Classificazione II

Prof. Matteo Golfarelli
Alma Mater Studiorum - Università di Bologna

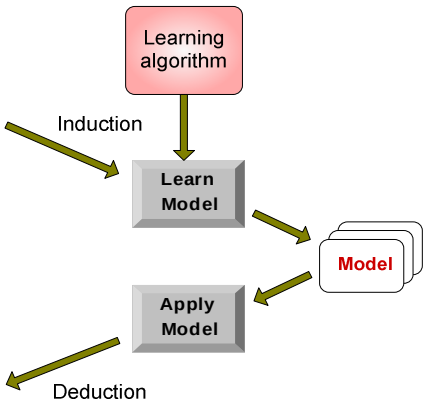
Classificazione: Definizione

- 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 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



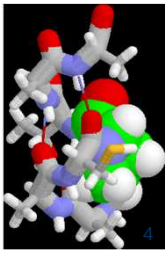
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Applicazioni

- Predire se una cellula tumorale è benigna o maligna in base alle sue caratteristiche
- Classificare se una transazione con carta di credito sia o meno fraudolenta

- Classificare le strutture proteiche secondarie in alpha-helix, beta-sheet, or random coil
- Classificare le news in base all'argomento:finanza, meteo, sport, intrattenimento, ecc.



Tecniche di classificazione

- Alberi decisionali o Decision Tree
- **Regole di decisione**
- **Nearest-neighbor**
- Reti Bayesiane
- Reti neurali
- Support Vector Machines

Classificatori basati su regole

- Classificano i record utilizzando insiemi di regole del tipo "if…then…"
- Una regola ha la forma: (*Condizione*) $\rightarrow$ *y*
  - ✓ dove
    - *Condizione* è una congiunzione di predicati su attributi
    - *y* è l'etichetta di classe
  - ✓ *LHS*: antecedente o condizione
  - ✓ *RHS*: conseguente o etichetta della classe
  - ✓ Esempi di regole:
    - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
    - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Evade=No

Costruire un modello significa identificare un insieme di regole

Vantaggi e svantaggi

- Espressivi quanto un Decision Tree
- Facili da interpretare
- Facili da generare
- Rapidi nella classificazione di nuove istanze

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

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 una istanza **x** se i valori di **x** soddisfano l'antecedente di *r*

r1: (Give Birth = no) ∧ (Can Fly = yes) → Birds

r2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes

r3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals

r4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles

r5: (Live in Water = sometimes) → Amphibians

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

La regola *r1* copre hawk => Bird

La regola *r3* copre grizzly bear => 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 = (*Status*=Single) → No

Coverage(*r*) = 40%, *Accuracy*(*r*) = 50%

Come funziona un classificatore basato su regole?

r1: (Give Birth = no) ∧ (Can Fly = yes) → Birds

r2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes

r3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals

r4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles

r5: (Live in Water = sometimes) → 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 lemure 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 dei classificatori basati su regole

- 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 da al più 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 dei classificatori basati su regole

- 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 discordanti
  - ✓ 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

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 utilizzata dai principali algoritmi di costruzione delle regole(RIPPER, C4.5rules)

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 per esempio i decision tree (C4.5rules)

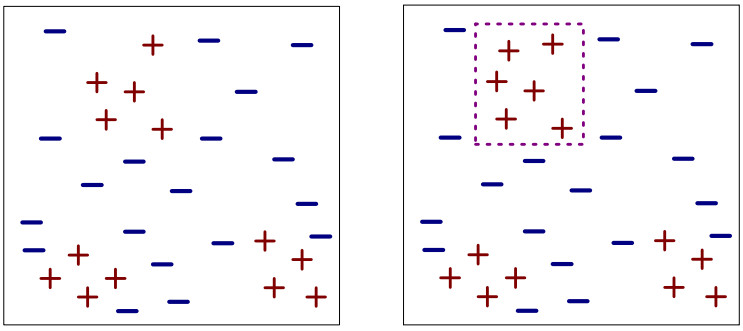
Metodi diretti: sequential covering

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

```
set R = ∅
for each class y ∈ Y0 – {yk} do
 stop=FALSE;
  while !stop do
    r =  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 ∪ r // Add r at the bottom of the rule list
   end while
end for
R = R ∪ {{}->yk}// Add the default rule at the bottom of the rule list
PostPruning(R);
```

