Classificazione e Predizione

Gianluca Amato

Corso di Laurea Specialistica in Economia Informatica
Università “G. D'Annunzio” di Chieti-Pescara
ultimo aggiornamento: 12/05/08

Introduzione ai concetti di Classificazione e Predizione

Classificazione e Predizione

- Classificazione e predizione sono processi che consistono nel creare dei modelli che possono essere usati
  - per descrivere degli insiemi di dati;
  - per fare previsioni future
- Molti algoritmi sono stati sviluppati per risolvere problemi di classificazione e predizione da settori diversi della comunità scientifica:
  - Apprendimento automatico (machine learning)
  - Statistica
  - Neurobiologia
- Con lo sviluppo del concetto di data mining si è posto particolare interesse al problema della scalabilità di questi algoritmi.

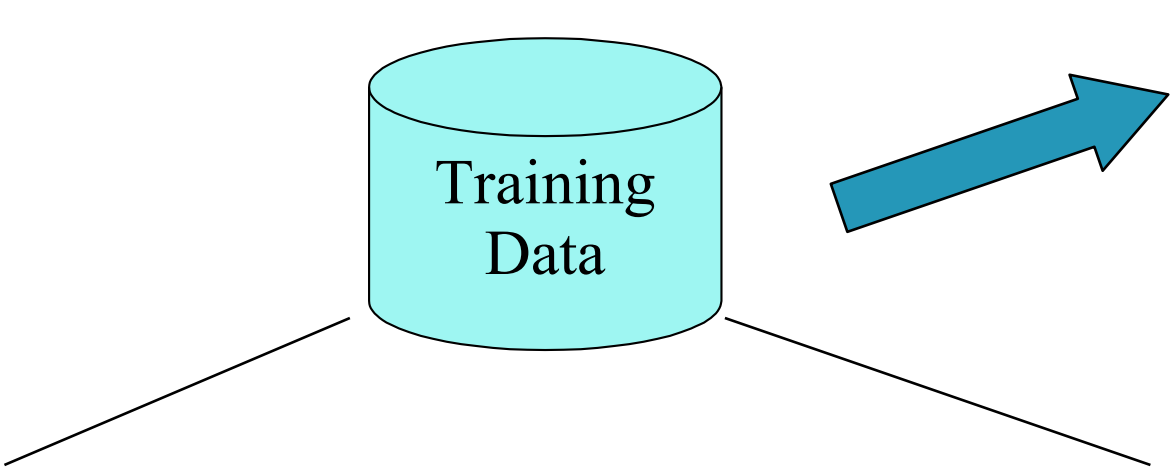
Classificazione (1)

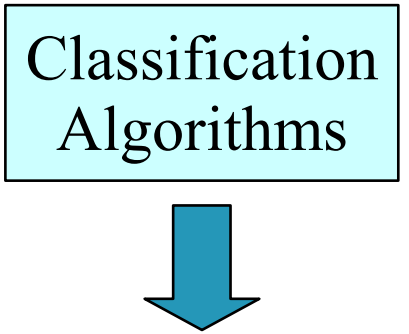
- Il processo di classificazione può essere visto come un processo a tre fasi.
- Fase 1 (addestramento): si produce un modello da un insieme di addestramento.
  - Si assume che ogni istanza in input faccia parte di una tra un numero predefinito di classi diverse.
  - La classe per ogni istanza si trova consultando un specifico attributo, chiamato "class label attribute" (attributo classificatore).
  - Proprio perché la classe di ogni istanza è fornita in input nella fase di addestramento, si parla apprendimento supervisionato.
    - Vedremo in futuro anche esempi di apprendimento non supervisionato.
  - Il risultato può essere presentato in varie forme: alberi di decisione, regole, formule matematiche, etc..

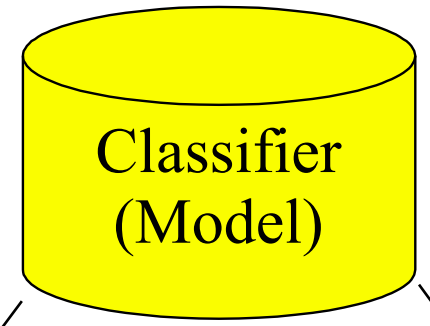
Classificazione (2)

- Fase 2 (stima dell'accuratezza): si stima l'accuratezza del modello usando un insieme di test.
  - ad esempio, si misura la percentuale di errori commessi sull'insieme di test
  - è importante che l'insieme di test sia indipendente dall'insieme usato per il campionamento, per evitare stime troppo ottimistiche.
- Fase 3 (utilizzo del modello): si classificano istanze di classe ignota.
  - Siamo di fronte a istanze di cui si conoscono tutti gli attributi tranne quello di classificazione

Costruzione del modello



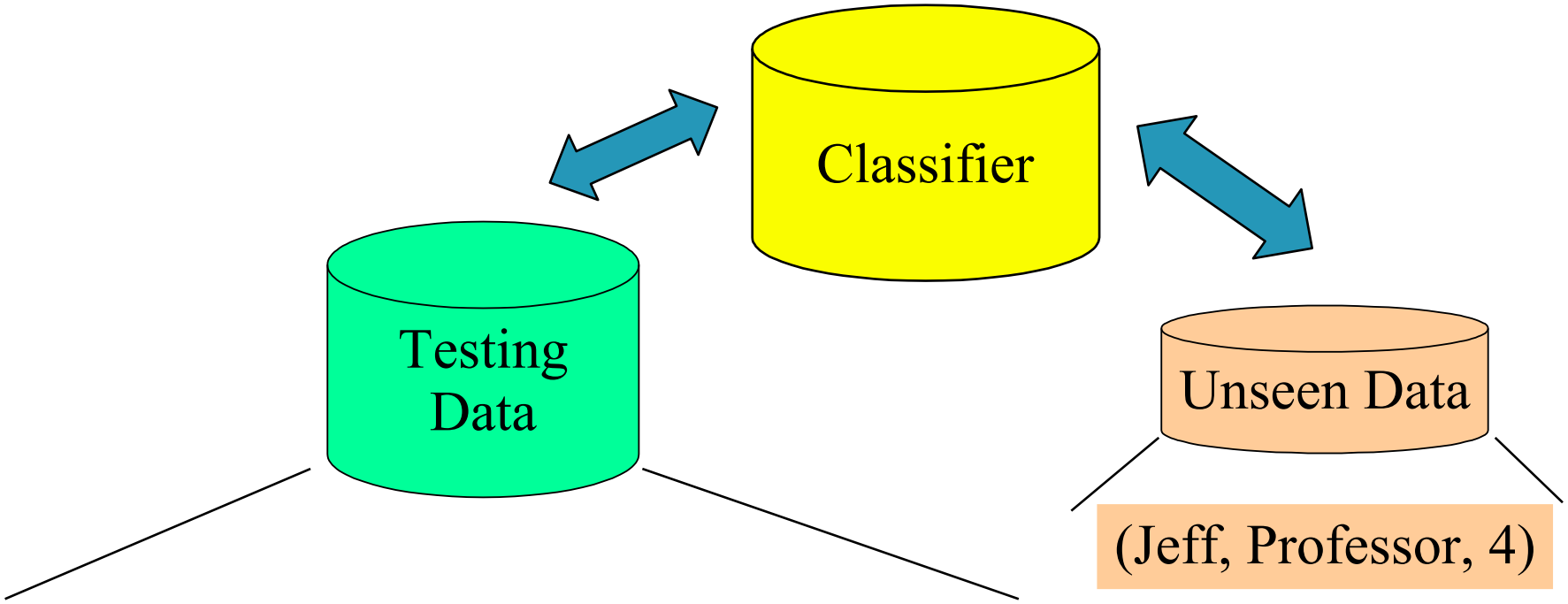




NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Test e Utilizzo del modello



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Classificazione vs Predizione

- Dal punto di vista etimologico, predizione sarebbe il concetto generale, che si divide in
  - Classificazione quando la classe è un valore nominale;
  - Regressione quando la classe è un valore numerico.
- Nella pratica però il termine Predizione viene usato come sinonimo di Regressione.

Problematiche legate ai metodi di classificazione e predizione

Preparazione dei dati (1)

- Data cleaning: pre-elaborare i dati in modo da
  - Eliminare il rumore
    - usando tecniche di smoothing
  - Eliminare eventuali outliers
    - dati con caratteristiche completamente diverse dal resto degli altri, probabilmente dovuti ad errori nei dati o a casi limite
  - Trattare gli attributi mancanti
    - Ad esempio, sostituendo un attributo mancante con la media dei valori per quell'attributo (nel caso di attributo numerico) o la moda (per attributi nominali)
- Anche se la maggior parte degli algoritmi di classificazione hanno dei meccanismi per eliminare rumore, outliers e attributi mancanti, una pulizia ad-hoc produce un risultato migliore.

Preparazione dei dati (2)

- Analisi di rilevanza degli attributi
  - Nota anche col termine feature selection dal termine usato nella letteratura di machine learning.
  - L'obiettivo è ottimizzare le prestazioni
    - L'idea è che il tempo impiegato per effettuare una analisi di rilevanza e l'addestramento sull'insieme di attributi ridotti è minore del tempo per effettuare l'addestramento su tutti gli attributi
  - Può anche migliorare la qualità del modello.
- Trasformazione dei dati
  - Ad esempio, generalizzare alcuni attributi secondo una gerarchia dei concetti...
  - ...oppure normalizzarne altri
    - Per esempio, ridurre il range di variabilità di un attributo numerico all'intervallo [0,1] sostituendo 0 al valore minimo, 1 al massimo e gli altri di conseguenza.

