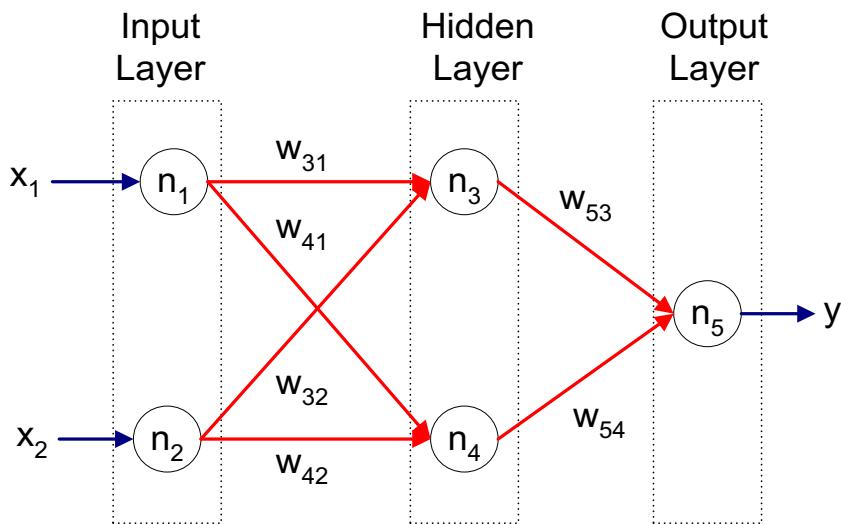
- Diversi tipi di funzione di attivazione (f)

$$Y = f(\sum_i w_i X_i)$$

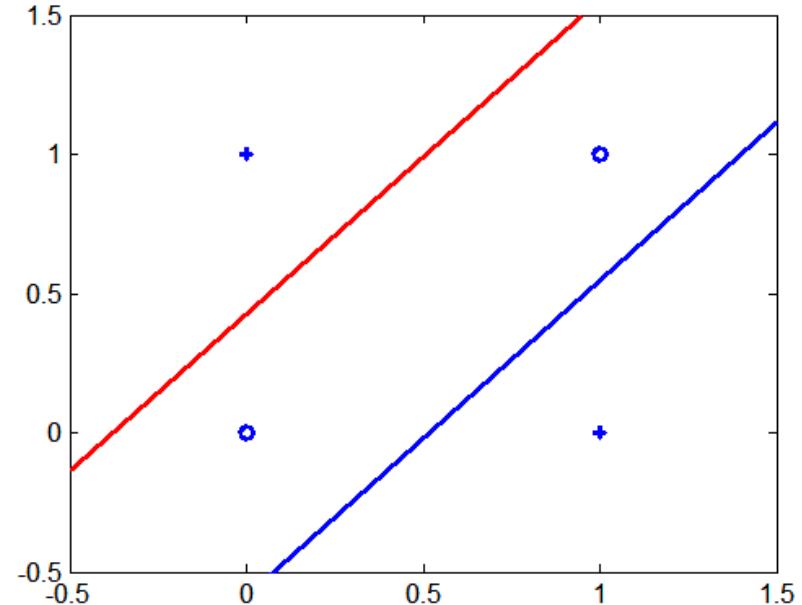


Rete Neurale Multilivello

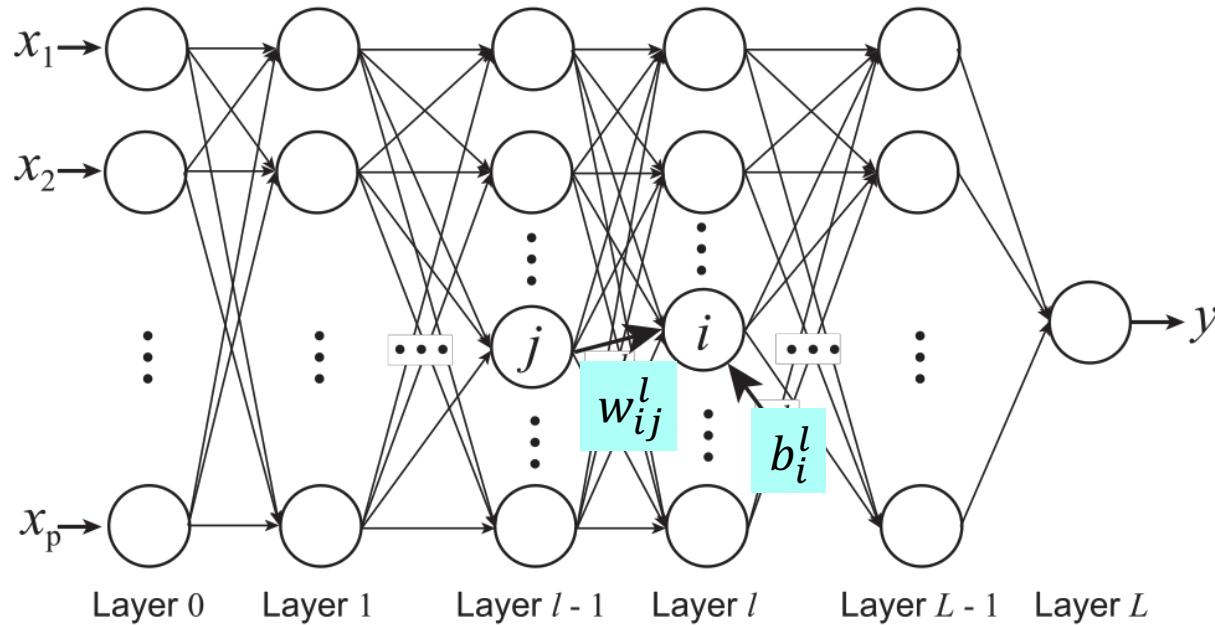
- La rete neurale multilivello può risolvere qualsiasi problema di classificazione che coinvolga superfici decisionali non lineari



XOR Data



Rete multi-livello



$$a_i^l = f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)$$

Activation value at node i at layer l

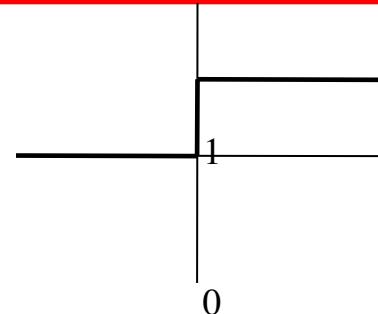
Activation Function

Linear Predictor

Funzioni di attivazione

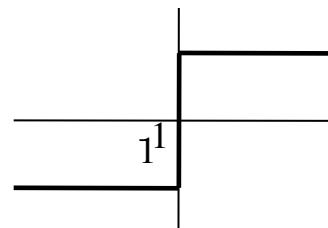
● Funzione gradino

$$gradino_t(x) = \begin{cases} 1 & x > t \\ 0 & altrimenti \end{cases}$$



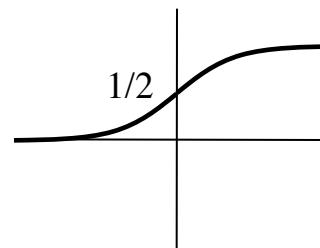
● Funzione segno

$$sign(x) = \begin{cases} +1 & x \geq 0 \\ -1 & altrimenti \end{cases}$$



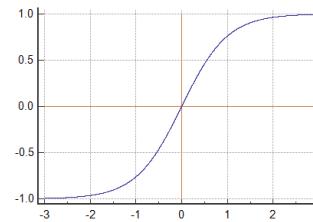
● Funzione sigmoidea

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

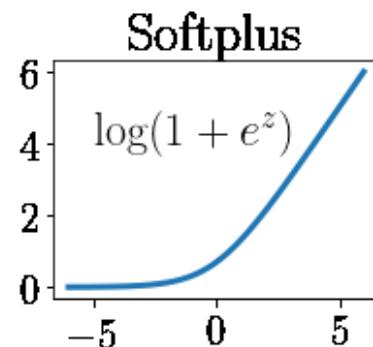
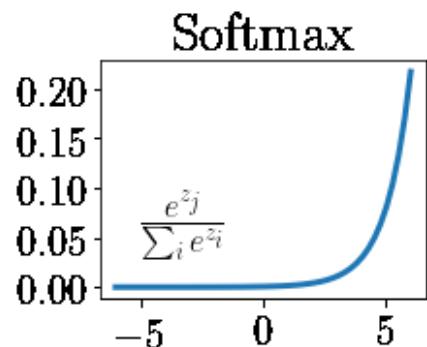
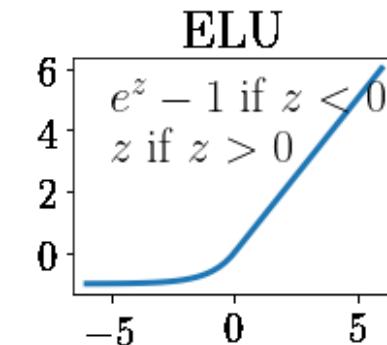
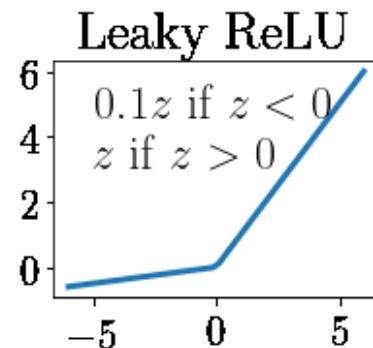
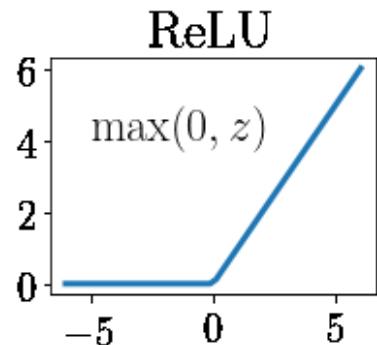
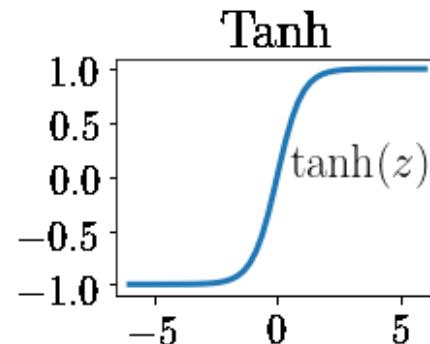
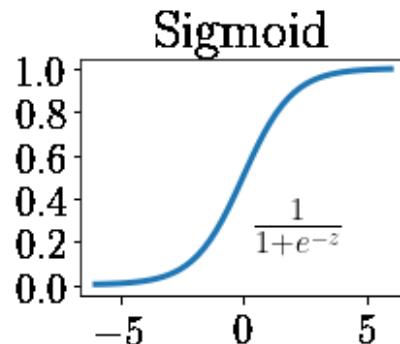
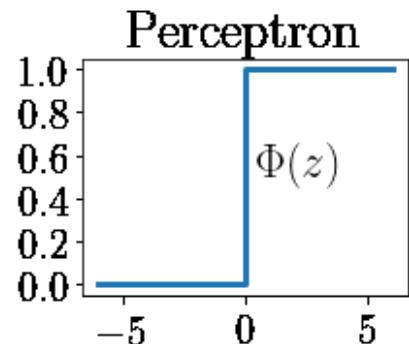


● Funzione tang. Iperbolica

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

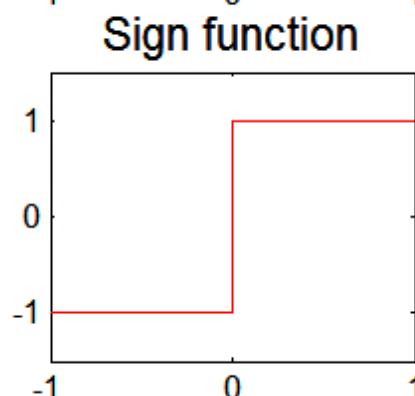
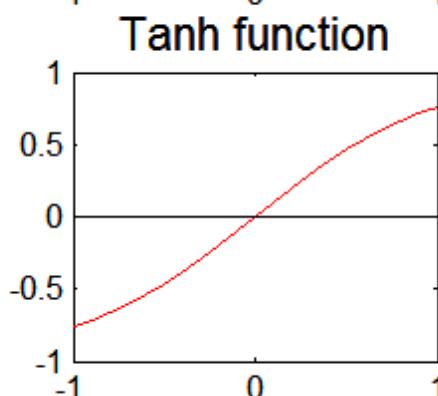
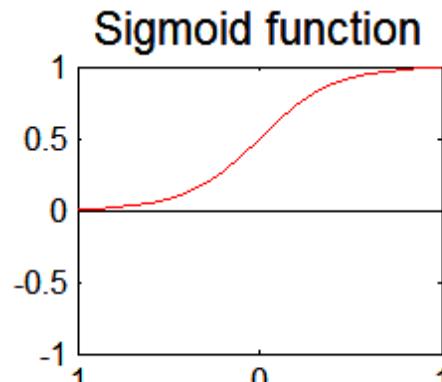
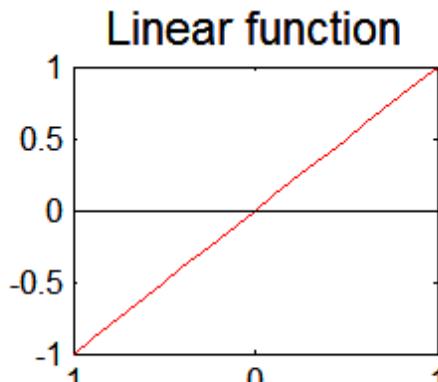


Funzioni di attivazione



Funzioni di attivazione e derivata

$$a_i^l = f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)$$



$$a_i^l = \sigma(z_i^l) = \frac{1}{1 + e^{-z_i^l}}$$

$$\frac{\partial a_i^l}{\partial z_i^l} = \frac{\partial \sigma(z_i^l)}{\partial z_i^l} = a_i^l(1 - a_i^l)$$

$$a_i^l = \tau(z_i^l) = \frac{e^{z_i^l} - e^{-z_i^l}}{e^{z_i^l} + e^{-z_i^l}}$$

$$\frac{\partial a_i^l}{\partial z_i^l} = \frac{\partial \tau(z_i^l)}{\partial z_i^l} = 1 + a_i^l$$

Addestramento di Reti Neurali Multilivello

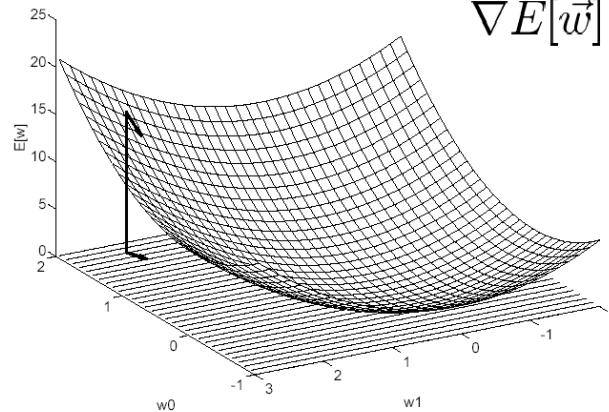
- Possiamo applicare la regola di apprendimento del percettrone a tutti i nodi, inclusi i nodi nascosti?
- La regola di apprendimento del percettrone calcola l'errore $e = y - f(w, x)$ e aggiorna i pesi di conseguenza
 - ◆ Problema:
 - ◆ Come determinare il vero valore di y per i nodi nascosti?
- Approssima l'errore dei nodi nascosti mediante l'errore dei nodi di output
 - ◆ Problemi:
 - ◆ Non è chiaro in che modo la regolazione nei nodi nascosti influisce sull'errore generale
 - ◆ Nessuna garanzia di convergenza alla soluzione ottimale