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

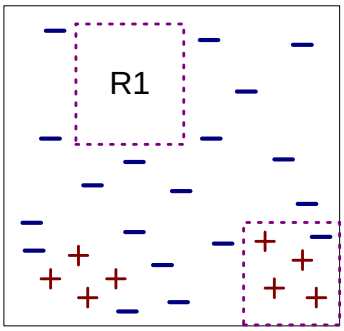
Sequential covering: esempio



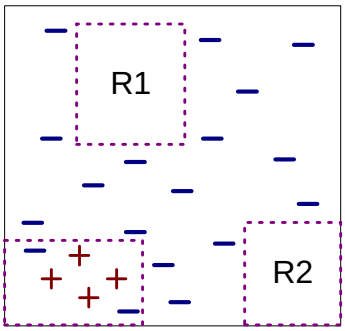
(i) Original Data (ii) Step 1

- 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

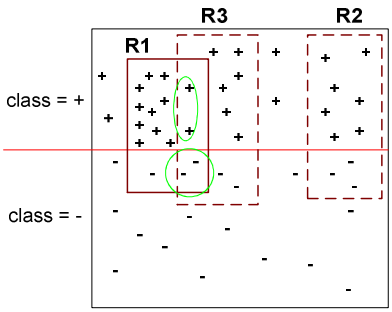
Eliminazione delle istanze dal training set

- 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: a evitare di sottostimare l'accuracy della prossima regola
    - Le istanze negative conteggiate per la regola 1 non devono essere conteggiate anche per la regola 3

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

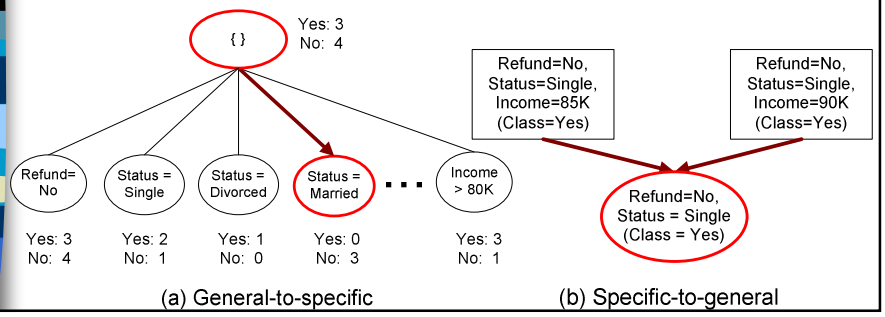
Dopo aver scelto R1

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



Learn-One-Rule

- L'obiettivo dell'algoritmo è trovare una regola che copra quanti più possibili esempi positivi e quanto meno possibili (o nessun) esempio negativo
  - ✓ 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 predicato ossia una possibile coppia (Attributo,Valore)



Learn-One-Rule

- E' necessario un criterio per scegliere quale predicato aggiungere
  - *n* : numero di istanze coperte dalla regola
  - n_r : numero di istanze correttamente classificate dalla regola
  - *k* : numero di classi
  - ✓ Accuratezza: limitata applicabilità poiché non considera la coverage della regola

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

    - *r1*: copre 50 esempi positivi e 5 esempi negativi → Accuracy(*r1*) = 90.9%
    - *r2*: copre 2 esempi positivi e 0 esempi negativi → Accuracy(*r2*) = 100%

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

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

    - Se la coverage è 0 (n_r=*n*=0) e assumendo distribuzione uniforme dei dati , l'indice si riduce alla probabilità a priori della classe (1/*k*)
    - Se la coverage è alta l'indice tende asintoticamente al valore dell'accuracy $\frac{n_r}{n}$
    - Per valori bassi di coverage *k* tende a ridurre il valore dell'indice

Learn-One-Rule

- ✓ Metriche che considerano il supporto della regola ossia il numero di esempi positivi coperti dalla regola

$$FoilGain(r, r') = t\left(\log_2 \frac{p'}{p'+n'} - \log_2 \frac{p}{p+n}\right)$$

  - *r* / *r'*: regola prima / dopo l'inserimento del nuovo predicato
  - *t*: numero di esempi positivi coperti sia da *r* sia da *r'*
  - *p* / *p'* : numero di esempi positivi coperti da *r* / *r'*
  - *n* / *n'* : numero di esempi negativi coperti da *r* / *r'*
  - L'indice è proporzionale a *t* e a (*p'*)/(*p'*+*n'*) quindi tende a favorire regole che hanno elevato coverage e accuracy

Training set composto da 100 esempi positivi e 400 negativi
R1: A → + (copertura: 4 esempi positivi e 1 negativo)
R2: A → + (copertura: 30 esempi positivi e 10 negativi)
R3: A → + (copertura: 100 esempi positivi e 90 negativi)