Valutazione degli algoritmi di classificazione

- Accuratezza nella predizione
- Velocità
  - Tempo per costruire il modello
  - Tempo per usarlo
- Robustezza
  - Abilità del modello di fare previsioni corrette anche in presenza di dati errati o mancanti
- Scalabilità
  - Caratteristica degli algoritmi che sono efficienti non solo per piccoli insiemi di dati ma anche per grossi database.
- Interpretabilità
  - Possibilità di assegnare un significato intuitivo al modello generato.

Classificazione con alberi di decisione

Classificazione con gli alberi di decisione

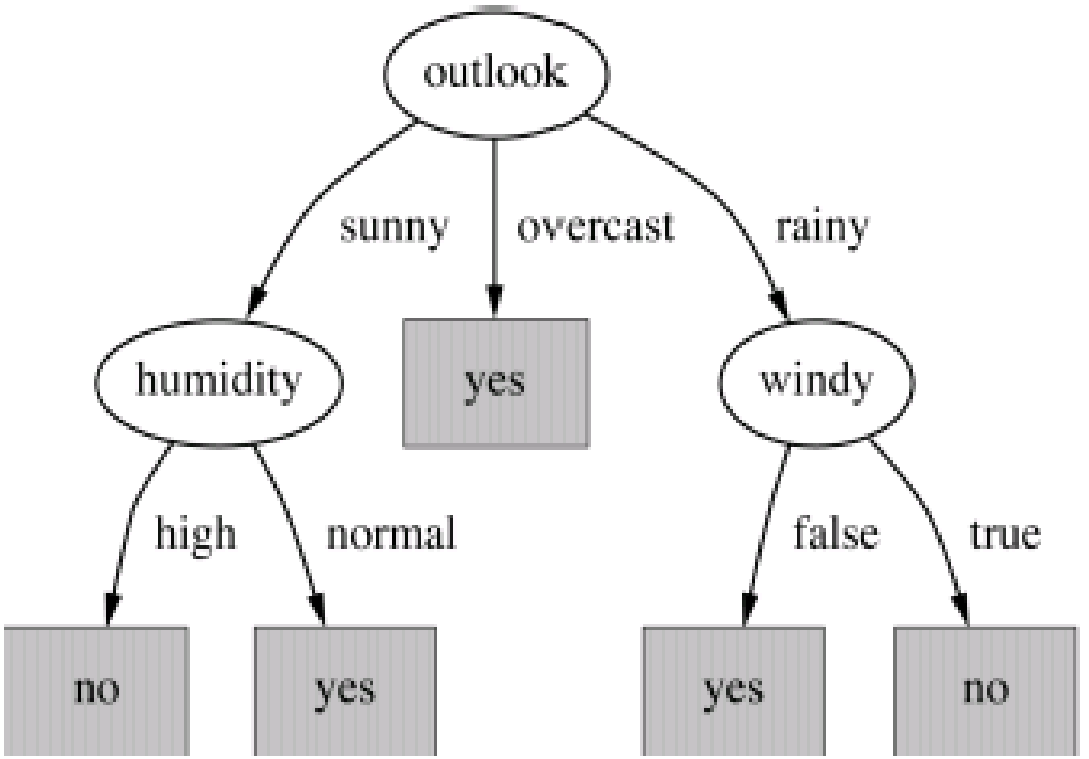
- Albero di decisione: un albero in cui
  - I nodi interni rappresentano test sugli attributi.
  - I rami rappresentano i risultati del test al nodo padre.
  - Le foglie rappresentano una classe o una distribuzione di probabilità per le classi.
- L'apprendimento è un processo a due fasi
  - Costruzione dell'albero
  - Sfoltimento (pruning) dell'albero
    - Identificare e rimuovere quei rami che rappresentano rumori nei dati o outliers.
- Uso dell'albero
  - Controllare i valori degli attributi della nuova istanza seguendo il percorso che parte dalla radice fino ad arrivare ad una foglia.

Insieme di addestramento

Il set di dati weather-nominal

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	No	No
Sunny	Hot	High	Yes	No
Overcast	Hot	High	No	Yes
Rainy	Mild	High	No	Yes
Rainy	Cool	Normal	No	Yes
Rainy	Cool	Normal	Yes	No
Overcast	Cool	Normal	Yes	Yes
Sunny	Mild	High	No	No
Sunny	Cool	Normal	No	Yes
Rainy	Mild	Normal	No	Yes
Sunny	Mild	Normal	Yes	Yes
Overcast	Mild	High	Yes	Yes
Overcast	Hot	Normal	No	Yes
Rainy	Mild	High	Yes	No

Albero di decisione



Algoritmo ID3 (1)

- È uno degli algoritmi “storici” per l'induzione di alberi di decisione.
  - È un algoritmo greedy (in ogni momento fa la “mossa” che massimizza la “bontà” dell'albero)
- L'albero è costruito in maniera top-down usando una politica nota come divide et impera (dividere un problema in sotto-problemi più piccoli)
  - si parte con un albero costituito dalla sola radice
    - alla radice sono assegnate tutte le istanze di addestramento
  - si sceglie un attributo (come?)
  - si creano tanti nodi (figli della radice) quanti sono i possibili valori dell'attributo scelto
    - ogni istanze di addestramento sarà assegnate al figlio appropriato
  - si procede ricorsivamente usando come radici i nuovi nodi

Algoritmo ID3 (2)

- Condizioni di arresto
  - tutte le istanze di un nodo appartengono alla stessa classe
    - il nodo diventa un foglia con la corrispondente classe assegnata
  - non ci sono più attributi sui quali dividere l'albero
    - il nodo diventa un foglia a cui viene assegnata la classe più comune tra le istanze ad esso assegnate
- Classi o distribuzioni di probabilità
  - in alternativa, invece che assegnare una classe ad un nodo, si può assegnare a distribuzione di probabilità dell'attributo classe, relativamente alle istanze ad esso assegnate.
    - Es: se all'ultimo nodo sono assegnate 3 istanze, con classi rispettivamente yes, yes e no, assegniamo al nodo la distribuzione di probabilità p tale che p(yes)=2/3 e p(no)=1/3.

Come scegliere l'attributo

- Lo schema generale visto si adatta alla maggior parte degli algoritmi di induzione di alberi di decisione.
  - cambia il metodo con cui si sceglie l'attributo.
  - l'obiettivo è comunque quello di ottenere un albero piccolo e preciso.
- Alcune possibilità:
  - guadagno di informazione: usato dall'algoritmo ID3
  - gain ratio
  - indici di Gini:usati da IBM Intelligent Miner

Guadagno di Informazione

- Procedimento simile a quello già visto:
  - Si considera l'insieme di istanze S associate al nodo da dividere, e si calcola l'entropia H(S).
  - Per ogni attributo A (categoriale) si calcola il guadagno di informazione $I(A)=H(S)- H(S|A)$
  - Si sceglie l'attributo che da il guadagno maggiore.

Esempio (1)

H(Istanze) = H(9/14,5/14)= 0.940 bit

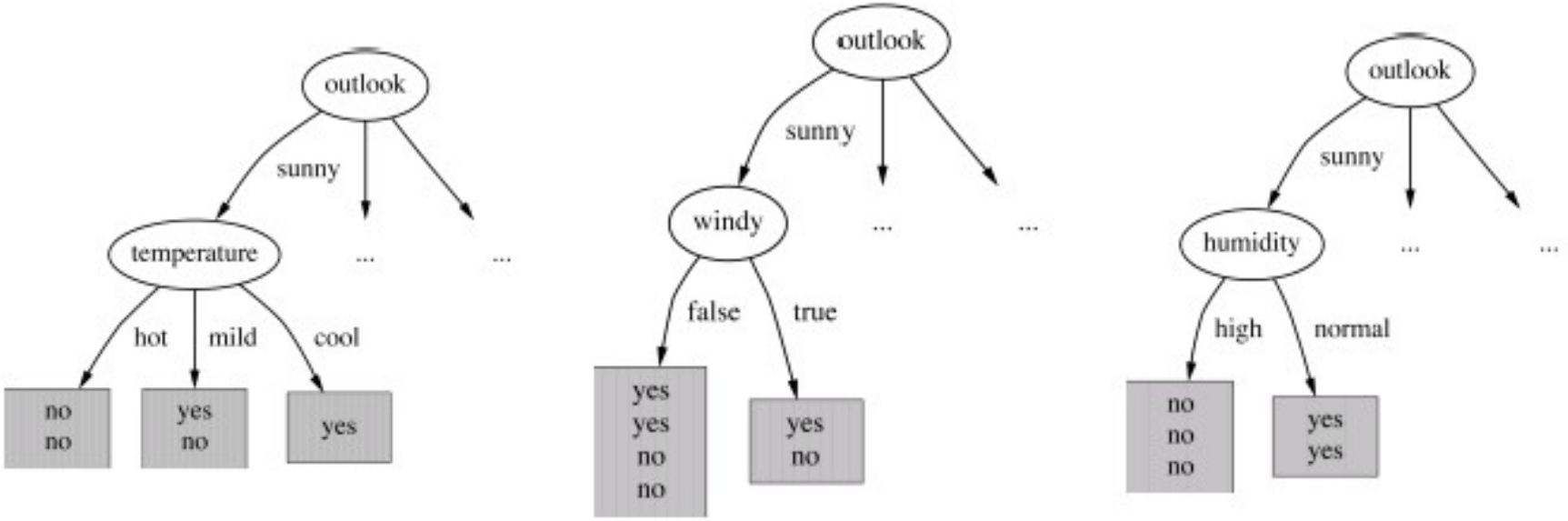
H(Istanze | Outlook=Sunny)=H(2/5,3/5)=0.971 bit
H(Istanze | Outlook=Overcast)=H(1,0)=0 bit
H(Istanze | Outlook=Rainy)=H(3/5,2/5)=0.971 bit
H(Istanze | Outlook)=5/14 * 0.971 + 4/14 *0 + 5/14 * 0.971= 0.693