Convergenza di ANN multilivello

- Problema: come ci si comporta in caso di situazioni **non linearmente separabili**?
- Soluzioni
 - Utilizzare la **discesa del gradiente (delta rule)** invece della regola di addestramento del percettrone
 - **reti multistrato feed forward o reti ricorrenti**

Discesa del gradiente

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$



ANN Multilivello con Gradiente Decrescente

- Aggiornamento dei pesi: $w_j^{(k+1)} = w_j^{(k)} - \lambda \frac{\partial E}{\partial w_j}$
- Errore:
$$E = \frac{1}{2} \sum_{i=1}^N \left(t_i - f\left(\sum_j w_j x_{ij}\right) \right)$$
- La funzione di attivazione f deve essere differenziabile

Discesa del gradiente (Gradient descent)

- Loss Function: misura gli errori nei nodi:

- $E(w) = \sum_{k=1}^n Loss(y_k - \hat{y}_k)$

- Errore quadratico (squared error):

- $Loss(y_k - \hat{y}_k) = (y_k - \hat{y}_k)^2$

oppure

- $Loss(y_k - \hat{y}_k) = \frac{(y_k - \hat{y}_k)^2}{2}$

- Gradient descent: modifica I pesi in modo da minimizzare l'errore:

- $w_{ij}^l \leftarrow w_{ij}^l - \lambda \frac{\partial E}{\partial w_{ij}^l},$

dove $\lambda = learning rate$, $b_i^l = w_{i0}^l$

- La funzione di attivazione f deve essere differenziabile

Calcolo dei gradienti

$$\frac{\partial E}{\partial w_{ij}^l} = \sum_{k=1}^n \frac{\partial \text{Loss}(y_k, \hat{y}_k)}{\partial w_{ij}^l}.$$

$$\begin{aligned}\hat{y} &= a^L \\ a_i^l &= f(z_i^l) = f\left(\sum_j w_{ij}^l a_j^{l-1} + b_i^l\right)\end{aligned}$$

- Ciascun elemento può essere riscritto come (chain rule):

$$\frac{\partial \text{Loss}}{\partial w_{ij}^l} = \frac{\partial \text{Loss}}{\partial a_i^l} \times \frac{\partial a_i^l}{\partial z_i^l} \times \frac{\partial z_i^l}{\partial w_{ij}^l}.$$

- Per la funzione sigmoidea:

$$\frac{\partial \text{Loss}}{\partial w_{ij}^l} = \delta_i^l \times a_i^l (1 - a_i^l) \times a_j^{l-1},$$

$$\text{where } \delta_i^l = \frac{\partial \text{Loss}}{\partial a_i^l}.$$

- Come calcolare questi valori per tutti i livelli?

Backpropagation

- Per il livello di output L:

$$\delta^L = \frac{\partial \text{Loss}}{\partial a^L} = \frac{\partial (y - a^L)^2}{\partial a^L} = 2(a^L - y).$$

- Per un livello “hidden” l (usando la chain rule):

$$\delta_j^l = \sum_i (\delta_i^{l+1} \times a_i^{l+1} (1 - a_i^{l+1}) \times w_{ij}^{l+1}).$$

- I gradienti al livello l sono calcolati usando i gradienti al livello $l + 1$
- Inizia dal livello L e propaga all’indietro (“backpropagation”) i gradienti al livello precedente
- Usa la discesa del gradiente per modificare i pesi a ciascuna epoch
- Per l’epoch successive usa i pesi modificati per calcolare la funzione Loss e il gradiente.
- Itera fino a convergenza (loss non cambia) o per un numero fissato di epoch

Backpropagation (esempio)

INPUT

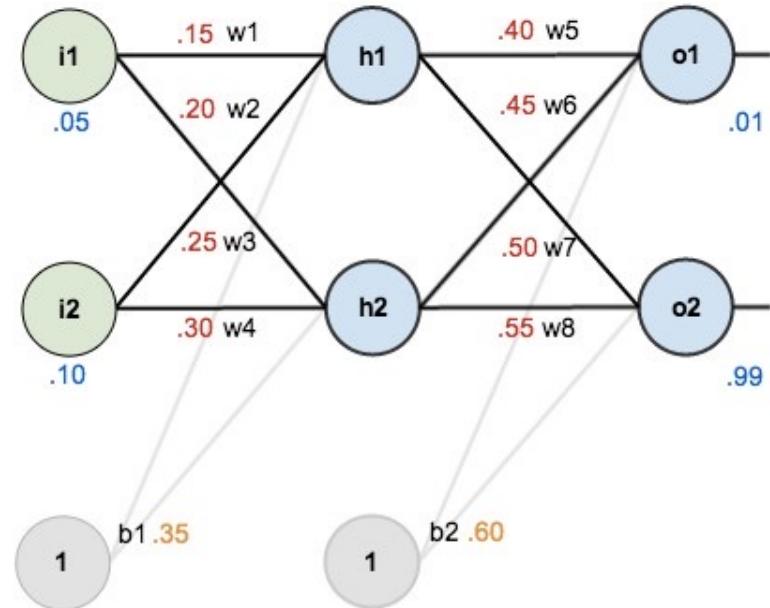
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



1. Forward step – Calcolo errore totale

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

Backpropagation (esempio)

INPUT

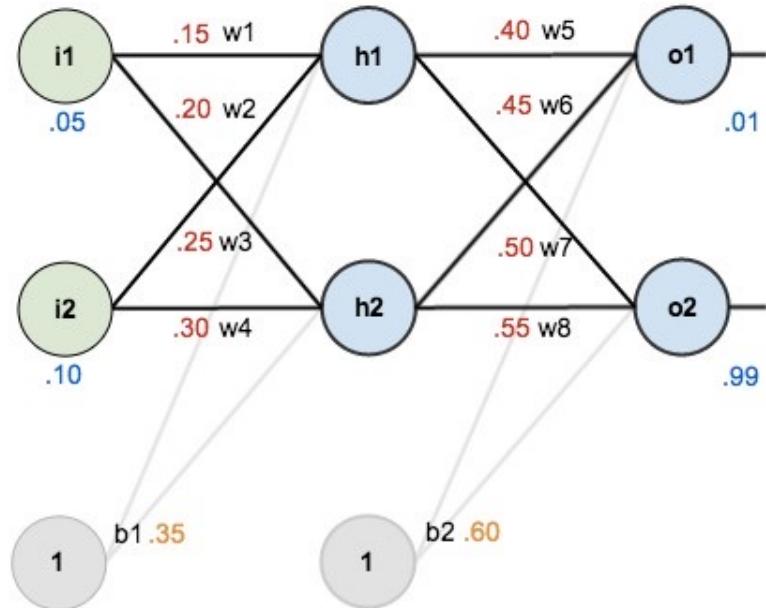
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



1. Forward step – Calcolo errore totale

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

Backpropagation (esempio)

INPUT

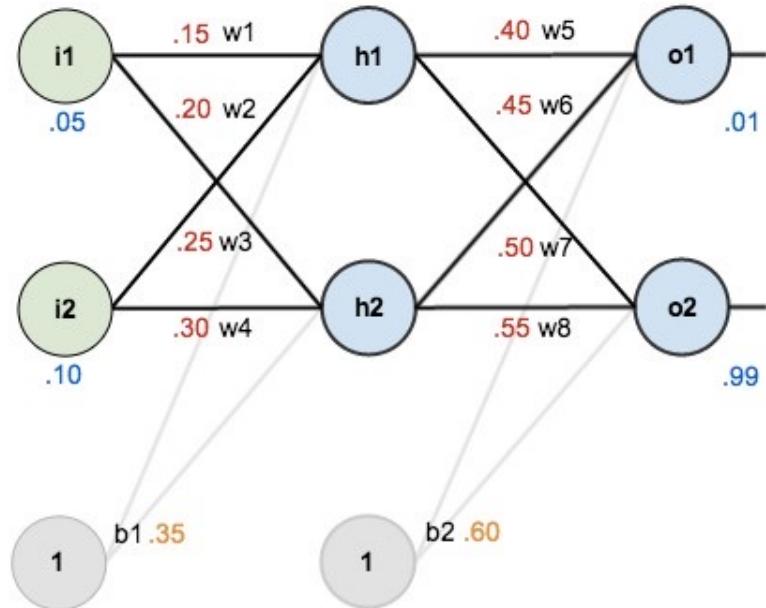
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$

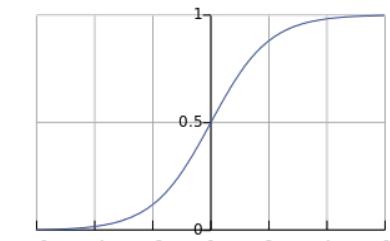


1. Forward step – Calcolo errore totale

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.5933$$



$$f(x) = \frac{1}{1 + e^{-x}}$$

Backpropagation (esempio)

INPUT

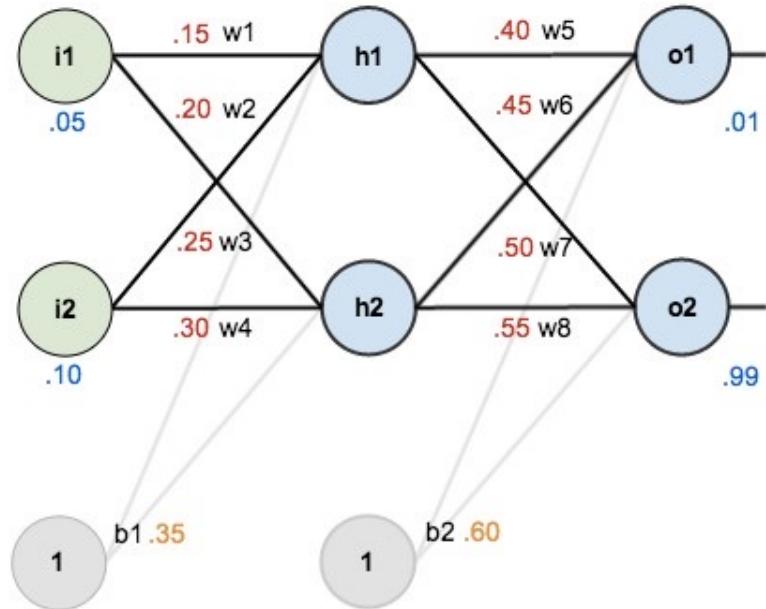
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



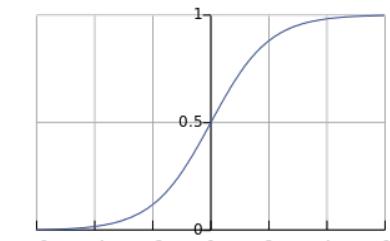
1. Forward step – Calcolo errore totale

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.5933$$

Repeat for $h_2 = 0.596$; $o_1 = 0.751$; $o_2 = 0.773$



$$f(x) = \frac{1}{1 + e^{-x}}$$

Backpropagation (esempio)

INPUT

$$i_1 = 0.05$$

$$i_2 = 0.10$$

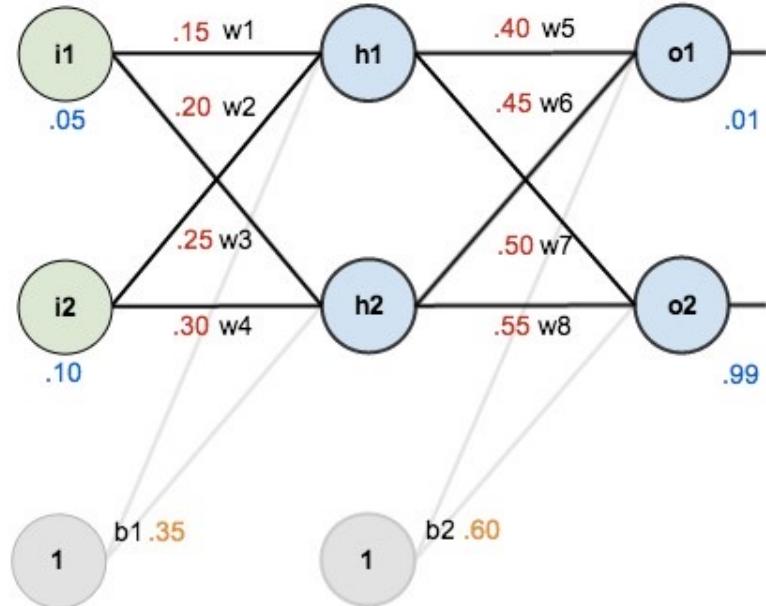
TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$

1. Forward step – Calcolo errore totale

We have o_1, o_2



Backpropagation (esempio)

INPUT

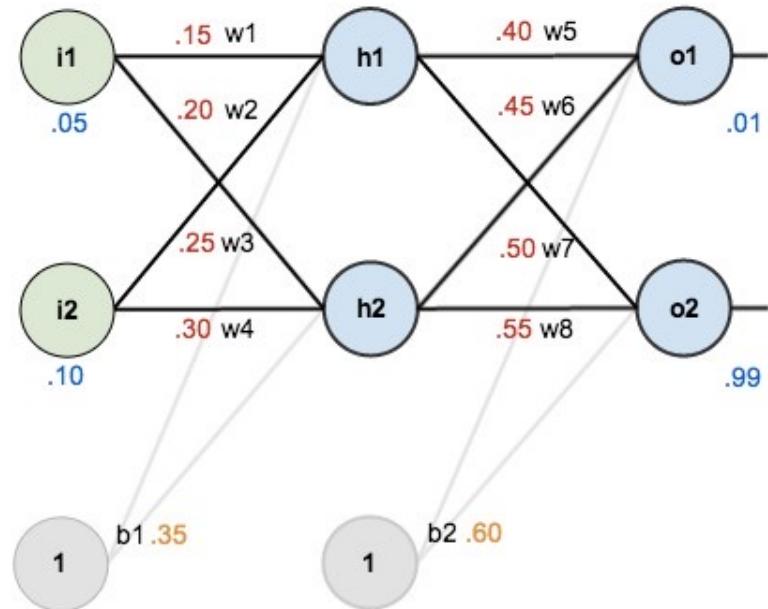
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



1. Forward step – Calcolo errore totale

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

Backpropagation (esempio)

INPUT

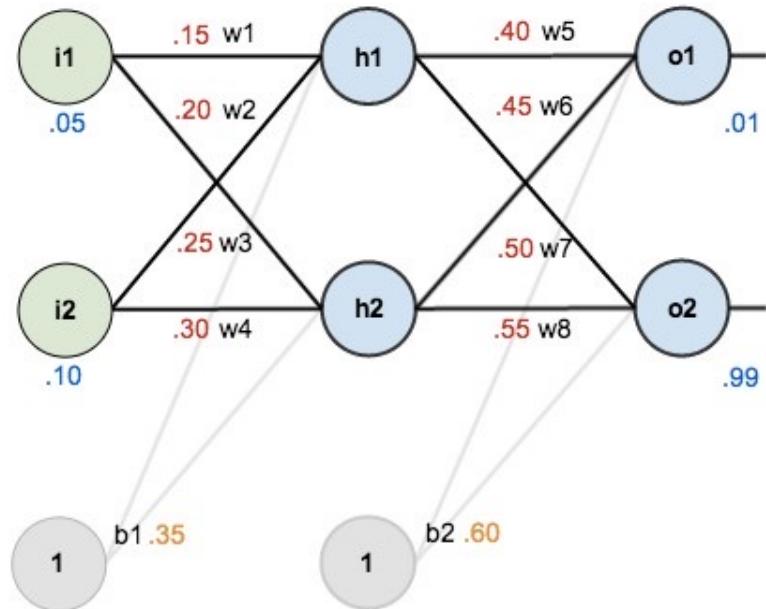
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