Determinare quale è la regola migliore sulla base dell'accuracy e del FoilGain

Learn-One-Rule

- Criterio di stop nell'estensione di una regola
  - ✓ Calcola il gain
  - ✓ Se il gain non è significativo stop!
- 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 uno dei predicati dalla regola
    - Elimina il predicato la cui rimozione comporta il massimo miglioramento dell'error rate sul validation set
    - Ripeti i passi precedenti sino a che continuano a determinare un miglioramento

Metodi diretti: RIPPER

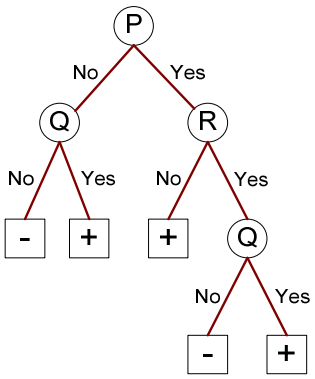
- Approccio basato sul sequential covering
  - ✓ Una sua versione denominata **JRip** è implementata in WEKA
- Per problemi a 2 classi, scegli una delle classi come classe positiva e l'altra come classe negativa
  - ✓ Impara le regole per la classe positiva
  - ✓ La classe negativa sarà la classe di default
- Per problemi multi-classe
  - ✓ Ordina le classi in base a un criterio di rilevanza della classe (frazione delle istanze che appartengono alla classe)
  - ✓ Impara le regole a partire dalla classe più piccola e considerando il resto delle istanze come classe negativa
  - ✓ Ripeti l'operazione con le classi successive
- Dopo l'aggiunta di una regola calcola il description length
  - ✓ Interrompi la costruzione di nuove regole quando il nuovo valore di description length è superiore di d bits rispetto a quello precedente

Metodi diretti: RIPPER

- Learn-one-rule:
  - ✓ Inizia con una regola vuota
  - ✓ Aggiungi predicati finchè determinano un miglioramento del FoilGain
  - ✓ 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 i predicati 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 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 pruninig

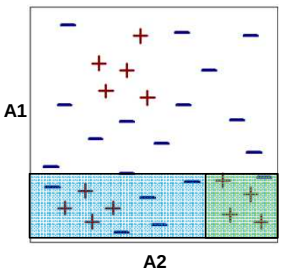


Rule Set

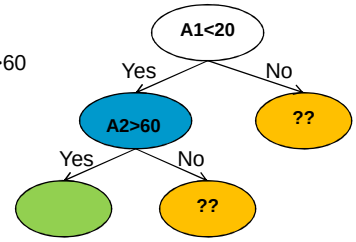
r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

Alberi decisionali vs regole

- Sebbene l'espressività delle due tecniche sia similare 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 split scelto in ogni nodo dell'albero cosidera la qualità di tutti i figli generati
  - ✓ il criterio con cui viene aggiunto un predicato alla regola valuta solo la bontà della classe che si viene a determinare

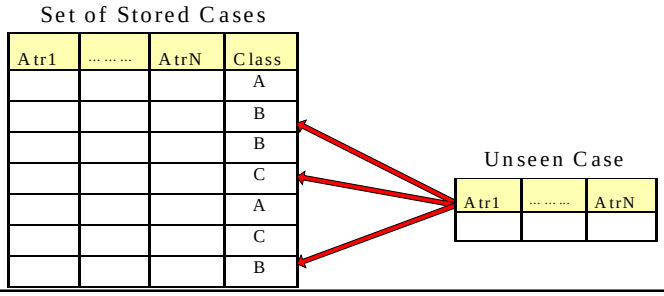


R1: A1<20
R1: A1<20 and A2>60



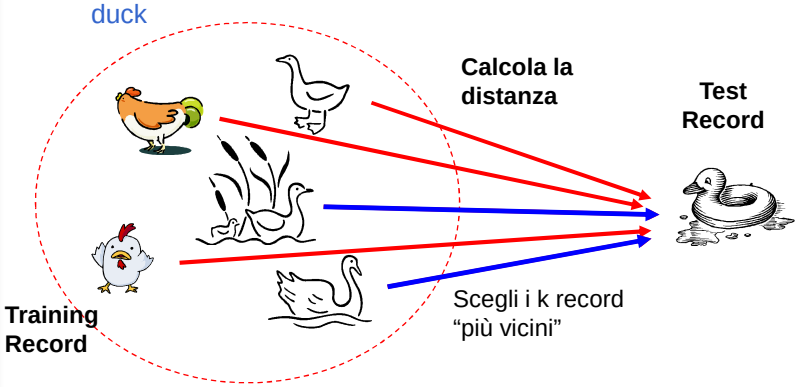
Classificatori Instance-Based

- Non costruiscono modelli ma classificano i nuovi record sulla base della loro somiglianza rispetto agli esempi nel training set
- Sono per questo detti lazy -pigri- learners in contrapposizione agli eager –diligenti, impazienti- learners (rule based, alberi decisionali, reti neurali, ecc.)
  - ✓ Rote-learner: classifica un record solo se coincide con uno del training set
  - ✓ Nearest-Neighbor: classifica il record in base ai più simili del training set