Gain(Outlook)= 0.247
Gain(Temperature)=0.029
Gain(Humidity)=0.152
Gain(Windy)=0.048

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	No	No
Sunny	Hot	High	Yes	No
Overcast	Hot	High	No	Yes
Rainy	Mild	High	No	Yes
Rainy	Cool	Normal	No	Yes
Rainy	Cool	Normal	Yes	No
Overcast	Cool	Normal	Yes	Yes
Sunny	Mild	High	No	No
Sunny	Cool	Normal	No	Yes
Rainy	Mild	Normal	No	Yes
Sunny	Mild	Normal	Yes	Yes
Overcast	Mild	High	Yes	Yes
Overcast	Hot	Normal	No	Yes
Rainy	Mild	High	Yes	No

Esempio (2)

- Dobbiamo decidere l'attributo su cui dividere per il ramo Outlook=Sunny



- Gain(Temperature)=0.571
- Gain(Humidity)=0.971
- Gain(Windy)=0.020

Attributi a molti valori (1)

- Il guadagno di informazione ha un difetto: predilige gli attributi con molti valori diversi
  - più sono i sotto-insiemi generati per un attributo, più la loro cardinalità è bassa;
  - è più probabile che insiemi più piccoli siano puri (bassa entropia) rispetto insiemi più grandi
  - caso limite: se A è una chiave primaria del mio insieme di dati, allora $H(S|A)=0$ e il guadagno di inf. è massimo
- Rischio di overfitting

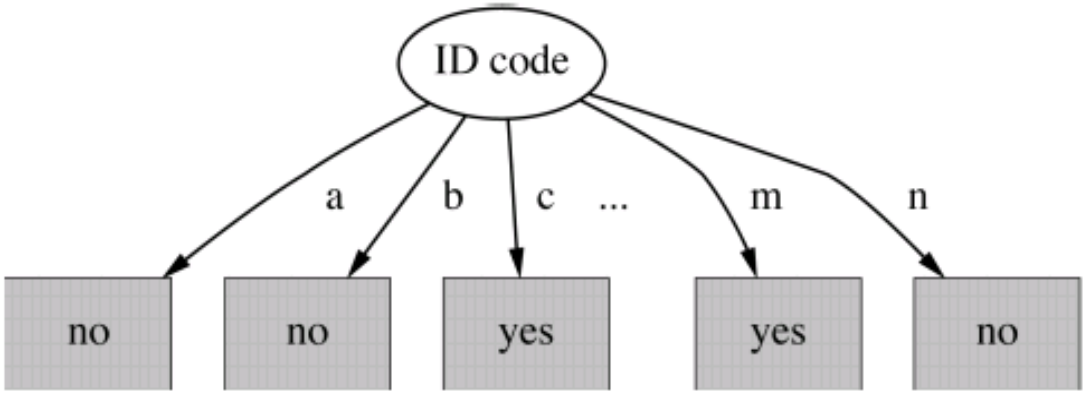
Attributi a molti valori (2)

Il set di dati weather-nominal con ID code

ID code	Outlook	Temp	Humidity	Windy	Play
A	Sunny	Hot	High	No	No
B	Sunny	Hot	High	Yes	No
C	Overcast	Hot	High	No	Yes
D	Rainy	Mild	High	No	Yes
E	Rainy	Cool	Normal	No	Yes
F	Rainy	Cool	Normal	Yes	No
G	Overcast	Cool	Normal	Yes	Yes
H	Sunny	Mild	High	No	No
I	Sunny	Cool	Normal	No	Yes
J	Rainy	Mild	Normal	No	Yes
K	Sunny	Mild	Normal	Yes	Yes
L	Overcast	Mild	High	Yes	Yes
M	Overcast	Hot	Normal	No	Yes
N	Rainy	Mild	High	Yes	No

Attributi a molti valori (3)

- Albero generato scegliendo ID code come attributo



- H(S)=0.941 bit
- H(S|A)=1/14 H(0,1) + ... + 1/14 H(1,0) + 1/14 H(0,1) = 0 bit
- Gain(A)=0.941 bit è il valore massimo

Gain ratio (1)

- Il Gain Ratio è una leggera modifica del guadagno di informazione che tiene conto del numero di valori degli attributi
  - in particolare, prende in considerazione la informazione intrinseca del partizionamento creato dall'attributo
- L'informazione intrinseca di un partizionamento di S negli insiemi S_1, ... S_n è l'entropia delle istanze, quando si condidera come classe l'indice i=1..n dell'insieme di appartenenza.
  - esempio: informazione intrinseca di ID code Intrinsic(A)=H(1/14, 1/14...., 1/14)= 3.807 bit
- Il gain ratio di un attributo si definisce come

$$GRatio(A) = \frac{Gain(A)}{Intrinsic(A)}$$

Gain ratio (2)

- Per il Gain Ratio si ottengono i seguenti risultati:
  - Outlook: 0.156 Temperature: 0.021
  - Humidity: 0.152 Windy: 0.049
  - ID code: 0.246
- Nonostante il nuovo metodo ID code è sempre superiore
  - occorre creare filtri ad hoc per rimuovere questi tipi di attributi
- Problemi con il Gain Ratio
  - può sovra-compensare, scegliendo un attributo solo perché la sua informazione intrinseca è bassa
  - si può risolvere il problema scegliendo l'attributo con Gain Ratio maggiore tra tutti quelli che hanno guadagno di informazione superiore alla media.

Sfoltire gli alberi

- L'albero generato può essere troppo specifico per l'insieme dei dati di addestramento (fenomeno detto overfit)
  - Ci sono troppi rami e alcuni riflettono soltanto il rumore dei dati di ingresso;
  - Il risultato è una scarsa accuratezza sulle istanze nuove.
- Sfoltire l'albero serve a rimuovere questi rami poco affidabili.
- Due approcci
  - Pre-pruning
  - Post-pruning

Pre-pruning

- Consiste nell'interrompere la costruzione dell'albero prima di costruirlo.
  - Si individua una misura del livello di associazione tra un attributo e la classe, in un determinato nodo dell'albero.
  - Si divide un nodo solo se esiste un attributo che ha un livello di associazione che supera una soglia prefissata.
  - Esempio: ci si può fermare quando il guadagno di informazione dell'attributo scelto è comunque sotto una certa soglia.
- Il metodo di pre-pruning ha vari problemi:
  - Come scegliere il valore di soglia?
  - Early stopping

Early stopping

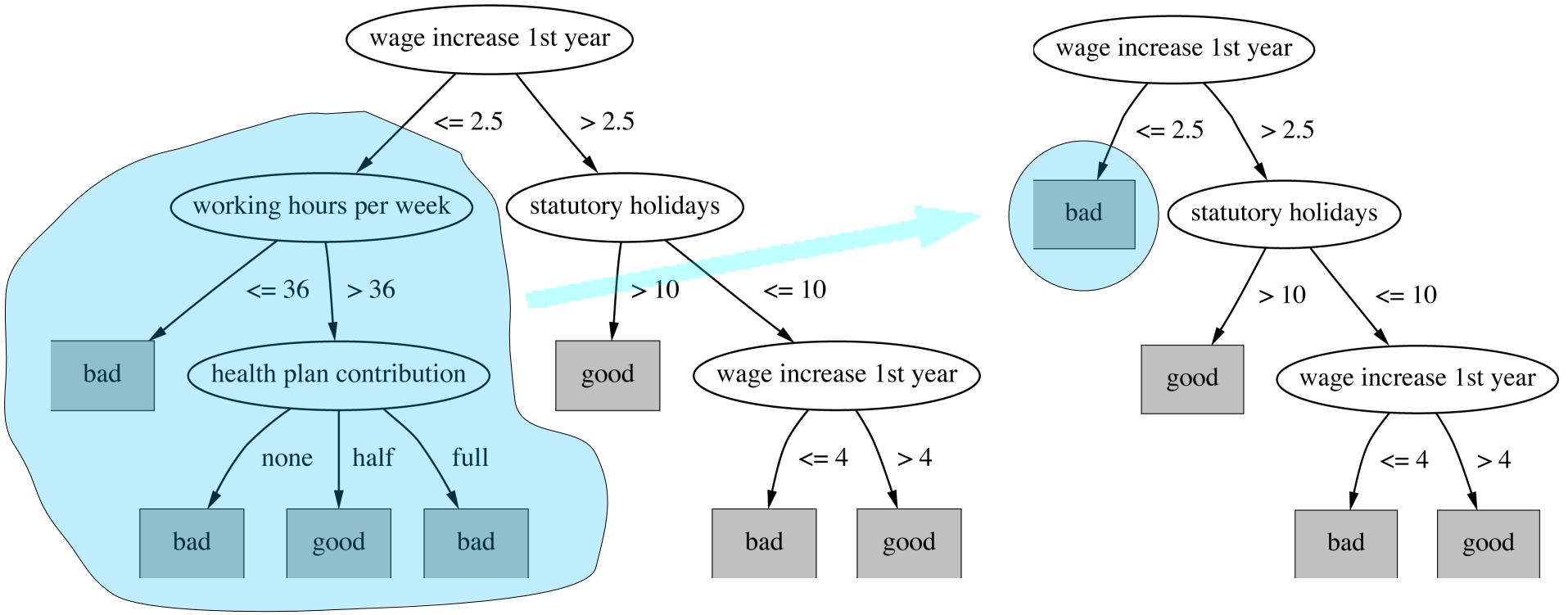
- Una interruzione prematura della costruzione dell'albero.
- Esempio classico, funzione XOR
  - Nessun attributo da solo è correlato in maniera significativa con il risultato dello XOR
  - Ma la coppia di input della funzione ha una correlazione fortissima.
  - Il prepruning rischierebbe di non accorgersi della situazione e non espandere l'albero radice.
- D'altronde, problemi di questo tipo sono rari.
- ... e gli algoritmi basati su pre-pruning sono più efficienti.