1. Forward step – Calcolo errore totale

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

$$E_{o1} = \frac{1}{2} (target_{o1} - out_{o1})^2 = \frac{1}{2} (0.01 - 0.7514)^2 = 0.2748$$

Backpropagation (esempio)

INPUT

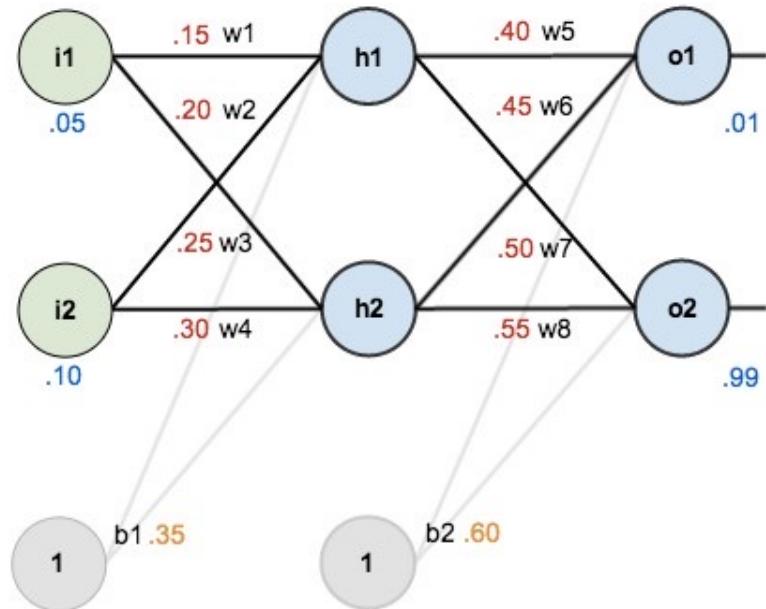
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



1. Forward step – Calcolo errore totale

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

$$E_{o1} = \frac{1}{2} (target_{o1} - out_{o1})^2 = \frac{1}{2} (0.01 - 0.7514)^2 = 0.2748$$

$$E_{o2} = 0.02356$$

Backpropagation (esempio)

INPUT

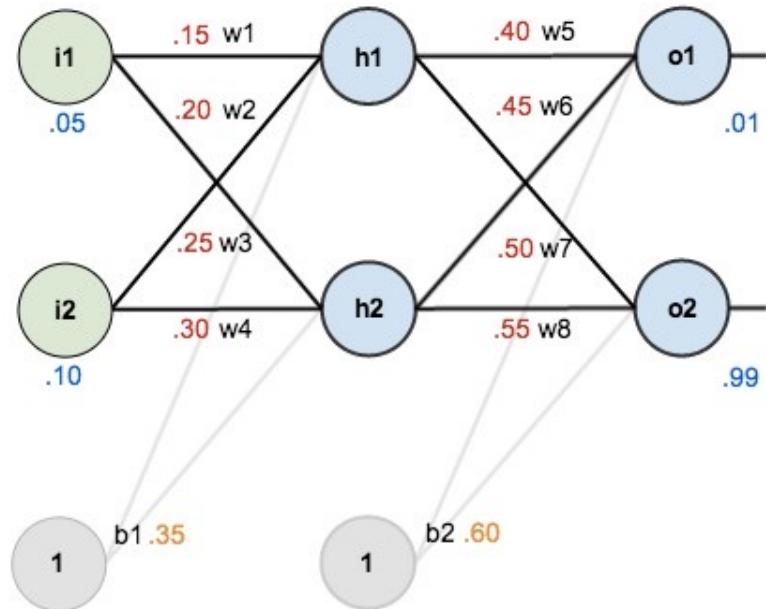
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



1. Forward step – Calcolo errore totale

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

$$E_{o1} = \frac{1}{2} (target_{o1} - out_{o1})^2 = \frac{1}{2} (0.01 - 0.7514)^2 = 0.2748$$

$$E_{o2} = 0.02356$$

$$E_{total} = E_{o1} + E_{o2} = 0.2748 + 0.02356 = 0.29836$$

Backpropagation (esempio)

INPUT

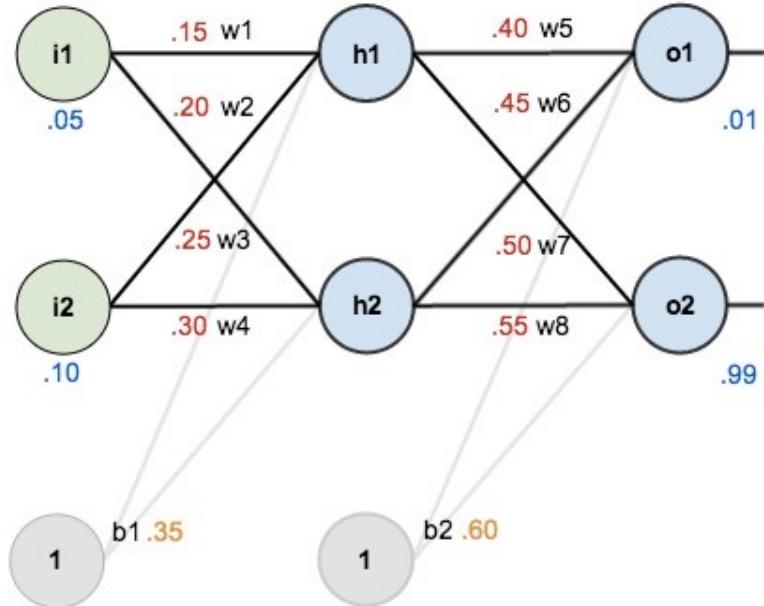
$i_1 = 0.05$

$i_2 = 0.10$

TARGET

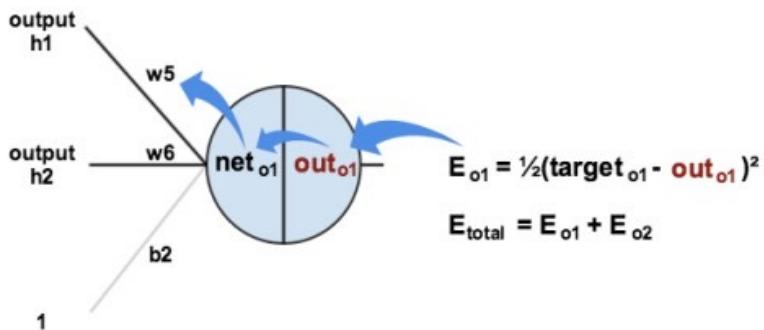
$o_1 = 0.01$

$o_2 = 0.99$



2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?



Backpropagation (esempio)

INPUT

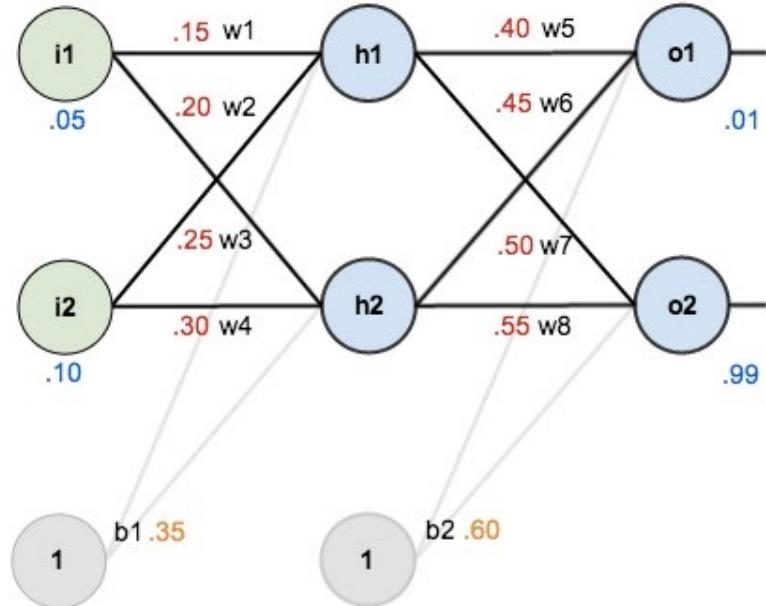
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

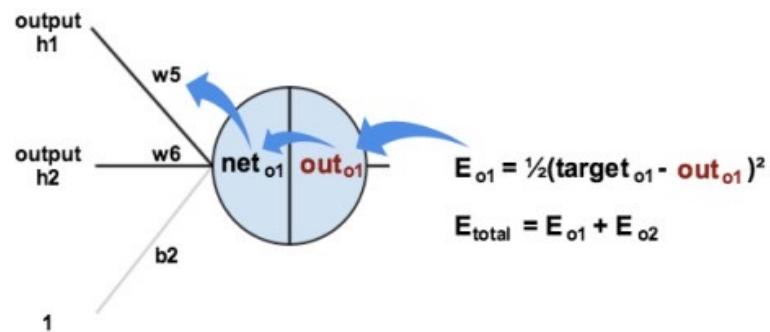
$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$



Backpropagation (esempio)

INPUT

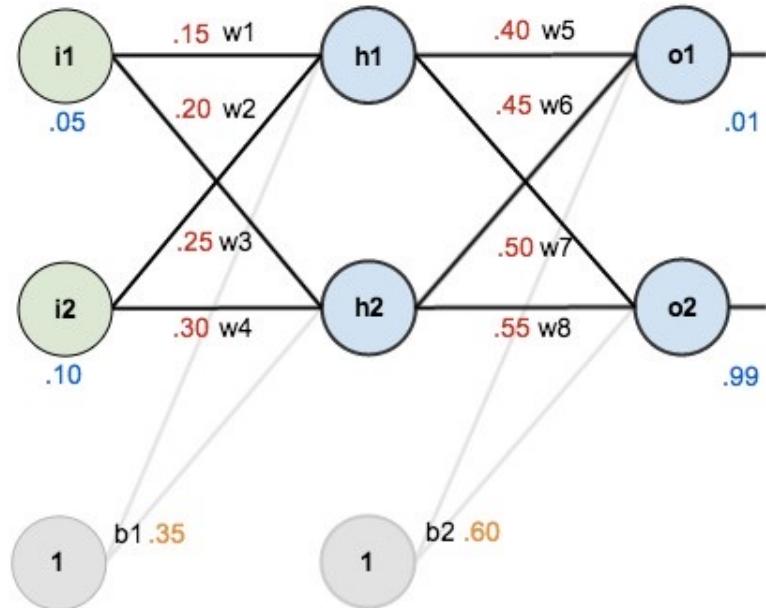
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

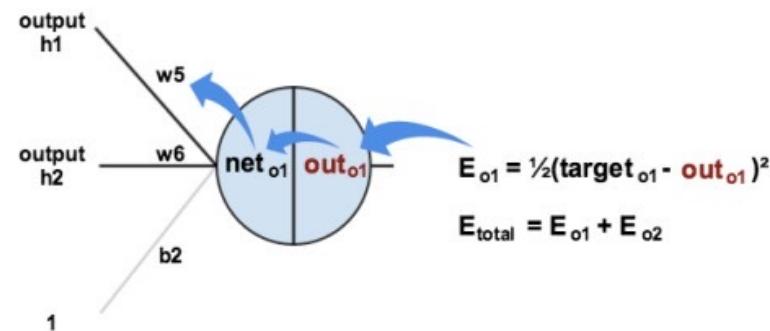
$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \boxed{\frac{\partial E_{total}}{\partial out_{o1}}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$



Backpropagation (esempio)

INPUT

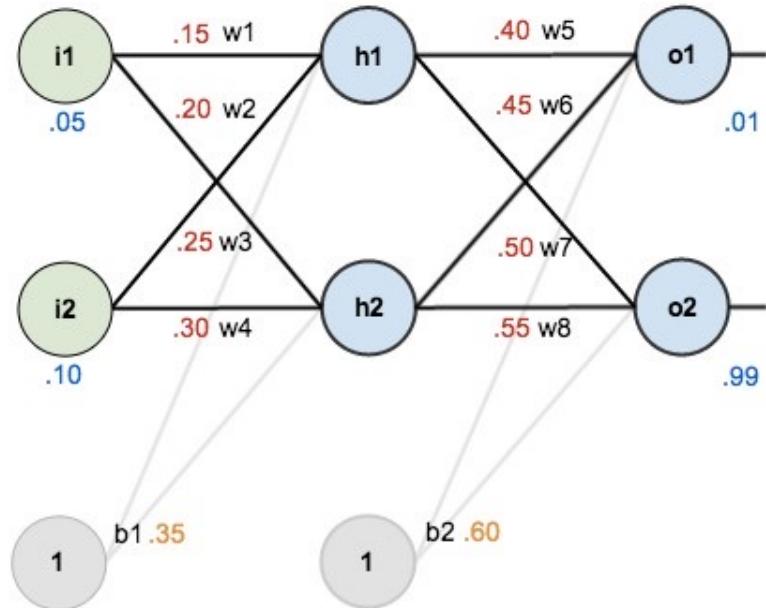
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$

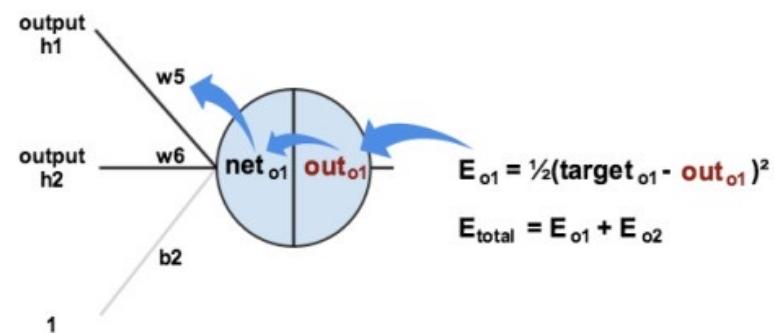


2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \boxed{\frac{\partial E_{total}}{\partial out_{o1}}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$



Backpropagation (esempio)

INPUT

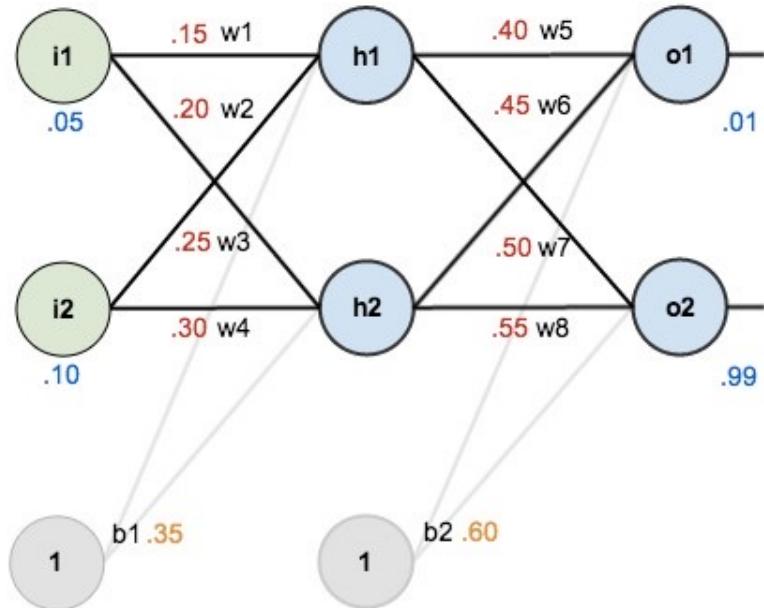
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