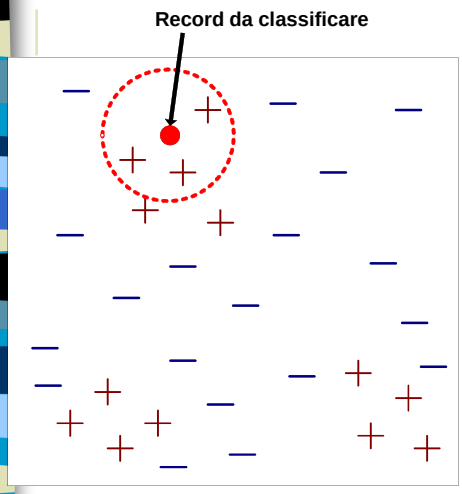


Classificatori Nearest Neighbor

- Utilizzano i k punti “più vicini” (nearest neighbors) per effettuare la classificazione
- Idea di base:
  - ✓ If it walks like a duck, quacks like a duck, then it’s probably a duck

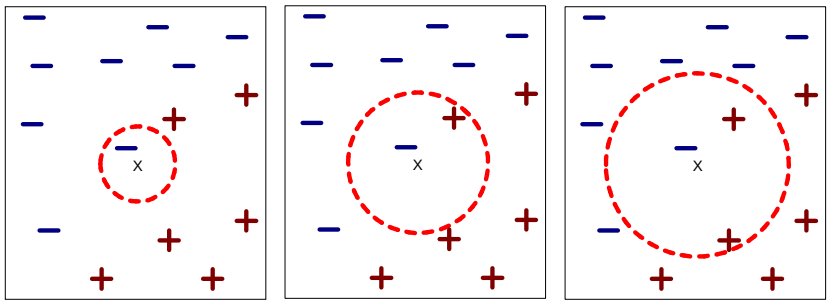


Classificatori Nearest Neighbor



- Richiedono:
  - Un taining set
  - Una metrica per calcolare la distanza tra i record
  - Il valore di *k*, ossia il numero di vicini da utilizzare
- Il processo di classificazione:
  - Calcola la distanza rispetto ai record nel traing set
  - Identifica *k* nearest neighbors
  - Utilizza le label delle classi dei nearest neighbor per determinare la classe del record sconosciuto (es. scegliendo quella che compare con maggiore 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* distance da **x**

K-Nearest Neighbor

- La classificazione di un record **z** è ottenuta con un processo di *majority voting* tra i *k* elementi D_z del training set *D* più vicini (o simili) a **z**

$$\bar{y} = \arg\max_{y \in \mathbf{Y}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(y_i = y)$$

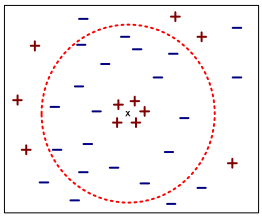
  - ✓ Y è l'insieme delle label di classe
  - ✓ I() restituisce 1 se il suo argomento è TRUE, 0 altrimenti

- Tutti i vicini hanno lo stesso peso
  - ✓ L'algoritmo è molto sensibile al valore di *k*
  - ✓ Questo rischio può essere ridotto pesando il contributo dei vicini in base alla distanza $w = 1/d(\mathbf{z},\mathbf{x}_i)$

$$\bar{y} = \arg\max_{y \in \mathbf{Y}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w \times I(y_i = y)$$

K-Nearest Neighbor

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



- Per operare correttamente gli attributi devono avere la stessa scala di valori e vanno quindi normalizzati in fase di pre-processing
  - ✓ Esempio: su quale attributo una differenza di 0.5 vale di più?
    - l'altezza di un adulto varia 1.5m to 2.1m
    - il peso di un adulto varia da 40kg a 150kg
    - Lo stipendio di una persona varia da 10K€ to 1M€