Post-pruning

- Post-pruning: costruisce tutto l'albero e poi rimuove i rami peggiori
  - È il metodo più comune.
  - Utilizzato anche in C4.5, un algoritmo che ha avuto grossa influenza nel campo della ricerca e che è una evoluzione di ID3.
- I punti importanti sono:
  - Le operazioni da applicare sull'albero per sfoltirlo
    - Subtree-replacement
    - Subtree-raising
  - Il criterio per decidere dove sfoltire

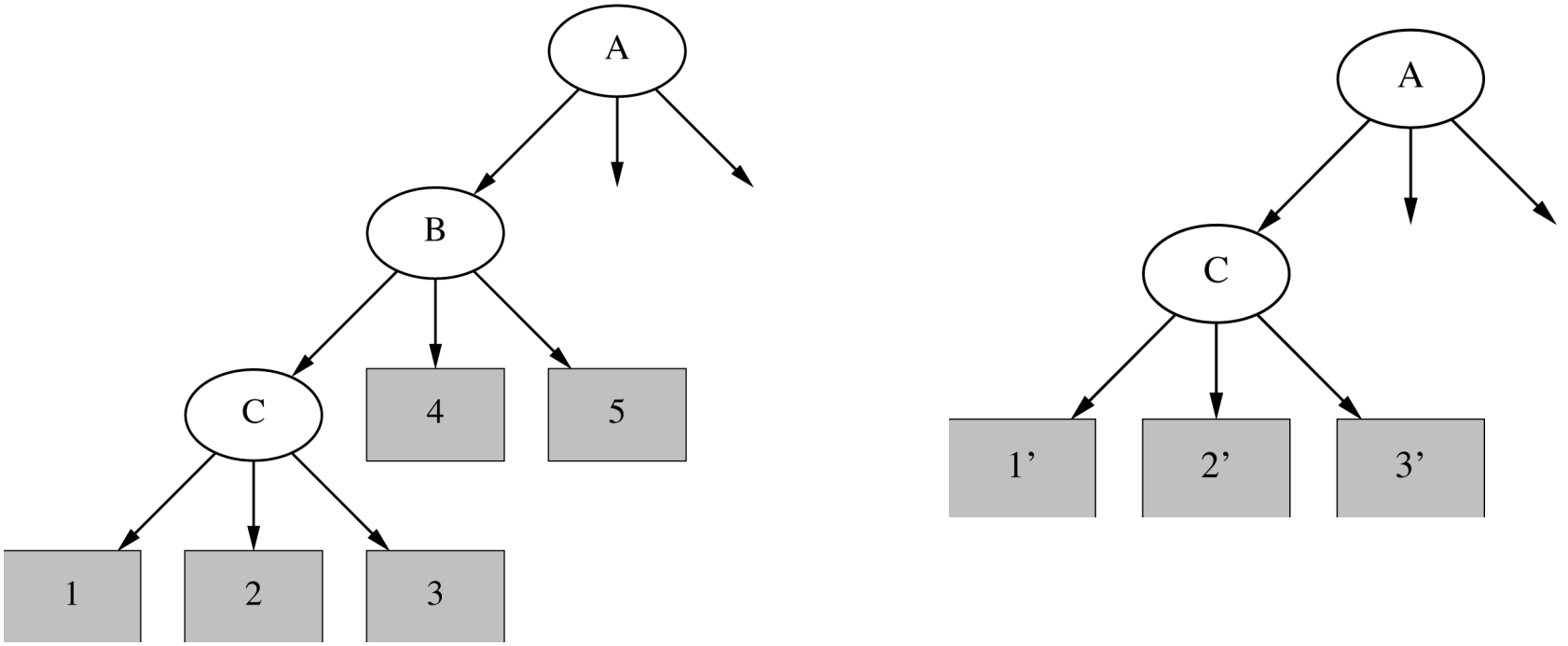
Subtree-replacement

- Subtree-replacement: l'idea è rimpiazzare un sottoalbero con una foglia.
  - Peggiora le prestazioni dell'albero sull'insieme di addestramento, ma può migliorarle su un insieme di dati indipendente.



Subtree-raising

- Subtree-raising: una operazione molto più costosa... e dalla utilità più discussa.
  - Nell'esempio, si elimina il nodo B e lo si sostituisce con C.
  - Le istanze nei sotto-albero 4 e 5 vanno riclassificate dentro C.
  - Di solito si applica quando C è il sotto-albero di B più numeroso.



Quando effettuare il post-pruning?

- Si utilizza un insieme di dati indipendente sia dal dato di testing che dal quello di training. Si parla di reduced-error pruning.
  - Si calcola il tasso di errore dell'albero prima e dopo l'operazione do pruning (calcolato sul nuovo insieme di dati).
  - Si sceglie di effettuare il pruning se il tasso di errore diminuisce.
- In effetti l'algoritmo C4.5 utilizza lo stesso insieme di addestramento per stimare il tasso di errore
  - considerazioni di tipo statistico per controbilanciare il fatto di usare lo stesso insieme di dati adoperato per costruire l'albero.

Trattamento di dati numerici (1)

- Estendiamo l'algoritmo ID3 per gestire dati numerici, limitandoci a test binari del tipo "A<*v*".
- Supponiamo di usare l'insieme di dati sul tempo atmosferico, nella versione con valori numerici. Dividendo le istanze in base alla temperatura, otteniamo

temp.:	64	65	68	69	70	71	72	75	80	81	83	85
classe:	yes	no	yes	yes	yes	no	yes	yes	no	yes	yes	no
							no	yes				

- Si dice che tra due valori consecutivi c'è uno split-point: è possibile separare l'insieme delle istanze in due sottoclassi, quelle con temperatura inferiore allo split-point è quelle con temperatura superiore.
- Tipicamente si posizionano gli split-point a metà tra due valori numerici consecutivi: 64.5, 66.6, 68.5, etc..

Trattamento di dati numerici (2)

- Per ogni valore di split-point *v* si calcola il guadagno di informazione che si ottiene separando i dati con il test "temperatura < *v*".
  - Per "temperatura < 71.5" abbiamo 6 istanze, 4 sì e 2 no
    - Dunque H(play | temperatura < 71.5)=H(4/6, 2/6)
  - Per "temperatura >= 71.5" abiamo 8 istanze, 5 sì e 3 no
    - Dunque H(play | temperatura >= 71.5)=H(5/8, 3/8)
  - Dunque l'entropia condizionata dallo split-point 71.5 è la media pesata delle entropie delle due classi:
    - H(play | split temperatura in 71.5) =
      6/14 * H(4/6,2/6) + 8/14 * H(5/8,3/8) = 0.939 bit
- Si calcola il guadagno di informazione per tutti gli split-point, e si sceglie il migliore, che diventa il guadagno di informazione dell'attributo temperatura.

Trattamento di dati numerici (3)

- Per il resto di procede normalmente. Se l'attributo temperatura è il più conveniente si procede a costruire l'albero col test "temperatura < *v*" dove *v* è lo split-point migliore.
- Una differenza con gli attributi nominali:
  - Non si incontrano mai due test sullo stesso attributo nominale in un singolo percorso da radice a foglia.
    - Non avrebbe senso perché il risultato del secondo test sarebbe sempre uguale al risultato del primo.
  - Si possono avere due test sullo stesso attributo numerico in un singolo percorso da radice a foglia, ma su split-point diversi.
    - Potrebbe rendere l'albero difficile da leggere.
  - Alternative:
    - Discretizzare l'attributo numerico prima di applicare l'algoritmo di classificazione
    - Test a più vie per valori numerici

Valori mancanti

- Un possibile approccio è considerare un valore mancante come un valore proprio, diverso da tutti gli altri
  - Ha senso quando il fatto che un valore manchi ha un significato preciso.
- L'algoritmo C4.5 divide una istanza con dati mancanti in pezzi.
  - Nel momento in cui si deve applicare un test sull'attributo X all'istanza I:
    - Supponiamo il test su X ha come risultato $\{v_1, ..., v_n\}$.
    - Supponiamo la percentuale di istanze in cui il test ha risultato v_i è p_i.
    - L'istanza I viene divisa “concettualmente” in tante istanze $I_1, .. I_n$.
    - Le istanze in I_i hanno peso p_i e danno risultato v_i
  - Si procede normalmente tenendo conto che adesso ogni istanza ha un peso. I concetti di information gain o gain ratio rimangono invariati, pur di considerare istanze pesate.