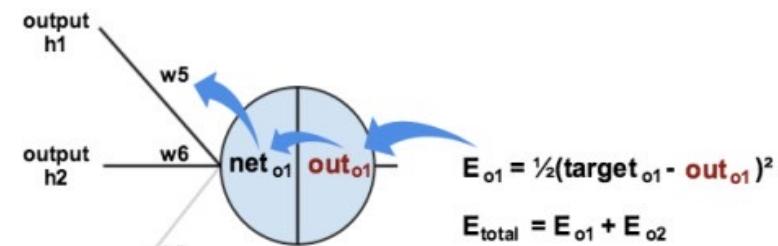
2. Backward step – Modifica pesi

Di quanto w5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \boxed{\frac{\partial E_{total}}{\partial out_{o1}}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2} (target_{o1} - out_{o1}) * -1 + 0 = -(0.01 - 0.751) = 0.741$$



Backpropagation (esempio)

INPUT

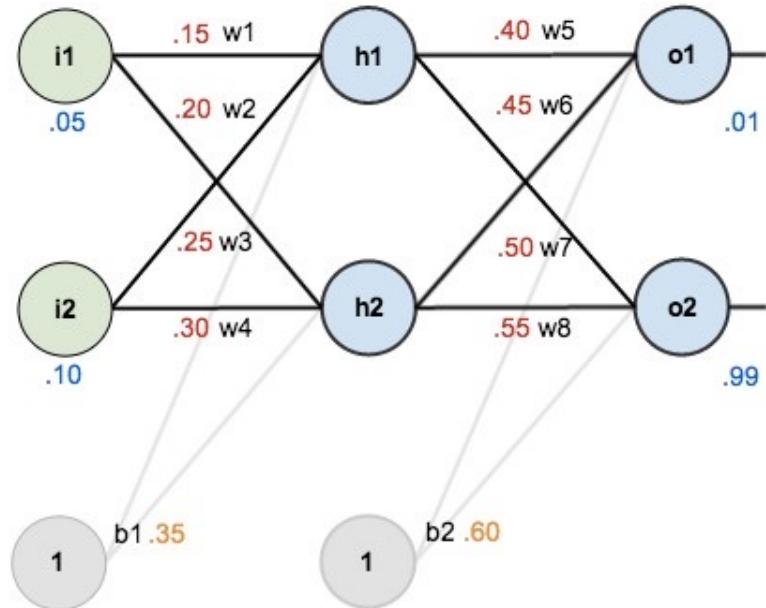
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \boxed{\frac{\partial out_{o1}}{\partial net_{o1}}} * \frac{\partial net_{o1}}{\partial w_5}$$

Backpropagation (esempio)

INPUT

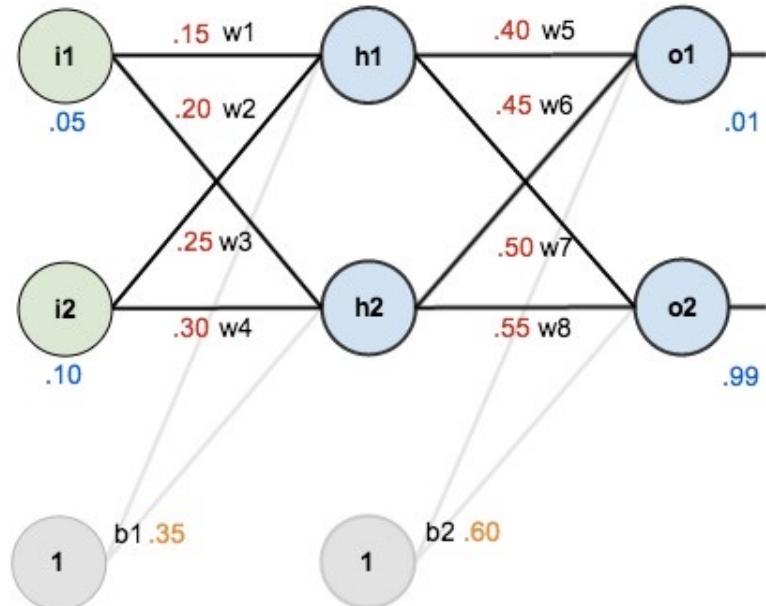
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$
$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

Backpropagation (esempio)

INPUT

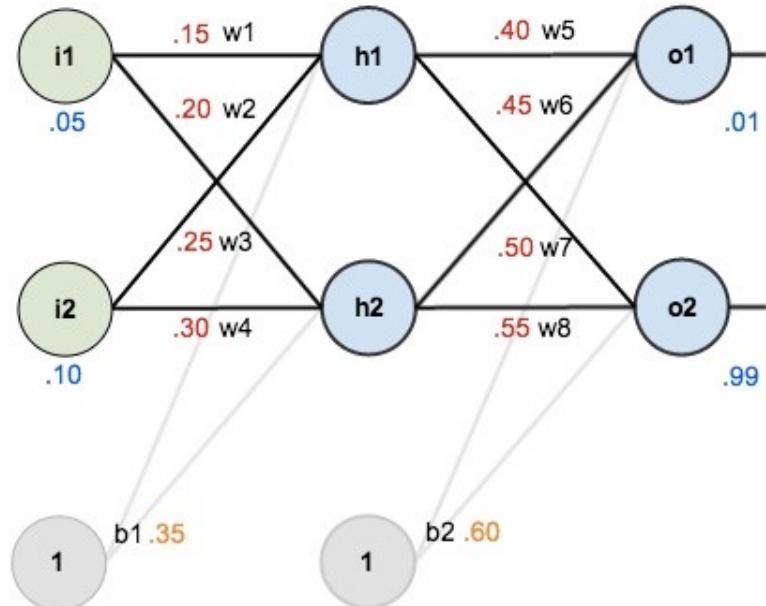
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1} (1 - out_{o1}) = 0.1868$$

Backpropagation (esempio)

INPUT

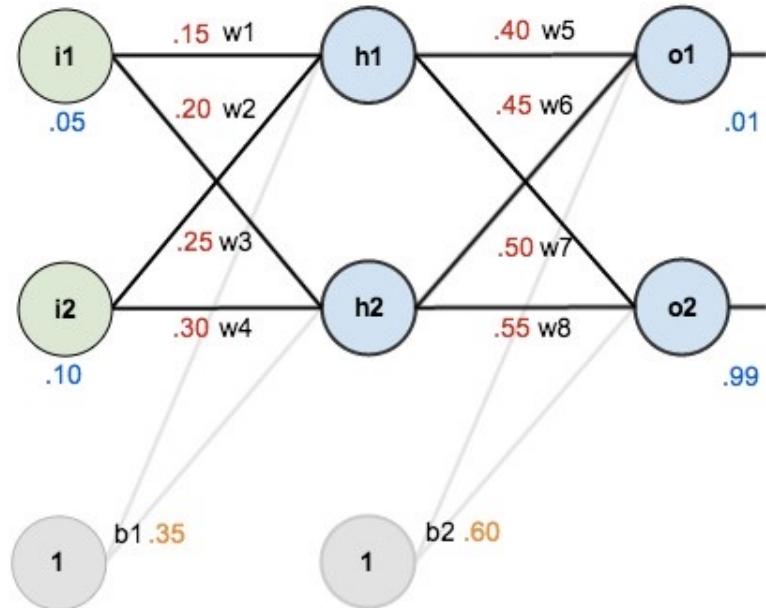
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \boxed{\frac{\partial net_{o1}}{\partial w_5}}$$

Backpropagation (esempio)

INPUT

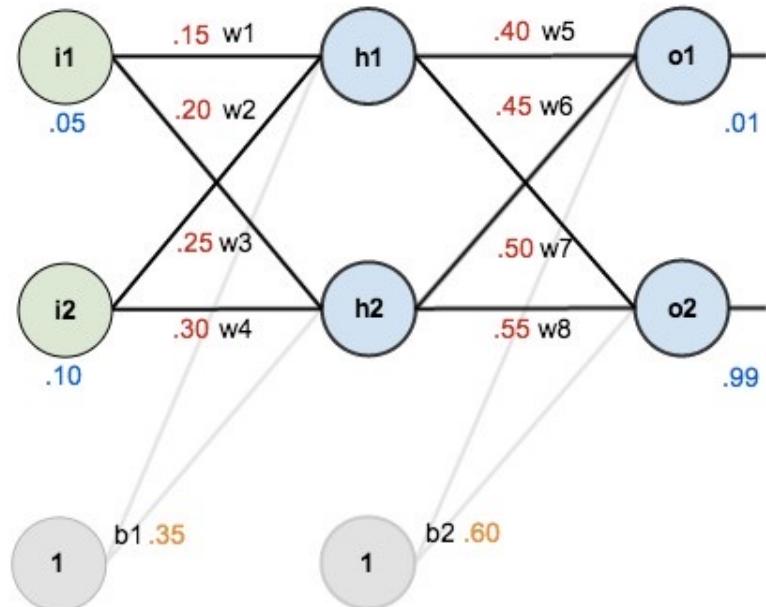
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

Backpropagation (esempio)

INPUT

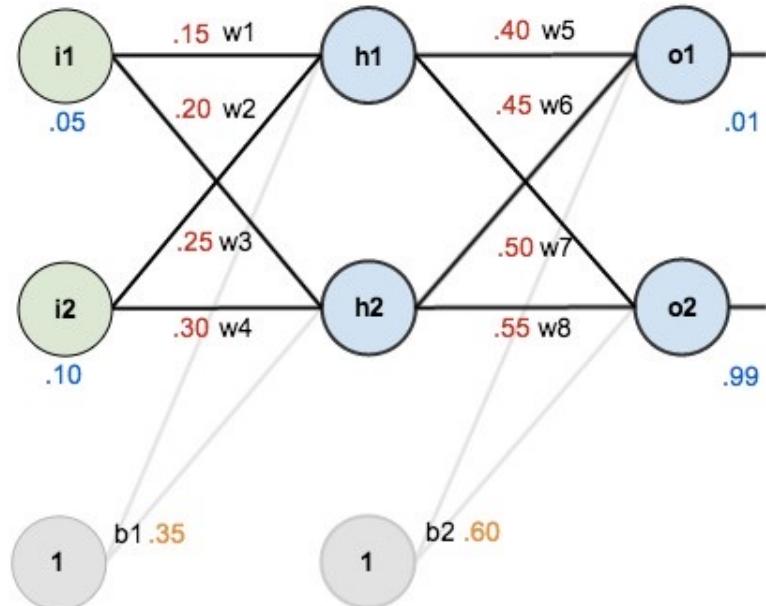
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = out_{h1} = 0.5933$$

Backpropagation (esempio)

INPUT

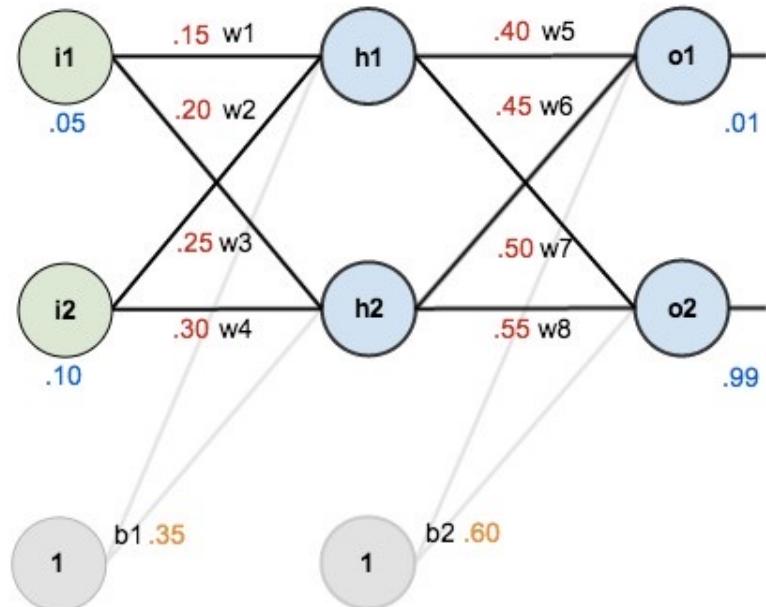
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

Backpropagation (esempio)

INPUT

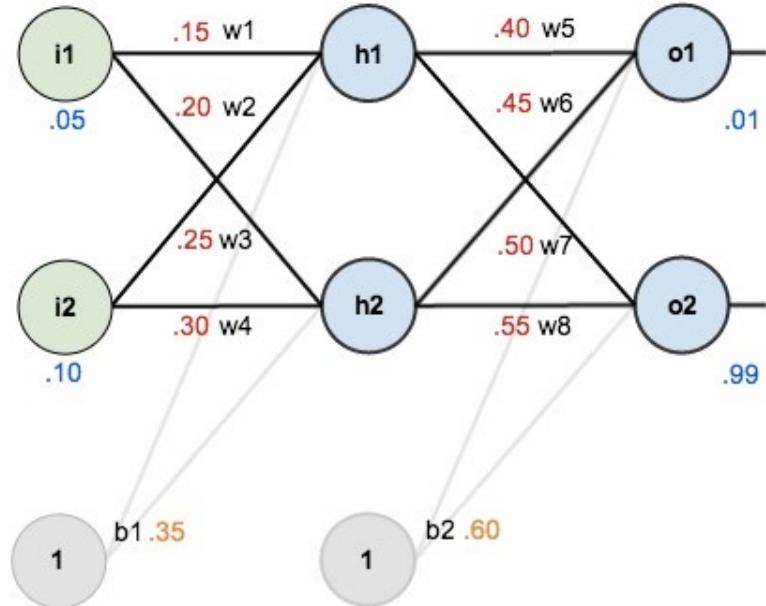
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w_5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.7414 * 0.1868 * 0.5933 = 0.0821$$

Backpropagation (esempio)

INPUT

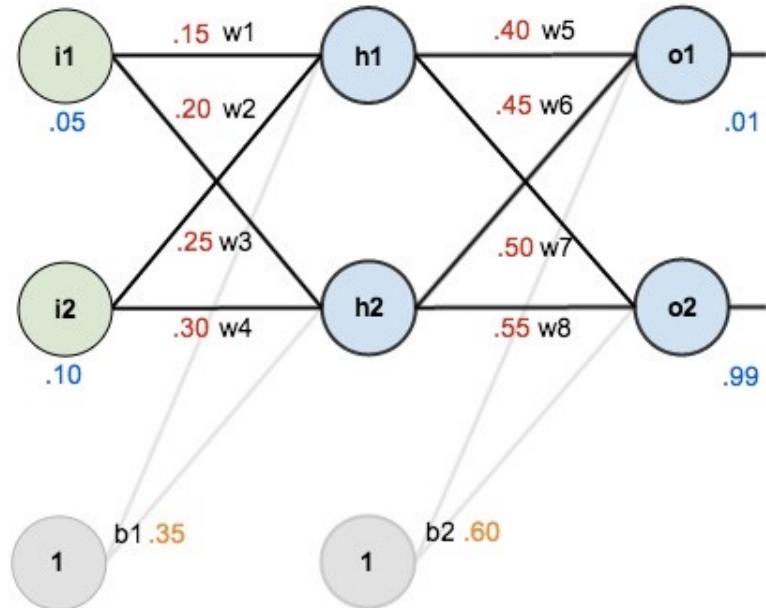
$$i_1 = 0.05$$

$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$



2. Backward step – Modifica pesi

Di quanto w5 influenza l'errore totale?