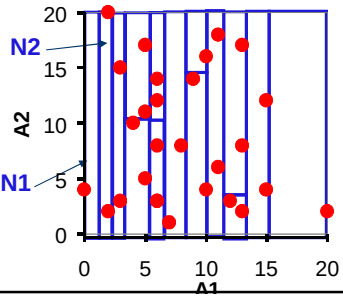
- Normalizzare la scala può non essere sufficiente in presenza di diverse distribuzioni dei dati
  - ✓ Mahalanobis distance

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 similarità o 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

Utilizzo di indici per query spaziali

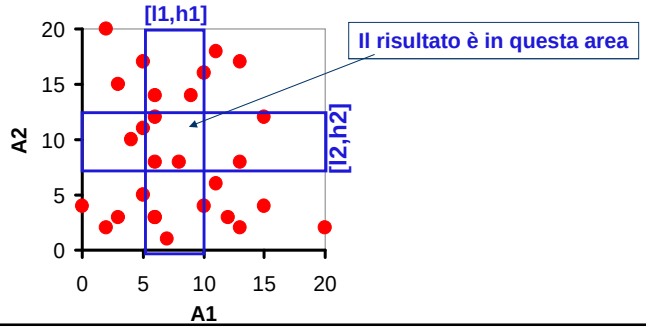
- Dati n attributi A1,…,An e in assenza di strutture dati specifiche si potrebbe pensare di costruire un B^+-tree multi-attributo, che organizzi (ordini) le tuple in base ai valori di A1,A2,…,An
- Considerando i nodi foglia del B^+-tree: N1→N2→N3 →…
  - ✓ La prima foglia, N1, conterrà le tuple con i più piccoli valori di A1 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 questa soluzione non cattura il concetto di distanza spaziale



Punti "vicini" potrebbero essere qui

Un'altra soluzione basata sui B$^+$-tree

- Assumiamo di conoscere, per esempio mediante le statisiche del DB, che i k-NN di un punto q stiano nell'(iper) rettangolo con lati [*l1*,*h1*] x [*l2*,*h2*] x…
- Allora possiamo utilizzare *n* query di range indipendenti `Ai BETWEEN li AND hi` su *n* indici distinti *A1*,*A2*,…,*An* 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



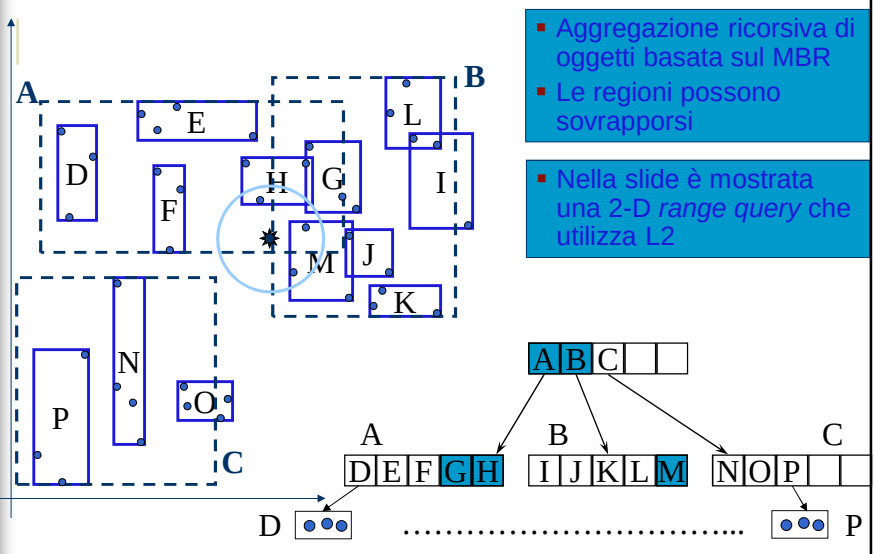
Indici multi-dimensionali (spaziali)

- Il B$^+$-tree multi-attributo mappa punti di $\mathbf{A} \subseteq \Re^D$ in punti di $\Re$
- 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

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 prinicipio ricorsivamente dalle foglie alla radice
- Gli R-tree organizzano gli oggetti in
  - ✓ Un insieme di intervalli multi-dimensionali sovrapposti (iper-rettangoli)
  - ✓ Applicano questo principio ricorsivamente dalle foglie alla radice
- Gli R-tree sono oggi disponibili in alcuni DBMS commerciali quali Oracle9i

R-tree: intuizione

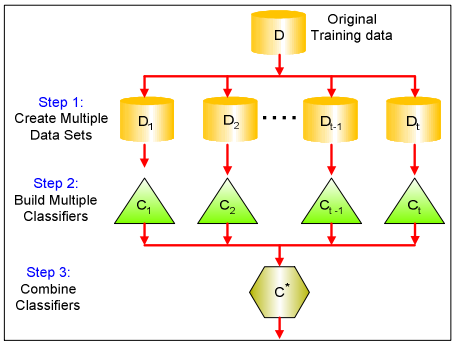


- Aggregazione ricorsiva di oggetti basata sul MBR
- Le regioni possono sovrapporsi