Regole di classificazione

Regole di classificazione

- Una regola di classificazione è una formula logica del tipo

  IF <antecedente> THEN <conseguente>

- L'antecedente è una serie di test, come i nodi di un albero di classificazione.
  - ... ma può anche essere una formula logica, in qualche formulazione evoluta.
- Il conseguente dà la classe da assegnare alle istanze che soddisfano l'antecedente.
- Esempio:
  - If outlook=sunny and humidity=high then play=yes

Alberi e regole di classificazione (1)

- Un albero di classificazione si può trasformare facilmente in un insieme di regole di classificazione:
  - Una regola è creata per ogni percorso dalla radice alle foglie
  - Ogni nodo interno del percorso è un test dell'antecedente.
  - La classe specificata sulla foglia è il conseguente.
- L'albero di decisione visto prima diventa:

  IF OutLook=”Sunny” AND Humidity=”High” THEN Play=”no”

  IF OutLook=”Sunny” AND Humidity=”Normal” THEN Play=”yes”

  IF OutLook=”Overcast” THEN Play=”yes”

  IF OutLook=”Rain” AND Windy=”true” THEN PLAY=”no”

  IF OutLook=”Rain” AND Windy=”false” THEN PLAY=”yes”
- Si può effettuare il pruning delle singole regole, eliminando le condizioni che non peggiorano (o magari che migliorano) l'accuratezza su un insieme di test distinto

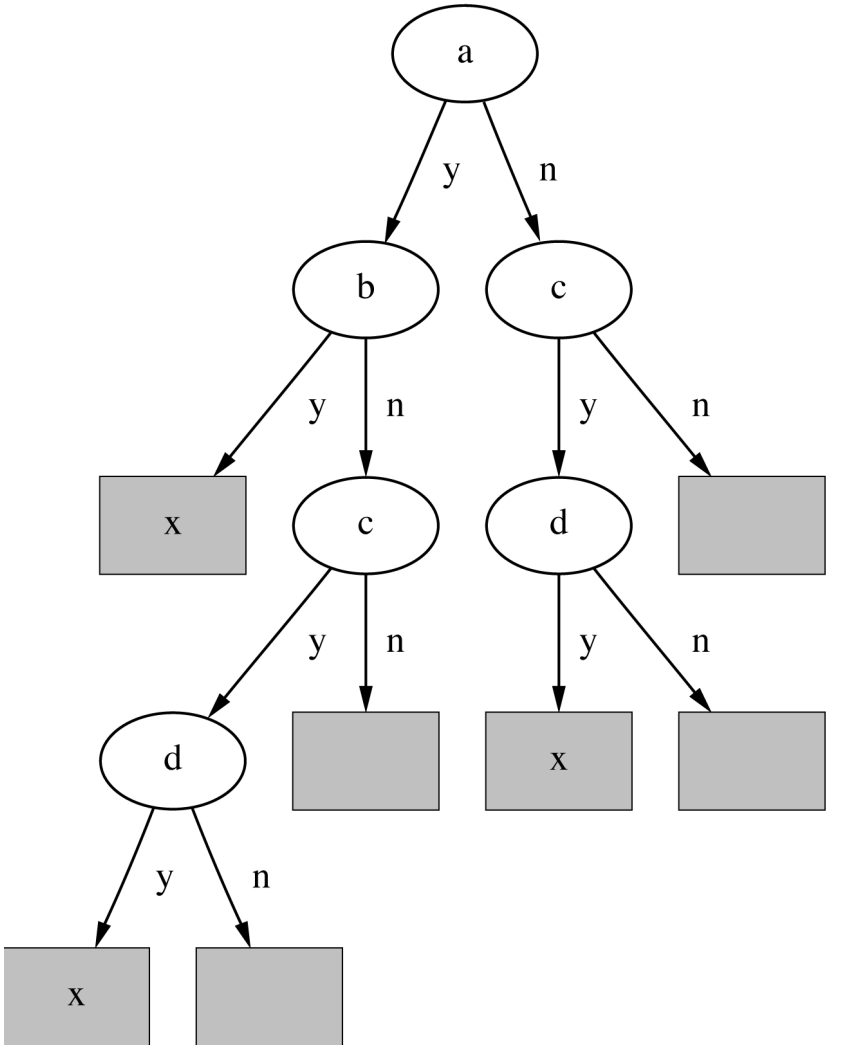
Alberi e regole di classificazione (2)

- Il procedimento inverso, da regole ad albero, non è così semplice, e può produrre alberi molto più di grossi del previsto.
  - In particolare, si ha il problema dei sottoalberi replicati
- Supponiamo di avere le seguenti regole

  if a and b then x

  if c and d then x

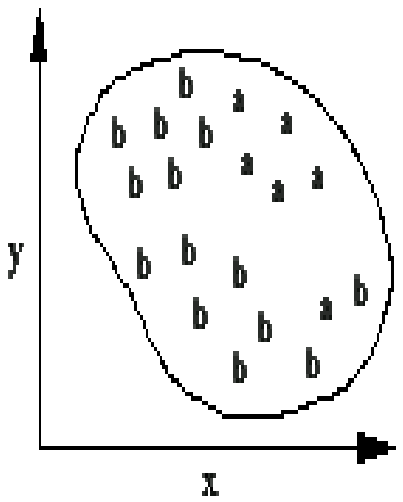
  si ottiene l'albero e destra.

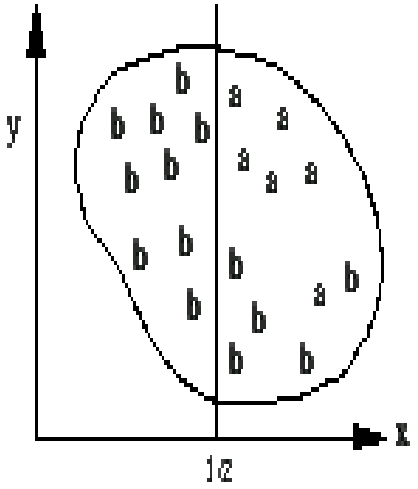


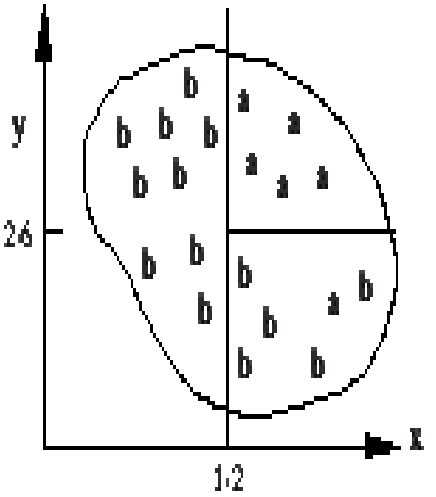
Algoritmi di copertura

- Spesso, la generazione di regole di decisione da alberi di copertura genera regole eccessivamente complicate.
- Ci sono algoritmi che generano direttamente un insieme di regole
  - Per ogni classe, trova l'insieme di regole che copre tutte le istanze di quella classe.
    - Determina una prima regola che copre alcune istanze della classe
    - Trova un'altra regola che copre alcune delle rimanenti..
    - ...e così via.
- Si parla di algoritmi di copertura proprio perché ad ogni passo coprono un sottoinsieme delle istanze della classe.

Esempio: generazione di una regola







If true then class = a

If x > 1.2 then class = a

If x > 1.2 and y > 2.6 then class = a

Copertura e Accuratezza

- Sia R una regola e
  - t: numero di istanze che soddisfano la premessa
  - p: numero di istanze che soddisfano premessa e conclusione
- Si definiscono allora i concetti di
  - Copertura: corrisponde al valore t
  - Accuratezza: corrisponde a p/t, ovvero la percentuale, tra le istanze che soddisfano la premessa, che soddisfano anche la conclusione
- Copertura e accuratezza corrispondo al supporto e alla confidenza per le regole associative.