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.7414 * 0.1868 * 0.5933 = 0.0821$$

$$w_5^{new} = w_5^{old} - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.0821 = 0.3589$$

Backpropagation (esempio)

INPUT

$$i_1 = 0.05$$

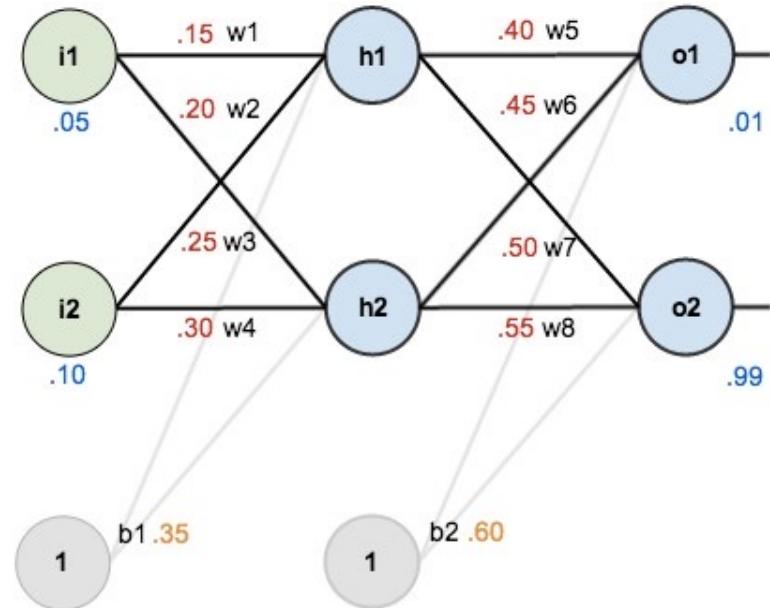
$$i_2 = 0.10$$

TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$

- Ripeti per w_6, w_7, w_8



Backpropagation (esempio)

INPUT

$i_1 = 0.05$

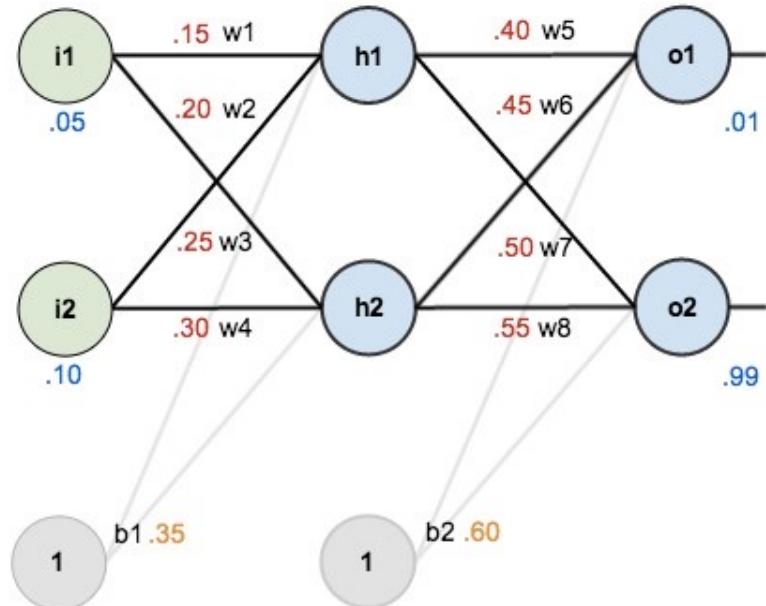
$i_2 = 0.10$

TARGET

$o_1 = 0.01$

$o_2 = 0.99$

- Ripeti per w_6, w_7, w_8
- Analogamente per w_1, w_2, w_3, w_4



Backpropagation (esempio)

INPUT

$$i_1 = 0.05$$

$$i_2 = 0.10$$

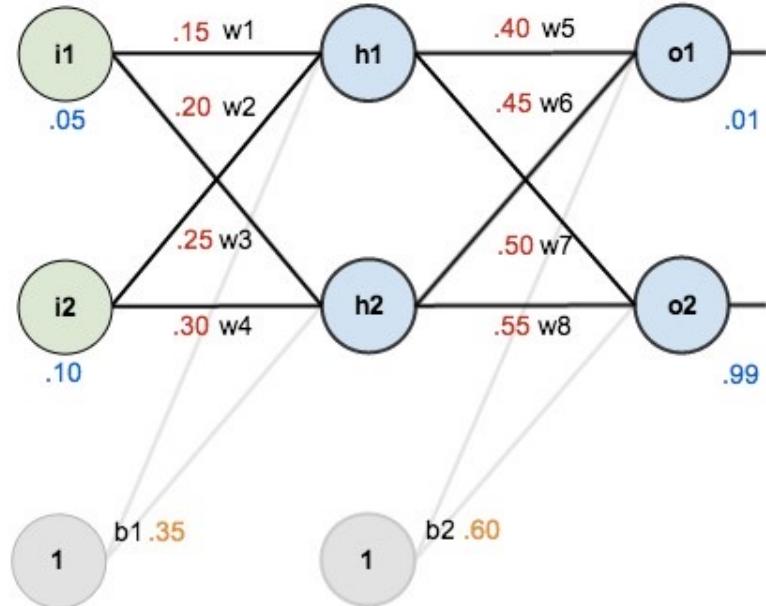
TARGET

$$o_1 = 0.01$$

$$o_2 = 0.99$$

- Ripeti per w_6, w_7, w_8
- Analogamente per w_1, w_2, w_3, w_4
- Calcola l'errore totale prima:

0.298371109



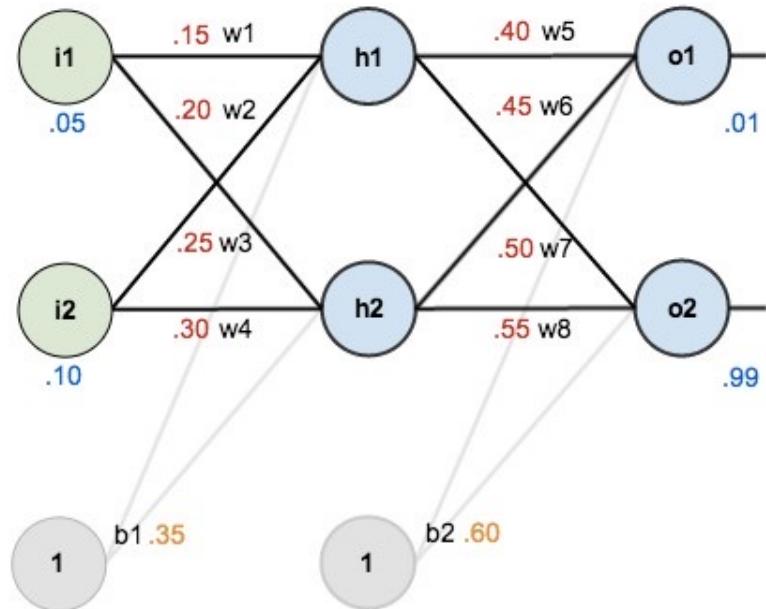
Backpropagation (esempio)

INPUT TARGET

$i_1 = 0.05$ $o_1 = 0.01$

$i_2 = 0.10$ $o_2 = 0.99$

- Ripeti per w_6, w_7, w_8
- Analogamente per w_1, w_2, w_3, w_4
- Errore totale prima dell'update: 0.298371109
- Errore totale dopo l'update: 0.291027924



Backpropagation (esempio)

INPUT TARGET

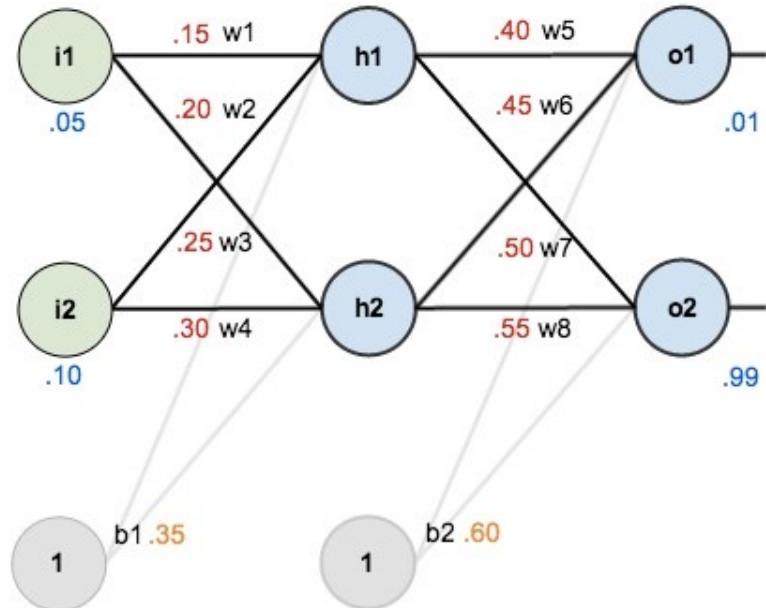
$i_1 = 0.05$

$i_2 = 0.10$

$o_1 = 0.01$

$o_2 = 0.99$

- Ripeti per w_6, w_7, w_8
- Analogamente per w_1, w_2, w_3, w_4
- Errore totale prima dell'update: 0.298371109
- Errore totale dopo l'update: 0.291027924
- Ripeti $\times 10000$: 0.000035085



Progettazione di ANN

- Numero di nodi nel livello di input
 - Un nodo di input per attributo binario / continuo
 - k o $\log_2 k$ nodi per ciascun attributo enumerato con k valori
- Numero di nodi nel livello di output
 - Un nodo di uscita per problemi con classi binarie
 - k o $\log_2 k$ nodi per problemi con k classi
- Numero di nodi nel livello nascosto
- Pesi iniziali

Caratteristiche delle ANN

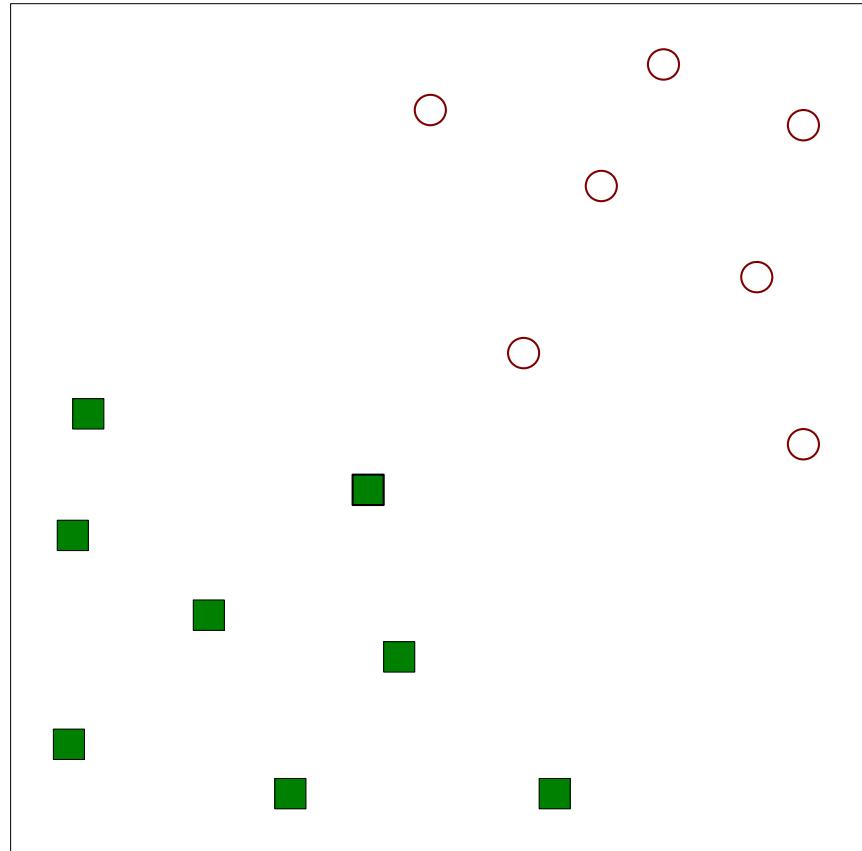
- ANN a più strati sono approssimatori universali ma potrebbero presentare il problema dell'overfitting se la rete è troppo complessa
- La discesa del gradiente può convergere al minimo locale
- La costruzione del modello può richiedere molto tempo, ma il test può essere molto veloce
- Può gestire attributi ridondanti perché i pesi vengono appresi automaticamente
- Sensibile al rumore nei dati del training set
- Difficoltà a gestire attributi mancanti

Sviluppi recenti

- Utilizzate nei problemi di deep learning e unsupervised feature learning
 - Cerca di apprendere automaticamente una buona rappresentazione dell'input da dati non etichettati
- Google Brain project
 - Apprendimento del concetto di gatto analizzando immagini di youtube non etichettate
 - Rete con un miliardo di connessioni

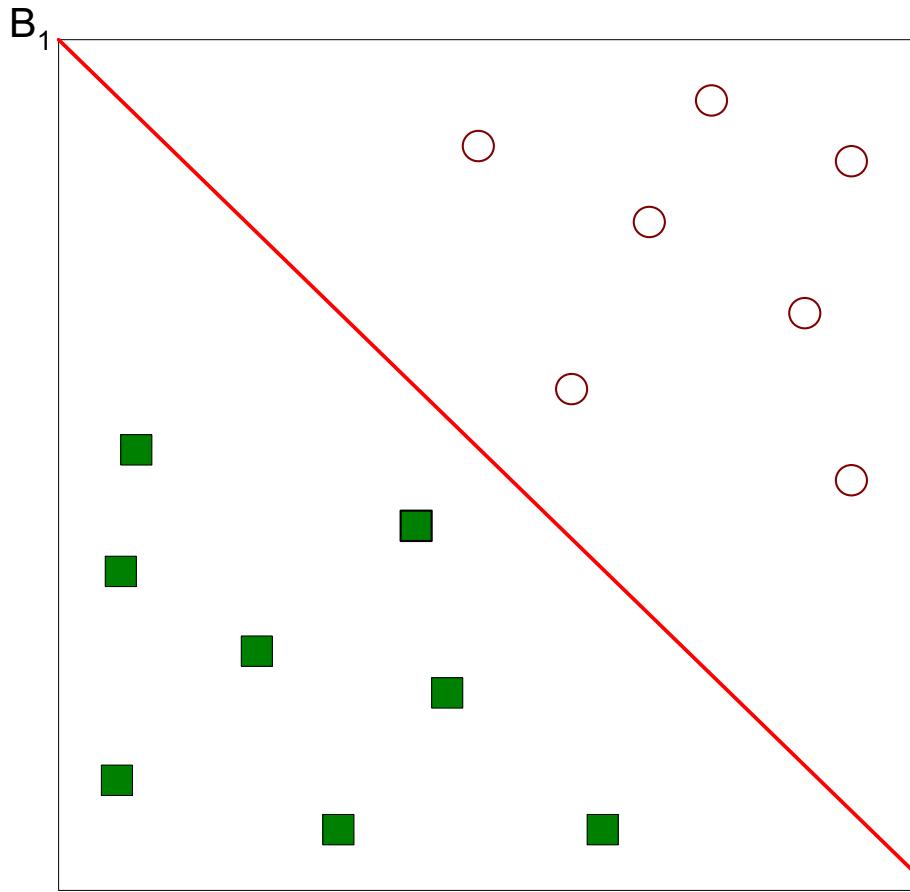
Support Vector Machines

Support Vector Machines



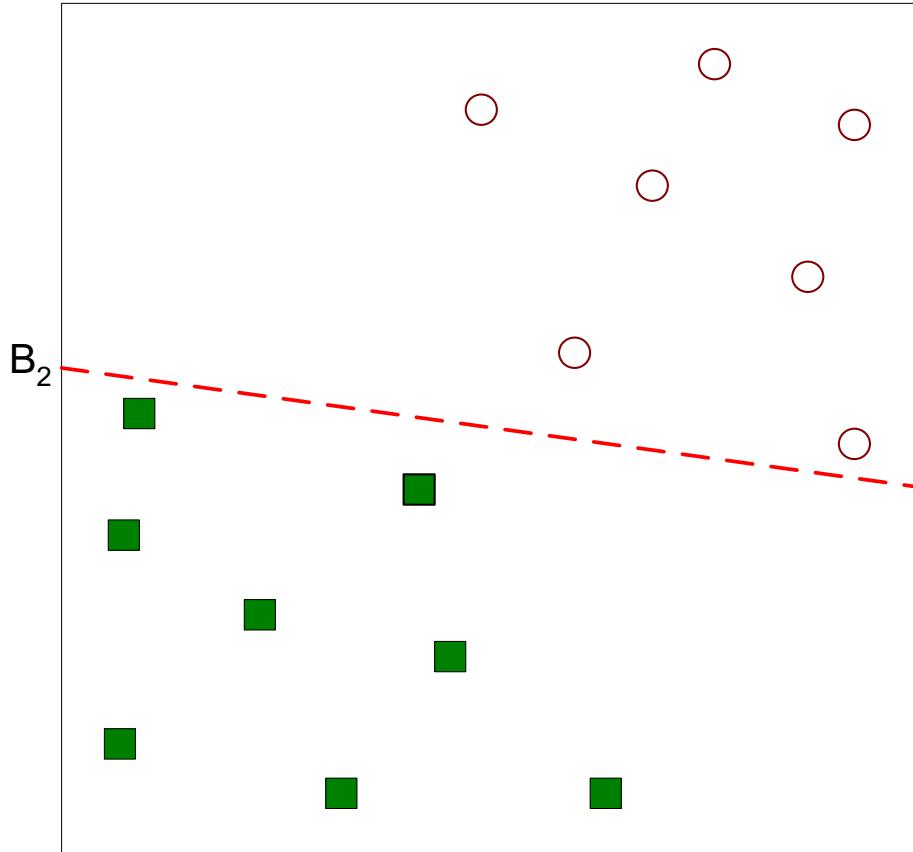
- Trova un iperpiano lineare (confine decisionale) che separerà i dati