- Nella slide è mostrata una 2-D *range query* che utilizza L2

Combinazione di classificatori

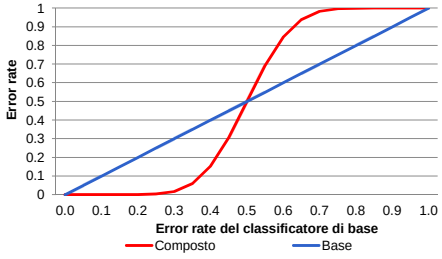
- Idea: costruire più classificatori di base e predire la classe di appartenza di un record aggregando le classificazioni ottenute
  - ✓ Il risultato del classificatore composto è definito per mezzo di una funzione che, per esempio, assegna il record alla classe che è stata "*votata*" dal maggior numero di classificatori



Principio di funzionamento

- Supponiamo di avere 25 classificatori semplici
  - ✓ Ogni classificatore ha un error-rate pari a ε = 0.35
  - ✓ Si assuma che i classificatori siamo indipendenti
    - Non c'è correlazione tra gli error-rate dei classificatori
- La probabilità che il classificatore composto dia un risultato errato è:

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



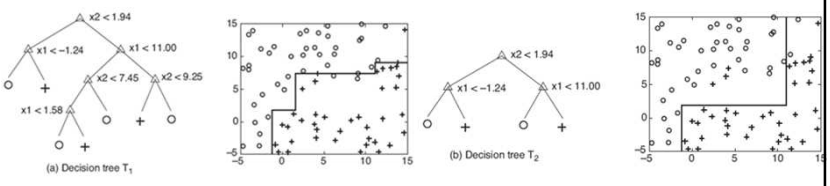
- Condizioni necessarie perchè il classificatore composto dia risultati migliori dei classificatori semplici sono:
  - ✓ Che i classificatori siano indipendenti
  - ✓ Che l'error-rate del singolo classificatore sia inferiore a 0.5

Come costruire classificatori composti

- Cambiando il training set: si costruiscono più training set a partire dal data set dato
  - ✓ Bagging e Boosting
- Cambiando gli attributi utilizzati: i singoli classificatori sono basati su un sottoinsieme degli attributi
  - ✓ Utile quando gli attributi sono fortemente ridondanti
    - Random Forest
- Cambiando le classi considerate:
  - ✓ Si partizionano le classi in due gruppi A0 e A1 e si trasforma il problema dato in un problema binario. Le classi che appartengono ad A0 sono classificate come 0 le rimanenti come 1.
  - ✓ I diversi classificatori sono costruiti risuddividendo le classi in sottoinsiemi diversi
  - ✓ La classificazione del classificatore composto si ottiene incrementando di 1 il punteggio delle classi che appartengono al sottoinsieme scelto.
  - ✓ Il record è infine assegnato alla classe che ottiene il punteggio maggiore
    - Error-Correcting Output Coding
- Cambiando i parametri dell'algoritmo di learning:
  - ✓ Topologia e pesi di una rete neurale
  - ✓ Alberi decisionali con politiche di scelta random degli attributi da utilizzare

Decomposizione Bias-Variance-Noise

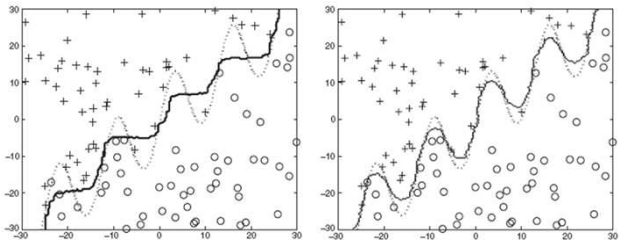
- Un modello formale per analizzare l'errore commesso da un classificatore
  - ✓ Probabilità che un classificatore sbagli la sua previsione
- L'errore commesso da un classificatore dipende da:
  - ✓ **BIAS:** capacità del classificatore scelto nel modellare gli eventi e nell'estendere la previsione agli eventi non presenti nel training set
    - Tipi di classificatori diversi hanno capacità diverse nel definire i decision boundary tra le classi
    - Per esempio decision tree diversi possono avere capacità diverse