Un semplice algoritmo di copertura

- Si parte da una regola di base senza condizioni.
- Si migliora la regola aggiungendo dei test che ne massimizzano l'accuratezza.
- Problema simile a quello degli alberi di decisione: decidere su che attributi dividere.
  - Gli alberi di decisione massimizzano la purezza della divisione e tengono tutte le classi in considerazione. I test hanno più di un possibile risultato.
  - I metodi di copertura massimizzano l'accuratezza della regole e considerano una sola classe. I test sono sempre binari.
- Ogni nuovo test riduce la copertura della regola.
- Ci si ferma se l'accuratezza è 1 o non si può dividere

Esempio: lenti a contatto (1)

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	No	Reduced	None
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Reduced	None
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Reduced	None
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	No	Reduced	None
Presbyopic	Myope	No	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Reduced	None
Presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Esempio: lenti a contatto (2)

- Regola cercata: if ? Then reccomendation = hard.
- Test possibili:
  - Age=Young 2/8
  - Age=Pre-presbyopic 1/8
  - Age=Presbyopic 1/8
  - Spectacle Prescription=Myope 3/12
  - Spectacle Prescription=Hypermetrope 1/12
  - Astigmatism=no 0/12
  - Astigmatism=yes 4/12
  - Tear Prod. Rate=reduced 0/12
  - Tear Prod. Rate=normal 4/12

Esempio: lenti a contatto (3)

- Regola migliore aggiunta
  - If astigmatism=yes then reccomendation=hard
- Istanza coperte dalla nuova regola

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Esempio: lenti a contatto (4)

- Stato corrente:
  - If astigmatism=yes and ? then reccomendation=hard
- Test possibili:
  - Age=Young 2/4
  - Age=Pre-presbyopic 1/4
  - Age=Presbyopic 1/4
  - Spectacle Prescription=Myope 3/6
  - Spectacle Prescription=Hypermetrope 1/6
  - Tear Prod. Rate=reduced 0/6
  - Tear Prod. Rate=normal 4/6

Esempio: lenti a contatto (5)

- Regola migliore aggiunta
  - If astigmatism=yes and tear prod. rate=normal then reccomendation=hard
- Istanze coperte dalla nuova regola

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Esempio: lenti a contatto (6)

- Stato corrente
  - If astigmatism=yes and tear prod. rate=normal and ? then reccomendation=hard
- Test possibili
  - Age=Young 2/2
  - Age=Pre-presbyopic 1/2
  - Age=Presbyopic 1/2
  - Spectacle Prescription=Myope 3/3
  - Spectacle Prescription=Hypermetrope 1/3
- Tra “Age=Young” e “Spectacle Prescription=Hypermetrope” scegliamo il secondo perché ha copertura maggiore.

Esempio: lenti a contatto (7)

- Regola finale:
  - If astigmatism=yes and tear prod. rate=normal and Spectacle Prescription=Myope then reccomendation=hard
- Seconda regola per “hard lenses”
  - ottenuta dalle istanze non coperte dalla prima regola
  - If age=young and astigmatism=yes and tear prod. rate=normal then reccomendation=hard.
- Queste regole coprono tutte le istanze delle lenti rigide.
  - Il processo è ripetuto per gli altri valori della classe.

L'algoritmo PRISM

- Quello che abbiamo descritto è l'algoritmo PRISM.
  - Se possibile, genera solo regole perfette, con accuratezza del 100%

```
Per ogni classe C
    inizializza E con l'insieme di tutte le istanze
    creare una regola R con una parte sinistra vuota che predice C
    finché R è perfetta (o non ci sono più attributi da usare)
        per ogni attributo A non menzionato in R, e ogni valore v
            considera di aggiungere la regola A=v al lato sinistro di R
            seleziona il valore v che massimizza l'accuratezza
                (in caso di più valori massimali, considera quello che
                massimizza anche il supporto)
            aggiungi A=v alla regola R
        rimuovi le istanze coperte da R in E
```

Regole contraddittorie? (1)

- Se ci sono due istanze uguali, tranne per l'attributo classe, vengono generate regole contraddittorie:

X	Y	Classe
a	b	y
a	b	n
a	b	n
a	a	y
b	b	n

- si generano le seguenti regole:
  - `if Y=a then Classe=y`
  - `if X=a and Y=b then Classe=y`
  - `if X=b then Classe=n`
  - `if Y=b and X=a then Classe=n`

Regole contraddittorie (2)

- Le regole 2 e 4 sono contraddittorie!
  - cosa fare quando vogliamo classificare l'istanza X=a, Y=b?
- Soluzione:
  - considero le regole nell'ordine in cui le ho determinate, ed applico la prima la cui condizione è verificata
    - in questo caso, classifico X=a, Y=b come classe y
  - in alternativa, potrei memorizzare l'accuratezza di tutte le regole:
    - la regola `if X=a and Y=b then Classe=y` ha accuratezza 1/3
    - la regola `if Y=b and X=a then Classe=n` ha accuratezza 2/3
    - considero tutte le regole che si possono applicare, è scelgo quella che ha accuratezza migliore
    - in questo caso, classificherei X=a, Y=b come classe n

Separate and Conquer

- Metodi come PRISM sono noti come algoritmi separate and conquer.
  - Viene identificata una regola
  - Le istanze coperte dalla regola vengono separate dal resto
  - Le istanze che rimangono vengono conquistate.
- La differenza con i metodi divide and conquer (come gli alberi di decisione)
  - I sottoinsiemi separati non devono più essere considerati.

Classificatori Bayesiani

Cosa sono i classificatori Bayesiani?

- Sono metodi statistici di classificazione.
- Hanno le seguenti caratteristiche:
  - Predicono la probabilità che una data istanza appartenga ad una certa classe.
  - Sono metodi incrementali: ogni istanza dell'insieme di addestramento modifica in maniera incrementale la probabilità che una ipotesi sia corretta.
    - La conoscenza già acquisita può essere combinata facilmente con le nuove osservazioni (basta aggiornare i conteggi).
    - Usati, ad esempio, in Mozilla, Thunderbird o SpamAssassin per riconoscere lo spam dalle mail “buone”.

Funzionamento dei classificatori Bayesiani

- Sia X una istanza da classificare, e $C_1,...C_n$ le possibili classi. I classificatori Bayesiani calcolano $P(C_i \mid X)$ come

$$P(C_i|X)=\frac{P(X|C_i)P(C_i)}{P(X)}.$$

- Si sceglie la classe C_i che massimizza $P(C_i|X)$.
  - P(X) è uguale per tutte le classi per cui non occorre calcolarla;
  - $P(C_i)$ si può calcolare facilmente sull'insieme dei dati di addestramento
    - si contano la percentuale di istanze di classe C_i sul totale
  - come si calcola $P(X \mid C_i)$?

Classificatori naive

- Assunzione dei classificatori naive: indipendenza degli attributi.
- Se X è composta dagli attributi $A_1=a_1 ... A_m=a_m$, otteniamo

$$P(X|C_i)=\prod_{j=1}^{m} P(A_j=a_j|C_i)$$

- Se A_j è categorico, $P(A_j=a_j \mid C_i)$ viene stimato come la frequenza relativa delle istanze che hanno valore $A_j=a_j$ tra tutte le istanze di C_i
- Se A_j è continuo, si assume che $P(A_j=a_j \mid C_i)$ segua una distribuzione Gaussiana, con media e varianza stimata a partire dalla istanze di classe C_i.

Esempio (1)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	**P(sunny|n) = 3/5**
P(overcast|p) = 4/9	**P(overcast|n) = 0**
P(rain|p) = 3/9	**P(rain|n) = 2/5**
temperature	
P(hot|p) = 2/9	**P(hot|n) = 2/5**
P(mild|p) = 4/9	**P(mild|n) = 2/5**
P(cool|p) = 3/9	**P(cool|n) = 1/5**
humidity	
P(high|p) = 3/9	**P(high|n) = 4/5**
P(normal|p) = 6/9	**P(normal|n) = 2/5**
windy	
P(true|p) = 3/9	**P(true|n) = 3/5**
P(false|p) = 6/9	**P(false|n) = 2/5**

Esempio 2

- Arriva un nuovo campione X = <rain, hot, high, false>
  - $P(X|p)\cdot P(p) = P(rain|p)\cdot P(hot|p)\cdot P(high|p)\cdot P(false|p)\cdot P(p) = 3/9\cdot 2/9\cdot 3/9\cdot 6/9\cdot 9/14 = 0.010582$
  - $P(X|n)\cdot P(n) = P(rain|n)\cdot P(hot|n)\cdot P(high|n)\cdot P(false|n)\cdot P(n) = 2/5\cdot 2/5\cdot 4/5\cdot 2/5\cdot 5/14 = 0.018286$
- Il campione X è classificato nella classe n (don't play) in quanto la verosimiglianza per N è più elevata

Il problema delle frequenze nulle

- Cosa succede se un certo valore di un attributo non si verifica mai per una data classe (ad esempio Humidity non è mai high per la classe p)
  - allora P(humidity=”high” | p)=0
  - quando arriva una nuova istanza X con humidity=high, sarà sempre P(p | X)=0 indipendetemente da quanto siano probabili i valori per gli altri attributi.
- Per risolvere questo problema si usa sommare 1 ai conteggi per le coppie (attributo, valore).
  - in questo modo nessuna probabilità è mai nulla

Valori mancanti

- Per le istanze di addestramento
  - nel contare la probabilità dell'evento Attributo=valore, non si considerano le istanze con Attributo mancante.
- Per le istanze da classificare
  - si calcola la verosimiglianza solo sulla base degli attributi noti
  - dato X=<?, cool, high, true> si ha
    - verosimiglianza di “p” = 3/9 * 3/9 * 3/9 * 9/14 = 0.0238
    - verrosimiglianza di “n” = 1/5 * 4/5 * 3/5 * 5/14 = 0.0343
    - P(p|X)=0.0238 / (0.0238+0.0343)=41%
    - P(n |X)=59%

Il caso continuo (1)