Support Vector Machines



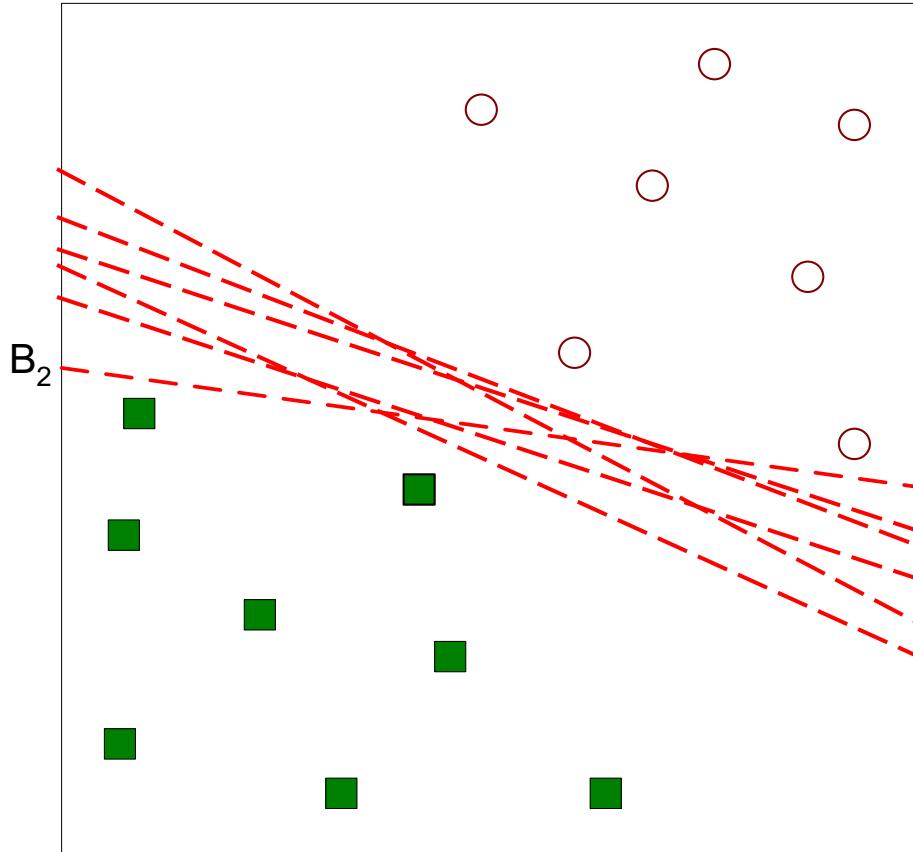
- Una possibile soluzione

Support Vector Machines



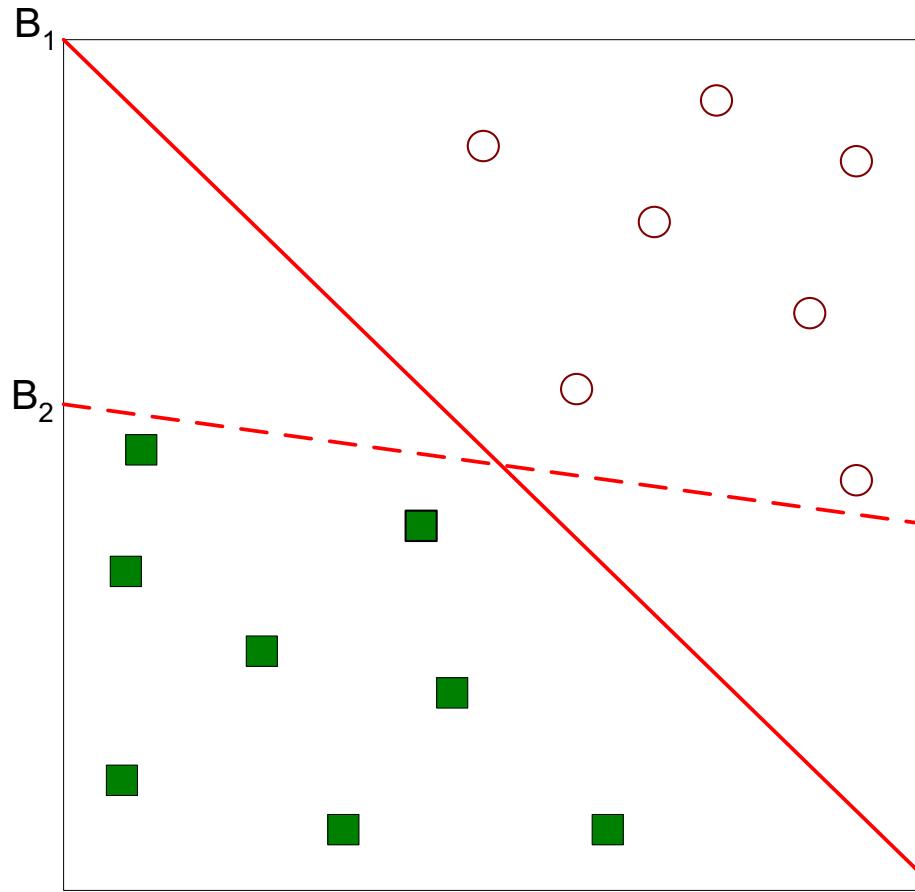
- Un'altra possibile soluzione

Support Vector Machines



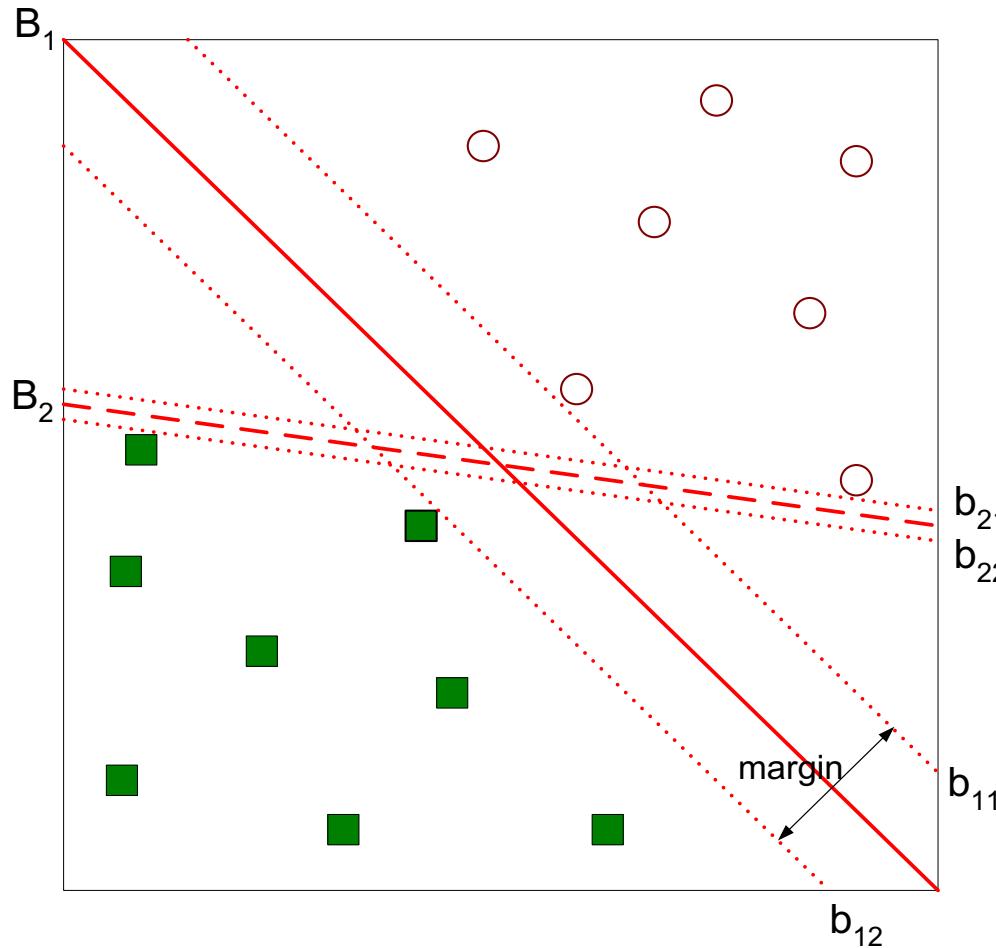
- Altre possibili soluzioni

Support Vector Machines



- Qual'è la migliore? B_1 o B_2 ?
- In base a quale criterio diciamo che è migliore?

Support Vector Machines



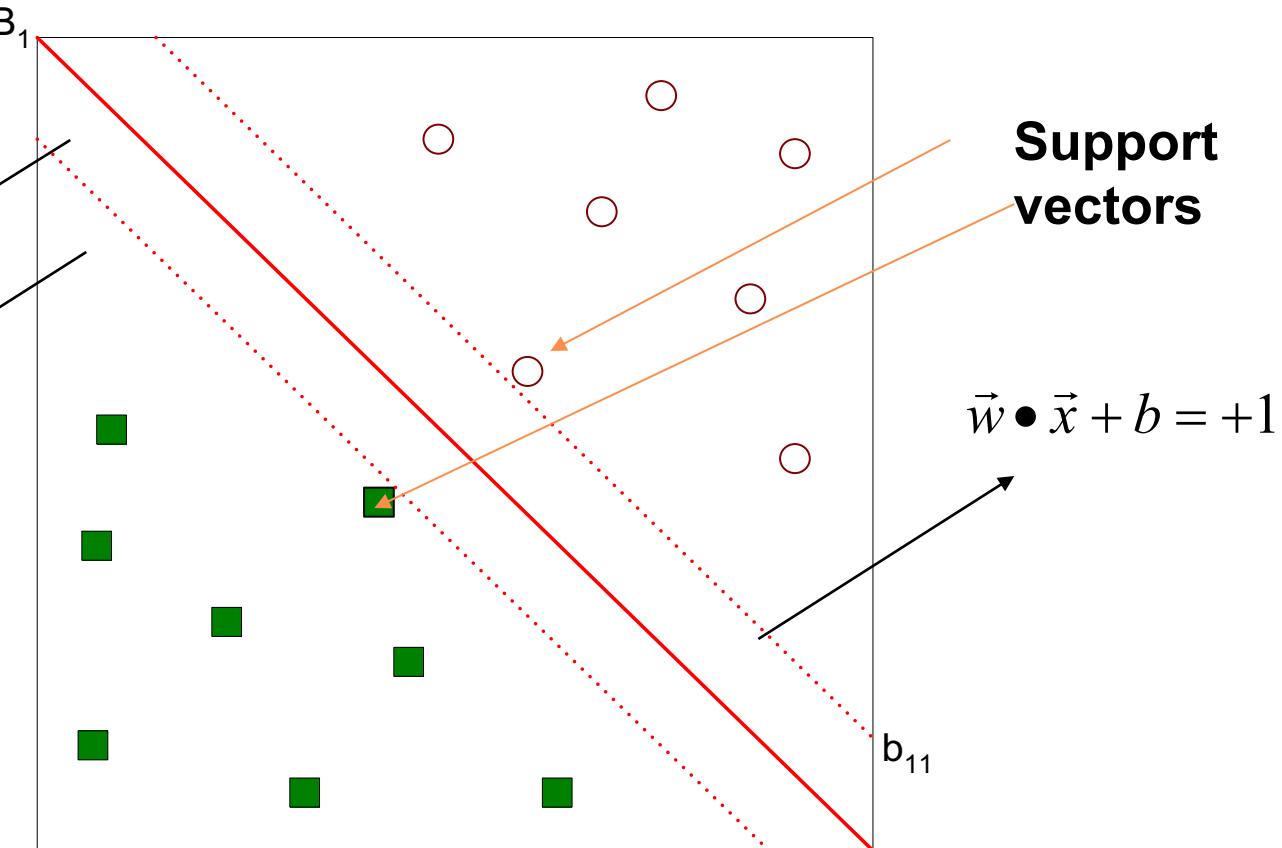
- Trova un iperpiano che **massimizzi il margine** → B1 è migliore di B2