  - ✓ **VARIANCE:** capacità del training set nel rappresentare il data set reale
    - Training set diversi possono determinare decision boundary diversi
  - ✓ **NOISE:** non-determinismo delle delle classi da determinare
    - Istanze set con gli stessi valori degli attributi possono determinare classi diverse

Decomposizione Bias-Variance-Noise

- Tipi di classificatori diversi hanno capacità intrinsecamente diverse nel modellare i margini delle regioni
  - ✓ 100 training set ognuno contenente 100 esempi ottenuti a partire da una suddivisione in regioni predefinita (linea trattegiata)
  - ✓ La linea nera rappresenta la media della linea di separazione media ottenuta dai 100 classificatori



- La differenza tra la vera linea di separazione e quella media rappresenta il bias del classificatore
  - ✓ Il bias del 1-NN è inferiore
  - ✓ Tuttavia i k-NN sono maggiormente sensibili alla composizione del training set e quindi presenteranno un maggiore variance

Multi-classificatori

- Si utilizzano classificatori diversi (es. Alberi decisionali + k-nearest neighbor) per ridurre il **bias** dell'errore
  - ✓ I classificatori devono essere inidipendenti: non c'è correlazione (o ce ne è poca) tra gli errori commessi tra due classificatori
  - ✓ Classificatori diversi possono operare su sottoinsiemi distinti di attributi su cui hanno performance ideali

- La classe di appartenenza è decisa mediante un meccanismo di voting
  - ✓ Classe votata dal maggior numero di classificatori
  - ✓ Il voto può essere pesato in base alla confidenza del classificatori nel caso in cui questo la fornisca
    - Es. Es il classificatore C1 vota per la classe X. In fase di addestramento C1 ha commesso 25 su 100 errori per la classe X. Il classificatore C2 vota per la classe Y. In fase di addestramento C2 ha commesso 10 su 100 errori per la classe Y.

Il record è assegnato alla classe Y

Bagging

- Permette di costruire classificatori composti che associano un evento alla classe più votata dai classificatori base
- Ogni classificatore è costruito mediante bootstrap di un medesimo trainining set

```
// k = numero di cicli di bootstrap N = card. del training set
// δ()=1 se l'argomento della funzione è TRUE, 0 altrimenti

for i=1 to k do
 Crea un training set D_i di dimensione N
 Allena un classificatore C_i sul training set D_i
end for
```

$$C^*(\mathbf{x}) = \arg\max_y \sum_i \delta(C_i(\mathbf{x}) = y)$$

- Il bagging migliora l'errore di generalizzazione riducendo la componente **variance**
  - ✓ Il bagging sarà quindi particolarmente utile per quei tipi di classificatori sensibili alle variazioni del training set

Bagging: un esempio

- Classificatore di base: albero decisionale binario a un livello
  - ✓ Può solo fare scelte del tipo x≤s →-1 x>s →1 dove s è lo split point
- Il data set

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

- L'accuratezza del classificatore base non può superare 70%
  - ✓ x≤0.3 →1 x>0.3 →-1
  - ✓ x≤0.7 →-1 x>0.7 →1

Bagging: le sessioni

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9	x <= 0.35 ==> y = 1
y	1	1	1	1	-1	-1	-1	-1	1	1	x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1	x <= 0.65 ==> y = 1
y	1	1	1	-1	-1	1	1	1	1	1	x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9	x <= 0.35 ==> y = 1
y	1	1	1	-1	-1	-1	-1	-1	1	1	x > 0.35 ==> y = -1

Bagging Round 4:

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

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1	x <= 0.35 ==> y = 1
y	1	1	1	-1	-1	-1	-1	1	1	1	x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1	x <= 0.75 ==> y = -1
y	1	-1	-1	-1	-1	-1	-1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1	x <= 0.75 ==> y = -1
y	1	-1	-1	-1	-1	1	1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1	x <= 0.75 ==> y = -1
y	1	1	-1	-1	-1	-1	-1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1	x <= 0.75 ==> y = -1
y	1	1	-1	-1	-1	-1	-1	1	1	1	x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9	x <= 0.05 ==> y = -1
y	1	1	1	1	1	1	1	1	1	1	x > 0.05 ==> y = 1

Bagging: i risultati

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
True Class	1	1	1	-1	-1	-1	-1	1	1	1

- Il bagging permette di ottenere il comportamento di un albero decisionale a due livelli

Disegnare l'albero decisionale a due livelli corrispondente al risultato del bagging