- Vediamo un esempio di utilizzo del classificatore naive Bayes con valori continui

Outlook			Temperature			Humidity			Windy		Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3		83	85		86	85	False	6	2	9	5
Overcast	4	0		70	80		96	90	True	3	3		
Rainy	3	2		68	65		80	70					
				...	...		...	...					
Sunny	2/9	3/5	*mean*	73	74.6	*mean*	79.1	86.2	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	*std dev*	6.2	7.9	*std dev*	10.2	9.7	True	3/9	3/5		
Rainy	3/9	2/5											

- Esempio: $f(temperature = 66 | Y) = \frac{1}{\sqrt{2\pi}\,6.2} e^{\frac{-(66-73)^2}{2*6.2^2}}$

Il caso continuo (2)

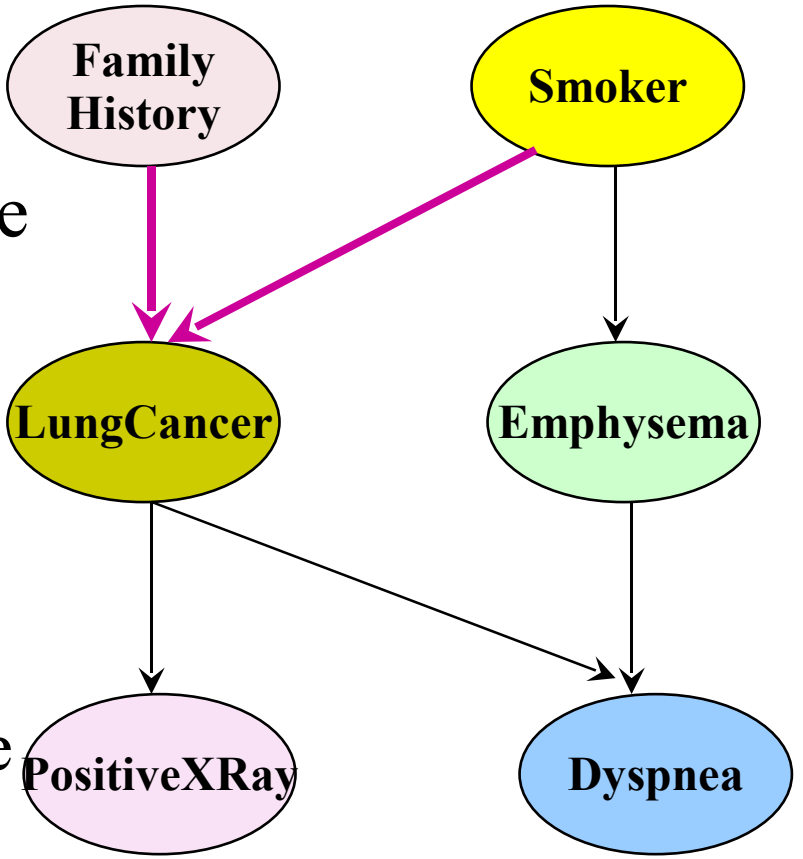
- Supponiamo di avere la nuova istanza X=<sunny,66,90,true>
  - Verosim. di “p”=2/9 * 0.0340 * 0.0221 * 3/9 * 9/14 = 0.000036
  - Verosim. di “n”=3/5 * 0.0291 * 0.0380 * 3/5 * 5/14 = 0.000136
  - P(p | X)=0.000036/(0.000036+0.000136)=20.9%
  - P(n | X)=79.1%
- Valori mancanti
  - nell'insieme di addestramento: i valori mancanti non vengono considerati per la stima di media e varianza.
  - nella nuova istanza: come per gli attributi categoriali, non si considera l'attributo mancante.

Vantaggi e svantaggi

- L'assunzione di indipendenza
  - rende i calcoli possibili
  - consente di ottenere classificatori ottimali quando è soddisfatta...
  - ...ma è raramente soddisfatta in pratica
- Una possibile soluzione: reti Bayesiane.
  - Consentono di considerare le relazioni causali tra gli attributi.
- In realtà, si è visto che anche quando l'ipotesi di indipendenza non è soddisfatta, il classificatore naive Bayes fornisce spesso ottimi risultati.

Reti Bayesiane (1)

- Uno dei componenti di una rete bayesiana è un grafo diretto aciclico nel quale ogni nodo è una var. casuale e ogni arco rappresenta una dipendenza probabilistica.
- Sia X un nodo e **Y** l'insieme dei suoi genitori:
  - X è indipendente da tutte le variabili aleatorie che non discendono da X se sono noti i valori di **Y**.
  - Se sono noti i valori di FamilyHistory e Smoker, allora LungCancer è indipendente da Emphysema



Reti Bayesiane (2)

- Un altro componente di una rete Bayesiana è la tabella di probabilità condizionata.

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

- Esprime i valori di P(LungCancer | FamilyHistory, Smoker).
- La probabilità totale di una tupla (z_1,... z_n) corrispondente alle variabili casuali Z_1...Z_n è

$$P(z_1, \ldots, z_n) = \prod_{i=1}^{n} P(z_i | Parents(Z_i)).$$

Fase di apprendimento

- Gli algoritmi di apprendimento per le reti bayesiane si differenziano a seconda di due caratteristiche:
  - struttura della rete data a priori oppure no;
  - presenza di "variabili nascoste":
    - sono nodi che non corrispondono né a un attributo del nostro insieme di dati né a un attributo classe.
- Struttura della rete data a priori:
  - assenza di variabili nascoste: l'addestramento avviene come per i metodi bayesiani naive
  - presenza di variabili nascoste: metodi di "discesa del gradiente", in maniera simile a quanto si fa per le reti neurali.
- Struttura della rete ignota a priori:
  - vari algoritmi di apprendimento che non esaminiamo

Fase di classificazione

- Esaminiamo solo la fase di classificazione in assenza di "variabili nascoste".
  - sia Y è l'attributo classe e **X** il vettore degli altri attributi
  - in presenza di una nuova istanza **x** si vuole calcolare, per ogni valore di classe y, la

$$P(Y=y|\boldsymbol{X}=\boldsymbol{x})=P(Y=y,\boldsymbol{X}=\boldsymbol{x})/P(\boldsymbol{X}=\boldsymbol{x})$$

  - siccome P(**X**=**x**) è costante per tutte le y, basta calcolare la verosimiglianza P(Y=y, **X**=**x**)
  - si sceglie come risultato la classe che ha verosimiglianza maggiore

Metodi basati sulle istanze

Metodi basati sulle istanze

- Memorizzano le istanze che fanno parte dell'insieme di addestramento e rimandano tutte le elaborazioni al momento in cui una nuova istanza deve essere classificata.
  - Si parla di lazy evaluation (valutazione pigra) contrapposta alla eager evalution degli altri metodi.
- Esempi tipici: *k*-nearest neighbor, case-based reasoning.
- Vantaggi:
  - Tipicamente danno luogo a metodi incrementali
  - Hanno accuratezza maggiore perché lo "spazio delle ipotesi" è più grande
- Svantaggi:
  - Computazionalmente pesante

k-nearest neighbor (1)

- Supponiamo le istanze siano formate da *n* attributi (escluso l'attributo classe)
  - Ogni istanza è un punto in uno spazio n-dimensionale.
  - Tra due istanze si definisce una distanza con la solita formula

$$d(X,Y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

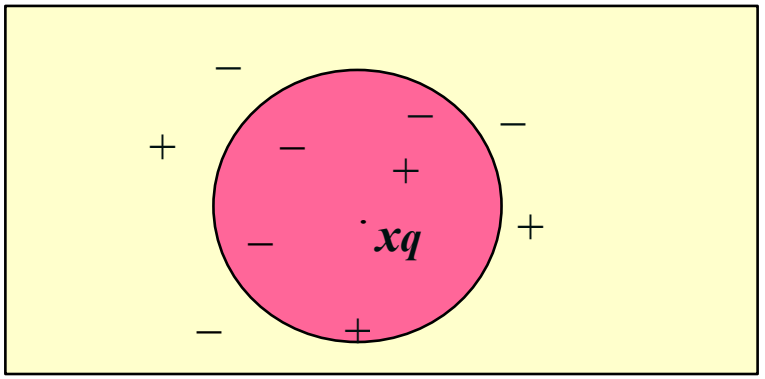
    - nasce la necessità di normalizzare i dati in modo da dare a tutti gli attributi lo stesso peso
  - Se gli attributi sono discreti ($x_i - y_i$) si definisce come 0 se i due valori sono uguali, 1 altrimenti...
    - ma ci sono altre scelte possibili..

k-nearest neighbor (2)