Support Vector Machines

Dopo aver effettuato una variazione di scala

$$\vec{w} \bullet \vec{x} + b = 0$$

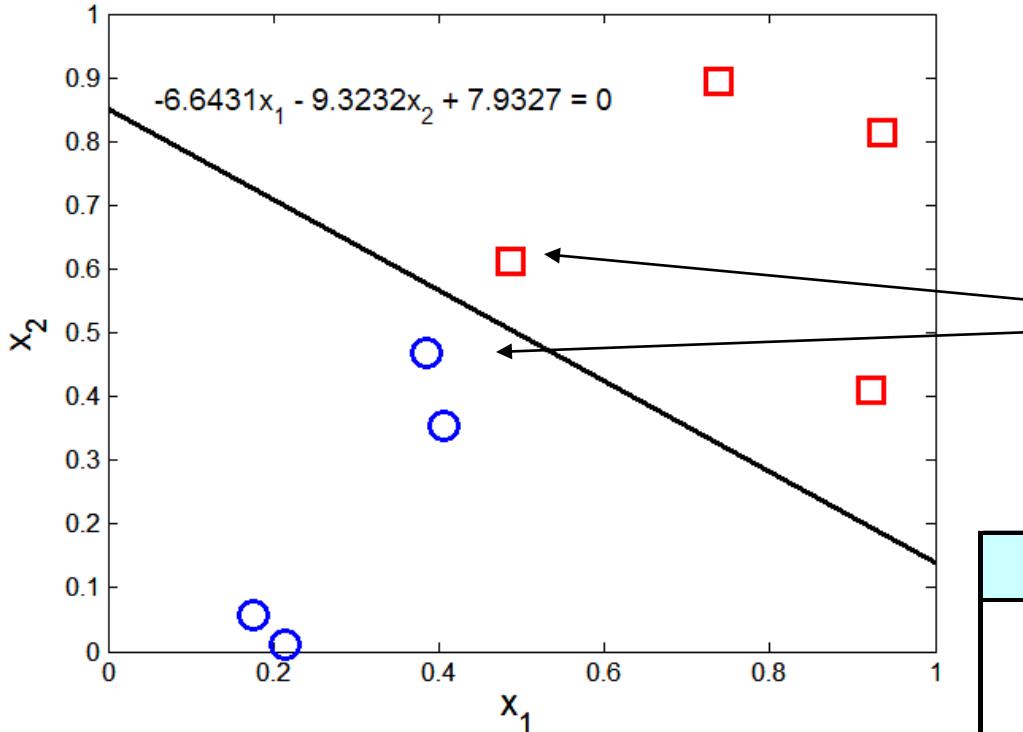
$$\vec{w} \bullet \vec{x} + b = -1$$



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

Il problema consiste nel massimizzare la distanza tra gli iperpiani

Linear SVM - Esempio



L'iperpiano di separazione è determinato solo dai «support vectors»

Support vectors

x1	x2	y
0.3858	0.4687	1
0.4871	0.611	-1
0.9218	0.4103	-1
0.7382	0.8936	-1
0.1763	0.0579	1
0.4057	0.3529	1
0.9355	0.8132	-1
0.2146	0.0099	1

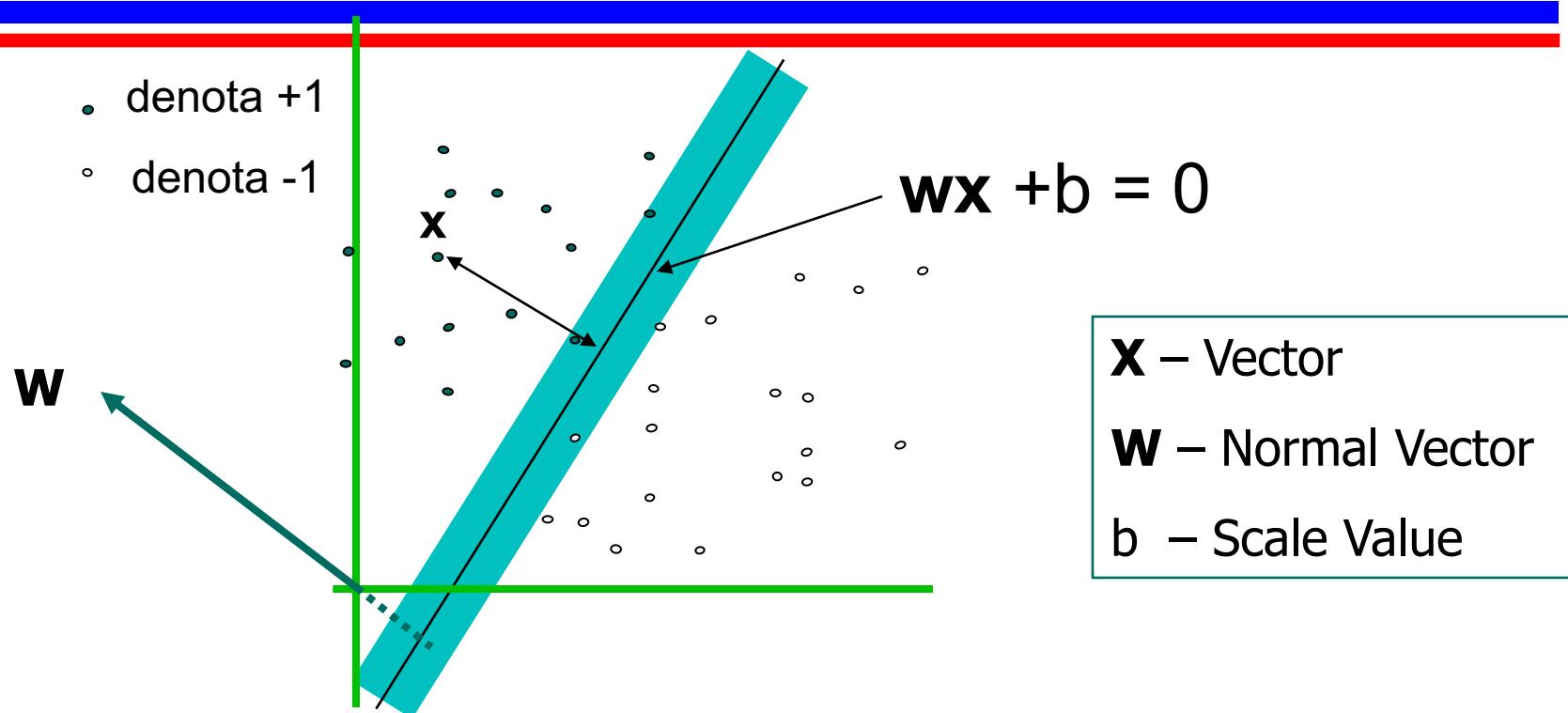
Linear SVM

- Linear model:

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

- Apprendere il modello è equivalente a determinare i valori di w e b
 - Come si determinano i valori di w e b dal training dataset?

Valutazione del margine

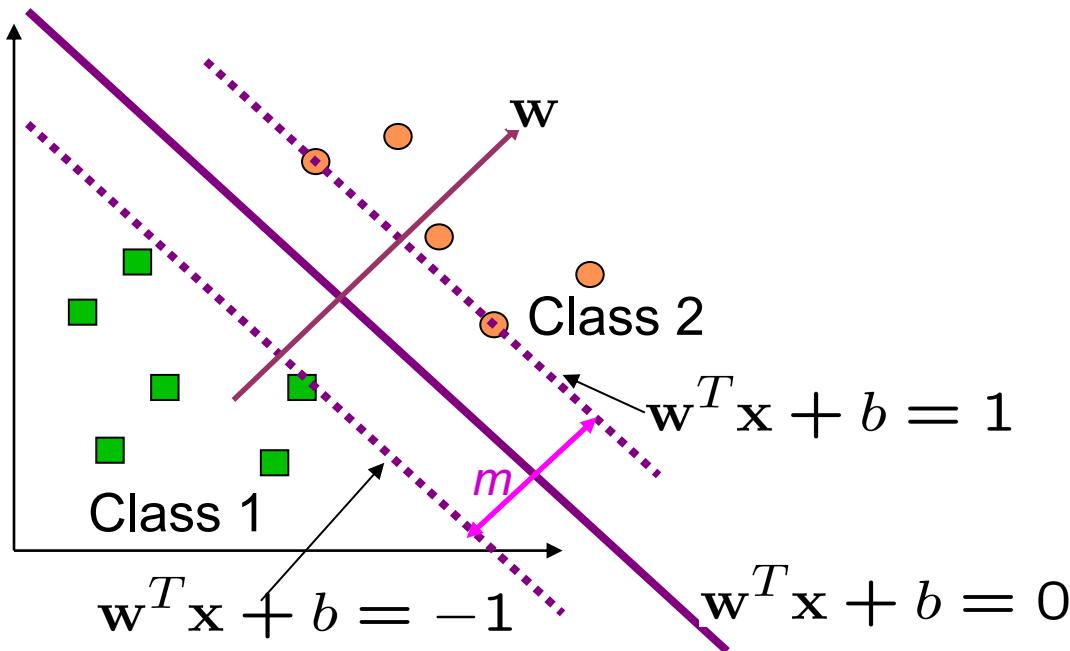


- Date 3 iperpiani paralleli r : $\mathbf{w}\mathbf{x} = 0$, $r': \mathbf{w}\mathbf{x}+b'$, $r'': \mathbf{w}\mathbf{x}+b''$ e un punto x

$$\bullet d(x, r') = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\sqrt{\|\mathbf{w}\|^2}} = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\sqrt{\mathbf{w} \cdot \mathbf{w}}} \quad d(r', r'') = \frac{|b' - b''|}{\|\mathbf{w}\|}$$

Valutazione del margine

- L'iperpiano di separazione dovrebbe essere il più possibile lontano dalle istanze di entrambe le classi
 - Distanza tra un support vector e l'iperpiano $\mathbf{w}\mathbf{x} = -b$ è $b/\|\mathbf{w}\|$
 - Vogliamo massimizzare il margine, $m = 2/\|\mathbf{w}\|$



$$m = \frac{2}{\|\mathbf{w}\|}$$

Linear SVM

- Linear model:

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

- Apprendere il modello è equivalente a determinare i valori di w e b
 - Come si determinano i valori di w e b dal training dataset?

Linear SVM

- Dato il dataset $\{x_1, \dots, x_n\}$ e sia $y_i \in \{-1, 1\}$ la classe di x_i
- L'iperpiano di separazione dovrebbe classificare tutti i punti correttamente $\Rightarrow y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \text{ for all } i$
- Infatti: se $y = -1$, abbiamo $(\mathbf{w} \cdot \mathbf{x} + b) < -1$ e se $y = 1$, abbiamo $(\mathbf{w} \cdot \mathbf{x} + b) > 1$. Per i support vectors, abbiamo $(y(\mathbf{w} \cdot \mathbf{x} + b) = 1)$.
- Il confine decisionale può essere trovato risolvendo il seguente problema di ottimizzazione vincolata

$$\underset{\mathbf{w}}{\text{minimize}} \frac{||\mathbf{w}||}{2}$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{for all } i$$

Learning Linear SVM

- L'obiettivo è massimizzare $m = \frac{2}{\|\mathbf{w}\|}$
- Che è equivalente a minimizzare $\frac{1}{m} = \frac{\|\mathbf{w}\|}{2}$

— Soggetta ai seguenti vincoli:

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathbf{w}} \bullet \vec{\mathbf{x}}_i + b \geq 1 \\ -1 & \text{if } \vec{\mathbf{w}} \bullet \vec{\mathbf{x}}_i + b \leq -1 \end{cases}$$

— equivalenti a

$$y_i(\mathbf{w} \bullet \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

- Problema di ottimizzazione vincolato
 - ◆ Risolvibile con il metodo moltiplicatore di Lagrange

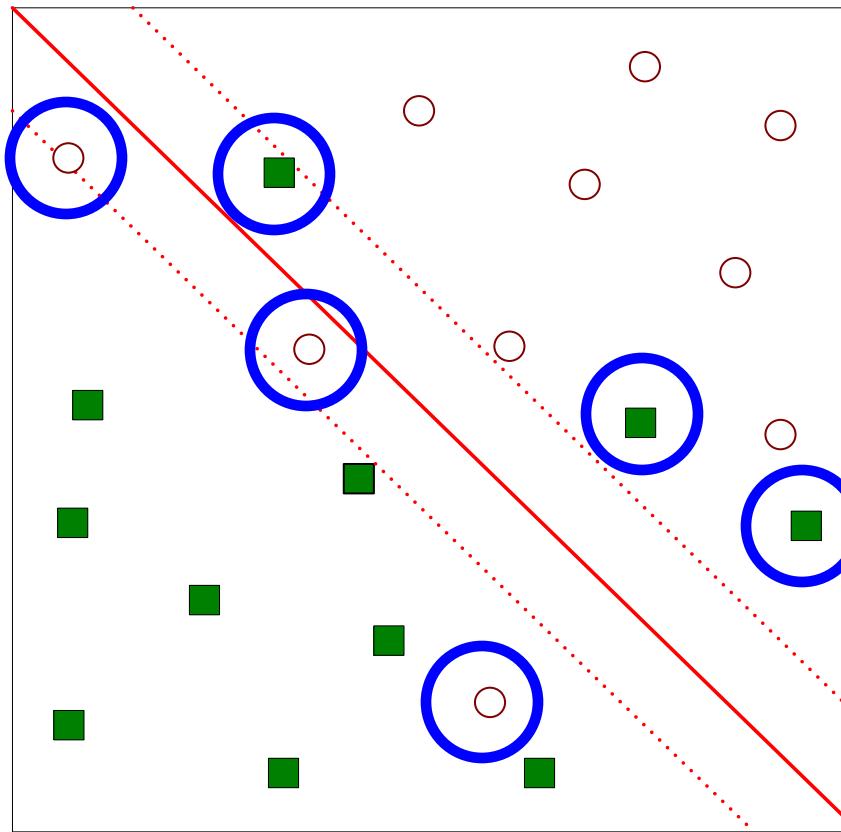
Learning Linear SVM

- L'iperpiano di separazione dipende solo dai punti “support vectors”
 - Se si dispone di dataset con gli stessi vettori di supporto, il confine decisionale non cambia
 - Come verranno classificati i record, usando SVM, una volta trovati w e b ? Dato un record di test, x_i

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

Support Vector Machines

- Cosa succede se il problema non è linearmente separabile ?



Support Vector Machines

- Cosa succede se il problema non è linearmente separabile ?
 - Introduci ulteriori variabili (slack) e riformula il problema

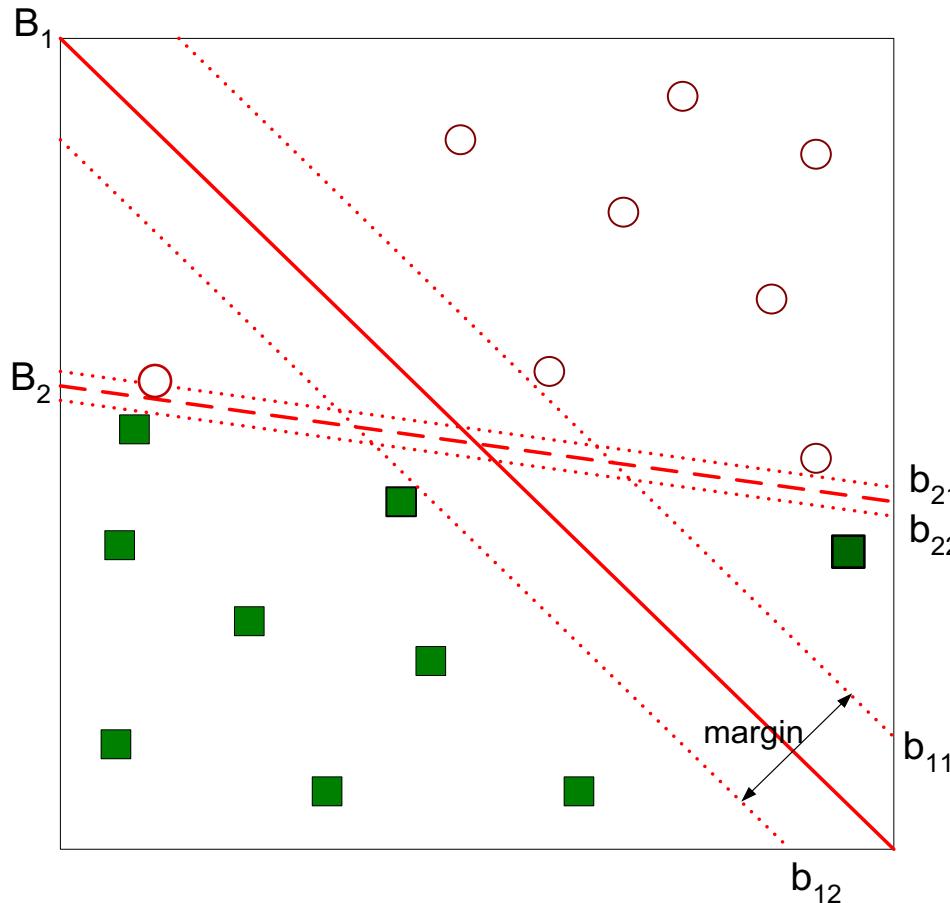
◆ minimize:

$$L(w) = \frac{\|\vec{w}\|^2}{2} + C \left(\sum_{i=1}^N \xi_i^k \right)$$

◆ Subject to:

$$y_i = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$$

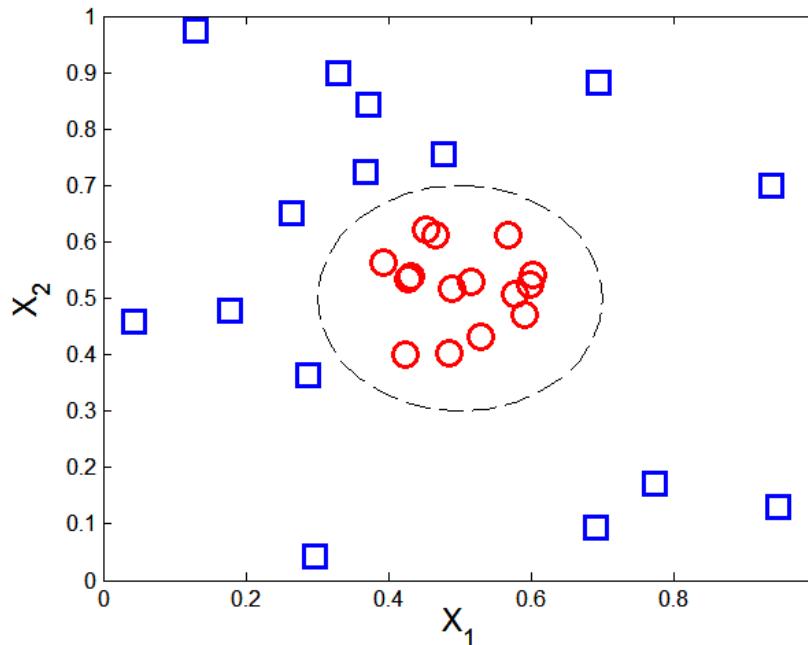
Support Vector Machines



- Trova l'iperpiano che ottimizza entrambi i fattori

Nonlinear Support Vector Machines

- Separazione non lineare



Trovare nuove forme (equazioni) di separazione

$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

SVM non lineari

- Problemi di progettazione:
 - Tipo di funzione utilizzata?
Come effettuare il calcolo in spazi n.dimensionali?
- Può essere risolto come problema di ottimizzazione convessa, per i quali sono disponibili algoritmi efficienti per trovare i minimi globali della funzione obiettivo (molti altri metodi usano approcci greedy e trovano soluzioni localmente ottimali)
- L'overfitting può essere affrontato massimizzando il margine
- Difficoltà nel gestire i valori mancanti
- Robusto per rumore
- Elevata complessità computazionale

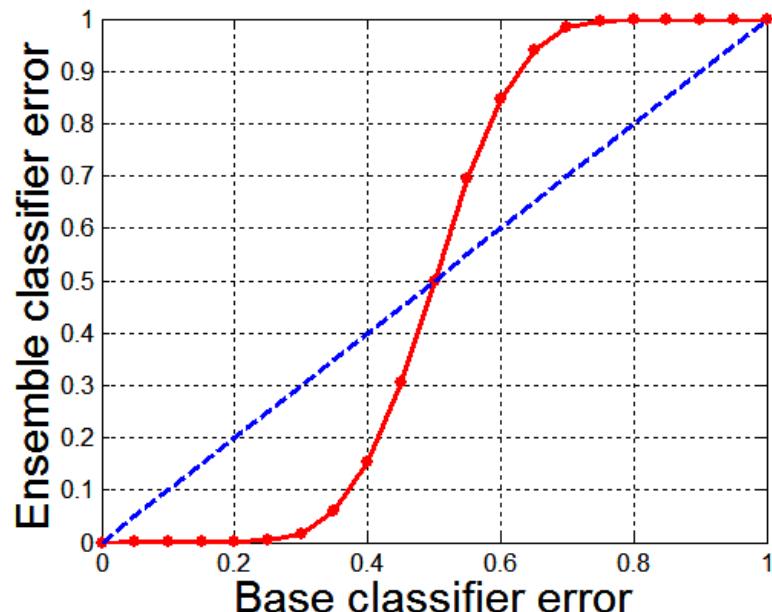
Ensemble Techniques

Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers

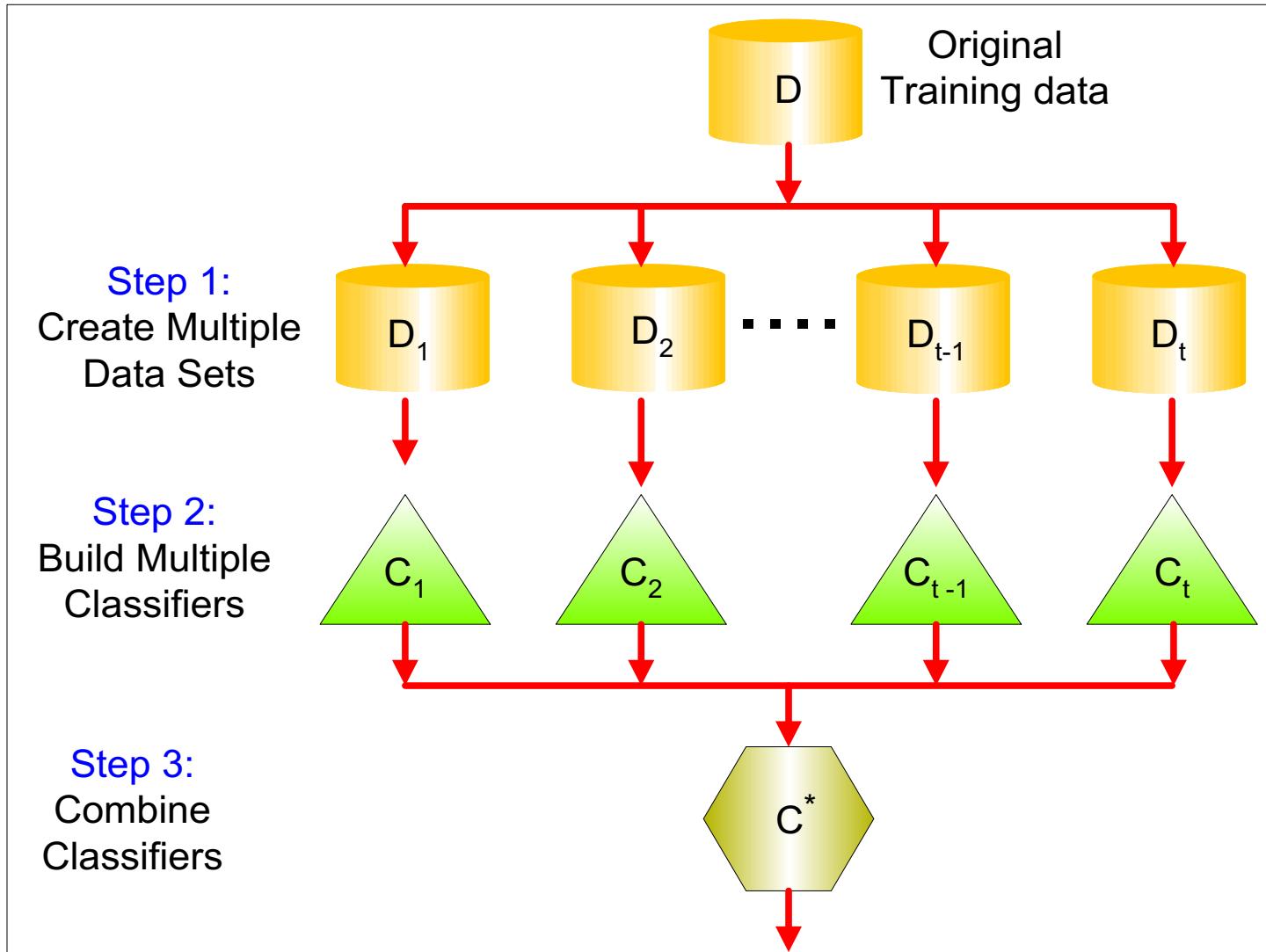
Why Ensemble Methods work?

- Suppose there are 25 base classifiers
 - Each classifier has error rate, $\varepsilon = 0.35$
 - Assume errors made by classifiers are uncorrelated
 - Probability that the ensemble classifier makes a wrong prediction:



$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

General Approach



Types of Ensemble Methods

- Manipulate data distribution
 - Example: bagging, boosting
- Manipulate input features
 - Example: random forests
- Manipulate class labels
 - Example: error-correcting output coding

Bagging

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Each sample has probability $1 - (1 - 1/n)^n$ of being selected.

Bagging Algorithm

Algorithm 5.6 Bagging Algorithm

- 1: Let k be the number of bootstrap samples.
- 2: **for** $i = 1$ to k **do**
- 3: Create a bootstrap sample of size n , D_i .
- 4: Train a base classifier C_i on the bootstrap sample D_i .
- 5: **end for**
- 6: $C^*(x) = \arg \max_y \sum_i \delta(C_i(x) = y)$, { $\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.}

Bagging Example

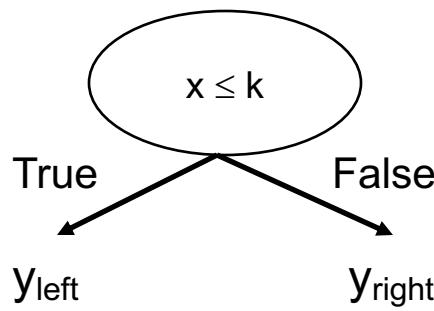
- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump

- Decision rule: $x \leq k$ versus $x > k$
- Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$
 $x > 0.35 \rightarrow y = -1$

Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

error = 0.2

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$x \leq 0.7 \rightarrow y = 1$

$x > 0.7 \rightarrow y = 1$

error = 0.3

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

error = 0.3

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \rightarrow y = 1$

$x > 0.3 \rightarrow y = -1$

error = 0.2

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

error = 0.3

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$
error = 0.1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$
error = 0.1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$
error = 0.2

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$
error = 0.2

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$x \leq 0.05 \rightarrow y = 1$
 $x > 0.05 \rightarrow y = 1$
error = 0.0

Bagging Example

- Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Predicted
Class

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights
 - Unlike bagging, weights may change at the end of each boosting round

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost

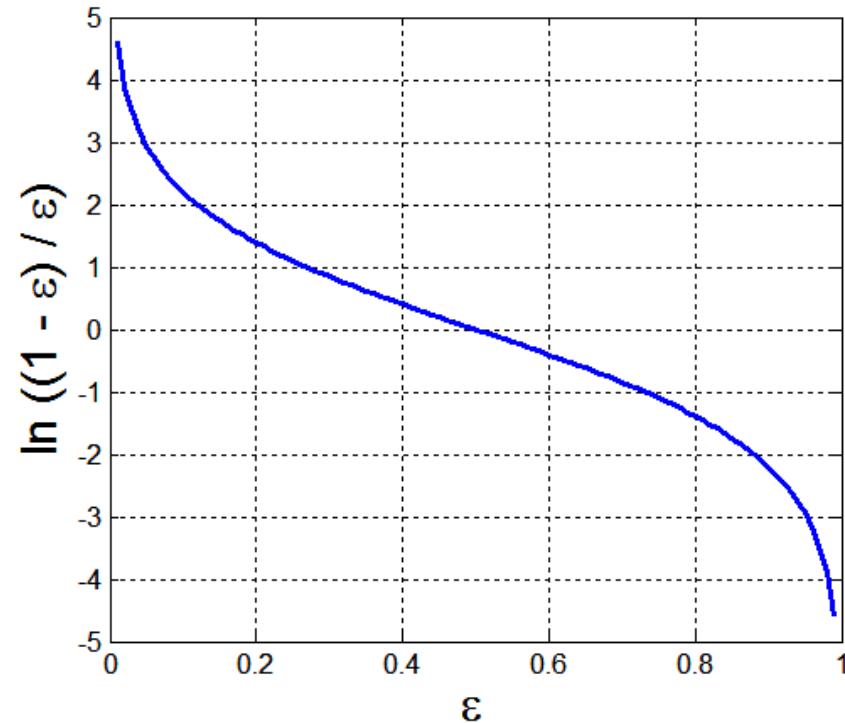
- Base classifiers: C_1, C_2, \dots, C_T

- Error rate:

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$



AdaBoost Algorithm

- Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases} \quad (5.88)$$

where Z_j is the normalization factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

AdaBoost Algorithm

Algorithm 5.7 AdaBoost Algorithm

```
1:  $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ . {Initialize the weights for all  $n$  instances.}
2: Let  $k$  be the number of boosting rounds.
3: for  $i = 1$  to  $k$  do
4:   Create training set  $D_i$  by sampling (with replacement) from  $D$  according to  $w$ .
5:   Train a base classifier  $C_i$  on  $D_i$ .
6:   Apply  $C_i$  to all instances in the original training set,  $D$ .
7:    $\epsilon_i = \frac{1}{n} [\sum_j w_j \delta(C_i(x_j) \neq y_j)]$  {Calculate the weighted error}
8:   if  $\epsilon_i > 0.5$  then
9:      $w = \{w_j = 1/n \mid j = 1, 2, \dots, n\}$ . {Reset the weights for all  $n$  instances.}
10:    Go back to Step 4.
11:  end if
12:   $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$ .
13:  Update the weight of each instance according to equation (5.88).
14: end for
15:  $C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$ .
```

AdaBoost Example

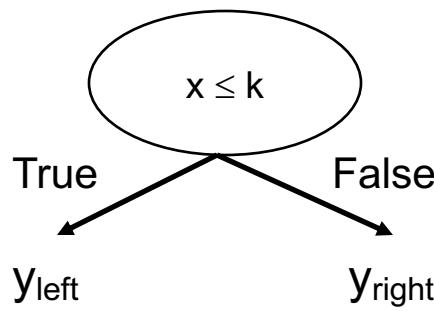
- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump

- Decision rule: $x \leq k$ versus $x > k$
- Split point k is chosen based on entropy



AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1



Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1



- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

AdaBoost Example

Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

Classification

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

Random Forest Algorithm

- Construct an ensemble of decision trees by manipulating training set as well as features
 - Use bootstrap sample to train every decision tree (similar to Bagging)
 - Use the following tree induction algorithm:
 - ◆ At every internal node of decision tree, randomly sample p attributes for selecting split criterion
 - ◆ Repeat this procedure until all leaves are pure (unpruned tree)

Characteristics of Random Forest

- Base classifiers are unpruned trees and hence are *unstable classifiers*
- Base classifiers are *decorrelated* (due to randomization in training set as well as features)
- Random forests reduce variance of unstable classifiers without negatively impacting the bias
- Selection of hyper-parameter p
 - Small value ensures lack of correlation
 - High value promotes strong base classifiers
 - Common default choices: \sqrt{d} , $\log_2(d + 1)$

Gradient Boosting

- Constructs a series of models
 - Models can be any predictive model that has a differentiable loss function
 - Commonly, trees are the chosen model
 - ◆ XGboost (extreme gradient boosting) is a popular package because of its impressive performance
- Boosting can be viewed as optimizing the loss function by iterative functional gradient descent.
- Implementations of various boosted algorithms are available in Python, R, Matlab, and more.