Boosting

- Un approccio iterativo che permette di adattare progressivamente la composizione del training set al fine di concentrarsi sui record classificati in modo incorretto
  - ✓ Inizialmente, tutti gli N record hanno lo stesso peso (1/N)
  - ✓ Diversamente dal bagging, i pesi possono cambiare alla fine del round di boosting in modo da aumentare la probailità del record di essere selezionato nel training set
    - Vengono aumentate le probabilità dei record che sono difficili da classificare ossia che nel precedente round di boosting sono stati classificati in modo incorretto
  - ✓ Il risultato finale si ottiene combinando il risultato le predizioni fatte dai diversi classificatori
- Le tecniche di boosting si differenziano in base a come:
  - ✓ sono aggiornati i pesi dei record del training set
  - ✓ sono combinate le predizioni dei classificatori
- AdaBoost è una delle tecniche di boosting più utilizzate

AdaBoost

- Siano C_1, C_2, …, C_T i T classificatori di base ognuno utilizzato a un ciclo di boost $j \in [1,T]$. Sia ε_j l'error rate:

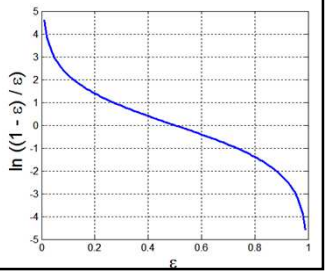
$$\varepsilon_j = \frac{1}{N}\sum_{i=1}^{N} w_i \delta\left(C_j(\mathbf{x}_i) \neq y_i\right)$$

  - ✓ $(\mathbf{x}_i, y_i)$ i=1..N record del training set
  - ✓ w_i è il peso del i-esimo elemento del training set
  - ✓ δ()=1 se l'argomento è TRUE, 0 altrimenti

- L'importanza di un classificatore è definita come:

$$\alpha_j = \frac{1}{2}\ln\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$$

- α_j assume valori positivi quando l'error rate è prossimo a 0
- α_j assume valori negativi quando l'error rate è prossimo a 1

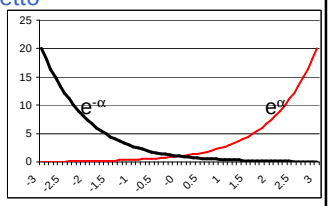


AdaBoost

- La regola per l'aggiornamento dei pesi del record *i* ad ogni ciclo di boost *j* è:

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

  - ✓ Z_j è un fattore di normalizzazione per assicurare che $\sum_i w_i^{j+1} = 1$
  - ✓ Il peso dei record classificati in modo corretto si riduce, quello dei pesi classificati in modo incorretto aumenta



- Se un ciclo di boost produce un classificatore con error rate maggiore di 50%, i pesi sono riportati 1/n
- Il record è assegnato alla classe che massimizza la somma pesata:

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

AdaBoost

```
1. w={w_i=1/N i=1..N}
2. for j=1 to T do // numero dei cicli di boost
3.    Crea un tr. set D_j campionando con replacement in base a w
4.    Allena un classificatore C_j utilizzando D_j
5.    Applica C_j a tutti gli esempi contenuti in D
6.    ε_j=1/N Σ_i w_i δ(C_j(x_i)≠y_i)  // Calcola l'errore pesato
7.    if ε_j > 0.5 then
8.       w={w_i=1/N|i=1,2,…,N};  // reset!
9.    else
10.      α_j=1/2 ln((1-ε_j)/ε_j);
11.      Aggiorna i pesi w;
12.   end if;
13.end for;
14. C*(x) = arg max Σ_{j=1}^{T} α_j sign(C_j(x) = y) // sign restituisce 1/-1 se la
          classificazione è corretta/sbagliata
```

AdaBoost: un esempio

- Classificatore di base: albero decisionale binario a un livello
  - ✓ Può solo fare scelte del tipo x≤s →-1 x>s →1 dove s è lo split point
- Il data set

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

- L'accuratezza del classificatore base non può superare 70%
  - ✓ x≤0.3 →1 x>0.3 →-1
  - ✓ x≤0.7 →-1 x>0.7 →1

AdaBoost: un esempio

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

Split point: 0.75
ε_1 =0.03
α_1=1.738

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

Split point: 0.05
ε_2 =0.004
α_2=2.7784

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

Split point: 0.30
ε_3 =0.00027
α_3=4.1195

(a) Training records chosen during boosting

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

(b) Weights of training records

AdaBoost: un esempio

- Il dataset

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

- Il risultato

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

α_1=1.738
α_2=2.7784
α_3=4.1195

- ✓ 5.16 = -1.738 + 2.7784 + 4.1195
- ✓ Il classificatore non commette errori ed ha un comportamento ottenibile con un albero a due livelli