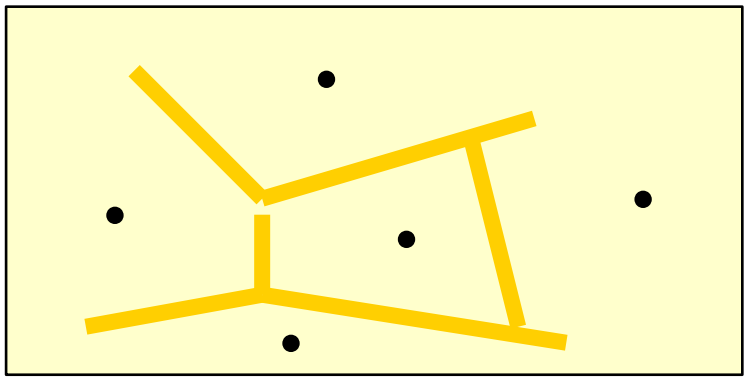
- Data una nuova istanza da classificare
  - si considerano le *k* istanze dell'insieme di addestramento più vicine
  - la classe predetta è quella più comune tra queste *k* istanze.
- Vantaggi:
  - Robustezza
  - Incrementalità
- Svantaggi:
  - Alto costo computazionale durante l'uso
    - Richiede tecniche di indicizzazione sofisticate per trovare i possibili "vicini" senza dover cercare tutte le istanze
  - Gli attributi irrilevanti vengono pesati come gli altri.. può causare instabilità.
    - Necessità di una fase iniziale che elimini quelli irrilevanti

Esempio

- Il punto x_q nella figura a destra:
  - è classificato come + se k=1
  - è classificato come – se k=5



- Lo spazio delle ipotesi viene diviso in poligoni convessi
  - ogni poligono contiene istanze che vengono classificate allo stesso modo
  - a destra viene visualizzata la superficie di decisione indotta da un classificatore 1-NN (diagrammi di Vonoroi)



Case-based reasoning

- Le istanze non sono rappresentate da punti nello spazio euclideo ma da complesse descrizioni simboliche (ad esempio da grafi).
  - Usato quando gli oggetti da classificare sono complessi, non trattabili come un semplice insieme di attributi:
    - Sentenze
    - Disegni tecnici
  - Le istanze sono trasformate in queste strutture complesse e immagazzinate.
  - Se si vuole classificare una nuova istanza, la si trasforma nella descrizione simbolica e si usano quelle più simili per determinare la classe.
    - La somiglianza non è data da una semplice distanza, ma da altre proprietà (ad esempio, avere sottografi comuni).

Altri metodi di classificazione

Reti neurali

- Vantaggi
  - La predizione è generalmente molto accurata
  - Metodo robusto: è poco sensibile a rumore nei dati
- Svantaggi
  - Tempi di addestramento molto lunghi
  - Difficile comprendere il significato del modello addestrato
    - che significato intuitivo dare a tutti i pesi?
    - recentemente sono stati proposti metodi per estrarre regole da reti neurali addestrate.
  - Non è facile da integrare con conoscenze a priori.

Regole associative

- Le regole associative possono essere utilizzate per la classificazione.
  - age =young & student=no => buys_computer=yes [sup: 30%, conf: 80%]
  - se buys_computer è l'attributo classe, allora la regola di prima da un criterio per classificare una istanza!
- Algoritmo CBA
  - determina tutte le regole del tipo itemset => y dove y è una possibile classe
  - ordina le regole sulla base del supporto e della confidenza
  - le nuove istanze vengono classificate sulla base della prima regola che le soddisfa.

Pattern emergenti (1)

- Un pattern emergente per una classe C è un itemset il cui supporto aumenta di molto spostandosi da una classe all'altra.
  - Sia C_1 la classe con “buys_computer=no” e C_2 la classe “buys_computer=yes”
  - L'itemset {age=young, student=no} è un tipico pattern emergente:
    - Il supporto è 0.2% su C_1 e 57.6% su C_2
  - Il rapporto tra i due supporti è il tasso di crescita.
    - Determina il “potere discriminante” del pattern emergente.

Pattern emergenti (2)

- CAEP (classification by aggregating emerging patterns):
  - durante l'addestramento
    - per ogni classe C, determina i pattern emergenti di C, ovvero quelli che hanno un supporto molto alto in C rispetto alle altre classi
  - quando deve classificare una nuova istanza X
    - per ogni classe C si determina un punteggio in base ai pattern emergenti per C che occorrono in X
    - si sceglie la classe che ha punteggio più elevato

Predizione

Predizione e regressione

- Il metodo più comune per affrontare problemi di predizione è il metodo della regressione, noto dalla statistica.
- Il più semplice metodo di regressione è la regressione lineare semplice:
  - date delle istanze con attributi numerici X e Y, si vuole determinare Y come funzione lineare di X, ovvero Y=a + bX.
  - i coefficienti a e b, noti come coefficienti di regressione, possono essere determinati con il metodo dei minimi quadrati, ovvero minimizzando l'errore

$$\sum_i (a + b\,x_i - y_i)^2$$

dove x_i e y_i sono i valori di X e Y per l'istanza *i*-esima.

Perché i minimi quadrati ? (1)

- Assumiamo di avere
  - fissato un insieme x_i di valori per la variabile indipendente
  - Y_i sia una variabile casuale ottenuta da x_i tramite una funzione f, sommando poi un termine di errore e_i
    - ovvero: $Y_i = f(x_i) + e_i$
  - e_i è una variabile casuale gaussiana di media nulla e varianza σ^2
  - le variabili e_i sono tra loro indipendenti
    - quindi lo stesso vale delle Y_i
- Assumiamo di osservare, per la variabile casuale Y_i, il valore y_i
- Vogliamo calcolare la stima di massima verosimiglianza per la funzione f.

Perché i minimi quadrati? (2)

- La stima di massima verosimiglianza è che quella che massimizza la probabilità di ottenere i valori y_i
- poiché le Y_i sono indipendenti, abbiamo
  - $P(Y_1=y_1, \ldots Y_n=y_n) = \Pi_{i=1..n} P(Y_i=y_i)$
  - $P(Y_i=y_i)=\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(y_i-f(x_i))^2}$
- La stima di massima verosimigliava è la funzione f che massimizza $\Pi_{i=1..n} P(Y_i=y_i)$
  - poiché la funzione logaritmo è monotona, essa è equivalente alla funzione che massimizza $\Sigma_{i=1..n} \log P(Y_i=y_i)$

Perché i minimi quadrati? (3)

- Si tratta allora di massimizzare

$$\sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(y_i - f(x_i))^2$$

- Eliminando le costanti che non dipendono da f è cambiando segno, ciò è equivalente a minimizzare

$$\sum_{i=1}^{n} (y_i - f(x_i))^2$$

ovvero proprio la somma dei minimi quadrati

Risoluzione dei minimi quadrati

- se f(x)=a+bx, la minimizzazione della verosimiglianza può essere risolta in modo analitico ottenendo le stime

$$b = \frac{\sum_{i=1}^{n} (y_i - \bar{y}) x_i}{\sum_{i=1}^{n} x_i (x_i - \bar{x})} \qquad a = \bar{y} - b\,\bar{x}$$

dove $\bar{x}$ e $\bar{y}$ sono i valori medi degli x_i e degli y_i.

Altri tipi di regressione

- Regressione lineare multipla
  - $Y=a + b_1X_1 + ... + b_nX_n$
- Regressione non-lineare
  - Ad esempio, $Y=a + b_1X + b_2X^2 + b_3X^3$
  - È spesso possibile ricondurre il tutto a un problema di regressione lineare introducendo delle nuove variabili:
    - Ad esempio poniamo $X_1=X$, $X_2=X^2$, $X_3=X^3$
    - Il problema diventa $Y = x + b_1X_1 + b_2X_2 + b_3X_3$

Regressione e classificazione

- È possibile utilizzare metodi di regressione anche per problemi di classificazione
  - addestramento: per ogni possibile modalità dell'attributo classe, si determina un nuovo attributo che è la variabile caratteristica o funzione indicatrice di quella modalità. Per ognuno di questi nuovi attributi si calcola una diversa curva di regressione.
  - utilizzo: quando disponiamo di una nuova istanza da classificare, si predice il valore delle variabili caratteristiche utilizzando le curve di regressione. Alla fine, si sceglie la classe la cui funzione indicatrice ha una predizione più alta.
- Altri metodi di regressione appositamente studiati per problemi di classificazione:
  - regressione logistica

k-nearest neighbor e predizione

- Il metodo *k*-nearest neighbor può essere utilizzato anche per problemi di predizione.
  - Si calcola la media delle classi delle *k* istanze più vicine, invece della moda.
- In alternativa, per problemi di predizione si può utilizzare il metodo della regressione locale pesata (weighted local regression)
  - data una istanza x, si considerano i k punti più vicini a x
  - si effettua una regressione sulla base di questi punti
    - eventualmente pesando in maniera diversa punti lontani e punti vicini
  - si ottiene come risultato il valore della funzione di regressione sulla istanza x.

Conclusioni

Conclusioni

- La classificazione è un problema studiato diffusamente
  - principalmente nelle comunità di ricerca di statistica, apprendimento automatico e reti neurali.
- La classificazione è uno degli strumenti più usati.
- La scalabilità è un problema fondamentale nell'uso dei classificatori per grandi insiemi di dati.
  - Ci si aspettano buoni risultati dalla combinazione delle tecniche attuali con tecniche derivate dalle basi di dati.
- Direzioni di ricerca: classificazione per dati non-relazionali, come testo, dati spaziali, dati multimediali, etc..

Bibliografia

- Ian H. Witten, Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.
  - capitolo 4, 6
- Jaiwei Han, Micheline Kamber. *Data Mining: Concepts and Techniques (2nd edition)*. Morgan Kaufmann
  - capitolo 6
- Remco R. Bouckaert. *Bayesian Network Classifiers in Weka*. http://www.cs.waikato.ac.nz/~remco/weka.bn.pdf
  - sezione 1